

N° d'ordre : 397/2021



UNIVERSITÉ SULTAN MOULAY SLIMANE
Faculté des Sciences et Techniques, Béni Mellal



Centre d'Etudes Doctorales en Sciences et Techniques

Formation doctorale : Mathématiques et Physique Appliquées

THÈSE

Présentée par

Fatima ES-SABERY

Pour l'obtention du grade de

Docteur

Spécialité : *Informatique*

***Analyse des sentiments par apprentissage automatique dans un
environnement big data***

Soutenue le 15/01/2022 à 10h30, devant la commission d'examen composée de :

Pr Cherki DAOUI,	Professeur à la FST, Béni Mellal, Maroc	Président.
Pr Abdelghafour ATLAS,	Professeur à l'ENSA, Marrakech, Maroc	Rapporteur.
Pr Najlae IDRISSE,	Professeur à la FST, Béni Mellal, Maroc	Rapporteur.
Pr Mohammed ERRITALI ,	Professeur à la FST, Béni Mellal, Maroc	Rapporteur.
Pr Hicham MOUNCIF,	Professeur à la FP, Béni Mellal, Maroc	Examineur.
Pr Abdellatif HAIR,	Professeur à la FST, Béni Mellal, Maroc	Directeur de Thèse.

Remerciements

Je n'aurais jamais pu réaliser ce travail doctoral sans le soutien d'un grand nombre de personnes dont la générosité, la bonne humeur et l'intérêt manifestés à l'égard de ma recherche m'ont permis de progresser dans cette phase délicate.

Je souhaiterais tout d'abord remercier infiniment mon directeur de thèse monsieur **Abdellatif Hair**, pour ses conseils avisés, sa patience, sa disponibilité, sa confiance et son soutien tout au long de ces cinq années, y compris pendant les moments de doute. L'équilibre qu'il a su trouver entre liberté dans les pistes de recherches étudiées et soutien régulier et conséquent aux moments clés ont été un réel moteur dans mes travaux. Je tiens également à remercier Monsieur **Abdelghafour ATLAS**, Madame **Najlae IDRISI** et Monsieur **Mohammed ERRITALI** pour le temps et l'intérêt qu'ils ont porté à ce travail en acceptant d'en être rapporteurs. Un immense merci également à Monsieur **Cherki DAOUI** pour m'avoir fait l'honneur d'accepter de présider le jury de ma thèse. Je remercie également Monsieur **Hicham MOUNCIF** d'avoir accepté de participer au jury de ma thèse, c'est un honneur.

Je présente mes sincères remerciements à tous mes enseignants, durant toutes les années de mes études, qui ont participé à ma formation et pour leurs conseils et encouragements.

J'adresse mes remerciements à tous les membres du laboratoire TIAD et LMACS.

Une thèse est une aventure au long cours, et celle-ci n'est vraiment possible que lorsque l'on est bien entouré. Pour cela, je tiens à remercier toute les membres de ma famille mes parents, mes frères et sœurs pour m'avoir soutenu et guidé et pour leurs enthousiasme concernant ce projet scientifique.

Que tous ceux qui m'ont apporté leurs aides, de près ou de loin dans l'élaboration de ce travail, trouvent ici l'expression de mes sincères gratitude.

Table des matières

Liste des Publications	11
Résumé	12
Abstract	12
Chapitre 1 Introduction Générale	20
1.1 Contexte général	20
1.2 Motivations et objectifs	23
1.3 Plan de la thèse	25
Chapitre 2 Généralités sur le Big Data	27
2.1 Introduction	27
2.2 Aperçu historique du Big Data	28
2.3 Définition du Big Data	29
2.3.1 Gartner - Définition des 3Vs	30
2.3.2 IBM - Définition des 4Vs	31
2.3.3 Yuri Demchenko - Définition des 5Vs	31
2.3.4 Microsoft - Définition des 6Vs	32
2.4 Sources du Big Data	32
2.5 Évolutivité	34
2.5.1 Évolutivité horizontale	34
2.5.2 Évolutivité verticale	34
2.6 Hadoop	35
2.6.1 Architecture d'Hadoop	35
2.6.1.1 Gestion et suivi	36
2.6.1.2 Système de fichiers distribués d'Hadoop	36
2.6.1.3 YARN	38
2.6.1.4 Modèle de programmation MapReduce	38
2.6.2 Modules d'accès aux données d'Hadoop	40
2.6.2.1 Pig	40

2.6.2.2	Hive	40
2.6.3	Modules d'intégration de données d'Hadoop	41
2.6.3.1	HBase	41
2.6.3.2	Sqoop	41
2.6.3.3	Flume	41
2.6.4	Modules de contrôle et de maintenance d'Hadoop	42
2.6.4.1	Oozie	42
2.6.4.2	Zookeeper	42
2.6.5	Autres modules de l'écosystème Hadoop	43
2.6.5.1	Mahout	43
2.6.5.2	Apache Kafka	43
2.7	Big Data et système de calcul en clusters	43
2.7.1	Système de calcul en clusters	44
2.7.2	Modèle de programmation MapReduce	46
2.7.3	Système de fichiers distribués	48
2.7.3.1	Système de fichiers de Google	48
2.7.3.2	Système de fichiers distribués d'Hadoop	50
2.7.4	Solutions de Big Data	51
2.7.4.1	Défaillances des nœuds	51
2.7.4.2	Complexité des calculs	51
2.7.4.3	Goulets d'étranglement	52
2.7.4.4	Programmation distribuée	53
2.8	Conclusion	54
Chapitre 3 Notions Fondamentales de Fouille de Données		55
3.1	Introduction	55
3.2	Types de fouille de données	57
3.3	Méthodes supervisées	58
3.4	Classification	59
3.4.1	Classification binaire	60
3.4.2	Classification multiclassés	61
3.5	Prétraitement de données	61
3.6	Extraction de caractéristiques	64
3.6.1	N-gramme	64
3.6.2	Sac de Mots	65
3.6.3	TF-IDF	65
3.6.4	GloVe	66

3.6.5	Word2Vec	67
3.6.6	FastText	68
3.7	Sélection de caractéristiques	69
3.7.1	Gain d'information	69
3.7.2	Rapport de Gain d'information	70
3.7.3	Indice de Gini	70
3.7.4	Khi-Carré	71
3.8	Algorithmes de classification	72
3.8.1	Naïve bayésienne	72
3.8.2	Arbres de décision	73
3.8.3	Méthode des k plus proches voisins	76
3.8.4	Machines à vecteurs de support	77
3.8.5	Réseau de neurones	78
3.8.5.1	Réseau de neurones à propagation avant	79
3.8.5.2	Réseau neuronal convolutif	80
3.8.6	Système flou de Mamdani	84
3.9	Mesures d'évaluation pour la classification	85
3.10	Conclusion	90

Chapitre 4 Extraction d'opinion pour la classification des tweets à l'aide d'un classificateur ID3 amélioré et Hadoop

91

4.1	Introduction	91
4.2	Travaux connexes	95
4.3	Processus de la détection de polarité d'opinions	98
4.3.1	Motivation	98
4.3.2	Matériels et méthodes	99
4.3.2.1	Pré-traitement de données	100
4.3.2.2	Extraction des caractéristiques	100
4.3.2.3	Selection des caractéristiques	101
4.3.3	Classification	102
4.3.3.1	Algorithme ID3 amélioré	102
4.3.3.2	Attribut pondéré	103
4.3.3.3	Fonction de corrélation pondérée	103
4.4	Parallélisation de l'approche proposée	104
4.5	Expérimentations et résultats	107
4.5.1	Influence de pré-traitement des données	107
4.5.2	Évaluation des extracteurs de caractéristiques	111

4.5.3	Analyse des sélecteurs de caractéristiques	115
4.5.4	Évaluation de performances de notre approche	117
4.5.5	Examen de notre approche en termes de complexité, conver- gence et stabilité	119
4.5.5.1	Complexité	119
4.5.5.2	Convergence	121
4.5.5.3	Stabilité	123
4.6	Conclusions	124

Chapitre 5 Amélioration de l’algorithme d’arbre de décision C4.5 basé sur le MapReduce et le système de règles floues 125

5.1	Introduction	125
5.2	Travaux connexes	127
5.3	Première approche hybride	129
5.3.1	Motivation	129
5.3.2	Méthodologie de recherche	131
5.3.2.1	Méthodes de fuzzification	133
5.3.2.2	Algorithme d’arbre de décision C4.5 parallèle et flou . . .	135
5.3.2.3	Règles floues	137
5.3.2.4	Méthodes de raisonnement floues	140
5.3.2.5	Méthode de raisonnement flou classique	140
5.3.2.6	Méthode générale de raisonnement flou	141
5.4	Deuxième approche hybride	144
5.4.1	Motivation	144
5.4.2	Vectorisation des données	148
5.4.3	Extraction et sélection des caractéristiques	150
5.4.4	Phase de fuzzification des données	153
5.4.5	Arbre de décision C4.5 flou	154
5.4.6	Méthode générale de raisonnement flou	154
5.4.7	Parallélisation de la deuxième approche sous Hadoop	154
5.5	Expériences de simulation et analyse	155
5.5.1	Ensembles de données expérimentales	155
5.5.2	Résultats et discussions	157
5.5.3	Étude comparative	161
5.5.4	Première expérience	162
5.5.5	Deuxième expérience	166
5.5.6	Troisième expérience	167

5.5.7	Quatrième expérience	169
5.5.8	Cinquième expérience	170
5.5.9	Sixième expérience	172
5.6	Conclusion	173
Chapitre 6 Classification des tweets Basée sur un classificateur d'appren-		
tissage en profondeur parallèle et floue		174
6.1	Introduction	174
6.2	Travaux connexes	177
6.3	Phases de la méthode proposée	180
6.3.1	Motivation	180
6.3.2	Pré-traitement et représentation des données	183
6.3.3	Réseau neuronal convolutif	184
6.3.4	Réseau de neurones à propagation avant	187
6.3.5	Système flou de Mamdani	190
6.3.5.1	Processus de fuzzification	193
6.3.5.2	Définition des règles floues	195
6.3.5.3	Processus d'inférence	197
6.3.5.4	Défuzzification	198
6.4	Parallélisation de la méthode proposée	200
6.5	Exemple de l'application	203
6.6	Expériences et résultats	212
6.6.1	Première expérience	213
6.6.2	Deuxième expérience	215
6.6.3	Troisième expérience	217
6.6.4	Quatrième expérience	223
6.6.5	Cinquième expérience	226
6.6.5.1	Complexité	226
6.6.5.2	Convergence	228
6.6.5.3	Stabilité	229
6.7	Conclusions	230
Conclusion Générale et Perspectives		234
Bibliographie		252

Liste des Publications

Publications dans des Revues Internationales

1. **Fatima Es-Sabery**, Khadija Es-Sabery, Junaid Qadir, Beatriz Sainz-De-Abajo, Abdellatif Hair, Begoña García-Zapirain & Isabel De La Torre-Díez, *A MapReduce Opinion Mining for COVID-19-Related Tweets Classification Using Enhanced ID3 Decision Tree Classifier*, IEEE Access, 9 : 58706–58739, 2021.
<https://doi.org/10.1109/ACCESS.2021.3073215>.
2. **Fatima Es-Sabery**, Abdellatif Hair, Junaid Qadir, Beatriz Sainz-De-Abajo, Begoña García-Zapirain & Isabel De La Torre-Díez, *Sentence-Level Classification Using Parallel Fuzzy Deep Learning Classifier*, IEEE Access, 9 : 17943–17985, 2021.
<https://doi.org/10.1109/ACCESS.2021.3053917>.
3. **Fatima Es-Sabery**, Khadija Es-Sabery, Abdellatif Hair, Junaid Qadir, Beatriz Sainz-De-Abajo, Begoña García-Zapirain & Isabel De La Torre-Díez, *Sentiment Analysis Using A MapReduce Fuzzified Hybrid Approach Based On C4.5 Decision Tree and Convolutional Neural Network Classifiers*, (être soumis à) International Journal of IEEE Access, 2021.
4. **Fatima Es-Sabery** & Abdellatif Hair, *A MapReduce C4.5 Decision Tree Algorithm Based on Fuzzy Rule-Based System*, Fuzzy Information and Engineering, 1–28, 2020. <https://doi.org/10.1080/16168658.2020.1756099>.
5. **Fatima Es-Sabery**, Hicham Ouchitachen & Abdellatif Hair, *Energy Optimization of Routing Protocols in Wireless Sensor Networks*, International Journal of Informatics and Communication Technology, 6 :76, 2017.
<https://doi.org/10.11591/ijict.v6i2.pp76-85>.

Chapitres de Livres

1. **Fatima Es-Sabery**, Khadija Es-Sabery & Abdellatif Hair, *A MapReduce Improved ID3 Decision Tree for Classifying Twitter Data*, In book : Business Intelligence, Springer, Cham, 160-182, 2021. https://doi.org/10.1007/978-3-030-76508-8_13.
2. **Fatima Es-Sabery** & Abdellatif Hair, *An Enhanced Energy-Efficient Hierarchical LEACH Protocol to Extend the Lifespan for Wireless Sensor Networks*, In book : Emerging Trends in ICT for Sustainable Development, Springer, Cham, 191-200, 2021. https://doi.org/10.1007/978-3-030-53440-0_21.

Communications dans les Conférences Internationales à Comité de Lecture

1. **Fatima Es-Sabery**, Khadija Es-Sabery, Hamid Garmani & Abdellatif Hair, *Sentiment Analysis of Covid19 Tweets Using A MapReduce Fuzzified Hybrid Classifier Based On C4.5 Decision Tree and Convolutional Neural Network*, The 5th International Conference of Computer Science and Renewable Energies (ICCSRE'2021), July 22-24, 2021, Agadir, Morocco.
2. **Fatima Es-Sabery** & Abdellatif Hair, *A Parallel Rule-Based Fuzzy System to Classify Data Based on Fuzzy Logic and Decision Tree*, The 3th International Conference on Advanced Intelligent Systems for Sustainable Development (AI2SD'2020), December 21-26, 2020, Tangier, Morocco.
3. **Fatima Es-Sabery** & Abdellatif Hair, *Big Data Solutions Proposed for Cluster Computing Systems Challenges : A survey*, The 3rd International Conference on Networking, Information Systems & Security (NISS2020), March 31, 2020, Marrakech, Morocco.
4. **Fatima Es-Sabery** & Abdellatif Hair, *Evaluation and Comparative Study of the Both Algorithm LEACH and PEGASIS Based on Energy Consumption*, The 3rd International Conference on Networking, Information Systems & Security (NISS2020), March 31, 2020, Marrakech, Morocco.
5. **Fatima Es-Sabery** & Abdellatif Hair, *An Improved ID3 Classification Algorithm Based On Correlation Function and Weighted Attribute*, The 2019 International Conference on Intelligent Systems and Advanced Computing Sciences (ISACS'2019), December 26-27, 2019, Taza, Morocco.

6. **Fatima Es-Sabery** & Abdellatif Hair, *Energy Consumption by LEACH and PE-GASIS Protocols in wireless Sensor Networks*, The International Conference on Intuitionistic Fuzzy Sets and Mathematics Sciences, April 11-13, 2018, Ifrane, Morocco.
7. **Fatima Es-Sabery**, Hicham Ouchitachen & Abdellatif Hair, *Energy Optimization of Routing Protocols in Wireless Sensor Networks*, The 3rd International Conference CBI'17. March 29-31, 2017, Beni Mellal, Morocco.

Résumé

L'analyse des sentiments est un domaine du traitement du langage naturel qui vise à capturer le sentiment humain dans un texte donné. Avec la diffusion croissante des sites d'achat en ligne, des sites de micro-blogging et des plateformes de médias sociaux, l'analyse des sentiments a suscité l'intérêt de milliers de chercheurs scientifiques. En effet, dans cette thèse, nous avons étudié et analysé le problème d'analyse des sentiments en utilisant les technologies big data et les techniques de la fouille de données spécifiquement les techniques d'apprentissage automatique et la théorie de la logique floue. Pour résoudre ce problème d'analyse des sentiments, nous avons proposé trois approches hybrides pour effectuer la classification des sentiments avec une haute performance. Dans la première contribution, nous proposons une approche innovante de classification basée sur un arbre de décision ID3 pondéré et mis en œuvre sous Hadoop. Dans la deuxième, nous proposons un nouveau modèle parallèle pour l'analyse des sentiments, qui combine le réseau neuronal convolutif, l'algorithme d'arbre de décision C4.5 et le système à base de règles floues. Et dans la troisième, un nouveau classificateur d'apprentissage en profondeur parallèle et floue est proposé.

Mots clés :

ID3, Hadoop, Fouille de données, Big data, Réseau neuronal convolutif, Analyse des sentiments, Réseau de neurones à propagation avant, Logique floue, C4.5.

Abstract

Sentiments analysis is a field of natural language processing that aims to capture human sentiment in the given text. With the ever-spreading of online purchasing websites, micro-blogging sites, and social media platforms, sentiments analysis in online social media platforms has picked the interest of thousands of scientific researchers. In this thesis, we studied and analyzed the problem of sentiment analysis using Big Data technologies and data mining techniques, specifically machine learning and fuzzy logic theory techniques. To solve the problem of sentiment analysis, we proposed three hybrid approaches to perform sentiment classification with high performance. In the first contribution, we introduce an innovative improved weighted ID3 decision tree classification approach. In the second contribution, we propose a new model for sentiment analysis, which combines the convolutional neural network, C4.5 decision tree algorithm, and fuzzy rule-based system. And in the third contribution, a new fuzzy deep learning classifier is suggested for improving the performance of data-sentiment classification.

Key Words :

ID3, Hadoop, Data mining, Big data, Convolutional neural network, Sentiment analysis, Feedforward neural network, Fuzzy logic, C4.5.

Liste des Acronymes

NLP : Natural Language Processing

SVM : Support Vector Machine

LR : Logistic regression

NB : Naive Bayes

RF : Random Forest

LSTM : Long short-term memory

MLP : Multilayer perceptron

CNN : Convolutional neural network

NFS : Network File System

YARN : Yet Another Resource Negotiator

FFNN : Feedforward neural network

RNN : Recurrent Neural Network

CAPF : Classificateur d'apprentissage en profondeur flou

MFS : Mamdani fuzzy system

DT : Decision tree

Liste des figures

2.1	Histoire du Big Data et les outils d'analyse du Big Data.	30
2.2	Définitions 3Vs, 4Vs, 5Vs et 6Vs du Big Data.	32
2.3	Une minute d'Internet en 2020 [1].	33
2.4	Architecture de Hadoop.	36
2.5	Architecture "Maître et esclave" de Hadoop.	38
2.6	Modèle de programmation MapReduce.	39
2.7	Architecture d'un système informatique avec multiples racks.	45
2.8	Schéma de modèle de programmation MapReduce.	47
2.9	Architecture de SFG.	49
2.10	Architecture de HDFS.	50
3.1	Un aperçu des étapes composant le processus KDD.	56
3.2	Taxonomie des méthodes de fouille de données.	58
3.3	Le processus d'apprentissage supervisé.	60
3.4	Étapes de base de l'étape de prétraitement des données.	62
3.5	Les sous-structures linéaires d'un ensemble de termes en appliquant Glove.	67
3.6	Modèle de skip-gram et le modèle de sacs de mots continus.	67
3.7	Processus d'application de la méthode Word2Vec.	68
3.8	Exemple d'arbre de décision.	74
3.9	Illustration de la structure d'un réseau de neurones.	79
3.10	Structure d'un réseau de neurones à propagation avant.	80
3.11	Réseau neuronal convolutif.	81
3.12	Représentation graphique de la fonction d'activation ReLU.	82
3.13	Illustration des fonctions de pooling-moyenne et de pooling-maximale.	83
3.14	Représentation graphique de la fonction d'activation du softmax.	83
3.15	Architecture globale d'un systèmes d'inférence floue.	84
3.16	Matrice de confusion pour un problème de classification binaire.	85
3.17	Matrice de confusion pour un problème de classification multi-classes.	88
4.1	Les étapes de base de la fouille d'opinion.	93
4.2	Architecture de notre processus pour la détection de polarité d'opinions.	99

4.3	Les étapes de base de la fouille d'opinion sur les tweets sous Hadoop	106
4.4	Taille du jeu de données et temps d'exécution de chaque tâche de pré-traitement appliquée au Sentiment140.	108
4.5	Taille du jeu de données et temps d'exécution de chaque tâche de pré-traitement appliquée au COVID-19_Sentiments.	109
4.6	TC de l'extraction d'opinions en appliquant N-gram sur Sentiment140 sous Hadoop.	112
4.7	TC de l'extraction d'opinions en appliquant N-gram sur COVID-19 Sentiments sous Hadoop.	112
4.8	TC en appliquant l'extracteur Sac de Mots sur COVID-19 Sentiments et Sentiment140 sous Hadoop.	113
4.9	TC en appliquant l'extracteur TF-IDF sur Sentiment140 et COVID-19 Sentiments sous Hadoop	114
4.10	TC en appliquant l'extracteur GloVe sur les deux jeux de données sous Hadoop	114
4.11	TC en appliquant l'extracteur Word2Vec sur Sentiment140 et COVID-19-Sentiments sous Hadoop.	115
4.12	TC en appliquant l'extracteur FastText sur Sentiment140 et COVID-19-Sentiments sous Hadoop.	116
4.13	Structure finale de la méthode hybride proposée sous le cadre Hadoop.	117
4.14	TC en appliquant notre classificateur et d'autres approches.	118
4.15	TR en appliquant notre classificateur et d'autres approches.	118
4.16	TE en appliquant notre classificateur et d'autres approches.	119
4.17	Taux de convergence en appliquant notre classificateur sur COVID-19-Sentiments et Sentiment140.	123
5.1	Organigramme de notre algorithme amélioré.	133
5.2	Un exemple d'arbre de décision flou	139
5.3	Méthode classique de raisonnement flou.	142
5.4	Méthode générale de raisonnement flou.	143
5.5	Structure générale de la deuxième approche hybride.	148
5.6	Modèle de Continuous Bag-Of-Words et l'approche Skip-Gram.	149
5.7	Extraction et sélection de caractéristiques à l'aide d'un réseau neuronal convolutif.	150
5.8	Illustration des fonctions de mise en commun moyenne et de mise en commun maximale.	152
5.9	Parallélisation de notre deuxième approche en utilisant Hadoop.	154

5.10	TC en mettant en œuvre l'approche 1 et l'approche 2.	157
5.11	TC en mettant en œuvre l'approche 1 et C4.5.	158
5.12	TE en mettant en œuvre C4.5+MapReduce et C4.5	159
5.13	TC et TE dans les cas (a) jeu de données n.1, (b) jeu de données n.2, (c) jeu de données n.3	160
5.14	TE en secondes de notre approche et d'autres techniques	161
5.15	Résultats obtenus par notre comparaison	162
5.16	TC en appliquant la deuxième approche hybride et les autres approches.	163
5.17	TR en appliquant la deuxième approche et les autres approches.	165
5.18	TE consommé par l'approche hybride et les autres méthodes.	167
5.19	TE en mettant en œuvre l'approche hybride sur un nombre différent de nœuds de Hadoop.	168
5.20	Taux de convergence de l'approche hybride appliquée sur les deux jeux de données.	171
6.1	Précision des approches AA en comparaison avec la précision des modèles AP par rapport à la taille des données.	175
6.2	Structure globale de notre classificateur d'apprentissage en profondeur pa- rallèle et floue.	182
6.3	Exemple d'application d'un filtre sur un caractère n-gramme du mot "fuzzy".	186
6.4	Exemple d'application de la couche de pooling.	186
6.5	Représentation graphique de la fonction d'activation sigmoïde.	189
6.6	Réseau de neurones à propagation avant avec et sans l'opération de décro- chage.	190
6.7	Les composantes du système flou de Mamdani.	193
6.8	Représentation graphique de la fonction triangulaire.	194
6.9	Représentation graphique de la fonction trapézoïdale.	195
6.10	Représentation graphique de la fonction gaussienne.	196
6.11	Architecture parallèle de notre contribution utilisant MapReduce.	202
6.12	Matrice de plongement de mots du tweet S.	204
6.13	Application des étapes du CNN.	204
6.14	Application des étapes du FFNN.	205
6.15	Degré d'appartenance de chaque terme linguistique.	206
6.16	Degré d'appartenance de chaque étiquette de classe.	211
6.17	Valeur défuzzifiée calculée à l'aide de l'examen des règles dans Matlab.	212
6.18	TC et TR de tous les CAPFs évalués de notre contribution.	218
6.19	TE de tous les CAPFs évalués de notre contribution.	221

6.20 Architecture de notre proposition après des multiples expériences. 223

6.21 TE de notre approche illustrée dans la figure 6.20. 224

6.22 TC et TE en appliquant notre approche et d'autres méthodes sur Senti-
ment140. 225

6.23 TC et TE en appliquant notre approche et d'autres approches sur COVID-
19_Sentiments. 225

6.24 Taux de convergence de notre CAPF31 pour les deux jeux de données. . . 228

Liste des tableaux

3.1	Échantillon des données météorologiques	75
4.1	Articles de recherche les plus récents de la fouille d'opinion.	97
4.2	Analyse de la taille du jeu de données (TJD) et du temps d'exécution des données avec Hadoop (TEA) et sans Hadoop (TES) dans le cas de Sentiment140.	109
4.3	Analyse de la taille du jeu de données (TJD) et du temps d'exécution des données avec Hadoop (TEA) et sans Hadoop (TES) dans le cas de COVID-19_Sentiments.	110
4.4	TE sans et avec les techniques de prétraitement des données.	111
4.5	TC en appliquant ID3, C4.5, et notre ID3 amélioré (ID3A) avec différents sélecteurs (1,2,3,4) sur les deux jeux de données.	116
4.6	Résultats expérimentaux en terms de <i>TPV</i> , <i>TNF</i> , <i>TNV</i> , <i>TPF</i> , <i>PR</i> , <i>SK</i> , et <i>SF</i>	120
4.7	Complexité spatiale de notre classificateur et d'autres approches.	121
4.8	Complexité temporelle de notre classificateur et d'autres approches.	122
4.9	Taux de convergence de notre classificateur et d'autres approches.	123
4.10	Stabilité de notre classificateur et d'autres approches.	124
5.1	Propriétés des ensembles de données.	157
5.2	Le résultat obtenu de <i>TPV</i> , <i>TNF</i> , <i>TNV</i> , <i>TPF</i> , <i>PR</i> , <i>SK</i> , et <i>SF</i>	161
5.3	Résultats expérimentaux en termes de <i>TPV</i> , <i>TNF</i> , <i>TNV</i> , <i>TPF</i> , <i>PR</i> , <i>SK</i> , et <i>SF</i>	166
5.4	Complexité spatiale de l'approche hybride et les autres approches.	169
5.5	Complexité temporelle de l'approche hybride et les autres approches.	170
5.6	Cycle de convergence de l'approche hybride et les autres approches.	172
5.7	Stabilité de l'approche hybride et les autres approches.	172
6.1	Paramètres d'entrée et de sortie du système flou utilisé.	193
6.2	Ensemble de jetons qui représentent le tweet à classifier.	203

6.3	Paramètres de notre CAPF que nous avons utilisés pour évaluer les approches du plongement de mots.	214
6.4	TR et TE de chaque méthode du plongement de mots sans l'application du cadre Hadoop.	214
6.5	TR et TE de chaque méthode du plongement de mots avec l'application du Hadoop.	215
6.6	TR, TC et TE de l'agrégation des méthodes de fuzzification/défuzzification sans Hadoop.	216
6.7	TR, TC et TE de l'agrégation des méthodes de fuzzification/défuzzification avec Hadoop.	216
6.8	Configuration des paramètres de notre modèle d'apprentissage en profondeur (CNN+FFNN).	217
6.9	Configuration des paramètres pour nos CAPFs parallèles.	219
6.10	TC, TR et TE des différents CAPF parallèles proposés.	220
6.11	TPV , TNF , TNV , TPF , PR , SK , et SF pour notre CAPF et les autres approches.	226
6.12	Complexité spatiale de notre proposition CAPF31 et des autres approches.	227
6.13	Complexité temporelle de notre modèle CAPF31 et d'autres modèles.	227
6.14	Taux de convergence de notre CAPF31 et les autres approches.	229
6.15	Stabilité de notre classificateur CAPF31 et les autres approches.	229

Introduction Générale

1.1 Contexte général

De nos jours, l'Internet est un outil incontournable d'échange d'information tant au niveau personnel que professionnel. Le Web nous offre un monde de l'information prodigieux et a évolué de simples ensembles de pages statiques vers des services de plus en plus complexes. Ces services nous offrent l'achat de tous les produits, la lecture de son journal préféré en ligne, la rencontre de l'âme soeur, la discussion sur de multiples forums ou la possibilité de s'exprimer sur les blogs. L'Internet contient un nombre énorme d'informations, et pour la plupart d'entre nous, c'est le premier lieu pour trouver ces informations, réserver l'avion ou l'hôtel, acheter des produits, consulter les avis d'autres utilisateurs sur les produits qui nous intéressent, lire les commentaires avant de choisir le film à voir au cinéma, voir des propositions d'autres personnes avant de choisir les cadeaux de mariage etc. Le problème principal n'est plus de savoir si l'information se trouve sur le Web mais comment la trouver car le flux informationnel est extrêmement bruité. Un autre problème, non lié à Internet lui même mais plutôt à des considérations sociologiques, est que la globalisation nous envahit. Nous avons l'accès à beaucoup plus de produits que l'on ne peut pas en connaître. Ces produits peuvent être de différentes sortes : de la musique, des films, de la technologie, les transports, le commerce, le travail, l'école, etc. L'internet vient en aide aux utilisateurs et facilite énormément le référencement, la recherche et l'accès aux informations aux informations liées a chaque produit.

Due à l'utilisation enorme d'internet, surtout l'utilisation des réseaux sociaux, la quantité de données disponibles augmente de façon exponentielle. Plus de données offrent plus d'opportunités, mais de nombreux nouveaux problèmes et de grands défis se posent également. On estime que 90% des données humaines ont été générées au cours des deux dernières années. Si on ne considère que les données textuelles, 2 milliards d'utilisateurs actifs de Facebook mettent à jour environ 734 millions de statuts et commentaires par jour. Google doit maintenant traiter plus de 3.5 milliards de recherches par jour. Environ 23 milliards de SMS et 225 milliards de courriels électroniques sont envoyés chaque jour pour communiquer au travail et à des fins personnelles. Cette quantité, cette diversité et cette rapidité de diffusion des données sont sans précédent. Compte tenu du rythme croissant des données entrantes, du temps et de la capacité humaine d'analyse des informations limités, il existe un besoin urgent de créer des outils automatiques pour aider les humaines à gérer et exploiter ces données. Le Big Data a proposé des techniques pour gérer et traiter ces données volumineuses.

Le Big Data est un phénomène qui a vu le jour avec l'émergence de données volumineuses qu'on ne pouvait pas traiter avec des techniques traditionnelles. Les premiers projets de Big Data sont ceux des acteurs de la recherche d'information sur le web « moteurs de recherche » tel que Google et Yahoo. En effet, ces acteurs étaient confrontés aux problèmes de la scalabilité (passage à l'échelle) des systèmes et du temps de réponse aux requêtes utilisateurs très rapidement. D'autres sociétés ont suivi le même chemin comme Amazon, Facebook, Twitter, et Youtube. Le Big Data est devenu incontournable pour beaucoup d'acteurs industriels du fait de l'apport qu'il offre en qualité de stockage, de traitement et d'analyse de données. Ce phénomène du BigData est généralement défini selon quatre dimensions (Vs) : Volume, Vitesse, Variété et Véracité. Le volume désigne la gestion de gros volumes de données. La vitesse (vitesse) est le temps nécessaire pour collecter et traiter les données. La variété traite des données structurées, semi-structurées et non structurées. Enfin, la véracité permet de garantir la qualité et la fiabilité des données. De nos jours, les données représentent une richesse pour les entreprises et les administrations et contribuent à leur développement. La qualité de ces données représente un enjeu important. Le coût de la non-qualité peut en effet s'avérer très élevé : prendre une décision à partir de mauvaises informations peut nuire à l'organisation, à ses clients ou ses partenaires. La gouvernance des données est un sujet qui prend de l'importance dans les entreprises et les administrations. Elle permet l'amélioration des interactions entre les différents collaborateurs d'une ou plusieurs organisations concernées. La plupart des entreprises/organisations utilisent les méthodes de la fouille de données pour extraire automatiquement l'information utile de ces données et la mettre à disposition des décideurs. Mais à cause de la quantité volumineuse de données générées par jour, les opérations de fouille de données telles que les opérations analytiques, les opérations de traitement et les opérations de récupération sont très difficiles et prennent énormément de temps à cause de l'énorme volume de données. Une solution pour surmonter ces problèmes est l'utilisation des plateformes et outils Big Data dédiés à la gestion de ces données parmi lesquels la plateforme Hadoop qui est constitué de deux composants essentiels à savoir : MapReduce qui est un nouveau paradigme de programmation sur lequel sont effectués les calculs parallèles et distribués de grandes masses de données et HDFS qui est un système de gestion de fichiers distribué.

La fouille de données (Data Mining) est récemment devenue l'un des domaines les plus progressifs et les plus prometteurs pour l'extraction et la manipulation de données afin de produire des informations utiles. Des milliers d'entreprises utilisent chaque jour des applications de la fouille de données afin de manipuler, d'identifier et d'extraire des informations utiles à partir des enregistrements stockés dans leurs bases de données, leurs dépôts de données et leurs entrepôts de données. Donc, on peut dire que la tâche principale de la fouille de données c'est utiliser des méthodes pour extraire automatiquement l'information utile de ces données et la mettre à disposition des décideurs. Grâce à ces informations extraites à l'aide des outils de la fouille de données, les entreprises ont pu améliorer leurs activités en appliquant les modèles, les relations et les tendances qui se sont cachées ou qui n'ont pas été découvertes dans des quantités colossales de données. Par exemple, la fouille de données produit des informations qui permettent aux entreprises de créer des profils de clients actuels et potentiels afin de les aider à gagner et à conserver leurs clients. Parmi les autres utilisations de la fouille de données, citons le développement de stratégies de vente croisée et de marketing, la mise à jour de crimes

ou de fraudes possibles, et la découverte de modèles d'accès des utilisateurs à leurs sites web. Ainsi, la recherche menée au sein de la communauté de la fouille de données, au cours des dernières années, a donné naissance à de nombreuses techniques différentes qui pourraient convenir à des conditions spécifiques tels que les petits jeux de données, les jeux de données composés principalement d'enregistrements numériques et les jeux de données qui comportent un grand nombre de classes. Parmi les techniques proposées par la communauté de la fouille de données nous avons constaté : Naïve bayésienne, Arbres de décision (C4.5, ID3, CART, et forêts d'arbres décisionnels), Méthode des k plus proches voisins, Machines à vecteurs de support, Réseau de neurones à propagation avant, Réseau de neurones convolutifs.

Avec l'avènement de la technologie numérique et des appareils intelligents, une grande quantité de données numériques est générée chaque jour. Cette forte augmentation des données, tant en taille qu'en diversité, est principalement dû aux réseaux sociaux qui permettent à des millions d'utilisateurs de partager des informations, d'exprimer et de diffuser leurs idées et leurs opinions sur un sujet et montrer leurs attitudes envers un contenu. Toutes ces actions stockées sur les médias sociaux génèrent un ensemble massif d'opinions qui offre une opportunité pour les systèmes automatiques de fouille et d'analyse de données pour déterminer les tendances des internautes. Plusieurs chercheurs ont montré un vif intérêt pour l'exploitation de ces informations afin de prédire les comportements humains des domaines aussi variés que la médecine, la politique, le marketing, etc. Cette exploitation est principalement basée sur l'analyse des sentiments qu'elle vise à déterminer le sentiment des gens en analysant leurs messages et différentes actions sur les médias sociaux. Elle consiste à classer la polarité des messages en différents sentiments opposés tels que positif, neutre et négatif. Le domaine de recherche présenté dans cette thèse est l'Analyse des Sentiments dans un environnement BigDataMining (Big Data+Fouille de données).

L'analyse des sentiments exprimés dans les textes est un outil essentiel de nos jours, qui permet aux entreprises et aux gouvernements de mieux comprendre la manière dont sont perçus par les utilisateurs et les citoyens de nouveaux produits ou de nouvelles propositions. À l'ère des échanges sur le marché mondial impliquant des milliers de concurrents issus de cultures différentes, l'analyse des sentiments est l'application de Traitement Automatique du Langage Naturel (TALN) la plus populaire pour les entreprises. Les gouvernements modernes ont également besoin de cet outil pour adapter plus rapidement leur politique à la réaction en temps réel de leurs citoyens sur les médias sociaux. En ce qui concerne les individus, leur vision de voir être créé à long terme un assistant, une machine au comportement à la fois humain et convivial, capable de communiquer et d'aider l'homme dans la vie quotidienne (robots en tant que réceptionniste, assistants, ...) restera un rêve tant que les aspects sémantiques essentiels du langage naturel ne seront pas interprétés correctement par les modèles actuels. De plus, alors que le temps est devenu un facteur rare de nos jours, lire toutes les informations textuelles disponibles n'est pas réaliste et les besoins d'accéder à des résumés fidèles et naturels de tous ces documents, avec les arguments principaux exprimant les idées centrales clairement identifiés, est devenu un impératif que ne peuvent réaliser les méthodes traditionnelles basées sur les mots fréquents.

1.2 Motivations et objectifs

L'analyse des sentiments est un domaine de recherche intéressant qui rassemble les commentaires des clients sur un événement, un service ou un produit et révèle si les sentiments exprimés à propos de cet événement, de ce service ou de ce produit sont neutres, négatifs ou positifs. L'analyse des sentiments est généralement effectuée sur des données collectées à partir de plateformes de médias sociaux pour aider les entreprises/organisations à contrôler leurs marques, services, événements et produits en analysant les opinions, idées et attitudes exprimées dans les commentaires des clients et en comprenant les besoins des acheteurs. En général, l'analyse des sentiments aide les grandes organisations/entreprises à optimiser leurs tactiques de commercialisation, à renforcer la qualité de leurs produits/marques/événements et à améliorer les services offerts aux acheteurs. Notre thèse est motivée par le double rôle essentiel de l'analyse des sentiments pour aider à la fois les clients et les organisations/entreprises et leur importance significative dans différents domaines de notre vie quotidienne. Par exemple, les entreprises utilisent les outils d'analyse des sentiments dans la prise de décision pour améliorer la qualité de leurs marques, produits, services ou événements afin de répondre aux besoins et exigences des consommateurs, la supervision de leurs produits/événements /services, la consolidation des meilleures relations avec leurs clients, la conception et l'évolution de stratégies commerciales efficaces, l'amélioration et la conception de leurs produits/événements /services. Ainsi, les commentaires écrits par les utilisateurs sur les produits aident les nouveaux utilisateurs à prendre des décisions, comme acheter ce nouveau produit ou non.

La fouille d'opinion est un important domaine de recherche qui s'efforce de concevoir une technique computationnelle pour détecter, capturer et évaluer les idées et les opinions des gens sur une entité et ses divers aspects et pour extraire les émotions exprimées dans ces idées et opinions car ces sentiments exprimés à propos d'un produit, d'un événement ou d'un service ont une valeur commerciale importante. La fouille d'opinion est extrêmement précieuse pour les grandes entreprises/organisations ainsi que pour les individus. Dans le cadre de la fouille d'opinion, il existe plusieurs tâches de traitement automatique des langues, qu'il est utile ou non de mettre en œuvre selon les applications visées. Il est possible, par exemple, de chercher à détecter la présence ou non d'une opinion ou d'une appréciation, de vouloir classer des opinions exprimées en fonction de leur polarité sur l'axe positif négatif ou bien de leur intensité. D'autres travaux s'attachent à identifier l'objet sur lequel porte l'opinion ou bien la personne qui exprime cet opinion. Toutes ces tâches peuvent être réalisées à différents niveaux selon les applications envisagées : au niveau global du texte, au niveau très précis d'un aspect particulier ou à des niveaux intermédiaires comme la phrase, le paragraphe ou bien une thématique.

Nous nous intéressons dans cette thèse à la classification de l'opinion au niveau du texte. Pour créer un classifieur automatique d'opinion, il est possible de s'appuyer sur des règles définies manuellement, par exemple à partir de lexiques ou bien d'utiliser des approches à base d'apprentissage automatique, avec des modèles qui doivent alors être entraînés sur des corpus qui doivent être annotés. Dans la littérature, de nombreuses catégories d'approches ont été appliquées pour effectuer la classification des sentiments. Parmi ces techniques, nous trouvons des modèles d'apprentissage en profondeur, des méthodes d'apprentissage automatique et des procédures basées sur des dictionnaires. Les

performances des méthodes d'apprentissage automatique et des approches basées sur les dictionnaires sont inférieures à celles des modèles d'apprentissage en profondeur si nous les appliquons à un énorme jeu de données. Les études démontrent que les méthodes classiques d'apprentissage automatique et les techniques basées sur les dictionnaires sont meilleures pour une taille de données moindre. Lorsque le volume de données augmente au-delà d'un certain nombre, la précision des algorithmes classiques d'apprentissage automatique devient constante. En revanche, la précision des algorithmes d'apprentissage en profondeur augmente avec la croissance de la taille des données. Cette différence de performance est due à la manière adoptée pour extraire les caractéristiques à partir du jeu de données. Les techniques classiques d'apprentissage automatique et les approches basées sur des dictionnaires adoptent l'extraction manuelle des caractéristiques, ensuite elles appliquent les algorithmes d'apprentissage sur ces caractéristiques extraites, mais le problème est que cette méthode d'extraction manuelle extrait des caractéristiques incomplètes et prend beaucoup de temps pour produire le résultat final. Contrairement aux modèles d'apprentissage en profondeur qui adoptent un extracteur automatique de caractéristiques. Dans cette thèse, nous nous utilisons pour la classification des sentiments les deux catégories : méthodes d'apprentissage automatique traditionnelles et modèles d'apprentissage en profondeur qui sont mises en oeuvre en utilisant la plateforme Hadoop pour faciliter le stockage et le traitement des données volumineux.

Mais bien que l'utilisation des modèles d'apprentissage automatique les plus avancés, il est une ambiguïté enracinée dans le traitement du langage naturel qui a besoin de plus de solutions. D'après plusieurs études, la théorie de la logique floue est considérée comme la solution la plus efficace pour traiter les données imprécises. Beer démontre dans son article [2] que la force de la théorie de la logique floue est sa ressemblance avec le raisonnement humain et le langage naturel. Une propriété substantielle de la théorie de la logique floue est la technique utilisée pour le calcul avec des termes. Cette technique sert à transformer les termes en valeurs numériques pour le raisonnement, l'inférence et le calcul. La logique floue nous fournit un moyen éligible pour gérer les problèmes linguistiques au niveau des données floues. Zadeh et al. [3] discutent dans leurs travaux qu'aucune autre technique ne résout ces problèmes linguistiques. En conséquence, aux avantages introduits dans [3] sur la théorie de la logique floue dans le domaine des données floues, nous décidons d'employer cette théorie dans cette thèse pour traiter l'incertitude inhérente aux données des médias sociaux. L'idée de la théorie de la logique floue vise à analyser les données recueillies dans différents domaines d'une manière qui est similaire aux sentiments des êtres humains, contrairement aux méthodes d'apprentissage automatique. Son but principal est de transformer un problème blanc et noir en un problème gris.

En général, le processus de fouille d'opinion est constituée de cinq phases ; la première étape est la collecte de données. La deuxième phase, appelée pré-traitement des données, qui vise à supprimer les données indésirables et bruitées. La troisième phase est celle de la représentation des données qui convertit les tweets en données numériques. La quatrième étape est celle de la sélection des caractéristiques pour réduire la dimensionnalité des caractéristiques extraites. Enfin, la phase de classification qui vise à classer chaque tweet comme étant négatif, neutre ou positif. Dans cette thèse, nous étudions et analysons les performances de plusieurs techniques de pré-traitement des données en termes de taille du jeu de données et de temps d'exécution avant et après la mise en oeuvre du cadre Hadoop. Par conséquent, l'objectif principal de l'application de techniques de pré-traitement des

données sur le jeu de données de tweets est de supprimer le bruit, et d'améliorer la qualité des données. Ainsi, nous mettons en oeuvre et analysons les différents populaires extracteurs de caractéristiques, notamment N-gramme, Sac de Mots, TF-IDF, GloVe, Word2Vec et FastText pour déterminer l'extracteur le plus efficace en termes de taux de classification. De plus, nous étudions et analysons les performances de plusieurs selecteur de caractéristiques, notamment khi-carré, ratio de Gain, gain d'information, indice de gini et réseau neuronal convolutif. Enfin, nous proposons des algorithmes et des approches hybrides pour classer les tweets en différentes polarités de sentiments telle que positif, neutre et négatif.

1.3 Plan de la thèse

Le contenu principal de la thèse est divisé en six chapitres et une conclusion générale. Les préliminaires et la terminologie définie dans le deuxième chapitre et troisième chapitre vont permettre aux lecteurs de mieux positionner et comprendre les différentes contributions réalisées dans le cadre de ce travail. Le [deuxième chapitre](#) présente brièvement les notions fondamentales du Big Data. Nous présentons un aperçu historique du Big Data et nous décrivons les différentes définitions détaillées des concepts du Big Data. Ensuite, nous présentons les différentes sources du Big Data. Par la suite, nous allons discuter les deux types de l'évolutivité du Big data (évolutivité horizontale, évolutivité verticale). De plus, nous décrivons en détails l'architecture et les modules d'Hadoop. À la fin, nous définissons les solutions du Big Data proposées pour le système de calcul en clusters.

Dans le [troisième chapitre](#), nous avons présenté brièvement les notions fondamentales de fouille de données. Nous avons décrit les types de fouille de données. Ensuite, nous avons présente la définition des méthodes supervisées parmi ces méthodes nous avons décrit la méthode de classification. Ainsi, nous avons discuté les différentes techniques de prétraitement de données. De plus, nous avons présenté le processus d'extraction de caractéristiques et le processus de sélection de caractéristiques. Nous avons présenté et récapitulé les différents algorithmes de classification. Enfin, nous avons défini les différentes mesures d'évaluation utilisées dans notre travail pour vérifier la performance de la classification.

Dans le [quatrième chapitre](#), nous proposons une approche innovante pour la fouille d'opinion qui est intitulée "Extraction d'opinion pour la classification des tweets à l'aide d'un classificateur ID3 amélioré et Hadoop" et qui se compose principalement de quatre aspects : premièrement, nous avons étudié et analysé l'impact de différentes techniques de prétraitement de données sur la performance de processus de classification. Ensuite, nous avons mis en oeuvre et comparé plusieurs extracteurs de caractéristiques pour détecter et capturer efficacement les données pertinentes à partir des tweets analysés. De plus, nous avons mis en oeuvre et comparé plusieurs sélecteurs de caractéristiques pour réduire la dimensionnalité des caractéristiques élevées. Enfin, nous avons utilisé les caractéristiques obtenues pour effectuer la tâche de classification à l'aide de notre classificateur ID3 amélioré, qui vise à calculer le gain d'information pondéré au lieu du gain d'information adopté par l'ID3 traditionnel. Cette contribution est mise en oeuvre dans un environnement distribué sous Hadoop.

Ensuite, dans le [cinquième chapitre](#), nous avons développé un nouveau modèle d'analyse des sentiments, qui combine le réseau neuronal convolutif, l'algorithme d'arbre de décision C4.5, le système à base de règles floues et le framework Hadoop. Cette approche proposée se compose de six parties. Tout d'abord, nous avons appliqué plusieurs techniques de prétraitement pour éliminer les données bruyantes et les informations indésirables. Deuxièmement, nous avons vectorisé les tweets. Troisièmement, nous avons implémenté le réseau neuronal convolutif pour extraire et sélectionner automatiquement les caractéristiques pertinentes. Quatrièmement, nous avons fuzzifié la sortie du réseau neuronal convolutif en utilisant la méthode de fuzzification gaussienne afin de traiter les données incertaines et ambiguës. Ensuite, nous avons appliqué la méthode C4.5 pour créer l'arbre flou et générer les règles floues. Enfin, nous avons appliqué l'approche générale de raisonnement flou sur les règles floues pour classer les nouveaux tweets. Nous avons également mis en œuvre cette approche sous Hadoop.

Dans le [sixième chapitre](#), nous introduisons un nouveau classificateur d'apprentissage en profondeur parallèle et flou pour reconnaître la polarité (négatif, Positif, neutre). L'approche proposée dans ce chapitre comprend principalement cinq étapes : la première étape concerne les étapes de prétraitement des données, la deuxième est l'application de prolongement lexical de mots, la troisième est l'application du modèle d'apprentissage en profondeur hybride proposé qui combine le réseau neuronal convolutif et le réseau de neurones à propagation avant, et la quatrième est l'application de système flou de Mamdani qui est utilisée comme un classificateur flou. Enfin, nous utilisons Hadoop pour paralléliser notre approche afin de surmonter le problème du long temps d'exécution.

À la fin de ce document, nous présentons une [conclusion générale](#) dédiée à la synthèse des travaux effectués, ainsi que les principaux résultats obtenus.

Généralités sur le Big Data

2.1 Introduction

Personne ne peut dire exactement la quantité de données générées par jour mais il existe de bonnes hypothèses : en 2020, 2.3 trillions de gigaoctets de données ont été générés chaque jour selon IBM, et ils prévoient que ce chiffre augmentera à 2.75 trillions de gigaoctets par jour d'ici 2025 [4]. En outre, selon les prévisions d'un rapport d'IDC, le volume global de données devrait augmenter de manière exponentielle, passant de 4.4 zettaoctets à 44 zettaoctets entre 2013 et 2020. D'ici 2025, IDC prévoit qu'il y aura 150 zettaoctets de données [5]. Les données et les informations disponibles sont impossibles à collecter, à vérifier et à comprendre pour une seule personne. Grâce à l'augmentation des données disponibles, des nouvelles et originales méthodes ont été développées pour extraire des informations utiles. Dans chaque organisation au sein de chaque industrie, une vaste quantité de données sont générées qui contient diverses informations provenant de sources internes et externes telles que les transactions de données, les documents d'entreprise, les médias sociaux, les capteurs et autres dispositifs. Les entreprises peuvent tirer profit de l'analyse de leurs données pour satisfaire les besoins des clients, optimiser leurs opérations ou obtenir de nouvelles sources de revenus [6].

Puisque la popularisation du réseau a rapidement augmenté, de nombreuses nouvelles méthodes de diffusion de l'information sont apparues [7]. En outre, l'essor de la technologie du "Informatique en Nuage" et du "Internet des Objets" fait que les données augmentent et s'accumulent avec une vitesse sans précédent. Aujourd'hui, c'est l'ère de Big Data [8]. L'attribut des données a changé, qui n'est plus seulement une sorte d'objet en attente de traitement mais deviennent une ressource fondamentale et cruciale. Les universités, l'industrie et même le gouvernement ont montré un vif intérêt pour la technologie de Big Data, et beaucoup de scientifiques ont commencé une série de recherches sur le Big Data. L'application de technologie de Big Data aujourd'hui a couvert des domaines très vastes, tels que le commerce, l'industrie, l'informatique, les traitements médicaux, l'éducation, etc. La manière de parvenir à une meilleure gestion et utilisation de Big Data à susciter beaucoup d'attention [9, 10].

Le terme "Big Data" est devenu l'une des technologies les plus connus de cette époque, mais il n'existe pas de définition spécifique de Big Data. Habituellement, de nombreux experts en science des données aimeraient définir la procédure d'extraction, de transformation et de chargement pour une quantité massive de données comme le concept de "Big Data" [11]. L'élaboration commune de Big Data est basée sur les trois caractéristiques les

plus courantes des ensembles de données : la vitesse, la variété et le volume (ou 3Vs). Cependant, il ne contient pas toutes les caractéristiques de Big Data avec précision [12, 13]. Afin de présenter une connotation complète de Big Data, une étude du terme Big Data d'un point de vue historique sera présentée pour voir comment la technologie de Big Data a évolué du sens d'hier au sens d'aujourd'hui. Historiquement, le terme de Big Data est mal défini et assez vague. Ce n'est pas un terme précis et ne donne pas de sens exact plutôt que la notion de son volume. L'idiome "Big" est très élastique.

Le Big Data est quelque chose de si énormes et complexe qu'il est impossible pour les systèmes traditionnels et les outils d'entreposage de données traditionnels de les traiter et de travailler dessus [14]. Les données sont générées par des machines, et également générées par les humains. Avec la croissance des techniques et des services, ces grandes données se génèrent qui peuvent être structurées, semi-structurées et non structurées à partir des différentes sources [15, 16]. Il n'est pas possible de travailler sur ces données volumineuses en utilisant des requêtes traditionnelles du type SQL, ni d'utiliser le système de gestion de base de données relationnelle (SGBDR) pour le stockage. De sorte qu'une grande variété d'outils et de techniques de bases de données évolutives a évolué. Hadoop est un système de traitement de données distribué à source ouvert, et l'une des solutions les plus connues [17]. NoSQL a gagné une importance en tant qu'une base de données non relationnelle avec des logiciels tels que MongoDB, DynamoDB d'Apache [18].

Le reste de ce chapitre est construit comme suit : l'aperçu historique du Big Data est décrite dans la section 2.2. La section 2.3 présente les différentes définitions du Big Data. Les sources du Big Data sont décrites en détail dans la section 2.4. L'évolutivité du Big data est discuté ensuite dans la section 2.5. La section 2.6 présente l'architecture et les modules d'Hadoop. Dans la section 2.7, les solutions Big Data proposées pour le système de calcul en clusters sont discutées. Et enfin, la section 2.8 récapitule les technologies de Big Data.

2.2 Aperçu historique du Big Data

Plusieurs études ont été menées sur les points de vue et les évolutions historiques dans le domaine de "Big Data". le Gil Press [19] a fourni un bref historique de Big Data à partir de 1944, qui était basé sur le travail de Rider [20]. Il a couvert 68 ans d'histoire de l'évolution du Big Data entre 1944 et 2012 et a illustré 32 événements liés au Big Data dans l'histoire récente de la science des données. En comparaison avec la revue de Gil Press, Frank Ohlhorst [21] a établi l'origine du Big Data en 1880, lors du 10e recensement américain. Le véritable problème au XIXe siècle était une question de statistiques, à savoir comment enquêter et documenter 50 millions de citoyens nord-américains. Bien que le Big Data puisse contenir des calculs de certains éléments statistiques, ces deux termes "Big" et "Data" ont aujourd'hui des interprétations différentes.

De même, Winshuttle [22] pense que l'origine du Big Data remonte au XIXe siècle. Selon Winshuttle, si les ensembles de données sont si volumineux et complexes et dépassent les capacités traditionnelles de traitement et de gestion, alors ces ensembles de données peuvent être considérés comme du Big Data. Par rapport à la revue de Gil Press, l'étude

de Winshuttle met l'accent sur la planification des ressources d'entreprise et la mise en œuvre sur l'infrastructure en nuage. En outre, l'étude prévoit également une croissance des données jusqu'en 2020.

La plus longue période de revue historique pour Big Data appartient à la description de Bernard Marr [23]. Il a retracé l'origine de Big Data jusqu'à 18000 ans avant notre ère. Marr a fait valoir que nous devrions prêter attention aux fondements historiques des Big Data, qui sont des approches différentes pour l'homme de saisir, stocker, analyser et récupérer à la fois des données et des informations. De plus, Marr pense que la première personne qui a utilisé le terme "Big Data" est Erik Larson [24]. En revanche, Steve Lohr [25] n'est pas d'accord avec le point de vue de Marr. Il soutient que le simple fait d'adopter le terme à lui seul pourrait ne pas avoir la connotation de Big Data d'aujourd'hui car le terme "Big Data" est tellement générique que la recherche de son origine n'était pas seulement un effort pour trouver une référence précoce à ces deux mots utilisés ensemble. L'objectif était plutôt d'utiliser très tôt le terme qui suggère son interprétation actuelle ; c'est-à-dire non seulement beaucoup de données, mais aussi différents types de données traitées de manière nouvelle".

Une autre revue historique a été réalisée par Visualizing.org [26]. Elle s'est concentrée sur la chronologie de la façon dont pour mettre en œuvre le Big Data. Sa description historique est principalement déterminée par les événements liés à l'initiative "Big Data" de nombreuses sociétés Internet et informatiques, telles que Google, Youtube, Yahoo, Facebook, et Twitter. Il a souligné l'impact significatif de Hadoop dans l'histoire du Big data. Elle a principalement mis en évidence le rôle important de Hadoop dans le Big Data. Sur la base de ces études, nous présentons l'histoire de Big Data, Hadoop et son écosystème dans la figure 2.1.

2.3 Définition du Big Data

Intuitivement, ni le volume de données d'hier (taille absolue) ni celui d'aujourd'hui ne peut être défini comme "Big". De plus, le "Big" d'aujourd'hui peut devenir le "petit" de demain. Afin de clarifier précisément le terme "Big Data" et de trancher le débat, nous pouvons étudier et comprendre les différentes définitions ont proposé pour le "Big Data".

Big Data il-même est une conception très abstraite. Nous pouvons littéralement comprendre cela comme "données massives". Cependant, ce sens littéral est loin de la définition exacte. En fait, il n'existe pas de définition officielle de ce type pour le Big Data. Cependant, sur la base du résumé des propriétés et des fonctions du Big Data, nous pouvons conclure que la définition est celle de la théorie des "3Vs" [27] : Le Big Data est une sorte de taille de données chargées qui contient les propriétés de volume, de variété et de vitesse. En outre, la véracité, définie par IBM [28], peut également être incluse dans les propriétés du Big Data. Yuri Demchenko [29] en 2013 a étendu les attributs 4Vs d'IBM à 5 Vs, qu'il a ajouté l'attribut valeur. Ainsi Microsoft a étendu les attributs 5Vs d'Yuri Demchenko à 6Vs [30], qu'il a ajouté la visibilité.































- 1997. "The problem of Big Data", chercheurs de la NASA, article de Michael Cox et David Ellsworth 
- 1998. Google a été fondé 
- 1999. Fondation logicielle Apache (Apache Software Foundation) a été créée 
- 2000. Doug Cutting a lancé son projet de recherche d'indexation: Lucene 
- 2000. L. Page et S. Brin ont écrit l'article " L'anatomie d'un moteur de recherche web hypertextuel à grande échelle "
- 2001. Les 3V, l'article de Doug Laney «Gestion des données 3D: contrôle du volume, de la vitesse et de la variété des données. 
- 2002. Doug Cutting et Mike Caffarella ont fondé Nutch, un sous-projet de Lucene pour l'exploration de sites Web 
- 2003. Sanjay Ghemawat et al. publié "The Google File System (GFS)"
- 2003. Cutting et Caffarella ont adopté l'idée GFS et ont créé Nutch Distributed File System (NDFS) plus tard, il est devenu HDFS (Hadoop Distributed File System)
- 2004. Google a commencé à développer Big Table 
- 2004. Yonik Seeley a créé Solr pour un moteur de recherche de schémas orientés texte, dominant en lecture, orienté document et flexible 
- 2004. Jeffrey Dean et Sanjay Ghemawat ont publié "Simplified Data Processing on Large Cluster using MapReduce"
- 2005. Nutch a établi Nutch MapReduce
- 2005. Damien Katz a créé Apache CouchDB (Cluster of Unreliable Commodity Hardware), ancien Lotus Notes 
- 2006. Cutting et Caffarella ont lancé Hadoop ou un sous-projet de Nutch 
- 2006. Yahoo Research a développé Apache Pig fonctionne sur Hadoop 
- 2007. 10gen, une start-up a travaillé sur la plateforme en tant que service (PaaS). Plus tard, il est devenu MongoDB 
- 2007. Projet de Taste
- 2008. Apache Hive (étendre SQL), HBase (léger les données) et Cassandra (sans schéma) pour prendre en charge Hadoop   
- 2008. Mahout, un sous-projet de Lucene intégré Taste 
- 2008. Hadoop devient le projet ASF de haut niveau
- 2008. TUB et HPI ont lancé le projet Stratosphere et plus tard il est devenu Apache Flink 
- 2009. Hadoop combine HDFS et MapReduce. Tri d'un TB 62 secondes sur 1460 nœuds
- 2010. Google sous licence à ASF Hadoop  
- 2010. Apache Spark, une plateforme de calcul en cluster s'étend de MapReduce pour les primitives en mémoire 
- 2011. Apache Storm a été lancé pour une distribution cadre de calcul pour flux de données 
- 2012. Apache Dil pour moteur de requêtes SQL sans schéma pour Hadoop, NoSQL et Cloud Storage
- 2012. Phase 3 de Hadoop-Emergence de "Yet Another Resource Negotiator" (YARN) et Hadoop 2 
- 2013. Mesos est devenu un projet Apache de haut niveau 
- 2014. Spark avait plus de 465 contributeurs en 2014, les projets ASF les plus actifs sont  
- 2015. Entre Zettabyte ère 
- 2015. Google et Microsoft mènent des constructions massives de centres de données  
- 2017. Huawei et Tencent ont rejoint Alibaba dans la construction de centres de données majeurs en Chine. 
- 2018. Principaux opérateurs de centres de données ont commencé la migration vers des vitesses de données de 400G.
- 2019. Technologie Silicon Photonics a commencé à avoir un impact positif sur les architectures de réseau des datacenters.
- 2020. Edge Computing révisera le rôle du cloud dans les secteurs clés de l'économie. 
- 2021. Vitesses du centre de données devraient dépasser 1 000G.
- 2025. Centres de données seront de plus en plus intégrés aux appareils

FIGURE 2.1 – Histoire du Big Data et les outils d'analyse du Big Data.

2.3.1 Gartner - Définition des 3Vs

Depuis 1997, de nombreux attributs ont été ajoutés à Big Data. Parmi ces attributs, trois d'entre eux sont les plus populaires et ont été largement cités et adoptés. Le premier est ce qu'on appelle l'interprétation de Gartner ou 3Vs ; la racine de ce terme remonte à février 2001. Elle a été énoncée par Douglas Laney [27] dans son livre publié par le groupe Meta, que Gartner a ensuite acquis en 2004. Douglas a remarqué qu'en raison de l'essor des activités de commerce électronique, les données se sont développées selon trois dimensions, à savoir :

- **Volume** signifie que les données générées en volume cumulé et en flux. Ce volume ou ces ensembles de données à grande échelle représentent le défi le plus immédiat pour les structures informatiques conventionnelles, car ce volume de données est beaucoup plus difficile à traiter et à analyser par les logiciels et les algorithmes d'extraction traditionnels. C'est l'aspect qui vient à l'esprit de la plupart des gens lorsqu'ils pensent au Big Data. De nombreuses entreprises disposent déjà d'une grande quantité de données archivées sous forme de journaux mais n'ont pas la capacité de traiter ces données. L'avantage tiré de la capacité à traiter de grandes quantités d'informations est le principal attrait de l'analyse des Big Data.
- **Vitesse** fait référence à la vitesse croissante à laquelle ces données sont créées, donc à la vitesse croissante à laquelle les données peuvent être traitées, stockées et analysées par des bases de données relationnelles. La vitesse fait référence à la vitesse à laquelle de nouvelles données sont générées et à la vitesse à laquelle les données se déplacent. À propos du message des médias sociaux qui est devenu viral en quelques secondes en 1999, l'entrepôt de données de Wal-Mart stockait

1000 téraoctets (1000000 gigaoctets) de données. En 2012, il avait accès à plus de 2.5 pétaoctets (2500000 gigaoctets) de données. Chaque minute de chaque jour, nous téléchargeons des centaines d'heures de vidéo sur YouTube. Nous envoyons plus de 200 millions d'e-mails par le biais de Gmail.

- **Variété** implique un format distinct des données. L'aspect suivant des grandes données est leur variété. Les données volumineuses ne sont pas toujours structurées et il n'est pas toujours facile de les intégrer dans une base de données relationnelle. Cela signifie que la catégorie à laquelle appartiennent les Big Data est également un fait essentiel qui doit être connu des analystes de données qui traitent une variété de données structurées et non structurées, ce qui augmente considérablement la complexité du stockage et de l'analyse des Big Data. 90% des données générées sont des données non structurées.

Selon l'histoire de la chronologie de Big Data, la définition 3Vs de Douglas Laney a été largement considérée comme les attributs "communs" de Big Data mais il s'est abstenu d'attribuer ces attributs au terme "Big Data".

2.3.2 IBM - Définition des 4Vs

Une autre caractéristique véracité des Big Data qui a été ajoutée à la définition 3Vs par Gartner, et résulte avec la nouvelle définition 4Vs de Big Data d'IBM [28]. IBM définit le nouvel attribut comme suit :

- **Véracité** déduit l'incertitude des données. Lorsque nous traitons un grand volume, une grande vitesse et une grande variété de données, il n'est pas possible que toutes les données soient correctes à 100% ; il y aura des données sales. La qualité des données saisies peut varier considérablement. La précision de l'analyse des données dépend de la véracité des données sources.

La nouvelle caractéristique de véracité V a été ajoutée en raison de la nécessité de la qualité des grandes sources de données demandées par les clients ont commencé à faire face aux initiatives de Big Data.

2.3.3 Yuri Demchenko - Définition des 5Vs

Yuri Demchenko [29] en 2013 a étendu les attributs 4Vs d'IBM à 5Vs, qu'il a ajouté l'attribut valeur :

- **Valeur** est l'aspect le plus important dans le Big Data. Bien que la valeur potentielle des données volumineuses soit énorme, c'est très bien d'avoir accès à des données volumineuses, mais si nous ne pouvons pas les transformer en valeur, elles deviennent inutiles. Il devient très coûteux de mettre en place des systèmes d'infrastructure informatiques pour stocker des données volumineuses, et les entreprises vont exiger un retour sur investissement.

2.3.4 Microsoft - Définition des 6Vs

Afin de maximiser la valeur commerciale, Microsoft a étendu les attributs 5Vs de Yuri Demchenko à 6 Vs, ce qui a permis d'ajouter la visibilité :

- **Visibilité** : Nombreuses organisations ne manquent pas de données. Au contraire, la source de leur frustration est souvent liée au fait que des données précieuses restent enfouies ou cachées dans des systèmes de stockage et des applications difficiles à atteindre ou même à déchiffrer. S'asseoir sur des données précieuses qui peuvent fournir des avantages concurrentiels et convaincants sur le marché mais ne pas pouvoir les exploiter est l'un des problèmes que pose la visibilité des données. La visibilité fait référence aux données qu'un utilisateur autorisé peut localiser, consulter, traiter et sécuriser de manière fiable et rapide. La visibilité fournit également aux utilisateurs un contexte. En d'autres termes, si un élément de données a été "assemblé", la source, les processus, le calendrier et d'autres détails pertinents doivent également être rendus visibles à un utilisateur autorisé.

La figure 2.2 illustre l'ensemble des Vs expliqués précédemment pour le Big Data.

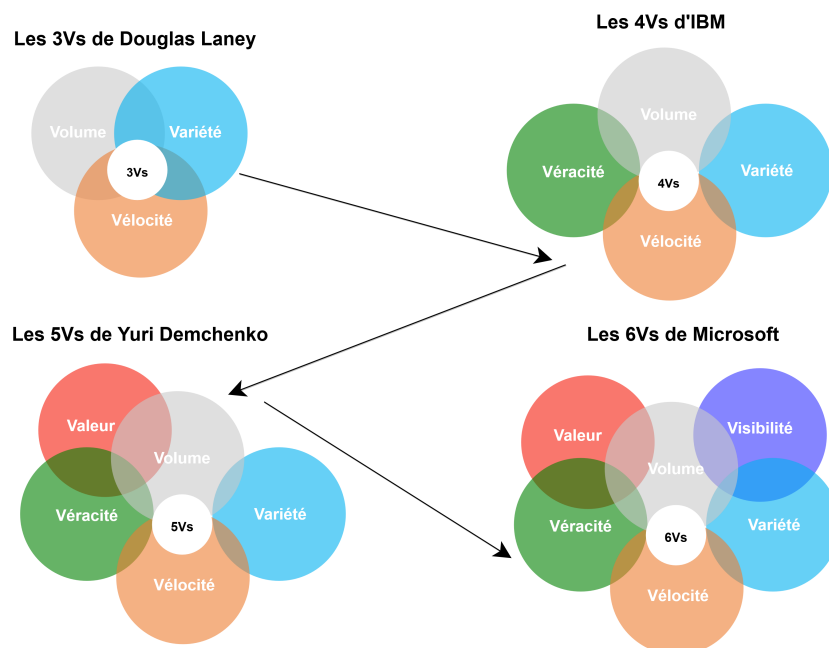


FIGURE 2.2 – Définitions 3Vs, 4Vs, 5Vs et 6Vs du Big Data.

2.4 Sources du Big Data

Aujourd'hui, le Big Data peut être trouvé dans de nombreux scénarios comme les données générées par l'identification radiofréquence, les journaux Web, les réseaux de capteurs, les données satellitaires et géospatiales, les textes internet et les données sociales des réseaux sociaux, l'indexation de la recherche Internet, les enregistrements des détails des appels, les textes et documents Internet, la science atmosphérique, l'astronomie, les

dossiers médicaux, la génomique, la biogéochimie, la biologie et autres surveillances complexes et/ou militaires, la recherche scientifique interdisciplinaire, les archives vidéo, les archives photographiques et le commerce électronique à grande échelle [31].

Premièrement, considérant que Wal-mart [32] gère 39 760 000 des transactions par heure, qui sont collectées dans d'énormes bases de données, et qui devraient comprendre plus de 5.5 pétaoctets de données. L'internet est considéré comme la plus grande source de Big Data, tant le volume de données générées en une minute sur l'internet en 2020 est énorme (figure 2.3). Une nouvelle étude prévoit que toutes les 60 secondes, Google reçoit plus de 483 300 000 requêtes, plus de 563 millions de messages électroniques seront envoyés, plus de 78 millions de vidéos seront regardées sur Youtube et 1 500 heures de vidéo y seront téléchargées, les utilisateurs de Facebook partageront plus de 1 189 200 000 contenus et les utilisateurs de Twitter génèrent plus de 80 millions de tweets [33].

Une source très importante de Big Data est L'Internet des Objets (IdO) [34]. L'IdO est un concept et un modèle qui prend en compte la présence étendue dans l'environnement ou une variété d'objets/choses qui sont capables de coopérer et d'interagir entre eux ou avec d'autres choses afin de produire des services/applications modernes par le biais de liens câblés et sans fil et de systèmes d'adressage uniques [1]. Le monde et le réel, où le numérique et le virtuel convergent pour créer des environnements intelligents qui rendent l'énergie, les transports, les villes et bien d'autres domaines plus intelligents. Le but de l'IdO est de permettre aux choses d'être reliées à n'importe quel endroit, n'importe quand, avec n'importe qui et n'importe quoi par n'importe quel chemin/réseau et n'importe quel service. Les objets de l'IdO se rendent identifiables et deviennent plus intelligents en prenant ou en permettant des décisions contextuelles sur la réalité, en ce sens qu'ils peuvent interconnecter des informations sur eux-mêmes et qu'ils peuvent contacter des informations qui ont été collectées par d'autres choses, ou ils peuvent être des composants, des objets ou des services complexes comme montré la figure 2.3.

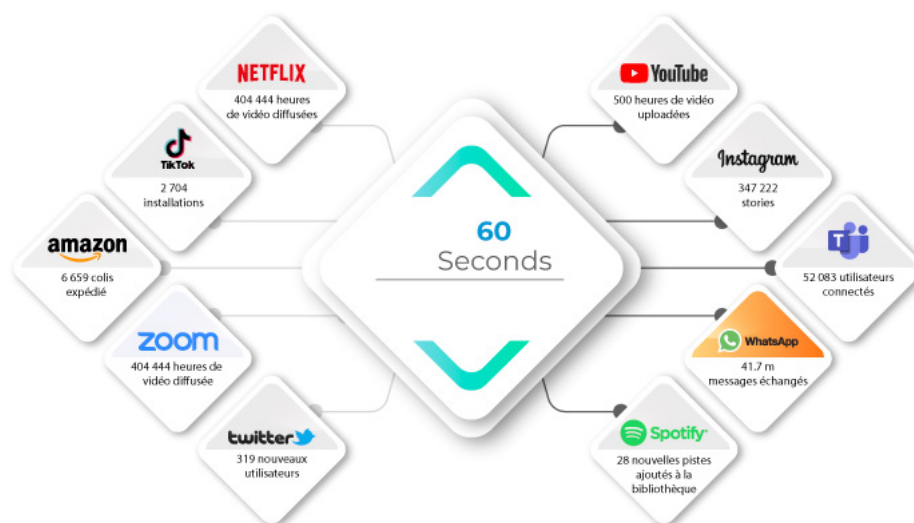


FIGURE 2.3 – Une minute d'Internet en 2020 [1].

2.5 Évolutivité

Actuellement, et avec des volumes de données étonnamment importants, l'extraction de grands ensembles de données est devenue l'une des tâches les plus longues et les plus difficiles à réaliser avec les outils d'extraction de données classiques ou même en utilisant une seule machine. D'où le besoin urgent de méthodes de mise à l'échelle. Ces méthodes de mise à l'échelle permettent de faire face à l'augmentation constante du volume de travail en développant les dispositifs de calcul. La définition de l'extensibilité est la capacité du système ou de la machine à faire face à une charge de travail croissante, ou sa capacité à se développer afin de s'étendre pour faire face à cette croissance croissante. Tout système est évolutif s'il a la capacité d'augmenter sa production totale tout en augmentant la charge de travail en ajoutant certaines capacités telles que le matériel ou en l'améliorant [35]. En général, il est difficile de définir l'extensibilité comme une caractéristique de tout système, mais dans chaque cas particulier il est nécessaire de définir des exigences spécifiques d'extensibilité dans ces tendances qui sont importantes. L'extensibilité est l'une des questions les plus importantes dans de nombreux domaines tels que les routeurs, les bases de données, les réseaux et les systèmes électroniques. On dit qu'un système qui améliore ses performances en ajoutant du matériel proportionnellement à la capacité ajoutée est un dispositif évolutif. Pour mettre à l'échelle un système, il y a deux façons : l'échelle horizontale et l'échelle verticale [36].

2.5.1 Évolutivité horizontale

Mise à l'échelle horizontale est l'ajout ou la suppression de nombreux nœuds informatiques dans le système. Avec la baisse continue des prix des appareils informatiques dont les performances sont de plus en plus efficaces, de nouvelles applications informatiques dans de nombreux domaines tels que les charges de travail en biotechnologie et l'analyse sismique ont été adoptées pour utiliser en même temps des systèmes de base peu coûteux mais très performants. Les spécialistes s'appuient désormais sur l'assemblage de centaines de petits ordinateurs et leur intégration dans un seul système pour obtenir une puissance de calcul supérieure. Ces progrès ont nécessairement conduit à la nécessité de disposer de types de logiciels permettant un contrôle efficace et une bonne manipulation de nombreux ordinateurs connectés entre eux, ainsi que de composants matériels tels que la mémoire de données partagée avec une exécution d'entrée/sortie supérieure. La taille de l'extensibilité à tout système est déterminée par le plus grand nombre de processeurs pouvant être impliqués dans ce système [36].

2.5.2 Évolutivité verticale

Mise à l'échelle verticale est l'ajout ou la suppression de nombreuses ressources d'un seul ordinateur, telles que la mémoire ou les unités centrales. Ce type de mise à l'échelle nous permet de mieux utiliser la technique de virtualisation car elle fournit de nombreuses ressources qui peuvent être partagées pour exécuter plus d'une application à la fois. Afin de construire des systèmes évolutifs pour l'exploitation de données à grande échelle, de

nouveaux outils ont été nécessaires pour analyser cet énorme volume de données. C'est pourquoi nous avons consacré la partie restante de ce chapitre à parler de certains des nouveaux outils utilisés dans l'analyse des données à grande échelle, qui permettent de construire des systèmes d'exploitation minière pouvant être évolutifs [36].

2.6 Hadoop

Hadoop est l'une des plateformes de travail libre. Il a été écrit en Java et a fait ses débuts en octobre 2003. Hadoop a donné un avantage très important en permettant la manipulation et le traitement d'ensemble de données massives grâce à l'utilisation de groupes d'ordinateurs utilisant des modèles logiciels simples. Hadoop est conçu pour étendre le traitement et le stockage d'énormes quantités de données en utilisant des milliers d'ordinateurs plutôt qu'un seul nœud [37].

L'essence de Hadoop se compose de deux parties : la première est la partie du stockage des données qui est connue sous le nom de HDFS. La seconde est la partie qui est responsable du processus de traitement des données par le biais du paradigme de programmation MapReduce. La nature du travail est basée sur la division des gros fichiers de données en petits blocs distribués sur les machines du cluster pour être traités en parallèle en écrivant des codes de programmation dans le MapReduce [38].

L'avantage d'adopter la plateforme Hadoop [39] est que "Hadoop est une plate-forme de calcul et de stockage distribuée, libre et open source. Elle a été créée pour permettre le stockage et le traitement de grandes quantités de données en utilisant des grappes de matériel de base". Cette déclaration décrit également le principe de base de l'architecture Hadoop qui se compose de trois éléments essentiels (voir la figure 2.4) : HDFS pour la fonction de stockage de fichiers, Map pour la fonction de distribution et Reduce pour la fonction de mise en commun.

Cependant, le principal inconvénient de Hadoop est qu'il traite toutes les charges de travail en mode par lot car "Hadoop est un cadre de traitement générique conçu pour exécuter des requêtes et d'autres opérations de lecture par lot sur des ensembles de données massives pouvant aller de dizaines de téraoctets à des pétaoctets" [40]. Cela signifie que la première version de Hadoop ne peut pas gérer les charges de travail de flux et d'interactivité.

2.6.1 Architecture d'Hadoop

Aujourd'hui, Hadoop est un ensemble de sous-projets connexes qui s'inscrivent dans le cadre des infrastructures de calcul distribué. Ces projets sont hébergés par la fondation d'Apache de logiciel qui apporte son soutien à une communauté de projets de logiciels à source ouvert. Bien que Hadoop soit surtout connu pour MapReduce et son HDFS, les autres sous-projets fournissent des services complémentaires, ou s'appuient sur le noyau pour ajouter des abstractions de plus haut niveau. Les sous-projets et leur place dans la pile technologique sont présentés dans la figure 2.4 et brièvement décrits ci-dessous :

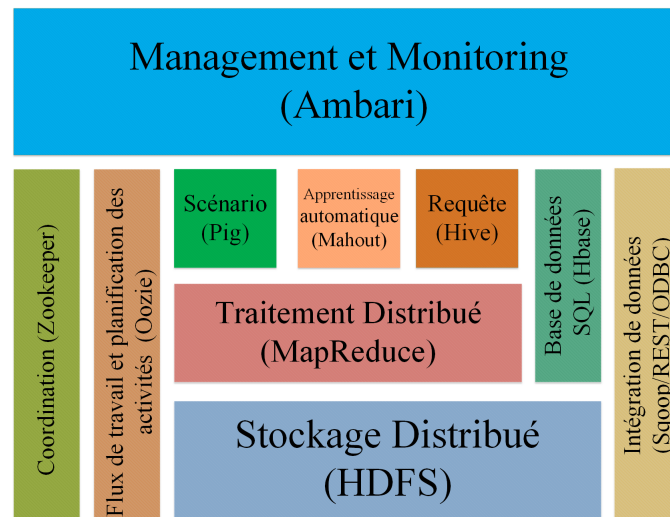


FIGURE 2.4 – Architecture de Hadoop.

2.6.1.1 Gestion et suivi

Contenant des utilitaires et des bibliothèques en Java qui sont nécessaires pour démarrer Hadoop, également il est nécessaire pour le fonctionnement d'autres unités telles que Hive, HBase, etc.

2.6.1.2 Système de fichiers distribués d'Hadoop

Lorsqu'un ensemble de données dépasse la capacité de stockage d'une seule machine physique, il devient nécessaire de le répartir sur plusieurs machines distinctes. Les systèmes de fichiers qui gèrent le stockage à travers un réseau de machines sont appelés systèmes de fichiers distribués. Comme ils sont basés sur un réseau, toutes les complications de la programmation en réseau se mettent en place, ce qui rend les systèmes de fichiers distribués plus complexes que les systèmes de fichiers sur disque ordinaires. Par exemple, l'un des plus grands défis consiste à faire en sorte que le système de fichiers tolère la défaillance d'un nœud sans subir de perte de données.

Hadoop est livré avec un système de fichiers distribué appelé HDFS. Il est considéré comme le secret du succès du système Hadoop, car il représente la classe responsable du stockage des données. Il divise d'abord les gros fichiers de données et les distribue ensuite aux différentes machines du système, ce qui permet d'accéder aux données et de les collecter à grande vitesse. Il produit de nombreuses copies des fichiers de données et les conserve sur un certain nombre de nœuds du cluster Hadoop jusqu'à la fin du processus de traitement [41].

Un cluster HDFS comporte deux types de nœuds fonctionnant selon un schéma maître-esclave : un nœud de nom (le maître) et un certain nombre de nœuds de données (les esclaves) [42]. Le nœud de nom gère l'espace de noms du système de fichiers. Il gère l'arborescence du système de fichiers et les métadonnées de tous les fichiers et répertoires de l'arborescence. Ces informations sont stockées en permanence sur le disque local sous

la forme de deux fichiers : l'image de l'espace de noms et le journal d'édition. Le nœud de nom connaît également les nœuds de données sur lesquels se trouvent tous les blocs d'un fichier donné, mais il ne stocke pas les emplacements des blocs de manière persistante, car ces informations sont reconstruites à partir des nœuds de données au démarrage du système.

Un client accède au système de fichiers au nom de l'utilisateur en communiquant avec le nœud de nom et les nœuds de données. Le client présente une interface du système de fichiers du type POSIX, de sorte que le code utilisateur n'a pas besoin de connaître le nœud de nom et les nœuds de données pour fonctionner [43]. Les nœuds de données sont les unités de travail du système de fichiers. Ils stockent et récupèrent des blocs lorsqu'ils en reçoivent l'ordre (par les clients ou le nœud de nom), et ils font périodiquement rapport au nœud de nom avec des listes de blocs qu'ils stockent.

Sans le nœud de nom, le système de fichiers ne peut pas être utilisé. En fait, si la machine qui exécute le nœud de nom était oblitérée, tous les fichiers du système de fichiers seraient perdus puisqu'il n'y aurait aucun moyen de savoir comment reconstruire les fichiers à partir des blocs sur les nœuds de données. Pour cette raison, il est important de rendre le nœud de nom résilient à l'échec, et Hadoop fournit deux mécanismes pour cela. Le premier moyen est de sauvegarder les fichiers qui constituent l'état persistant des métadonnées du système de fichiers. Hadoop peut être configuré de manière à ce que le nœud de nom écrive son état persistant dans plusieurs systèmes de fichiers. Ces écritures sont synchrones et atomiques. Le choix de configuration habituelle est d'écrire sur le disque local ainsi que sur une montagne NFS distante.

Il est également possible d'exécuter un nœud de nom secondaire [44] qui est malgré son nom n'agit pas comme un nœud de nom. Son rôle principal est de fusionner périodiquement l'image de l'espace de noms avec le journal d'édition pour éviter que ce dernier ne devienne trop volumineux. Le nœud de nom secondaire fonctionne généralement sur une machine physique séparée, car il nécessite beaucoup de CPU et autant de mémoire que le nœud de nom pour effectuer la fusion. Il conserve une copie de l'image de l'espace de noms fusionné, qui peut être utilisée en cas de défaillance du nœud de nom. Cependant, l'état du nœud de nom secondaire [43] est en retard sur celui du primaire, de sorte qu'en cas de défaillance totale des données primaires, la perte est presque garantie. Dans ce cas, la procédure habituelle consiste à copier les fichiers de métadonnées du nœud de nom qui se trouvent sur le NFS vers le nœud secondaire et à l'exécuter en tant que nouveau primaire.

Le HDFS fonctionne selon le modèle "maître et esclave", dans lequel le nœud de nom joue le rôle de maître qui suit le processus de stockage et de distribution des données, et le nœud de données joue le rôle d'esclave qui est basé sur la synthèse et l'assemblage des données des différents nœuds au sein du cluster Hadoop [45](voir la figure 2.5).

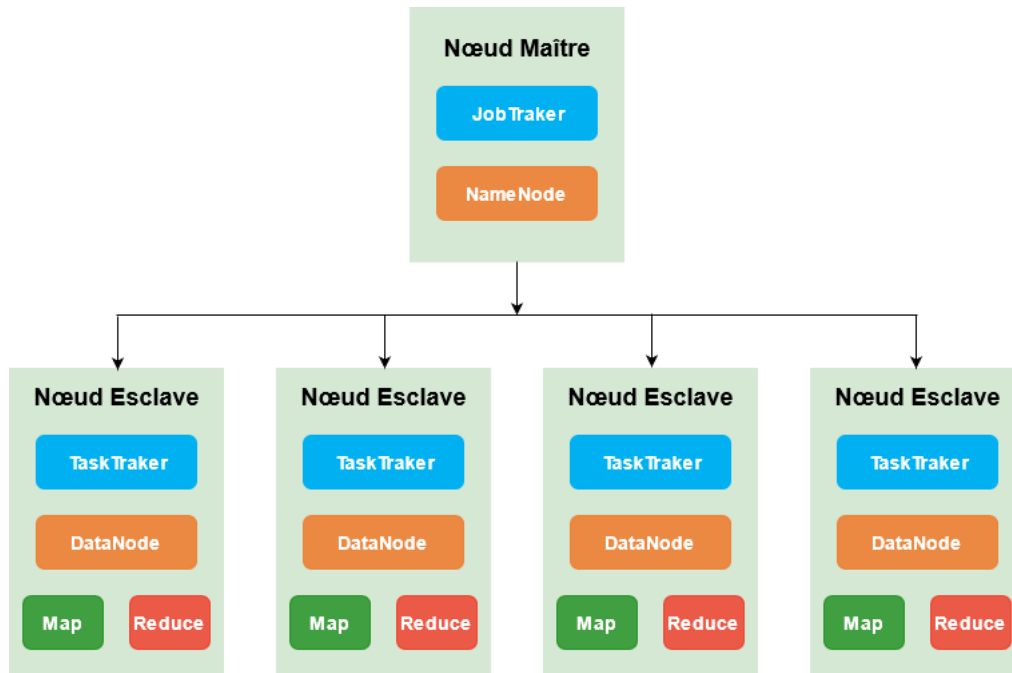


FIGURE 2.5 – Architecture "Maître et esclave" de Hadoop.

2.6.1.3 YARN

YARN est la technologie de gestion des ressources et de planification des tâches dans le cadre du traitement distribué à source ouvert Hadoop. YARN est l'un des principaux composants d'Apache Hadoop. Il est chargé d'allouer les ressources système aux différentes applications s'exécutant dans un cluster Hadoop et de planifier les tâches à exécuter sur les différents nœuds du cluster [46]. YARN est devenue un sous-projet d'Apache Hadoop au sein de la fondation d'Apache de logiciel en 2012 et a été l'une des principales fonctionnalités ajoutées dans Hadoop 2.0 qui a été publié en octobre 2013 [47].

2.6.1.4 Modèle de programmation MapReduce

MapReduce est un modèle de programmation basé sur Java a été créé par Google pour traiter les données stockées dans le HDFS. MapReduce désagrège la tâche de traitement des grandes données en tâches plus petites. MapReduce agrège les données et les résultats des différentes machines du cluster Hadoop pour terminer le processus d'analyse dans un modèle parallèle [48]. Le modèle MapReduce est principalement basé sur le YARN qui permet le traitement parallèle dans l'analyse de grandes données. En utilisant MapReduce, il est facile d'écrire des applications qui peuvent être mises en œuvre en utilisant un grand nombre de machines, en tenant compte de la gestion des pannes et des défauts. Le principe de base du travail dans MapReduce est que MapJob envoie une requête pour traiter les fichiers de données stockés dans le HDFS dans le cluster Hadoop, mais aussi que le JobReduce collecte tous les résultats en une seule sortie et les stocke dans le fichier de sortie [49]. La fonction Map dans le système Hadoop prend les entrées et les divise en parties indépendantes et les sorties de cette étape seront des entrées pour l'étape suivante.

Ensuite, le système Hadoop rassemble ces résultats dans un fichier de sortie et toutes les entrées et sorties des différentes tâches sont stockées dans le HDFS (voir la figure 2.6).

La stratégie de base de MapReduce est de diviser pour mieux régner. Afin de réaliser efficacement différentes applications gourmandes en données avec MapReduce sur le cadre HDFS, Dean et Ghemawat [50] ont présenté un processus en cinq étapes ou modèle de programmation comme le montre la figure 2.6.

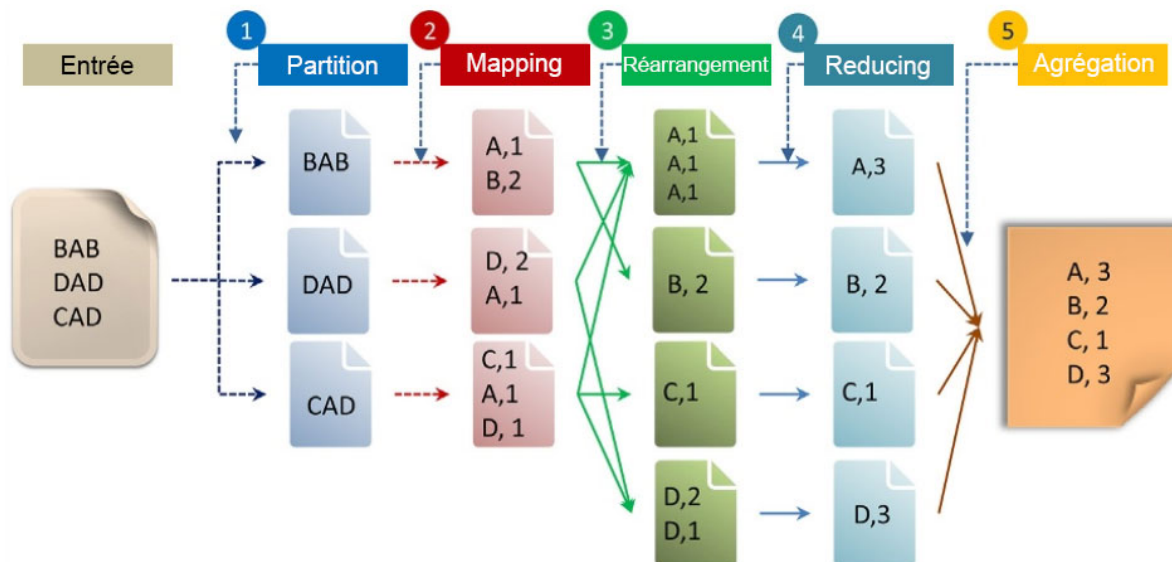


FIGURE 2.6 – Modèle de programmation MapReduce.

Lin et al [51] ont simplifié ce processus en le ramenant à trois étapes : Mapping, Regroupement et Reducing. Comme le montre la figure 2.6, la première étape consiste à diviser le fichier d'entrée en trois fichiers. La deuxième étape consiste à générer un processus d'une paire clé/valeur par un utilisateur (ou client) qui spécifie la fonction. Dans l'exemple ci-dessus, il s'agit de compter le nombre de lettres différentes (A, B, C et D) avec une quantité correspondante dans chaque fichier scindé. Le premier fichier scindé contient les lettres "BAB". La lettre "A" est comptée comme 1 et la lettre "B" dans la troisième étape. La fonction de brassage consiste à générer une paire clé/valeur intermédiaire, qui consiste à trier dans un seul fichier la même lettre (ou clé) et la même quantité (ou valeur) provenant de différents fichiers séparés. La quatrième étape consiste à fusionner toutes les valeurs intermédiaires (3, 2, 1 et 2) associées à la même lettre (A, B, C et D). L'étape finale consiste à agréger ces paires clé/valeur dans un fichier de sortie. Ici, la "clé" est égale aux différents types de lettres à compter et "valeur" est égal à la quantité de chaque lettre.

Du point de vue de la programmation, MapReduce a deux autres significations : "Mapping", c'est diviser pour distribuer et "Reducing", c'est mélanger et trier en parallèle. L'un des principaux avantages de MapReduce est sa capacité de traitement des données sans partage, ce qui signifie que tous les mappers peuvent traiter leurs données indépendamment.

La caractéristique du traitement des données sans partage permet à MapReduce d'exé-

cuter un programme simple sur des milliers, voire des millions de machines peu fiables et homogènes en parallèle et d'accomplir une tâche en très peu de temps. Théoriquement parlant, il permet à tout programmeur d'accéder instantanément (théoriquement) ou dans un délai acceptable (pratiquement) à un type de ressources informatiques presque illimité, par exemple une infrastructure en nuage. Plusieurs plateformes de calcul ont mis en œuvre leurs propres modèles de traitement MapReduce, comme Couch DB, Cloud MapReduce et Aneka [52].

2.6.2 Modules d'accès aux données d'Hadoop

2.6.2.1 Pig

Pig est conçu par Yahoo pour rendre le processus d'analyse de données volumineuses plus facile et plus efficace. Sa principale mission est de fournir un niveau élevé de flux de données dans le système de Hadoop et il se caractérise par sa facilité d'utilisation et son évolutivité. La caractéristique la plus importante de Pig est sa structure ouverte, qui permet de traiter facilement des données volumineuses en parallèle [53].

Pig augmente le niveau d'abstraction pour le traitement de grands ensembles de données. Avec MapReduce, il existe une fonction de Mapping et une fonction de Reducing, et il peut être difficile de trouver comment adapter votre traitement de données à ce schéma, qui nécessite souvent plusieurs étapes MapReduce. Avec Pig, les structures de données sont beaucoup plus riches, étant généralement multi-valeurs et imbriquées. Aussi l'ensemble des transformations que vous pouvez appliquer aux données sont beaucoup plus puissantes qui incluent des jointures, par exemple, qui ne sont pas pour les faibles de cœur dans MapReduce. Le Pig est formé de deux parties [54] :

- Langue utilisée pour exprimer les flux de données, appelée Pig Latin.
- Environnement d'exécution pour exécuter les programmes Pig Latin. Il existe actuellement deux qui sont : exécution locale dans une seule JVM et exécution distribuée sur un cluster Hadoop.

2.6.2.2 Hive

Hive a été développé par Facebook et est un entrepôt de données qui a été placé sur le dessus du Hadoop, fournissant un langage simple comme SQL appelé HiveQL, utilisé pour analyser, interroger et résumer les grandes données. La caractéristique la plus importante est qu'il rend le processus d'interrogation des données importantes plus rapide grâce à l'utilisation du processus d'indexation [55]. Hive a des principaux composants et des interactions avec Hadoop qui sont :

- **Interfaces externes** : Hive fournit à la fois des interfaces utilisateur comme la ligne de commande (CLI) et l'interface utilisateur web, et des interfaces de programmation d'applications (API) comme JDBC et ODBC.
- **Serveur thrift de Hive** expose une API client très simple pour exécuter les instructions HiveQL. Thrift est une plateforme pour les services multilingues, où un serveur écrit dans un langage (comme Java) peut également prendre en charge

des clients dans d'autres langages. Les clients Thrift de Hive générés dans différents langages sont utilisés pour construire des pilotes communs comme JDBC (java), ODBC (C++) et des scripts pour les pilotes écrits en php, perl, python, etc.

- **Métastore** est le catalogue du système. Tous les autres composants du Hive interagissent avec le métastore.

2.6.3 Modules d'intégration de données d'Hadoop

2.6.3.1 HBase

La composante HBase est une base de données orientée colonne qui stocke les données dans le HDFS. HBase prend en charge la lecture aléatoire et le calcul par lot en utilisant MapReduce. Nous pouvons créer d'énormes tables contenant des millions de lignes et de colonnes et les stocker dans le système Hadoop en utilisant HBaseNoSQL. La meilleure utilisation est lorsque vous devez lire ou écrire de manière aléatoire pour accéder à des données volumineuses [55].

HBase s'attaque au problème de l'échelle dans la direction opposée. Elle est construite à partir de la base pour mettre à l'échelle de façon linéaire en ajoutant simplement des nœuds. HBase n'est pas relationnel et ne supporte pas le SQL, mais étant donné l'espace de problème approprié, il est capable de faire ce qu'un SGBDR ne peut pas faire : héberger avec compétence de très grandes tables peu peuplées sur des clusters fabriqués à partir de matériel de base.

2.6.3.2 Sqoop

La tâche principale de sqoop est de transférer des données de ses sources externes et de les placer aux endroits désignés dans le système Hadoop, comme le HDFS également est utilisé pour transférer des données de l'intérieur du Hadoop pour les stocker dans des sources de données externes. sqoop est un transfert de données qui fonctionne en parallèle, ce qui le rend plus efficace dans l'analyse des données et accélère leur copie [56].

2.6.3.3 Flume

Les données des médias sociaux sont généralement acquises à l'aide de canaux. Flume est un logiciel à source ouvert développé par Cloudera pour servir de service d'agrégation et de déplacement de très grandes quantités de données dans un cluster Hadoop au fur et à mesure de leur production ou peu après. L'utilisation principale de Flume est de rassembler les fichiers journaux de toutes les machines du cluster pour les conserver dans un magasin centralisé tel que HDFS. Dans ce cas, nous devons créer des flux de données en construisant des chaînes de nœuds logiques et en les reliant à la source et aux puits. Par exemple, si vous voulez déplacer des données d'un journal d'accès apache vers le HDFS, vous devez créer une source par la queue access.log et utiliser un nœud logique pour l'acheminer vers un puits HDFS. La plupart des déploiements de canaux ont une

conception à trois niveaux. Le niveau des agents est constitué d'agents de canaux situés à proximité des sources de données à déplacer. Le niveau collecteur est constitué de plusieurs collecteurs qui collectent chacun des données provenant de plusieurs agents et les transmettent au niveau stockage [57] qui est constitué d'un système de fichiers comme HDFS.

2.6.4 Modules de contrôle et de maintenance d'Hadoop

2.6.4.1 Oozie

Apache Oozie est un logiciel de la fondation d'Apache servant à l'ordonnancement de flux dédié au logiciel Hadoop. Il est implémenté comme une application web Java exécuté dans un conteneur de servlets Java et est distribué sous la licence Apache 2.0.

Les flux de travail dans Oozie sont définis comme une collection de flux de contrôle et d'actions dans un graphe orienté acyclique. Les nœuds du flux définissent le début et la fin d'un flux de travail (Début, fin et des nœuds de défaillance), ainsi qu'un mécanisme destiné à contrôler la trajectoire d'exécution de flux de travail (décision, la fork et nœuds de jonction). Les nœuds d'action sont le mécanisme par lequel un flux de travail déclenche l'exécution d'une tâche de calcul ou d'un traitement. Oozie prend en charge différents types d'actions dont MapReduce à savoir les opérations HDFS, Pig, SSH, et l'envoi email. Oozie peut également être étendu pour supporter d'autres types d'actions.

Les flux de travail dans Oozie peuvent utiliser des paramètres en utilisant des variables comme \$inputDir définis dans flux de travail. Lorsqu'une tâche est lancée, les valeurs pour les paramètres doivent être fournies. Si elle est correctement paramétrée (en utilisant différents répertoires de sortie), une même tâche peut être utilisée simultanément par plusieurs processus [58].

2.6.4.2 Zookeeper

ZooKeeper est un service centralisé pour la maintenance des informations de configuration, le nommage, la synchronisation distribuée et la fourniture de services de cluster. Tous ces types de services sont utilisés sous une forme ou une autre par des applications distribuées. Chaque fois qu'ils sont mis en œuvre, il y a beaucoup de travail à faire pour corriger les bogues et les conditions de course qui sont inévitables. En raison de la difficulté de mise en œuvre de ces types de services, les applications lésinent généralement au départ, ce qui les rend fragiles en présence de changements et difficiles à gérer. Même lorsqu'elles sont bien faites, les différentes implémentations de ces services entraînent une complexité de gestion lors du déploiement des applications [59].

2.6.5 Autres modules de l'écosystème Hadoop

2.6.5.1 Mahout

L'objectif initial de Mahout était de construire une bibliothèque d'apprentissage automatique basée sur Java qui couvre tous les algorithmes ou techniques d'apprentissage automatique en théorie, mais elle peut principalement traiter trois types d'algorithmes d'apprentissage automatique en pratique :

- Filtrage collaboratif (moteurs de recommandation)
- Regroupement
- Classification

Si d'autres algorithmes d'apprentissage sont nécessaires, nous devons vérifier l'URL d'Apache Mahout [60] et découvrir si MapReduce peut prendre en charge un algorithme particulier ou non avant que cet algorithme puisse être appliqué dans un environnement à grande échelle. En d'autres termes, Mahout n'est pas une bibliothèque d'apprentissage automatique universelle. En plus des problèmes d'extensibilité, Hadoop est très lent pour les charges de travail d'apprentissage automatique. Il a conduit au développement d'écosystèmes complémentaires, tels que Hama, Storm, Spark et Flink, qui ont permis de remédier aux faiblesses des systèmes basés sur MapReduce.

2.6.5.2 Apache Kafka

Kafka est un projet de courtier en messages à source ouvert développé par la fondation d'Apache de logiciel et écrit en Scala [61]. Le projet vise à fournir une plateforme unifiée, à haut débit et à faible latence pour traiter les flux de données en temps réel. Un seul courtier Kafka peut gérer des centaines de mégaoctets de lecture et d'écriture par seconde provenant de milliers de clients. Afin de soutenir la haute disponibilité et l'extensibilité horizontale, les flux de données sont partitionnés et répartis sur un cluster de machines [61]. Kafka dépend de Zookeeper de l'écosystème Hadoop pour la coordination des nœuds de traitement. Les principales utilisations de Kafka sont les situations où les applications ont besoin d'un débit très élevé pour les messages tout en répondant aux exigences de faible latence, de haute disponibilité et d'extensibilité.

2.7 Big Data et système de calcul en clusters

SCC (Système de calcul en Clusters) vient de résoudre les problèmes de la technique standard. L'objectif de cette technique est d'améliorer les performances et la consommation d'énergie d'un seul processeur pour le stockage et l'extraction de grands ensembles de données, en utilisant la programmation parallèle pour lire et traiter les ensembles de données massives sur plusieurs disques et unités centrales. Ce qui rend ces systèmes un peu plus performants que la technique standard est l'organisation physique des nœuds de calcul dans le cluster. Actuellement, ce type de cluster ne résout pas entièrement le

problème car il comporte des difficultés, à savoir les défaillances des nœuds, la complexité des calculs, le goulot d'étranglement du réseau et la programmation distribuée. Tous ces problèmes surviennent lorsque nous exploitons et stockons le volume massif de données à l'aide de l'informatique en clusters. Pour résoudre ces défis, Google a inventé un nouveau cadre de traitement des données Big Data appelé MapReduce, pour gérer le traitement des données à grande échelle sûre de grands clusters de serveurs de produits de base.

Dans le modèle de calcul sous-jacent, la plupart des calculs sont effectués sur une seule unité centrale, avec sa mémoire principale, son cache et un disque local (un nœud de calcul). L'idée de ce modèle est que chaque algorithme est construit et mis en œuvre comme un flux d'instructions en série. Ces instructions sont exécutées sur une unité centrale d'un ordinateur et accèdent aux données qui sont en mémoire. Une seule instruction peut être exécutée à la fois, une fois que cette instruction est terminée, la suivante est exécutée. Il s'agit d'un modèle de base que nous utilisons pour mettre en œuvre toutes sortes d'algorithmes dans l'apprentissage automatique et les statistiques [62].

La question qui se pose est la suivante : que se passe-t-il si les données croissent plus vite que la vitesse de traitement et que tout ne peut pas tenir en mémoire en même temps ? Pour résoudre ce problème, la fouille de données arrive, avec l'idée que ses algorithmes classiques examinent le disque en plus de la mémoire [62]. Pendant le processus en cours, les données sont divisées en parties sur le disque, l'algorithme ne peut mettre en mémoire qu'une partie des données à la fois, les traiter par lots, et renvoyer les résultats partiels sur le disque. Selon le développement scientifique et technologique de ces dernières années, les données augmentent plus rapidement, et le principe des algorithmes classiques de fouille de données n'est plus efficace en ce qui concerne la durée d'exécution et le rapport d'accélération [63]. Par exemple : le réseau de capteurs sans fil est déployé pour apprendre quelque chose sur le comportement de leur environnement.

- 3.5 téraoctets de flux de données arrivent chaque jour
- Largeur de bande de lecture du disque = 50 Mo/secs
- Temps de lire = $3500000/50 = 70000$ secs = 19.44444 min
- Encore plus de temps pour faire quelque chose d'utile avec les données

Dans cet exemple, les nœuds de capteurs envoient chaque jour une grande quantité de données d'environ 3.5 téraoctets à la station de base. Si la station de base utilise le modèle de calcul classique, il faut 19.44444 min pour lire ces données massives sans faire quelque chose d'utile avec les données. C'est un temps long et c'est juste le temps de lire les données en mémoire [63]. La limitation fondamentale de ce modèle est la largeur de bande, la largeur de bande des données entre le disque et le CPU est égale à 50MB/secs. L'amélioration des performances/du rendement énergétique d'un seul processeur a été confrontée à divers défis découlant de la fin de la mise à l'échelle technologique classique [64]. Ainsi, l'importance d'exécuter efficacement les applications sur un système informatique parallèle/distribué n'a cessé de croître.

2.7.1 Système de calcul en clusters

Les performances des ordinateurs personnels et des postes de travail sont améliorées grâce aux SCC qui sont devenus populaires ces dernières années. Ce cluster devient financièrement et techniquement intéressant pour concevoir un supercalculateur basé sur

l'utilisation d'une technique standard en connectant plusieurs ordinateurs simples dans un réseau à haut débit [65]. Il est utilisé pour exécuter un seul programme (à forte intensité de calcul) sur plusieurs machines en parallèle. L'exemple le plus populaire d'ordinateur en cluster est celui de Beowulf, basé sur Linux [64]. Les SCCs sont conçus pour résoudre les problèmes de la technique classique, en utilisant la programmation parallèle pour lire et traiter les ensembles de données massives sur plusieurs disques et CPU. Ce qui rend ces systèmes un peu plus performants que la technique classique est l'organisation physique des nœuds de calcul en clusters [64,66]. Ainsi, l'architecture de calcul parallèle est organisée comme suit : l'architecture est constituée de plusieurs racks connectés par des commutateurs de base, et le commutateur de base a une bande passante élevée qui peut varier de 2 à 10 gigabits entre les racks. Chaque rack possède de 16 à 64 nœuds Linux de base (nœuds de calcul), et ces nœuds sont connectés par un commutateur avec une bande passante égale à un gigabit entre toutes paire de nœuds [67]. La figure 2.7 illustre l'architecture d'un système de calcul à grande échelle. Cependant, il peut y avoir beaucoup plus de racks et beaucoup plus de nœuds de calcul par rack.

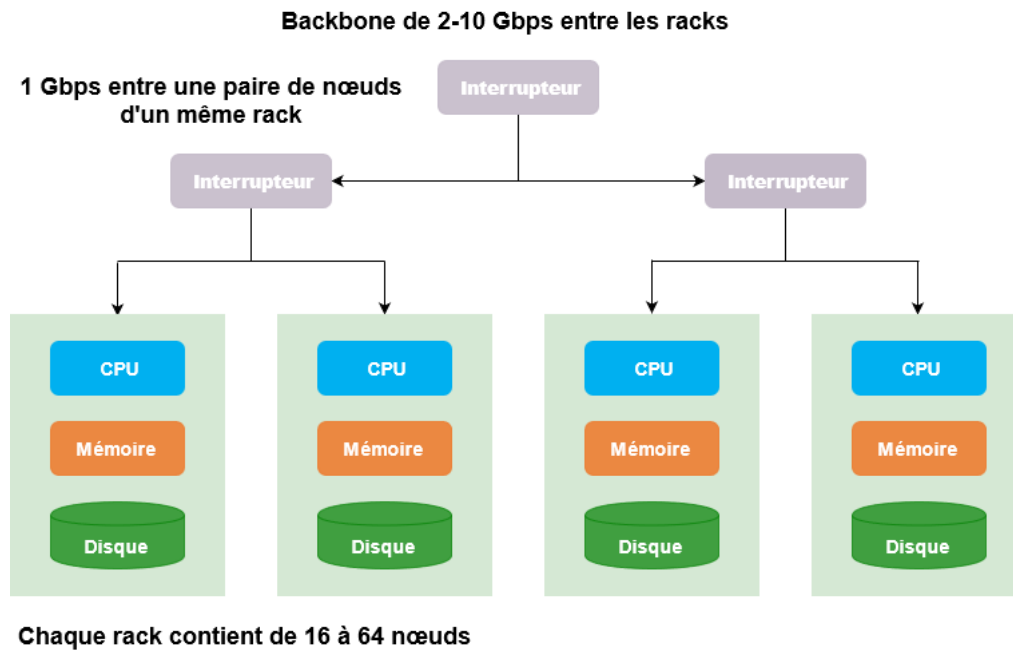


FIGURE 2.7 – Architecture d'un système informatique avec multiples racks.

Le calcul en clusters a été développée en 1960 par IBM [67] pour stocker et exploiter les énormes ensembles de données. Aujourd'hui, ce type de clusters ne résout pas entièrement le problème car il s'accompagne des défis indiqués ci-dessous :

- *Défaillances de nœuds* : est le premier et le plus important des défis. Ses modes de défaillance dominants sont la perte d'un rack entier et la perte d'un seul nœud. Par exemple, Google [68] utilise le calcul en clusters pour traiter la très grande quantité de recherche mondiale ; s'il utilise un million de serveurs par cluster, il va connaître mille défaillances par jour.
- *Complexité de Calculs* : la plupart des calculs de fouille de données appliqués à une quantité massive de données prennent des minutes, des heures, voire des jours sur

des milliers de nœuds de calcul [69]. Si nous devions interrompre et relancer ces calculs chaque fois qu'un nœud de calcul échoue, ces calculs pourraient ne jamais aboutir.

- *Goulot d'étranglement du réseau* : dans le SCC, le réseau peut devenir un goulot d'étranglement [66]. Par conséquent, ce problème du goulet d'étranglement se rapporte à une condition discrète dans laquelle le flux de données est limité par des nœuds de calcul individuels dans un rack ou des racks individuels [70].
- *Programmation distribuée* : le dernier défi des SCCs est que la programmation distribuée peut être très difficile, même les programmeurs sophistiqués ont du mal à écrire correctement des programmes distribués et à éviter les conditions de course et divers types de complications [69, 71].

Tous ces problèmes surviennent lorsque nous exploitons et stockons un énorme volume de données en utilisant le SCC. Pour résoudre ces problèmes, Google a inventé un nouveau modèle de programmation du traitement des données appelé MapReduce pour gérer le traitement des données à grande échelle sur de grands clusters de serveurs de produits de base [72]. Le modèle MapReduce répond à tous ces problèmes. Il résout le premier problème de défaillance des nœuds en stockant les données de manière redondante sur plusieurs nœuds de calcul grâce au système de fichiers distribués. Ainsi, les mêmes données sont stockées sur plusieurs nœuds de calcul, et si l'un de ces nœuds tombe en panne, les données sont toujours disponibles sur un autre nœud [73]. Pour le second problème, le système MapReduce divise les calculs en tâches. De telle sorte que si une tâche ne s'exécute pas jusqu'à la fin, elle peut être relancée sans affecter les autres tâches de calcul. Troisièmement, le système MapReduce résout le problème des goulets d'étranglement du réseau en rapprochant les calculs des données et en évitant de copier les données sur le réseau, ce qui réduit au minimum le problème des goulets d'étranglement [70]. Enfin, le modèle MapReduce fournit également un modèle de programmation très simple qui dissimule la complexité de toutes les tâches de son fonctionnement. Dans la section suivante, nous allons donc discuter en détail de ce modèle.

2.7.2 Modèle de programmation MapReduce

Actuellement, le Big Data Mining est la capacité d'extraire des modèles ou des informations utiles à partir de données à grande échelle. Pour traiter cette énorme quantité de données à l'aide d'un seul nœud de calcul, il est inefficace en temps réel. Pour résoudre ce problème, le cadre de traitement Big Data est déployé sur des ordinateurs en cluster dotés d'une plateforme de calcul à haute performance, et les tâches d'extraction de données sont déployées sur ce cluster d'ordinateurs en exécutant le cadre de haut niveau de données parallèles MapReduce. Ce dernier simplifie la conception et la mise en œuvre de systèmes de traitement des données à grande échelle [69]. Comment améliorer les performances de MapReduce et renforcer la nature en temps réel de la fouille de données à grande échelle est un sujet de recherche brûlant. Le modèle de programmation parallèle de MapReduce a été appliqué dans de nombreux algorithmes d'apprentissage automatique et de fouille de données. Les algorithmes de fouille de données doivent généralement parcourir les données d'apprentissage pour obtenir des statistiques permettant de résoudre ou d'optimiser les paramètres du modèle [74]. MapReduce est un style de calcul parallèle qui a été déployé

dans de nombreux systèmes ; par exemple, les systèmes les plus populaires qui mettent en œuvre ce modèle de programmation parallèle sont Google et le cadre Hadoop de la fondation d'Apache [73]. Il s'agit d'une implémentation associée pour l'extraction et la génération de données à grande échelle d'une manière tolérante aux défaillances du matériel. Le calcul dans ce modèle prend un ensemble de paires clé/valeur d'entrée et produit un ensemble de paires clé/valeur de sortie. L'utilisateur spécifie une fonction de Mapping qui traite un ensemble de paires de clé/valeur d'entrée afin de générer un ensemble de paires de clé/valeur intermédiaires. Enfin, la fonction de Reducing fusionne toutes les valeurs intermédiaires associées à la même clé intermédiaire. Les programmes écrits dans ce style fonctionnel sont automatiquement parallélisés et exécutés sur un grand groupe d'ordinateurs de base [75]. En bref, un calcul MapReduce s'exécute comme suit :

- **Tâche de Map** : est écrite par l'utilisateur, lire un ou plusieurs blocs d'un système de fichiers distribué comme données d'entrée, transformer le bloc de données en un ensemble de paires clés-valeurs intermédiaires et l'écrire sur le disque local.
- **Regroupement par clé** : le système MapReduce effectue cette fonction, dont le but est de regrouper toutes les valeurs intermédiaires associées à une même clé intermédiaire dans une liste de valeurs et de les transmettre à la tâche de Reduce.
- **Tâche de Reduce** : également est écrite par l'utilisateur, après qu'une tâche de Reduce ait reçu toutes les clés intermédiaires avec la liste de valeurs de toutes les tâches de la fonction de Map, elle travaille sur une clé à la fois et fusionne la liste de valeurs associées à cette clé dans un ensemble de valeurs éventuellement plus petit selon la manière de combinaison spécifiée par l'utilisateur pour la fonction de Reduce. Enfin, les tâches de Reduce écrivent une paire clé-valeur de sortie sur un système de fichiers distribué.

Le secret qui rend les tâches de calcul de MapReduce plus puissantes et plus populaires est que le contour reste le même, mais que les fonctions Map et Reduce sont modifiées pour s'adapter à chaque problème. La figure 2.8 illustre clairement les trois tâches de MapReduce expliquées précédemment :

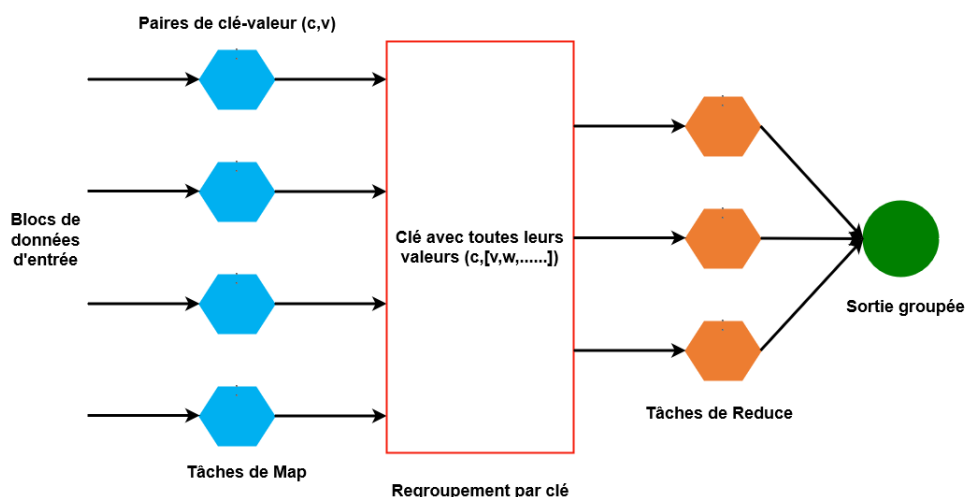


FIGURE 2.8 – Schéma de modèle de programmation MapReduce.

Le calcul dans ce schéma de calcul MapReduce (voir la figure 2.8) a commencé avec la paire clé-valeur à une entrée, et la fonction de Map produit plusieurs paires clé-valeur intermédiaires. Il peut s'agir d'une ou de plusieurs paires clé-valeur intermédiaires de

chaque paire clé-valeur d'entrée. L'étape suivante consiste à prendre ces paires de valeurs clés intermédiaires et à les regrouper par clé. Toutes les paires clé-valeur intermédiaires qui ont la même clé sont regroupées, et cela est fait en triant par clé puis en regroupant les valeurs avec la même clé, une fois qu'il produit ces groupes clé-valeur alors l'étape finale est Reducing, le Reducer regarde un seul groupe clé-valeur comme entrée et produit une sortie qui a la même clé. Une fois que nous avons appliqué le Reducer à tous les groupes clé-valeur intermédiaires nous obtenons la sortie finale stockée dans le système de fichiers distribués.

2.7.3 Système de fichiers distribués

Pour exploiter un cluster de calcul comme une plateforme de calcul a haute performance, le nouveau système de fichiers, appelé Système de Fichiers Distribué (SFD), est un peu différent des systèmes de fichiers classiques que l'on trouve sur les ordinateurs individuels qui ils gèrent les fichiers et les données sur plusieurs dispositifs de stockage [76]. Le SFD fournit un espace de noms de fichiers globaux où il fournit un accès transparent à plusieurs systèmes potentiellement distribués. Par conséquent, à travers l'espace de noms de fichiers globaux, le SFD assure la redondance et la disponibilité des données et résout le problème provoqué par les défaillances des nœuds de manière efficace et fiable. Le monde extérieur ne voit dans le SFD qu'un seul dispositif de stockage, et il n'est qu'une interface dans une plus large mesure [77]. En outre, les SFDs sont parfois appelés "clusters", ou "réseau", ou "nuage", ou "système de fichiers parallèle", ou "grille" et sont généralement utilisés comme suit :

- Les gros fichiers peuvent avoir une taille de plusieurs centaines de téraoctets et il n'est pas efficace d'utiliser le DFS pour les fichiers dont la taille est petite.
- Les fichiers sont rarement mis à jour. Ils sont plutôt lus comme des données pour l'exécution de certaines tâches de calculs, et probablement des données supplémentaires sont ajoutées aux fichiers de temps en temps. Par exemple, imaginons le scénario de google, lorsque google rencontre une nouvelle page web, il l'ajoute à son répertoire. Il ne va jamais mettre à jour le contenu d'une page web qu'il a déjà explorée.

De ce qui précède, on peut déduire que pour résoudre les premiers défis du SCC, une infrastructure de stockage redondant est fournie par le SFD qui stocke les données plusieurs fois sur plusieurs nœuds situés sur des racks différents. Il existe plusieurs implémentations de SFD, mais deux d'entre elles sont les plus populaires, telles que SFG (Système de fichiers de Google) et HDFS (Système de fichiers distribués de Hadoop) [78].

2.7.3.1 Système de fichiers de Google

SFG est le système de fichiers distribués inventé par Google pour rendre l'accès à des données massives plus efficaces et plus fiables en utilisant de grands groupes d'ordinateurs bon marché. SFG a été conçu pour gratifier les efforts de deux développeurs, Larry Page et Sergey Brin afin de satisfaire les exigences de Google en matière de stockage et d'utilisation des données, comme le moteur de recherche qui génère une grande quantité de données qui

doivent être conservées. Un nouveau successeur au système de fichiers de Google, nommé Colossus est sorti en 2010. Dans le SFG, chaque fichier est divisé en plusieurs blocs de données d'une taille fixe de 64 mégaoctets. Comme les racks des systèmes de fichiers ordinaires, qui sont très rarement écrasés ou réduits ; les fichiers sont généralement ajoutés ou lus [75]. SFG est également conçu de manière précise et efficace pour fonctionner sur des clusters de calcul en Google qui se composent de milliers de machines Linux de base, ce qui signifie que des précautions doivent être prises contre le taux élevé d'échec des nœuds individuels et la perte de données qui en découle [79]. Un cluster SFG se compose de plusieurs nœuds : un nœud appelé nœud maître et les autres nœuds sont des serveurs de blocs (BlocServeur) comme le montre la figure 2.9. Les fichiers sont divisés en blocs de taille fixe de 64 mégaoctets [75]. Chaque bloc est identifié par une étiquette unique de 64 bits (poignées de bloc) attribués par le nœud maître au moment de la création des blocs et le mappage du fichier en blocs est maintenu. Les serveurs de blocs stockent ces blocs sur les disques locaux en tant que fichiers Linux et écrivent ou lisent des données de blocs identifiées par une poignée de blocs et une plage d'octets. Chaque bloc est répliqué plusieurs fois sur différents serveurs de blocs pour plus de fiabilité. Par défaut, nous stockons trois répliques, bien que les utilisateurs spécifient différents niveaux de réplication pour différents secteurs de l'espace de noms des fichiers. En outre, le nœud maître conserve toutes les métadonnées du système de fichiers. Cela comprend l'emplacement actuel des blocs, la correspondance entre les fichiers et les blocs, l'espace de noms et les informations de contrôle d'accès. Il supervise également la migration des blocs entre les serveurs de blocs et communique périodiquement avec chaque serveur de blocs afin de recueillir son état et de lui donner ses instructions [79, 80]. Un client de SFG attaché à chaque application met en œuvre l'API du système de fichiers et interagit avec les serveurs de blocs un nœud maître pour écrire ou lire les données de l'application. En outre, le client de SFG communique avec le nœud maître pour élaborer toutes les métadonnées du système de fichiers pour l'application en cours d'exécution.

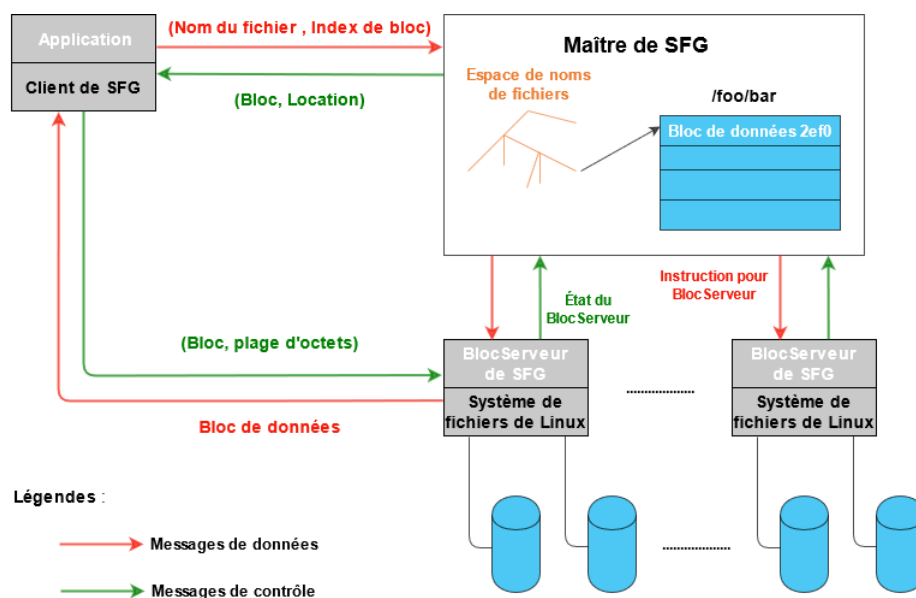


FIGURE 2.9 – Architecture de SFG.

2.7.3.2 Système de fichiers distribués d'Hadoop

Un système Hadoop fonctionne au-dessus d'un système de fichiers distribués appelé HDFS. Il s'agit d'un système de fichiers distribué, portable et évolutif écrit en Java. Jusqu'à présent, il s'agit d'un système de stockage à haute tolérance aux pannes, qui stocke de manière fiable d'énormes quantités de données sur plusieurs machines à faible coût et de manière redondante. Il permet ainsi de sauver le système d'éventuelles pertes de données ultérieures en cas de défaillance [73, 77]. Les données d'entrée de Hadoop sont stockées sous forme de fichiers dans le HDFS. Ainsi, les métadonnées des fichiers sont stockées sur le serveur NameNode et les données d'application sont stockées sur d'autres serveurs appelés DataNodes. Le HDFS utilise une architecture maître-esclave, comme le montre la figure 2.10. Le nœud maître contient un seul NameNode qui stocke les métadonnées de fichiers des blocs dans la mémoire, et les esclaves sont constitués d'un cluster de DataNodes qui stockent les blocs, chaque demande de bloc doit être traitée par les nœuds maîtres et l'envoi à l'esclave qui stocke le bloc associé [18]. La réponse à chaque demande de bloc est plus rapide ; cela est dû au fait que le nœud maître stocke toutes les métadonnées des blocs en mémoire. En plus des options de redondance disponibles pour le NameNode en raison de sa criticité et chacun DataNode utilise un protocole de bloc spécifique au HDFS afin de répliquer également les données sur le réseau. Le HDFS utilise des sockets TCP/IP pour la communication, et les clients utilisent des appels de procédure à distance (RPC) pour communiquer entre eux [81]. Le HDFS stocke des fichiers volumineux (généralement de l'ordre de gigaoctets à téraoctets) sur plusieurs machines. Ces fichiers sont divisés en blocs de données et stockés de manière dispersée dans le système. Comme le HDFS est conçu pour stocker une énorme quantité de données, la taille de chaque bloc de données est généralement de 64 mégaoctets. Les blocs sont répliqués par le HDFS, peut-être trois fois à trois nœuds de calculs différents situés sur des racks différents pour obtenir une tolérance aux fautes de données. La cohérence des répliques n'est pas prise en compte dans le HDFS car les blocs de données sont écrits une fois et lus plusieurs fois. Normalement, tant la taille des blocs de données que le degré de réplification peuvent être décidés par l'utilisateur [81, 82].

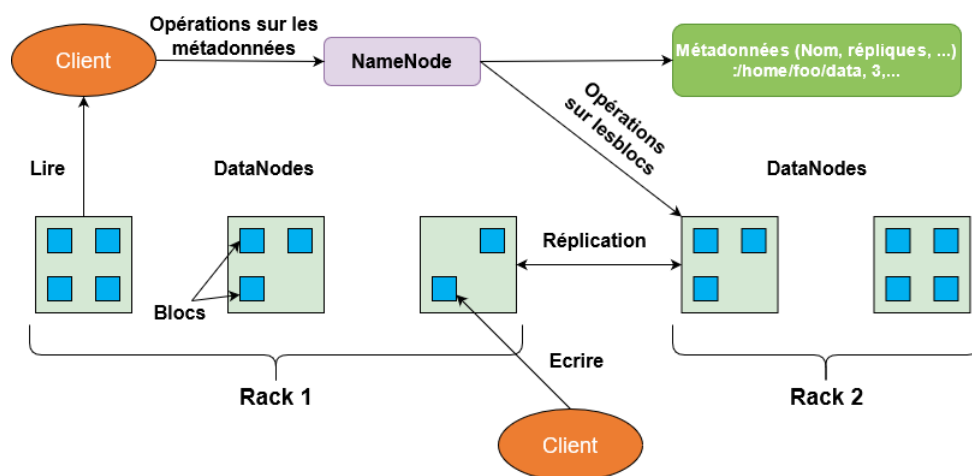


FIGURE 2.10 – Architecture de HDFS.

2.7.4 Solutions de Big Data

Dans cette section, je présenterai les solutions Big Data proposées pour surmonter les défis du SCC à savoir défaillances des nœuds, la complexité des calculs, le goulot d'étranglement du réseau et les problèmes de programmation distribuée.

2.7.4.1 Défaillances des nœuds

Dans un environnement où il y a des milliers de machines, la défaillance est inévitable, comme l'a montré Jeff Dean dans un délai de trois ans (1000 jours), "si vous avez 10000 serveurs, attendez-vous à en perdre dix par jour" [83]. Les défaillances de nœuds constituent le premier et le plus important défi du SCC ; ses principaux modes de défaillance sont la perte d'un rack entier (par exemple, le réseau reliant ses nœuds à un autre rack de nœuds et le monde extérieur tombe en panne) et la perte d'un seul nœud (par exemple, le disque de ce nœud tombe en panne). Par exemple, Google utilise un cluster de calcul pour traiter la très grande quantité de recherche mondiale ; s'il utilise un million de serveurs par cluster, il va connaître mille défaillances par jour. Le cadre MapReduce est conçu pour surmonter ce défi en tolérant les défaillances des nœuds avec grâce. Le nœud maître assure la détection et la gestion des défaillances des nœuds. En d'autres termes, le nœud maître envoie périodiquement un Ping à chaque processus de nœud pour détecter les nœuds défaillants, à un moment prédéfini, si aucune réponse n'est reçue d'un nœud, le nœud maître marque ce nœud comme étant défaillant. Toutes les tâches de Mapping qui ont été assignées à ce nœud défaillant sont réinitialisées à l'état de repos, même si elles ont été achevées et sont maintenant éligibles pour être reprogrammées sur d'autres nœuds dans lesquels sont stockées les mêmes données que le nœud défaillant [74]. La raison de la ré-exécution des tâches de Mapping complétées est que leur résultat est stocké sur le disque local du nœud défaillant et n'est donc pas disponible pour les tâches de Reducing. Dans ce cas, le nœud maître doit envoyer à chaque tâche de Reducing le nouvel emplacement de son entrée. La gestion d'un échec au niveau du nœud Reducer est plus simple que celle du nœud Mapper, les tâches en cours sont seulement remises à zéro et deviennent disponibles pour être reprogrammées plus tard sur un autre travailleur Reducer. La raison pour laquelle les tâches de Reducing terminées ne sont pas replanifiées est que leur résultat est stocké dans un système de fichiers global [82]. MapReduce est résistant aux défaillances de nœuds à grande échelle, mais la pire chose qui puisse arriver, c'est que le nœud sur lequel le maître s'exécute échoue. Dans ce cas, tout le processus MapReduce est interrompu et doit être redémarré [84].

2.7.4.2 Complexité des calculs

Dans le SCC, la plupart des calculs de la fouille de données s'appliquent à la quantité massive de données qui prennent des minutes, des heures, voire des jours sur des milliers de nœuds de calcul [69]. Si nous devons interrompre et relancer ces calculs à chaque fois qu'un nœud de calcul échoue, ces calculs pourraient ne jamais aboutir. La solution proposée par MapReduce pour résoudre ce problème est de diviser les calculs en plusieurs tâches. De telle sorte que si une tâche ne s'exécute pas jusqu'au bout, elle peut être relancée

sans affecter les autres tâches de calcul. Dans ce cadre, le calcul est divisé en trois grandes phases à savoir phase de Mapping, phase de Regroupement et phase de Reducing, comme nous l'avons expliqué plus haut dans la section 2.7.2. La tâche Map exécute une fonction de Map fournie par l'utilisateur et lit un bloc comme données d'entrée, qui contient plusieurs enregistrements de paires clé-valeur [65]. La tâche map combine les enregistrements avec la même clé qu'un tuple et l'écrit dans un fichier intermédiaire sur le disque local. La tâche Map produit plusieurs fichiers intermédiaires, et chacun d'entre eux sera transmise à une tâche de sauvetage ultérieurement [82]. Après la phase Map, la phase de regroupement supportée par le framework, trie les n-uplets dans un fichier intermédiaire par la clé, et transfère le fichier à une tâche Reduce. Lorsqu'une tâche Reduce reçoit tous les fichiers intermédiaires de toutes les tâches Map, elle applique la fonction Reduce fournie par l'utilisateur et traite les tuples dans tous les fichiers intermédiaires et enfin écrits un fichier de sortie sur HDFS [74]. Ce modèle de programmation exécute automatiquement ces fonctions en parallèle sur un nombre quelconque de nœuds de calcul, de sorte que le temps de traitement total soit aussi court que possible.

2.7.4.3 Goulets d'étranglement

Le réseau du SCC peut devenir un goulot d'étranglement. Par conséquent, ce problème du goulot d'étranglement se rapporte à une condition discrète dans laquelle le flux de données est limité par des nœuds de calcul individuels dans un rack ou des racks individuels. Le flux de données est contrôlé en fonction de la largeur de bande des différentes ressources du système [70], par exemple la largeur de bande du réseau disponible entre les différents nœuds d'un rack est d'un gigabit par seconde et celle disponible entre les individus peut varier de 2 à 10 gigabits par seconde [66]. Par exemple, si le système dans les progrès de l'apprentissage sur le SCC fournit dix téraoctets de données, pour déplacer ces données dans le rack, il y a une bande passante d'un gigabit par seconde entre toutes paire de nœuds de calcul, cette capacité du réseau rend les calculs plus complexes, en outre, le temps de calcul devient plus long. Il y aura alors un goulot d'étranglement dans le réseau. Pour comprendre les goulets d'étranglement sur les SCCs, nous devons d'abord apprendre la signification de deux mots : latence et bande passante. La bande passante est la quantité de données qui peut être transmise sur le matériel d'interconnexion dans une période de temps déterminée. La latence est le délai entre l'instant où une unité centrale demande une information et le moment où cette information commence à être disponible. La largeur de bande est mesurée en unités de taille de mémoire/temps (généralement des mégaoctets /seconde), tandis que la latence est mesurée en unités de temps (généralement des secondes à des nanosecondes) [70]. Le cadre de MapReduce résout le problème du goulet d'étranglement du réseau de deux façons : par l'évitement et par la détection. L'évitement des goulets d'étranglement fait partie de la planification des tâches. L'ordonnanceur minimise les goulets d'étranglement du réseau en rapprochant le calcul des données [66], ce qui signifie que l'ordonnanceur assigne les tâches de Mapping soit à des machines SFG où les blocs de données sont stockés (Mappers locaux), soit au moins à des machines SFG partageant le même rack physique avec les données (Mappers locaux de rack). Les opérations de ce planificateur augmentent la bande passante des cartes de 18 Go/s à 100 Go/s [65]. De plus, le MapReduce évite de copier les données sur les réseaux en évitant les interactions entre les parties du système (par exemple, la tâche de Mapping ne

peut pas se parler même pour la tâche de Reducing) et en stockant localement les fichiers de Mapping intermédiaires générés par la fonction Map. L'inconvénient est qu'il n'y a qu'une seule copie des données sur cette machine et que l'esclave chargé de Reducing ne doit parler qu'à celle-ci, ce qui réduit le problème des goulets d'étranglement du réseau. Une autre opération de MapReduce visant à éviter le goulot d'étranglement et à accélérer le temps d'exécution du travail global consiste à réduire la quantité de données à envoyer sur le réseau en appliquant une fonction de combinaison à la sortie d'une fonction Map. La deuxième façon est la détection des goulets d'étranglement, qui se fait dans le cadre du suivi de la tâche en cours. MapReduce détecte la largeur de bande du réseau en tirant parti du fait que le SFG divise chaque fichier d'entrée en blocs de 64 Mo et stocke plusieurs copies de chaque bloc (généralement 3 copies) sur différentes machines. Lorsqu'elle trouve une tâche excessivement longue, l'infrastructure lance de manière spéculative une tâche en double qui fait le même travail sur le bloc en double et utilise la sortie de la tâche qui se termine en premier [66]

2.7.4.4 Programmation distribuée

Le dernier défi des SCCs est que la programmation distribuée peut être très difficile ; même les programmeurs sophistiqués ont du mal à écrire correctement des programmes distribués. Nous avons donc besoin d'un modèle simple qui cache la majeure partie de la complexité de la programmation distribuée et qui facilite l'écriture de programmes distribués, quel que soit le type d'application fonctionnant sur le SCC [67]- [73]. Le paradigme de MapReduce offre à un programmeur ce modèle simple. Il est facile à utiliser car il offre au programmeur une API simple pour le calcul, le partitionnement des données d'entrée ou la gestion des données, le traitement des pannes de machine, la gestion de la communication inter-machine requise et la programmation de l'exécution du programme sur un ensemble de machines. Tous les programmes écrits dans le style fonctionnel de MapReduce sont automatiquement parallélisés et exécutés sur un grand nombre de machines. Par conséquent, le but principal de ce cadre est de permettre aux programmeurs n'ayant aucune expérience des systèmes distribués et parallèles de l'utiliser sans se soucier de la complexité sous-jacente [71]- [74].

2.8 Conclusion

Les données devenant de plus en plus volumineuse et complexe, nos bases de données traditionnelles sont limitées face à l'analyse et au traitement de ces données. Dans un souci de gain de temps, de nouvelles technologies sont venues pour soulager les entreprises génératrices d'un grand nombre de données. L'analyse de Big Data est sans aucun doute vouée à gagner une importance, certains parlent même de révolution.

A ce stade on peut dire que le Big Data est un écosystème large et complexe. Il nécessite la maîtrise des technologies matérielles et logicielles diverses (stockage, parallélisation des traitements, virtualisation, ...). Le Big Data demande de la compétence et de l'expertise dans la maîtrise et l'analyse des données. Les usages du Big Data sont très vastes qui touchent presque tous les secteurs d'activités (marketing, recherche, visualisation, ...).

Ce chapitre a été consacré à la présentation des notions fondamentales du Big Data, en particulier nous avons présenté un aperçu historique sur le Big Data pour comprendre comment le Big Data est apparu. Ensuite, nous avons décrit tous les différentes définitions du Big Data à savoir 3Vs, 4Vs, 5Vs, et 6Vs. Puis, nous avons discuté les différentes sources du Big Data en détail. En outre, nous avons présenté l'évolutivité du Big data et l'architecture des modules d'Hadoop. Aussi ce chapitre a décrit le fonctionnement du système de calcul en clusters et présente ses défis en cette ère de Big Data à savoir défaillances des nœuds, la complexité des calculs, le goulot d'étranglement du réseau et les problèmes de programmation distribuée. De plus, nous avons présenté les solutions Big Data les plus populaires proposées pour relever les défis du système de calcul en clusters. Nous avons montré également comment les technologies "Big Data" résolvent les problèmes liés au système de calcul en clusters. D'une manière générale, l'objectif principal de ce chapitre est de mieux comprendre les technologies "Big Data" et les défis du système de calcul en clusters et d'identifier les solutions essentielles en matière de Big Data dans ce domaine du système de calcul en clusters.

Dans le prochain chapitre, nous allons aborder et détailler un domaine qui commence à révolutionner le monde informatique qui est la fouille de données.

Notions Fondamentales de Fouille de Données

3.1 Introduction

La fouille de données (Data Mining) est récemment devenue l'un des domaines les plus progressifs et les plus prometteurs pour l'extraction et la manipulation de données afin de produire des informations utiles. Des milliers d'entreprises utilisent chaque jour des applications de la fouille de données afin de manipuler, d'identifier et d'extraire des informations utiles à partir des enregistrements stockés dans leurs bases de données, leurs dépôts de données et leurs entrepôts de données [85].

Grâce à ces informations extraites à l'aide des outils de la fouille de données, les entreprises ont pu améliorer leurs activités en appliquant les modèles, les relations et les tendances qui se sont cachées ou qui n'ont pas été découverts dans des quantités colossales de données. Par exemple, la fouille de données produit des informations qui permettent aux entreprises de créer des profils de clients actuels et potentiels afin de les aider à gagner et à conserver leurs clients. Parmi les autres utilisations de la fouille de données, citons le développement de stratégies de vente croisée et de marketing, la mise au jour de crimes ou de fraudes possibles, et la découverte de modèles d'accès des utilisateurs à leurs sites web [86].

Fouille de données est définie comme un ensemble de règles, de processus et d'algorithmes conçus pour générer des informations exploitables, extraire des modèles et identifier des relations à partir de grands ensembles de données [87]. La fouille de données comprend l'extraction, le traitement et la modélisation automatisés des données au moyen d'une série de méthodes et de techniques. En revanche, l'analyse des données fait référence aux techniques utilisées pour analyser et acquérir des renseignements à partir des données et se considère comme un domaine plus large [88] englobant un éventail plus large de méthodes qui comprennent à la fois les statistiques et la fouille de données.

Un certain nombre d'algorithmes ont été développés dans les domaines des statistiques, de l'apprentissage automatique et de l'intelligence artificielle pour soutenir et permettre l'extraction des données utiles à partir des jeux de données massives. Bien que les approches statistiques les précèdent, elles comportent intrinsèquement des limites, les plus connues étant des conditions rigides de distribution des données. Les techniques d'apprentissage automatique ont gagné en popularité car elles imposent moins de restrictions tout en dérivant des modèles compréhensibles à partir des données [87, 88]

Les projets de la fouille de données suivent généralement un processus ou une méthodologie structurée, comme Mariscal illustré dans le document [89]. Une méthodologie de la fouille de données spécifie les tâches, les entrées, les sorties, et fournit des directives et des instructions sur la manière dont les tâches doivent être exécutées. Ainsi, la méthodologie de la fouille de données fournit un ensemble de lignes directrices pour l'exécution d'un ensemble de tâches afin d'atteindre les objectifs d'un projet de la fouille de données [89,90].

Les bases des méthodologies structurées de la fouille de données ont été proposées pour la première fois par Fayyad et son équipe dans le document [91] et étaient initialement liés à la découverte de connaissances à partir des bases de données (KDD). KDD (Knowledge Discovery in Database) présente un modèle de processus conceptuel des théories et des outils de calcul qui soutiennent l'extraction d'informations utiles (connaissances) à partir du jeu de données [92].

Dans le cadre du KDD, l'approche globale de la découverte de connaissances comprend la fouille de données comme étape spécifique. En tant que telle, la KDD avec ses neuf étapes principales (présentées dans la figure 3.1) a l'avantage de prendre en compte le stockage, l'accès aux données, la mise à l'échelle des algorithmes, l'interprétation, la visualisation des résultats, et l'interaction homme-machine [93].

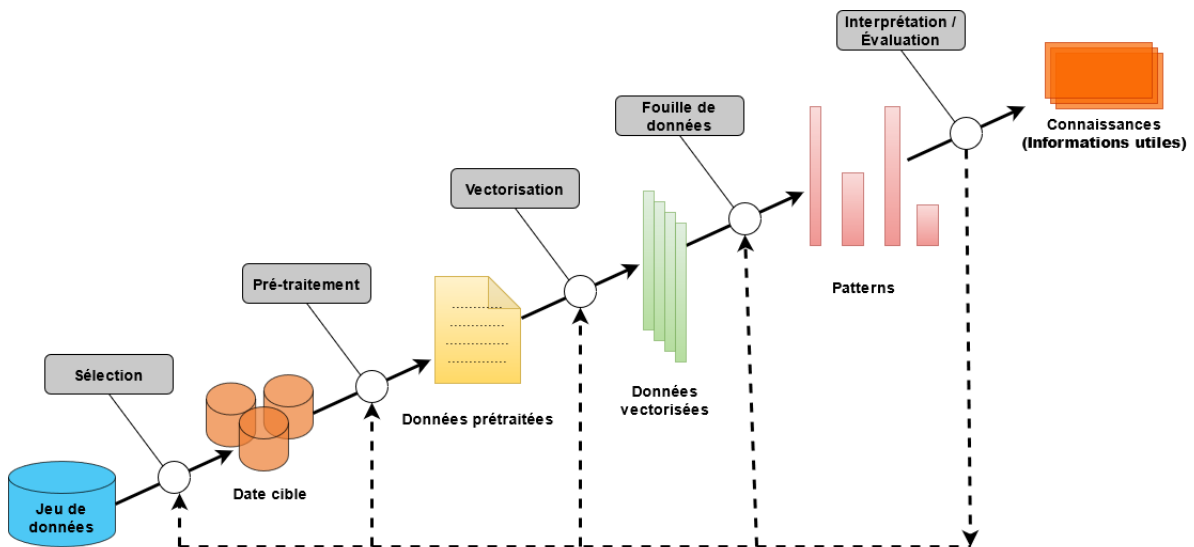


FIGURE 3.1 – Un aperçu des étapes composant le processus KDD.

Le reste de ce chapitre est construit comme suit : les types de fouille de données sont décrits dans la section 3.2. La section 3.3 présente les méthodes supervisées. Dans la section 3.4, la classification est décrite en détail. Les techniques de prétraitement de données sont discutés dans la section 3.5. La section 3.6 présente le processus d'extraction de caractéristiques. Dans la section 3.7, le processus de sélection de caractéristiques est discuté. La section 2.8 récapitule les différents algorithmes de classification. Dans la section 3.9, les mesures d'évaluation pour la classification sont décrites. Et enfin, la section 3.10 récapitule les différentes notions de fouille de données.

3.2 Types de fouille de données

Il est utile de distinguer deux grands types de la fouille de données : orientée vérification (le système vérifie l'hypothèse de l'utilisateur) et orientée découverte (le système trouve de nouvelles règles et de nouveaux modèles de manière autonome) [91]. La figure 3.2 illustre cette taxonomie avec chaque type a sa propre méthodologie.

Les méthodes de découverte qui identifient automatiquement des modèles à partir des données impliquent à la fois des méthodes de description et de prédiction. Les méthodes de description s'attachent à comprendre le fonctionnement des données sous-jacentes, tandis que les méthodes de prédiction visent à construire un modèle comportemental pour obtenir des échantillons nouveaux et invisibles et pour prédire les valeurs d'une ou plusieurs variables liées à l'échantillon. Cependant, certaines méthodes axées sur les prédictions peuvent également contribuer à la compréhension des données.

La plupart des techniques axées sur la découverte sont basées sur l'apprentissage inductif [94], où un modèle est construit explicitement ou implicitement en généralisant à partir d'un nombre suffisant d'exemples d'apprentissage. L'hypothèse sous-jacente de l'approche inductive signifie que le modèle formé est applicable à des exemples futurs invisibles. À proprement parler, toute forme d'inférence dans laquelle les conclusions ne sont pas déductivement sous-entendues par les prémisses peut-être considérées comme une induction.

Les méthodes de vérification, en revanche, évaluent une hypothèse proposée par une source externe (comme un expert, etc.). Ces méthodes comprennent les méthodes les plus courantes des statistiques traditionnelles, comme le test d'adéquation, le test t des moyennes et l'analyse de la variance. Ces méthodes sont moins associées à la fouille de données que leurs homologues axées sur la découverte car la plupart des problèmes d'exploration de données concernent la sélection d'une hypothèse (parmi un ensemble d'hypothèses) plutôt que le test d'une hypothèse connue. Les méthodes statistiques traditionnelles sont généralement axées sur l'estimation de modèles, par opposition à l'un des principaux objectifs de la fouille de données : l'identification de modèles [95].

Dans notre thèse, nous nous sommes concentré sur les méthodes de découverte et surtout, nous avons travaillé sur des méthodes de prédiction de type classification, comme le montre la figure 3.2.

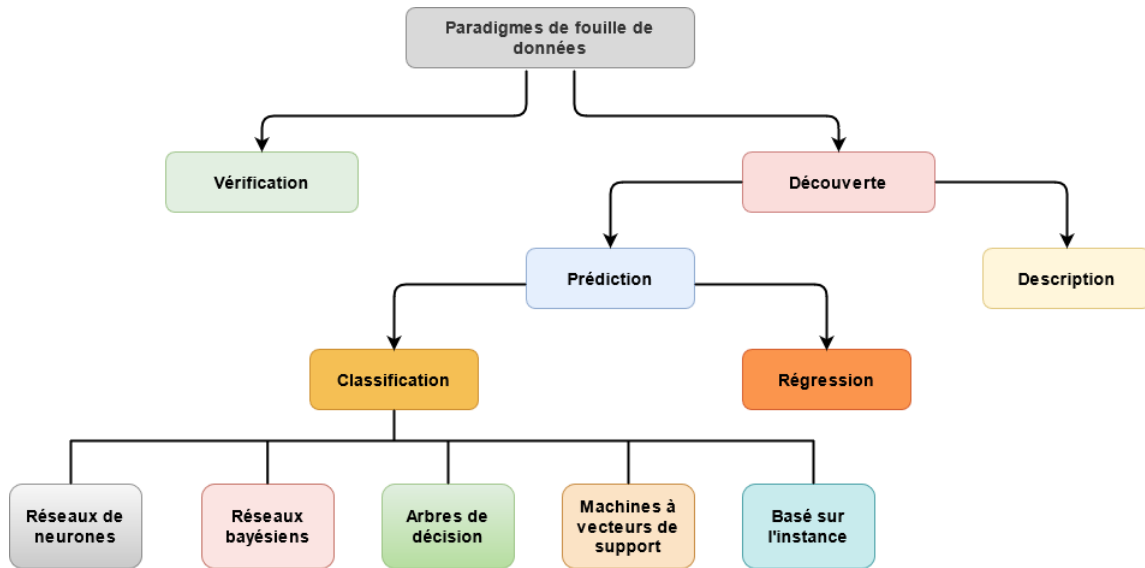


FIGURE 3.2 – Taxonomie des méthodes de fouille de données.

3.3 Méthodes supervisées

Dans la communauté de l'apprentissage automatique, les méthodes de prédiction sont communément appelées méthodes d'apprentissage supervisé. L'apprentissage supervisé s'oppose à l'apprentissage non supervisé, qui consiste à modéliser la distribution des instances dans un espace d'entrée typique de grandes dimensions. Selon [96], le terme "apprentissage non supervisé" fait référence à des "techniques d'apprentissage qui regroupent des instances sans attribut dépendant prédéfini". Ainsi, le terme "apprentissage non supervisé" ne couvre qu'une partie des méthodes de description présentées dans la figure 3.2. Par exemple, le terme couvre les méthodes de regroupement mais pas les méthodes de visualisation.

Les méthodes supervisées sont des méthodes qui tentent de découvrir la relation entre les attributs d'entrée (parfois appelés variables indépendants) et un attribut cible (parfois appelé variable dépendant). La relation découverte est représentée dans une structure appelée modèle. Habituellement, les modèles décrivent et expliquent des phénomènes, qui sont cachés dans le jeu de données, et qui peuvent être utilisés pour prédire la valeur de l'attribut cible lorsque les valeurs des attributs d'entrée sont connues. Les méthodes supervisées peuvent être mises en œuvre dans divers domaines tels que le marketing, la finance et la fabrication.

Il est utile de distinguer deux principaux modèles supervisés : les modèles de classification et les modèles de régression. Les modèles de régression font correspondre l'espace d'entrée à un domaine de valeur réelle. Par exemple, un régresseur peut prévoir la demande d'un certain produit compte tenu de ses caractéristiques. D'un autre côté, les classificateurs font correspondre l'espace d'entrée à des classes prédéfinies. Par exemple, les classificateurs peuvent être utilisés pour classer les consommateurs de prêts hypothécaires en bons (remboursement complet du prêt hypothécaire dans les délais) et mauvais (remboursement différé). Parmi les nombreuses possibilités de représentation des classifi-

cateurs, on trouve, par exemple, les machines à vecteurs de support, les arbres de décision, les résumés probabilistes, les fonctions algébriques, etc.

Cette thèse porte principalement sur les problèmes de classification. La classification est l'une des approches les plus étudiées, peut-être celle qui présente le plus d'intérêt pratique. Les avantages potentiels des progrès en matière de classification sont immenses puisque la technique a un grand impact sur d'autres domaines, tant dans la fouille des données que dans ses applications

3.4 Classification

La classification est la tâche qui consiste à attribuer des observations à l'une de plusieurs catégories prédéfinies également appelées classes. Il s'agit d'une tâche d'apprentissage supervisée consistant à apprendre une fonction à valeur discrète qui associe une observation à l'étiquette de classe correcte. Une observation est constituée d'un vecteur d'attributs F , appelés caractéristiques.

L'ensemble des observations ou des exemples avec pour chaque entrée le label de classe correcte est appelé *l'ensemble d'apprentissage*. Plus formellement, il contient des paires $\langle x, c \rangle \in \mathcal{D} * \mathcal{C}$, où \mathcal{D} est le domaine des observations et $\mathcal{C} = \{c_1, c_2, \dots, c_{|\mathcal{C}|}\}$ est l'ensemble des classes ou catégories prédéfinies. Lorsqu'une observation $x \in \mathcal{D}$ est étiquetée à une classe $c_i \in \mathcal{C}$ cette observation est appelée un exemple positif de c_i sinon elle est appelée un exemple négatif de c_i . Le classificateur est évalué sur un *ensemble de test* qui est disjoint de l'ensemble d'apprentissage. C'est important car si vous utilisez les mêmes données pour l'apprentissage et le test, l'évaluation des performances peut être trop précise. Aussi, le classificateur (également appelé hypothèse ou modèle) peut également être utilisé pour prédire des cas des exemples inconnus.

Définition de la classification

La classification est la tâche d'approximation de la fonction cible inconnue $\Phi : \mathcal{D} \mapsto \mathcal{C}$ par le biais d'une fonction $\phi : \mathcal{D} \mapsto \mathcal{C}$ appelé le classificateur, qui attribue à chaque observation $x \in \mathcal{D}$ l'un des labels de classe prédéfinis $c \in \mathcal{C}$

La méthode d'apprentissage supervisé peut maintenant être formellement définie comme une fonction $\mathcal{D} * \mathcal{C} \mapsto \Phi$ quel que soit le jeu de données d'apprentissage utilisées, et produit comme sortie le classificateur Φ . L'ensemble du processus d'apprentissage supervisé est présenté dans la figure 3.3. Tout d'abord, les données sont divisées en deux ensembles disjoints : le jeu de données d'apprentissage et le jeu de données de test. Ensuite, les méthodes d'extraction et de sélection des caractéristiques sont appliquées. La flèche allant de l'extraction et de la sélection des caractéristiques dans la phase d'apprentissage à la phase de test indique que certaines informations sont passées de la phase d'apprentissage à la phase de test. Par exemple, les caractéristiques sélectionnées pendant la phase d'apprentissage doivent être les mêmes que pendant la phase de test. Il existe deux types de classification : la classification binaire et la classification multiclassées. La classification binaire et multiclassées visent toutes les deux à placer les observations dans une classe correcte. La classification binaire comprend deux classes, tandis que la classification multiclassées com-

prend deux classes ou plus. Certains algorithmes sont utilisés pour la classification binaire tandis que d'autres sont utilisés pour la classification multiclassées. Il est même possible d'utiliser plusieurs algorithmes de classification binaire pour effectuer une classification multiclassées.

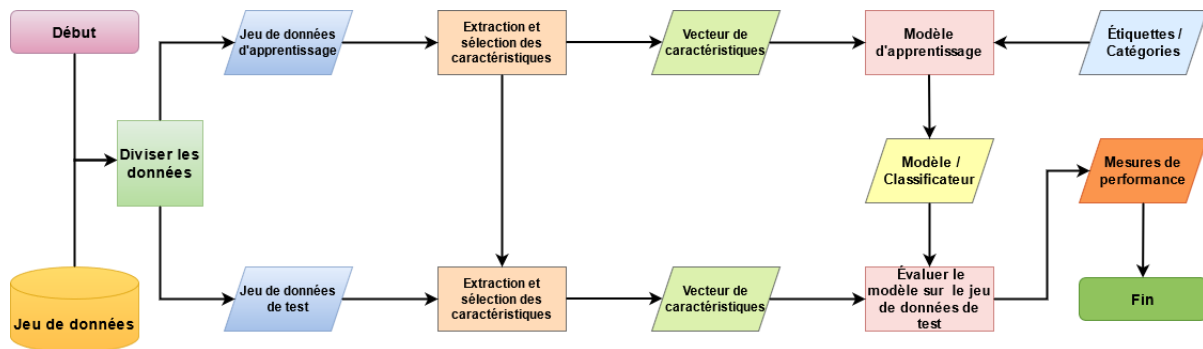


FIGURE 3.3 – Le processus d'apprentissage supervisé.

3.4.1 Classification binaire

Classification binaire est le type de modèle prédictif le plus courant. Les principaux décideurs des entreprises et autres organisations doivent souvent prendre des décisions cruciales rapidement, ce qui exige un système permettant d'arriver à une décision oui/non en toute confiance. Ces systèmes ne sont pas conçus pour faire les choses à moitié ou avec une certaine probabilité. Soit ils les font, soit ils ne les font pas.

Avec la modélisation prédictive d'une cible binaire, la probabilité qu'un événement se produise ou non est également une information très utile. Par exemple, avec une cible binaire, nous créons une situation en noir et blanc ; pleuvra-t-il demain - oui ou non ? Il pleuvra ou ne pleuvra pas, mais il est très peu probable que votre météorologue local (ou même le service météorologique national) vous donne une chance de voir de la pluie à 0 ou 100 % car, s'il pleut (100 %) ou ne pleut pas (0 %), l'estimation est implicitement le degré de confiance que nous avons dans notre prévision. Cette estimation de confiance est souvent beaucoup plus utile que la prédiction binaire elle-même. Votre comportement est-il différent si vous voyez 10 % de chances de pluie contre 40 % de chances de pluie ? Êtes-vous prêt à laisser vos fenêtres baissées ou le toit de votre cabriolet ? pour ces deux prévisions, puisque le pourcentage est inférieur à 50, indiquent qu'il est moins probable qu'il pleuve que non.

L'objectif de la classification binaire est d'optimiser une fonction $f(x)$ afin de minimiser la probabilité d'erreur de classification $P(y=f(x) < 0)$. Où y est l'étiquette de classe avec $+1$ pour le positif et -1 pour le négatif. Il existe de nombreuses méthodes de classification binaire influentes, telles que les méthodes du noyau [97] exemple des machines à vecteurs de support [98], méthodes d'ensemble [99], les méthodes d'ensemble comprennent la dynamisation [100], comme la forêt aléatoire [101] et les méthodes d'apprentissage approfondi sont basées sur des réseaux de neurones artificiels [102].

3.4.2 Classification multiclassées

Classification multiclassées ou nominale est très similaire à la classification binaire, à l'exception du fait qu'il y a maintenant plus de deux classes. La classification nominale est une extension de la classification binaire. Il existe plusieurs exemples où la classification nominale est courante, mais pour la plupart, c'est la plus rare des cibles. On peut voir un exemple d'un tel modèle dans l'industrie de la téléphonie mobile lorsqu'on examine le taux de désaffection des clients. Une entreprise peut ne pas être uniquement intéressée par la réponse binaire à la question de savoir si un compte reste actif ou non. Elle peut plutôt vouloir plonger plus profondément et examiner une réponse nominale de désabonnement volontaire (le client choisit d'annuler le contrat), de désabonnement involontaire (par exemple, le contrat a été résilié en raison de paiements en retard) ou d'un client actif.

Dans de nombreux cas, un problème de classification nominale se pose lorsqu'un cas d'exception est ajouté à ce qui pourrait être une décision binaire. C'est le cas, par exemple, de la prévention de la fraude à la carte de crédit. Lorsqu'une transaction par carte de crédit est initiée, l'émetteur de la carte dispose d'une fenêtre très courte pour accepter ou refuser la transaction. Bien que cela puisse être considéré comme un simple problème binaire, accepter ou refuser, il y a certaines transactions pour lesquelles la décision n'est pas aussi simple et peut tomber dans une zone grise. Un troisième niveau de réponse peut être ajouté pour indiquer que cette transaction doit faire l'objet d'un examen plus approfondi avant qu'une décision d'acceptation ou de refus puisse être prise.

Le problème de la classification nominale pose quelques complications supplémentaires du point de vue du calcul. Au lieu de calculer simplement une probabilité d'événement (P) et de prendre ensuite $1 - P$ pour arriver à la probabilité de non-événement, vous devez calculer la probabilité de l'événement $\#1(P_1)$, la probabilité de l'événement $\#2(P_2)$, et ainsi de suite jusqu'au dernier niveau qui peut être calculé en utilisant $1 - \sum_{i=1}^{n-1} P$ [103]. Il est également difficile de calculer le taux d'erreur de classification. Comme il existe de nombreux choix, les valeurs du rapport doivent être calibrées pour être facilement interprétées par le lecteur du rapport [104].

3.5 Prétraitement de données

Les tâches de prétraitement sont considérées comme la première phase des tâches de classification des textes, et le choix de techniques de prétraitement appropriées et efficaces peut améliorer les performances de classification. L'objectif principal de la procédure de prétraitement du texte est de préparer, normaliser, supprimer et nettoyer les données bruyantes de l'ensemble de données utilisé qui va être classifié. Les données bruyantes dans notre problématique sont celles qui ne contiennent aucune information utile pour la classification des sentiments. La technique de prétraitement transforme les données bruyantes des caractéristiques de haute dimension en caractéristiques de basse dimension afin d'obtenir autant de données utiles précises que possibles à partir de l'ensemble de données considéré. La phase de prétraitement du texte peut faire intervenir plusieurs techniques en fonction du problème de classification du texte et de la situation. Dans cette thèse, notre problématique de classification est la classification des sentiments des don-

nées collectées à partir de Twitter. Twitter permet à ses utilisateurs de ne diffuser que des messages de 140 caractères. En raison de cette règle restrictive, les utilisateurs de Twitter ont utilisé l'argot, les abréviations, les points d'exclamation, les liens, les répétitions, les signes de ponctuation pour exprimer leurs attitudes et leurs émotions dans un court tweet. En outre, les utilisateurs de Twitter sont vulnérables aux fautes d'orthographe et de typographie. Il n'est pas essentiel d'inclure toutes les expressions du tweet dans le processus d'apprentissage dans notre travail, et plusieurs d'entre elles devraient être supprimées, normalisées, nettoyées ou remplacées par d'autres. Il est donc nécessaire d'appliquer des techniques de prétraitement aux jeux de données utilisés. Leur absence de bruit est un facteur crucial pour augmenter l'efficacité de la classification des sentiments. La figure 3.4 illustre la vision la plus courante des techniques de prétraitement des données.

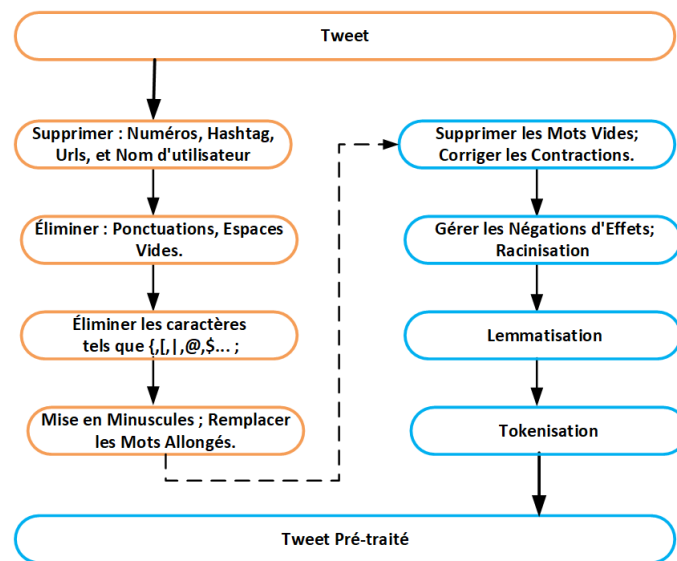


FIGURE 3.4 – Étapes de base de l'étape de prétraitement des données.

Comme illustré dans la figure 3.4, il y a de nombreuses étapes à effectuer pour ré-évaluer les données de manière exacte et extraire le véritable sens de ces données via le traitement des données. Voici donc les étapes de prétraitement suivies dans cette thèse :

Supprimer numéros, hashtag, urls, et nom d'utilisateurs : c'est une tactique populaire d'éliminer les chiffres, les URL, les hashtags et les noms d'utilisateurs dans la phrase de prétraitement, car ils ne contiennent aucune émotion.

Éliminer ponctuations, espaces vides et les caractères spéciaux : la première étape consiste à supprimer tous les espaces vides existant dans le tweet, puis à supprimer les trois signes de ponctuation, à savoir les points d'arrêt, d'interrogation et d'exclamation. Tous les caractères spéciaux trouvés sont supprimés car ils n'ont pas d'impact positif ou négatif sur le sentiment exprimé. Après toutes ces techniques de prétraitement présentées précédemment, nous n'avons gardé que les lettres minuscules et majuscules.

Mise en minuscules : à partir des étapes décrites précédemment, tous les caractères spéciaux autres que les lettres ont été supprimés. L'étape suivante est donc la mise

en minuscules. En d'autres termes, toutes les lettres conservées dans le tweet ont été transformées en minuscules, ce qui réduit la dimensionnalité des mots.

Remplacer les mots allongés : cette opération sert à supprimer la lettre qui se répète au moins plus de trois fois dans le mot allongé comme le mot "heuuuuuuuuureuse". Après avoir appliqué cette opération, le mot devient "heureuse" et normalisé avec au maximum deux lettres "u".

Supprimer les mots vides : les mots vides sont les mots dont l'occurrence est élevée dans le tweet publié. Ils sont supprimés car ils ne contiennent aucune émotion et il est jugé inutile de les traiter. Par conséquent, dans notre travail, tous les mots vides trouvés dans le tweet sont supprimés sur la base de la liste des mots vides déterminés par le paquet *NLTK* en *Python*.

Corriger les contractions : une tactique qui peut-être employée dans la procédure de prétraitement est la correction des contractions. Par exemple, pour des mots comme "isn't", le mot corrigé sera "is not", et pour "weren't", le mot corrigé sera "were not".

Gérer les effets de négation : cette approche remplace le mot précédé de NOT par son antonyme. L'antonyme signifie le sens opposé du mot remplacé. Le processus de cette approche sert à rechercher dans chaque tweet le mot précédé de NOT, puis à vérifier si ce mot a un antonyme dans le dictionnaire WordNet ; si c'est le cas, il remplace ce mot par son antonyme non ambigu. Par exemple, elle remplace le mot "not uglify" par "beautify".

Racinisation : est l'opération qui consiste à réduire la taille des mots en fusionnant plusieurs mots en un seul. Cette approche supprime les terminaisons des mots pour découvrir leur racine à partir d'un dictionnaire. Dans cette thèse, nous avons utilisé le *Stemmer Porter* du paquet *NLTK* en *Python*.

Lemmatisation : a le même rôle que la racinisation. C'est également une autre approche qui sert à déterminer les formes de la racine des mots. La différence entre les deux procédures réside dans le processus suivi pour détecter la racine ou le lemme des mots.

Tokenisation : est un processus qui divise les phrases en mots appelés jetons. Dans ce processus, les plus grands paragraphes de données d'analyse peuvent être divisés en phrases. Ensuite, ces phrases obtenues peuvent également être divisées en jetons. Dans ce travail, nous avons utilisé un tokeniseur *NLTK* fourni par *Python*.

Toutes les techniques présentées précédemment sont appliquées aux jeux de données utilisés dans cette thèse. En outre, nous construisons une table de correspondance avec 3000 mots et phrases contenant les mots, les abréviations et les mots d'argot pour remplacer les abréviations et les mots d'argot dans le tweet en cours de traitement par des mots corrects. Par exemple, on trouve ces mots d'argot et d'abréviation "ab/abt", "B" et "B4", qui désignent et remplacent respectivement par "about", "be" et "before". Dans la plateforme Twitter, les utilisateurs sont susceptibles de faire des fautes d'orthographe et de typographie qui peuvent rendre la procédure d'apprentissage plus difficile. En conséquence, pour améliorer l'efficacité du processus d'apprentissage, nous avons utilisé le correcteur orthographique et typographique de *Norvig*, qui les corrige automatiquement.

3.6 Extraction de caractéristiques

La plupart des classificateurs d'apprentissage automatique ne peuvent traiter que des données numériques ; le type de données que nous avons dans cette thèse sont des données textuelles. Et pour traiter ces données textuelles obtenues après avoir appliqué l'étape de prétraitement des données en utilisant des méthodes d'apprentissage automatique, ces données textuelles doivent être transformées en valeurs numériques. Ce processus est appelé vectorisation du texte ou phase d'extraction des caractéristiques. Il s'agit de l'une des problématiques fondamentales du TAL permettant les techniques d'apprentissage automatique d'examiner les données textuelles. Comme nous l'avons dit précédemment, l'extraction de caractéristiques est l'opération qui consiste à transformer les données d'entrée textuelles en un ensemble de caractéristiques numériques. L'efficacité de chaque classificateur d'apprentissage automatique dépend fortement des caractéristiques extraites. Par conséquent, il est essentiel de sélectionner les meilleures caractéristiques qui ont un impact positif sur le taux de la classification. Il existe plusieurs extracteurs de caractéristiques diverses pour représenter les données textuelles en données numériques, notamment *Sac de Mots*, *N-gramme*, *GloVe*, *Word2vec*, *FastText*, et *TF-IDF*. Ces différents extracteurs de caractéristiques conduiront à des résultats d'analyse différents et influenceront différemment les performances de classification. Les sous-sections suivantes présentent en détail les différentes méthodes de vectorisation mentionnées précédemment.

3.6.1 N-gramme

Dans les domaines de la probabilité, de la linguistique computationnelle et de la statistique, un N-gramme est une séquence adjacente d'éléments, puisque chaque élément contient N caractères obtenus en divisant le texte ou le mot analysé à l'aide de l'algorithme du N-gramme. Par exemple, le mot "WORD" aura les N-grammes suivants :

- Bigrammes (N = 2) : _W, WO, OR, RD, D_
- Trigrammes (N = 3) : _WO, WOR, ORD, RD_, D__
- Quadrigrammes (N = 4) : _WOR, WORD, ORD_, RD___, D_____

Le N-gramme est un extracteur de caractéristique manuel qui joue un rôle important en tant qu'extracteur distinctif de caractéristiques dans la classification de textes, l'extraction d'opinions, la détection de la cyber intimidation, le filtrage du spam, l'attribution et la vérification de l'identité de l'auteur, le vérificateur de plagiat et divers autres domaines d'application. La méthode du N-gramme aide à former un vecteur de représentation de mots utiles pour les termes inconnus, ce qui améliore la précision de la classification dans les tâches basées sur des données textuelles, où une part importante de mots inconnus apparaît dans le domaine de l'exploration d'opinion. Le principal avantage de la méthode du N-gramme est l'indépendance linguistique, c'est-à-dire que le potentiel de portage d'une méthode de représentation des données et d'un algorithme d'apprentissage automatique d'une langue à une autre n'est pas pris en considération. Le problème de la méthode N-gramme est qu'elle génère un nombre important de caractéristiques qui prendront plus de temps pour entraîner un algorithme supervisé sur celles-ci, mais la mise en œuvre du cadre Hadoop peut éviter ce problème.

3.6.2 Sac de Mots

L'extracteur de caractéristiques Sac de Mots est l'une des approches les plus couramment utilisées dans les tâches de représentation de données, que l'on appelle extracteur de caractéristiques. Elle est considérée comme une approche précieuse et simple de la représentation des données, qui permet d'associer un ensemble de phrases à classer en un vecteur numérique sous la forme suivante $S_v[x_1; x_2; \dots; x_n]$, où x_j décrit l'occurrence du j ème élément du vocabulaire collecté à partir du jeu de données fourni, c'est-à-dire qu'elle ne prend en compte que les doublons de mots, sans tenir compte de la morphologie et de la position des mots. Cette approche fait l'objet de recherches très poussées et possède une excellente capacité à sélectionner et à classer les caractéristiques en construisant des sacs pour chaque type d'exemple. Pour représenter les données, il est utilisé dans de nombreux domaines d'application tels que le traitement du langage naturel, La récupération d'informations, la classification de documents, la classification de phrases, la vision par ordinateur et l'exploration d'opinions. Par exemple, nous avons les trois commentaires sur un livre décrit ci-dessous :

- Commentaire A : Ce livre est très long et ennuyeux
- Commentaire B : Ce livre n'est pas ennuyeux et est plus court.
- Commentaire C : Ce livre est bon et agréable

Le vocabulaire de ces trois commentaires de livre se compose de treize mots qui sont : "Ce", "livre", "est", "très", "ennuyeux", "et", "long", "n'est", "pas", "plus", "court", "bon", et "agréable". Par conséquent, le vecteur numérique de chaque commentaire est créé par la méthode du Sac de Mots comme suit :

- **Vecteur de commentaire A** : [Ce :1, livre :1, est :1, très :1, ennuyeux :1, et :1, long :1, n'est :0, pas :0, plus :0 court :0, bon :0, agréable :0]
- **Vecteur de commentaire B** : [Ce :1, livre :1, est :1, très :0, ennuyeux :1, et :1, long :0, n'est :1, pas :1, plus :1 court :1, bon :0, agréable :0]
- **Vecteur de commentaire C** : [Ce :1, livre :1, est :1, très :0, ennuyeux :0, et :1, long :0, n'est :0, pas :0, plus :0 court :0, bon :1, agréable :1]

3.6.3 TF-IDF

Dans cette thèse, le TF-IDF est l'une des techniques utilisées pour vectoriser les données textuelles des tweets. Chaque tweet est traité comme un document. TF-IDF est utilisé pour la représentation des données car il prend en compte la fréquence d'un mot dans l'ensemble de la liste des documents. Dans chaque phrase (document), chaque terme est pondéré en fonction de sa présence dans la phrase, c'est-à-dire que chaque mot se voit attribuer un poids en fonction de sa présence dans cette phrase. Par conséquent, si un mot apparaît dans de nombreuses phrases, le poids attribué à ce mot est diminué, car il n'est pas utile pour discerner les phrases. TF-IDF forme une matrice dans laquelle les lignes représentent les phrases, les colonnes décrivent les mots et les valeurs indiquent l'importance des mots dans les phrases. TF signifie la fréquence des termes, qui calcule combien de fois un mot apparaît dans une phrase donnée. IDF signifie la fréquence inverse de document qui compte combien de fois ce mot apparaît dans un ensemble de phrases. Si un terme apparaît fréquemment dans une phrase donnée, mais qu'il apparaît également dans plusieurs autres phrases, ce terme n'est pas utile pour distinguer une phrase donnée.

$$W_{w,s} = \text{tf}_{w,s} \times \log\left(\frac{N}{\text{df}_w}\right) \quad (3.1)$$

Où :

- $W_{w,s}$: le poids du mot w dans la phrase s .
- $\text{tf}_{w,s}$: le nombre d'occurrences du mot w dans la phrase s .
- df_w : le nombre de phrases contenant le mot w .
- N : le nombre total de phrases.

D'après la formule numérique précédente, on peut observer que si un terme apparaît fréquemment dans une phrase donnée, mais qu'il n'apparaît pas dans diverses autres phrases. Son taux de TF sera élevé, et son IDF sera très proche de 1. Par conséquent, son taux de TF-IDF sera élevé. En revanche, si un terme apparaît fréquemment dans une phrase donnée, mais qu'il apparaît également dans diverses autres phrases, même si son taux TF sera élevé, son taux IDF sera proche de 0. Par conséquent, son taux TF-IDF sera très faible. Si un terme apparaît fréquemment dans une phrase donnée et qu'il existe également dans toutes les autres phrases, son taux IDF sera égal à 0, et son taux TF-IDF sera également égal à 0. Les termes pertinents, qui apparaissent fréquemment dans une phrase donnée, auront un taux TF-IDF élevé, tandis que les termes moins pertinents, qui apparaissent fréquemment à la fois dans une phrase donnée et dans diverses autres phrases, auront un taux TF-IDF faible. Ces taux représentent les caractéristiques qui seront utilisées ultérieurement par les classificateurs d'apprentissage automatique dans le processus d'apprentissage.

3.6.4 GloVe

Glove pour le plongement lexical du mot a été présenté par Jeffrey Pennington et al. [105], et a été soutenu par l'Université de Stanford. Ce modèle d'apprentissage est un algorithme non supervisé. Son objectif est de calculer le plongement lexical pour les mots distribués. Cette opération s'effectue en trouvant la similarité sémantique entre les mots, puis générer la matrice de comptage de cooccurrence mot par mot, c'est-à-dire la fréquence à laquelle ces mots apparaissent ensemble dans le corpus. Pour cela, Glove a été nommé le modèle basé sur le comptage. le plongement lexical des mots de ce modèle est obtenu en agrégeant la matrice de comptage des co-occurrences créée à partir d'un corpus, et le plongement lexical de mots résultants montrent pour chaque mot dans l'espace vectoriel des sous-structures linéaires importantes. En résumé, ce modèle intègre les deux méthodes, qui sont le modèle de la fenêtre du contexte local et la méthode de factorisation matricielle globale. Expérimentalement, GloVe donne de bons résultats sur la similarité des mots, la reconnaissance des entités nommées et les tâches d'analogie des mots par rapport à word2vec et FastText. La figure 3.5 illustre les sous-structures linéaires d'un ensemble de mots lors de l'application de la méthode de plongement de mots GloVe.

La représentation graphique illustrée dans la figure 3.5 est le résultat de l'application d'un schéma d'évaluation basé sur des analogies de mots qui est adopté par GloVe pour trouver l'espace vectoriel d'un mot inconnu est l'utilisé. Par exemple, la relation "the brother is to sister as uncle is to aunt" sont représentés en utilisant l'espace de représentation vectorielle qui est calculé par l'équation vectorielle **brother-sister=uncle-aunt**. Ce sys-

tème d'évaluation préfère les modèles qui fournissent des dimensions significatives, d'où le concept de multi-clusters des vectorisations distribuées.

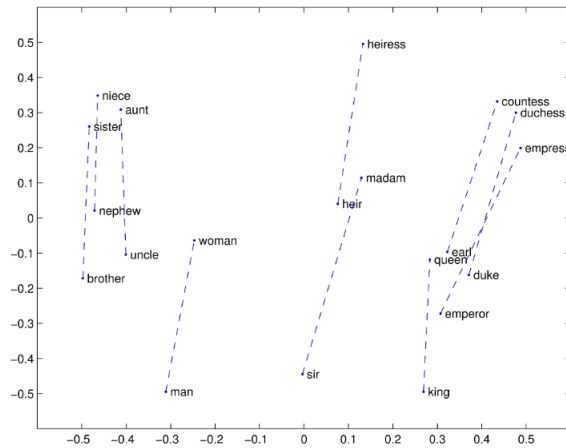


FIGURE 3.5 – Les sous-structures linéaires d'un ensemble de termes en appliquant GloVe.

3.6.5 Word2Vec

Word2Vec a été développé par une équipe de recherche chez Google sous la direction de Tomas Mikolov [106]. Il utilise le réseau de neurones à propagation avant avec une couche cachée pour extraire le vecteur de plongement lexical du mot à partir des données de texte/image en entrées. Cette méthode intègre le modèle skip-Gram qui vise à prédire les mots du contexte actuel en se basant sur les mots cibles en entrée, et le modèle de sacs de mots continus (SMC) visant à prédire un mot étant donné son contexte comme montré dans la figure 3.6. En pratique, le modèle SMC est plus rapide à apprendre, mais le modèle skip-gram donne généralement de meilleurs résultats. C'est pour cette raison qu'elle a été appelée réseau de neurones à deux couches. Son objectif est d'améliorer la capacité prédictive de plongement lexical des mots.

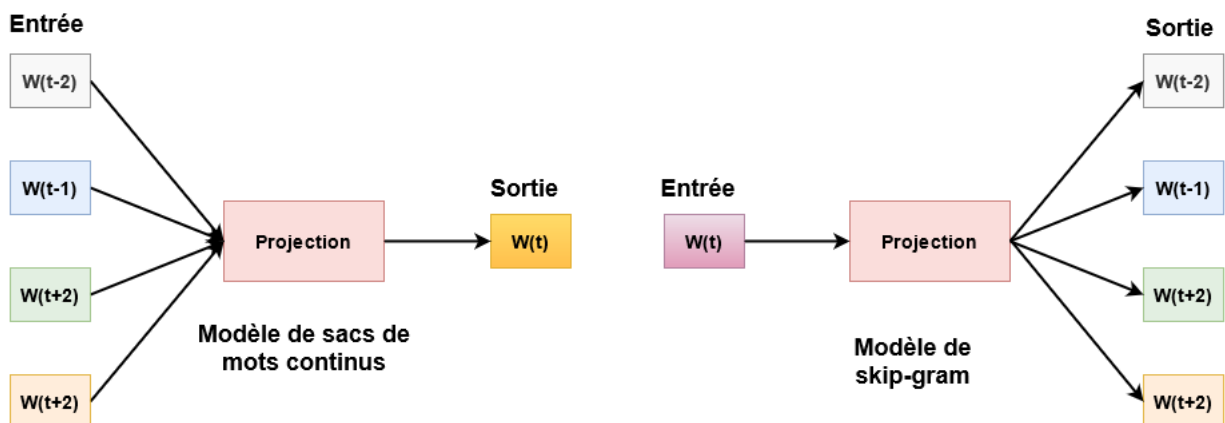


FIGURE 3.6 – Modèle de skip-gram et le modèle de sacs de mots continus.

La méthode Word2Vec prend en entrée un vaste corpus de données textuelles/images et génère en sortie une matrice de plongement lexical. Où chaque ligne de la matrice créée

représente le vecteur à plusieurs centaines de dimensions. Ce vecteur obtenu représente le plongement lexical du vecteur one-hot du jeton d'entrée (mot ou caractère) comme décrit dans la figure 3.7.

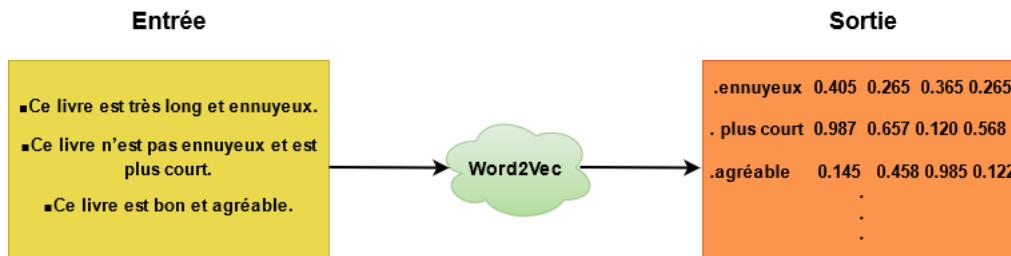


FIGURE 3.7 – Processus d'application de la méthode Word2Vec.

En résumé, Word2Vec est un simple réseau de neurones à propagation avant avec une couche cachée. Pendant le processus d'apprentissage, son objectif principal est d'ajuster leurs poids pour minimiser le taux d'erreur en diminuant la fonction de perte. Ces poids cachés sont utilisés comme des plongements lexicaux de mots. Word2Vec offre de meilleures performances dans la prédiction des polarités de sentiments, et ses performances sont meilleures sur des jeux de données massives.

3.6.6 FastText

FastText est un autre technique de plongement lexical de mots créés par l'équipe de recherche en intelligence artificielle de Facebook pour un apprentissage efficace de la vectorisation des mots. Cette méthode est considérée comme une extension de la méthode Word2Vec ; au lieu d'entraîner un ensemble de jetons (mots) directement comme dans la méthode Word2Vec, FastText entraîne chaque jeton comme un n-gramme de caractères. Par exemple, la représentation du mot "fuzzy" selon la méthode FastText avec n-gramme=2 est (f, fu, uz, zz, zy, y), où les parenthèses indiquent le début et la fin du mot représenté. Cela permet de détecter le sens des mots plus courts et aide le plongement lexical à apprendre les suffixes et préfixes du mot. Ainsi, une fois que le jeton entré a été divisé en appliquant la méthode N-gramme, soit le modèle skip-gram soit le modèle de sacs de mots continus est utilisé pour apprendre le plongement des mots. FastText fonctionne bien avec les mots inconnus et les mots hors vocabulaire. Ainsi, même si le mot est invisible dans les étapes d'apprentissages précédentes, cette méthode le décompose en caractères N-grammes pour calculer ses plongements lexicaux. Les méthodes Word2Vec et Glove ne parviennent pas à obtenir les plongements lexicaux pour les mots invisibles, contrairement à FastText qui peut apprendre les mots invisibles. C'est le point fort de cette méthode par rapport à d'autres techniques [107].

3.7 Sélection de caractéristiques

La sélection des caractéristiques est d'une importance considérable dans la classification des modèles, l'analyse des données, l'apprentissage automatique et l'exploration des données. Pour de nombreux problèmes de classification des modèles, une bonne méthode de sélection des caractéristiques peut réduire le coût de la mesure des caractéristiques, et peut augmenter l'efficacité du classificateur et la précision de la classification [108]. Par conséquent, la sélection des caractéristiques peut servir d'outil de prétraitement très important avant de résoudre les problèmes de classification.

La sélection des caractéristiques fait référence au processus de sélection des caractéristiques pertinents dans le texte, où chaque terme (mot/phrased) dans le texte représente généralement un élément. Les objectifs sont d'améliorer l'efficacité de la classification et l'efficacité en matière de calcul (en réduisant la dimensionnalité) [109]. Miner et al. [110] ont classé les approches de sélection des caractéristiques dans le domaine de la fouille de données selon qu'elles étaient basées sur :

1. **Théorie de l'information** : c'est la recherche du meilleur moyen de traiter les signaux et de comprimer et communiquer les données.
2. **Statistiques** : déterminer la corrélation statistique entre les termes et les étiquettes de classe des documents.
3. **Fréquence** : déterminer l'importance des termes en fonction de leur fréquence dans les documents.

Par rapport à la théorie de l'information et les méthodes statistiques, les méthodes de fréquences sont moins coûteuses en matière de calcul. Les approches les plus pertinentes en ce qui concerne ces catégories et le travail décrit dans cette thèse sont décrites dans les sous-sections suivantes, à savoir : *Gain d'Information* (GI), *Rapport de Gain d'Information* (RGI), *Indice de Gini* (IG), et *Khi-Carré* (KC).

3.7.1 Gain d'information

Le Gain d'information (GI) est basé sur la théorie de l'information, qui concerne le traitement et la compression des signaux et des données de communication, et a été introduit en 1948 par Claude Shannon [111] qui est considéré comme le "père" de la théorie de l'information. Selon Miner et al [110] "IG mesure combien l'incertitude sur la variable cible, appelée entropie, est réduite lorsque l'attribut est utilisé". En d'autres termes, étant donné que l'entropie est une mesure de l'incertitude concernant un ensemble d'apprentissage (ou la quantité d'informations requises pour attribuer un label de classe à une instance). L'IG est un indicateur de la quantité d'informations obtenues d'une entropie initiale à une nouvelle entropie d'un attribut. Il est calculé comme suit :

$$GI(A) = Info(D) - Info_A(D) \quad (3.2)$$

Où D est une partition de données qui comprend des instances au niveau d'un noeud N, et qui représente des tuples de la partition D. L'information nécessaire pour attribuer un

label de classe à une instance dans D , en d'autres termes l'entropie de D , est donnée par :

$$Info(D) = - \sum_{i=1}^m p_i \log_2(p_i) \quad (3.3)$$

Où une étiquette de classe peut avoir m valeurs différentes et p_i est la probabilité qu'une instance appartenant à D soit liée à une certaine classe. L'information nécessaire pour produire une classification correcte est donnée par :

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} * Info(D_j) \quad (3.4)$$

$\frac{|D_j|}{|D|}$ représente le poids de la j ème partition. Un attribut avec un GI élevé a une meilleure prédiction de l'occurrence de la variable cible. En général, les attributs sont classés en fonction de leur GI, et les attributs ayant des valeurs plus élevées (qui ont une meilleure capacité de prédiction par rapport aux étiquettes de classe) sont choisis.

3.7.2 Rapport de Gain d'information

Le GI est biaisé en faveur des attributs ayant un grand nombre de valeurs. Dans le cas le plus extrême, nous pouvons maximiser le gain d'information en traitant chaque observation comme une partition. Pour surmonter ce biais, plusieurs mesures correctives ont été proposées. La partition de l'information (Split-Info) est la plus couramment utilisée.

$$Split - Info(A) = - \sum_{v \in values(D_j)} \frac{|D_{j,v}|}{|D_j|} * \log \frac{|D_{j,v}|}{|D_j|} \quad (3.5)$$

$Split-Info(A)$ est l'information nécessaire après le partitionnement par l'attribut A , dans lequel D_j est un sous-ensemble de D induit par l'attribut A , et $D_{j,v}$ est un sous-ensemble de D_j de valeur v pour l'attribut A , et $Values(D_j)$ est l'ensemble des valeurs de l'attribut A pour les enregistrements dans D_j . L'opérateur de valeur absolue signifie la cardinalité de la mesure GI après le partitionnement par l'attribut A .

Le RGI est un rapport entre le GI et l'information intrinsèque. Il a été proposé par Ross Quinlan [112], pour réduire un biais vers les attributs à valeurs multiples en prenant en compte le nombre et la taille des branches lors du choix d'un attribut. Le RGI est calculé en utilisant le GI divisé par le Split-Info comme suit :

$$RGI(A) = \frac{GI(A)}{Split_Info(A)} \quad (3.6)$$

3.7.3 Indice de Gini

IG a été élaboré par Gini en 1912 [113] et entretient un lien strict avec la représentation de l'inégalité des revenus à l'aide de la courbe de Lorenz. En particulier, il mesure le ratio entre l'aire située entre la courbe de Lorenz et la droite d'équidistribution et l'aire de

concentration maximale. Donc, ce coefficient de Gini est une mesure de l'inégalité d'une distribution. Il est défini comme un rapport dont les valeurs sont comprises entre 0 et 1 : le numérateur est l'aire entre la courbe de Lorenz de la distribution et la ligne de distribution uniforme. Le dénominateur est la zone située sous la ligne de distribution uniforme. L'IG pour chaque valeur de l'attribut A est calculé comme suit :

$$Gini(A = i) = 1 - \sum_{i=1}^n (p_i)^2 \quad (3.7)$$

L'IG pour l'attribut A est la summation de l'IG de toutes ses valeurs, Il est calculé comme suit :

$$Gini(A) = \sum_{i=1}^V \frac{|D_i|}{|D|} Gini(A = i) \quad (3.8)$$

Où V est le nombre de valeurs de l'attribut A , $|D_v|$ est le nombre d'instance dont l'attribut A prend la valeur i . $|D|$ est le nombre de toutes les instances.

3.7.4 Khi-Carré

Le Khi-Carré (KC) est une technique statistique permettant d'estimer le degré de dépendance d'une variable conditionnelle par rapport à la variable cible correspondante. Un taux de KC élevé indique que l'association entre une variable conditionnelle et la variable cible correspondante est très forte. Alors qu'un faible taux de KC indique que l'association entre une variable conditionnelle et la variable cible correspondante est très faible. Et KC est égal à 0 dans le cas où la variable conditionnelle est indépendante de la variable cible correspondante. Le score KC est calculé en suivant les étapes décrites ci-dessous :

- Déterminer l'hypothèse nulle qui montre que la variable conditionnelle et la variable cible correspondante sont indépendantes, et définir l'hypothèse alternative qui indique que la variable conditionnelle est dépendante de la variable cible correspondante.
- Concevoir un tableau qui indique comment se répartissent les variables cibles correspondantes en lignes et les variables conditionnelles en colonnes. Le rapport de liberté pour le tableau créé est $(\text{ligne} - 1) \times (\text{colonne} - 1)$.
- Calculer les valeurs prédites pour toutes les cellules du tableau formé précédemment en utilisant la formule suivante : $e = n \times p$. Où e est la valeur prédite, n est le nombre de fois que la valeur de chaque cellule apparaît dans le tableau et p est la probabilité conjointe entre une variable conditionnelle et sa variable cible correspondante.
- Calculer le score KC en utilisant la formule suivante : $x_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$. Où c est le degré de liberté, O est la valeur observée, E est la valeur attendue et i indique la variable.
- Le taux de KC calculé ci-dessus peut être examiné par rapport au tableau de KC créé à l'étape 2 pour vérifier s'il se situe dans la cellule d'acceptation ou de rejet. Si

elle se situe dans la cellule de rejet, alors l'hypothèse nulle est déclinée, et l'hypothèse alternative est admise. S'il se situe dans la cellule d'acceptation, alors c'est l'inverse. Par conséquent, il est possible de déterminer si la variable conditionnelle et la variable cible correspondante sont indépendantes l'une de l'autre. S'ils sont autonomes, alors la variable conditionnelle peut être exclue.

3.8 Algorithmes de classification

Il n'existe pas d'algorithme de classification meilleure et unique qui puisse être appliqué efficacement à chaque problème de la fouille de données. Les raisons de cette situation ne sont pas claires, mais on suppose que cela est dû aux caractéristiques uniques des jeux de données à savoir : (i) le type de données, (ii) la taille des données, (iii) le nombre d'enregistrements et (iv) la répartition des classes. La recherche menée au sein de la communauté de la fouille de données, au cours des dernières années, a donné naissance à de nombreuses techniques différentes qui pourraient convenir à des conditions spécifiques tels que les petits jeux de données, les jeux de données composés principalement d'enregistrements numériques et les jeux de données qui comportent un grand nombre de classes. Parmi les techniques proposées par la communauté de la fouille de données nous avons constaté : (i) Naïve bayésienne (NB), (ii) Arbres de décision (C4.5, ID3, CART, et forêts d'arbres décisionnels), (iii) Méthode des k plus proches voisins (KNN), (iv) Machines à vecteurs de support (SVM), (v) Réseau de neurones à propagation avant (FFNN), (vi) Réseau de neurones convolutifs (CNN) (vii) et le système flou de Mamdani (MFS). Chacun d'eux est discuté en détail dans les sous-sections qui suivent.

3.8.1 Naïve bayésienne

La classification NB est un type de classification bayésienne probabiliste simple basé sur le théorème de Bayes avec une forte indépendance (dite naïve) des hypothèses. Elle met en œuvre un classifieur NB appartenant à la famille des classifieurs linéaires, qui a été proposé et nommé d'après Thomas Bayes [114]. Le théorème de Bayes s'exprime généralement sous la forme suivante :

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (3.9)$$

Où A est une hypothèse, B est une évidence, $P(A|B)$ est la probabilité postérieure de A conditionné par B , $P(A)$ et $P(B)$ sont les probabilités antérieures de A et B respectivement, et $P(B|A)$ est la probabilité postérieure de B conditionné par A .

Parmi les nombreuses implémentations du théorème de Bayes qui ont été proposées, la plus simple version de NB est la plus utilisée. La méthode NB combine les probabilités antérieures et conditionnelles pour calculer la probabilité de classifications alternatives [115]. Il est appelé "naïf" parce qu'il suppose "naïvement" que les attributs sont indépendants les uns des autres, de sorte que les probabilités peuvent être multipliées. Malgré l'hypothèse «naïve», l'algorithme NB s'est avéré être une forme efficace de classificateurs, surtout

lorsque la sélection des caractéristiques a été effectuée et qu'il ne reste que des attributs non redondants (indépendants).

Ce théorème montre un moyen de trouver la probabilité d' A étant donné de l'évidence B . Dans la classification, cela signifie que nous avons un élément A et une évidence de classe B . Une certaine méthode d'évaluation, telle qu'un seuil, est choisie pour déterminer si les éléments appartiennent ou non à la classe. Par exemple, si la probabilité est supérieure à 0.6 les éléments sont appartenus à la catégorie. Ce classificateur considère que toutes les caractéristiques sont indépendantes.

3.8.2 Arbres de décision

L'apprentissage par arbre de décision désigne une technique d'apprentissage supervisé basée sur l'utilisation d'un arbre de décision comme modèle prédictif [116]. Il est considéré comme un outil d'aide à la décision représentant un ensemble de choix sous la forme graphique d'un arbre. Les différentes décisions possibles sont situées aux extrémités des branches de l'arbre (les feuilles de l'arbre), et sont atteintes en fonction de décisions prises à chaque étape. L'arbre de décision est un outil utilisé dans des domaines variés tels que la sécurité, la fouille de données, et la médecine, etc. Il présente plusieurs avantages : la simplicité de compréhension, d'interprétation et la haute performance sur de grands jeux de données. Il s'agit de plus d'une représentation calculable automatiquement par des algorithmes d'apprentissage supervisé. L'idée générale des arbres de décision est de diviser récursivement et le plus efficacement possible les exemples de l'ensemble d'apprentissage par des tests définis à l'aide des attributs jusqu'à ce que l'on obtienne des sous-ensembles d'exemples ne contenant (presque) que des exemples appartenant tous à une même classe. Dans toutes les méthodes basées sur l'arbre de décision, nous trouvons les trois opérateurs suivants :

- *Décider si un nœud est terminal*, c'est-à-dire décider si un nœud doit être étiqueté comme un nœud feuille.
- *Sélectionner un test à associer à un nœud*, soit aléatoirement et soit en utilisant des critères statistiques (si ce nœud n'est pas terminal).
- *Affecter une classe à un nœud* si ce nœud est terminal.

Les méthodes vont différer par les choix effectués pour ses différents opérateurs, c'est-à-dire selon le choix d'un test (par exemple, utilisation du coût et de la fonction entropie) et le critère d'arrêt (quand arrêter la croissance de l'arbre c'est-à-dire quand décider si un nœud est terminal). Nous pouvons alors définir un schéma général d'algorithme, sans spécifier comment seront définis les trois opérateurs décrits plus haut, comme montré dans l'algorithme 8.

Avec un tel algorithme, nous décidons qu'un nœud est terminal lorsque tous les exemples associés à ce nœud, ou du moins la plupart d'entre eux sont dans la même classe. De plus, un arbre de décision parfait est un arbre dont tous les exemples de l'ensemble d'apprentissage soient correctement classifiés. Lorsque plusieurs classes sont en concurrence, nous pouvons choisir la classe la plus représentée dans l'ensemble de l'échantillon, ou en choisir une au hasard. La sélection d'un test à associer à un nœud est plus délicate.

Algorithm 1 APPRENTISSAGE GÉNÉRIQUE PAR ARBRE DE DÉCISION**Input** : Données : échantillon S

```

1 ;
  Initialiser l'arbre courant a l'arbre vide ; la racine est le noeud courant ;
  while noeud courant n'est pas terminal do
2   if le noeud est terminal then
3     | Affecter ce noeud à une classe ;
4     | return
5   else
6     | Sélectionner un test et créer autant de nouveaux noeuds fils qu'il y a de réponses
7     | possibles au test ;
8   end if
9   Passez au noeud suivant non explorer s'il en existe ;
10  Jusqu'à obtenir un arbre de décision ;
end while
Output : Arbre de décision

```

Tandis que, l'idéal serait de trouver un critère qui permet d'arrêter la croissance de l'arbre au bon moment. Malheureusement, dans l'état actuel des recherches, un tel critère n'a pu être trouvé. En outre, le risque d'arrêter trop tôt la croissance de l'arbre est plus important que de l'arrêter trop tard. Par conséquent, les méthodes utilisées pour la construction d'un arbre de décision peuvent contenir plusieurs anomalies liées au problème du surapprentissage (overfitting). Il s'agit de la déduction d'informations plus que supporte l'ensemble de données d'apprentissage.

Exemple illustratif : nous allons considérer un exemple très simple pour introduire les algorithmes d'apprentissage par arbres de décision. Nous prenons en entrée un ensemble de données classées, et nous fournissons en sortie un arbre où chaque nœud final (feuille) représente une décision (une classe) et chaque nœud non final (intermédiaire) représente un test. Chaque feuille représente la décision d'appartenance à une classe de données vérifiant tous les tests du chemin menant de la racine à cette feuille. L'exemple du tableau 3.1 montre un ensemble de données avec quatre attributs : Ciel, Température, Humidité, Vent et l'attribut à prédire Jouer.

L'arbre appris à partir de cet ensemble de donnée est le suivant (figure 3.8) :

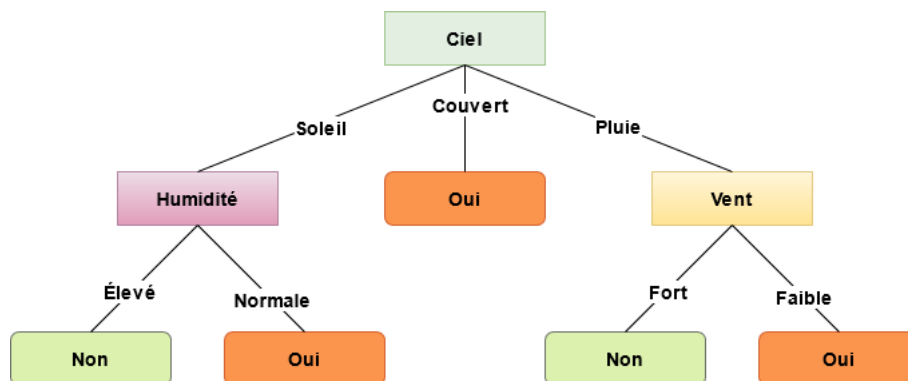


FIGURE 3.8 – Exemple d'arbre de décision.

TABLE 3.1 – Échantillon des données météorologiques

Jour	Ciel	Température	Humidité	Vent	Jouer
J1	soleil	chaud	élevé	faible	non
J2	soleil	chaud	élevé	fort	non
J3	couvert	chaud	élevé	faible	oui
J4	pluie	doux	élevé	faible	oui
J5	pluie	froid	normale	faible	oui
J6	pluie	froid	normale	fort	non
J7	couvert	froid	normale	faible	oui
J8	soleil	doux	élevé	faible	non
J9	soleil	froid	normale	faible	oui
J10	pluie	doux	normale	fort	oui
J11	soleil	doux	normale	fort	oui
J12	couvert	doux	élevé	fort	oui
J13	couvert	chaud	normale	faible	oui
J14	pluie	doux	élevé	fort	non

Nous pouvons remarquer que toutes les données ayant l'attribut Ciel="Soleil" et l'attribut Humidité="Normale" appartiennent à la classe ("oui"). En effet, toute nouvelle donnée peut être classée en testant ses valeurs d'attributs l'un après l'autre en commençant de la racine jusqu'à atteindre une feuille c'est-à-dire une décision.

Pour construire un tel arbre, plusieurs algorithmes existent : ID3 [117], CART [118], C4.5 [119], FA [120], etc. Nous commençons généralement par le choix du meilleur attribut puis nous affectons cet attribut à la racine. Pour chaque valeur de cet attribut, nous créons un nouveau noeud fils de la racine après nous classons les exemples dans les noeuds fils. Si tous les exemples d'un noeud fils sont homogènes, nous affectons leur classe au noeud, sinon nous commençons à partir de ce noeud. L'algorithme continue d'une manière récursive jusqu'à obtenir des noeuds qui contiennent des données homogènes.

L'algorithme ID3 : l'algorithme ID3 [117] construit l'arbre de décision récursivement. À chaque étape il calcule parmi les attributs restants pour la branche en cours, celui qui maximisera le gain d'information. C'est-à-dire l'attribut qui permettra le plus facilement de classer les exemples. À ce niveau de cette branche de l'arbre. Le calcul se fait à base de l'entropie de Shannon déjà présentée ci-dessus. L'algorithme suppose que tous les attributs sont catégoriels ; si des attributs sont numériques, ils doivent être discrétisés pour pouvoir l'appliquer.

L'algorithme CART : l'algorithme CART [118], construit un arbre de décision d'une manière analogue à l'algorithme ID3. Contrairement à ce dernier, l'arbre de décision généré par CART est binaire (un noeud ne peut avoir que 2 fils) et le critère de segmentation est l'indice de Gini [121] déjà présentée ci-dessus. A est un attribut binaire correspond un test binaire ou \hat{A} et un attribut qualitatif ayant n modalités, nous pouvons associer autant de tests qu'il y a de partitions en deux classes, soit $2n - 1$ tests binaires possibles. Enfin, dans le cas d'attributs continus, il y a une infinité de tests envisageables. Dans ce cas, on découpe l'ensemble des valeurs possibles en segments, ce découpage peut être réalisé par un expert ou de façon automatique.

L’algorithme C4.5 : L’algorithme C4.5 [119] est une amélioration de l’algorithme ID3, appelé aussi (J48), il prend en compte les attributs numériques ainsi que les valeurs manquantes. Cet algorithme s’appuie sur le rapport de gain d’information défini dans (l’équation (3.6)) et la fonction entropie (l’équation (3.3)) combiné avec une fonction SplitInfo (l’équation (3.5)) pour évaluer les attributs à chaque itération. Pour les attributs à valeur sur intervalle continu : l’algorithme permet de les gérer de la façon suivante : Si l’attribut C_i a un intervalle continu de valeurs. Nous examinons les valeurs de cet attribut dans les données d’apprentissage. Supposons que ces valeurs sont en ordre croissant, A_1, A_2, \dots, A_m . Ensuite pour chacune de ces valeurs, on partitionne les enregistrements entre ceux qui ont des valeurs de C_i inférieures ou égales à A_j et celles qui ont des valeurs supérieures à A_j . Pour chacune de ces partitions nous calculons le gain, ou le rapport de gain et nous choisissons la partition qui maximise le gain. Pour les attributs à valeurs manquantes : dans de nombreux problèmes concrets, il existe certains attributs dont les valeurs ne sont pas renseignées. Par exemple, si nous disposons des données de patients, il est très probable que toutes les mesures ne sont pas disponibles car elles n’ont pas pu être faites pour tous les patients. Pour classifier un exemple possédant des valeurs manquantes à l’aide d’arbres de décision, nous procédons comme dans le cas standard, lorsque nous rencontrons un test et que la valeur de l’attribut est manquante, nous considérons la branche majoritaire. Pour la phase d’apprentissage, nous supposons que la valeur de cet attribut suit la distribution des valeurs connues.

L’algorithme de forêts aléatoires : les forêts aléatoires ont été inventées par Breiman [120]. Elles sont composées (comme le terme forêt l’indique) d’un ensemble d’arbres décisionnels. Ces arbres se distinguent les uns des autres par le sous-échantillon de données sur lequel ils sont entraînés. Ces sous-échantillons sont tirés au hasard (d’où le terme aléatoire) dans le jeu de données initial. Elles sont en général plus efficaces que les simples arbres de décision mais possèdent l’inconvénient d’être plus difficilement interprétables. Leur construction se base sur le bootstrap (ou le bagging). On subdivise l’ensemble de données en plusieurs parties par le bootstrap puis on obtient un arbre de décision à partir de chaque partie.

3.8.3 Méthode des k plus proches voisins

La technique des KPPV est une procédure d’apprentissage supervisé, il est nécessaire d’avoir des données labellisées. À partir d’un ensemble E de données labellisées, il sera possible de classer (déterminer la classe) d’une nouvelle donnée (donnée n’appartenant pas à E). Cette méthode fonctionne en utilisant une certaine fonction de similarité pour comparer les nouvelles instances avec les instances existantes. La similarité est donnée par la représentation de chaque instance comme un point dans un espace à n dimensions où une instance inconnue est classée selon les instances classées les plus proches. La distance entre les points est généralement mesurée à l’aide de la distance euclidienne, mais d’autres mesures peuvent être utilisées, par exemple la distance de Mahalanobis ou la distance de Tchebychev. L’algorithme KPPV classe les instances inconnues en fonction de leur similarité avec les instances d’entraînement les plus proches dans un espace de caractéristiques. En raison de sa nature, il est plus coûteux en termes de calcul que les autres méthodes [122]. L’algorithme des KPPV ne nécessite pas de phase d’apprentissage

à proprement parler, il faut juste stocker le jeu de données d'apprentissage.

Soit un ensemble E contenant n données labellisées $E = (y_i, \vec{x}_i)$ avec i compris entre 1 et n , où y_i correspond à la classe (l'étiquette) de la donnée i et où le vecteur \vec{x}_i de dimension p ($\vec{x}_i = (x_{1i}, x_{2i}, \dots, x_{pi})$) représente les variables prédictives de la donnée i . Soit une donnée u qui n'appartient pas à E et qui ne possède pas d'étiquette (u est uniquement caractérisée par un vecteur \vec{x}_i de dimension p). Soit d une fonction qui renvoie la distance entre la donnée u et une donnée quelconque appartenant à E . Soit un entier k inférieur ou égal à n . Voici le principe de l'algorithme des KPPV :

- Calculer les distances entre la donnée u et chaque donnée appartenant à E à l'aide de la fonction d .
- Retenir les k données du jeu de données E les plus proches de u .
- Attribuer à u la classe qui est la plus fréquente parmi les k données les plus proches.

3.8.4 Machines à vecteurs de support

Les SVM ou séparateurs à vaste marge sont un ensemble de techniques d'apprentissage supervisé destinées à résoudre des problèmes de classification et de régression. Les SVM sont une généralisation des classificateurs linéaires. Les séparateurs à vaste marge ont été développés dans les années 1990 à partir des considérations théoriques de Vladimir Vapnik [123] sur le développement d'une théorie statistique de l'apprentissage. Les SVM ont été appliqués à de très nombreux domaines à savoir : bio-informatique, récupération d'informations, vision par ordinateur, et finance. Toutefois, les SVM se sont révélés très efficaces dans le contexte de la classification des textes [124]. Les SVM sont fondés sur deux idées clés : la théorie de la marge maximale et la théorie de fonction noyau. Ces deux concepts existaient depuis longtemps avant qu'ils ne soient combinés pour élaborer des SVM [125]. Ils permettent de résoudre les problématiques de discrimination non linéaire, et de reformuler le problème de classification en un problème d'optimisation quadratique.

La première idée clé est la notion de marge maximale. La marge est la distance entre la frontière de séparation et les échantillons les plus proches. Ces derniers sont appelés vecteurs supports. Dans les SVM, la frontière de séparation est choisie comme celle qui maximise la marge. Ce choix est justifié par la théorie de Vapnik-Chervonenkis (ou théorie statistique de l'apprentissage), qui montre que la frontière de séparation de marge maximale possède la plus petite capacité. Le problème est de trouver cette frontière séparatrice optimale, à partir d'un ensemble d'apprentissage. Ceci est fait en formulant le problème comme un problème d'optimisation quadratique, pour lequel il existe des algorithmes connus.

Afin de pouvoir traiter des cas où les données ne sont pas linéairement séparables, la deuxième idée clé des SVM est de transformer l'espace de représentation des données d'entrées en un espace de plus grande dimension (possiblement de dimension infinie), dans lequel il est probable qu'il existe une séparation linéaire. Ceci est réalisé grâce à une fonction noyau, qui doit respecter les conditions du théorème de Mercer, et qui a l'avantage de ne pas nécessiter la connaissance explicite de la transformation à appliquer pour le changement d'espace. Les fonctions noyau permettent de transformer un produit

scalaire dans un espace de grandes dimensions, ce qui est coûteux, en une simple évaluation ponctuelle d'une fonction. Cette technique est connue sous le nom de kernel trick (ou astuce du noyau).

Les SVM sont une forme d'algorithmes d'apprentissage supervisé qui peut être utilisée pour résoudre les problèmes de classification et de régression [125]. Comme le montre Joachims [126], la méthode SVM est donnée de bons résultats pour la classification des textes en raison de sa capacité à généraliser en grandes dimensions, ce qui est souvent le cas avec la catégorisation des textes. Les SVM sont également connus pour bien fonctionner dans les cas où le nombre d'attributs est supérieur au nombre des d'échantillons d'apprentissage.

3.8.5 Réseau de neurones

Un réseau de neurones artificiels ou réseau neuronal artificiel, est un système dont la conception est à l'origine schématiquement inspirée du fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques. Un réseau de neurones peut être décrit comme un réseau de couches (couches d'entrée, cachées et de sortie) où chaque couche est constituée de nœuds [127]. La figure 3.9 est une illustration de la structure d'un réseau de neurones. Tous les nœuds de la couche d'entrée sont connectés à tous les nœuds de la couche cachée suivante avec des poids ajustables. Le réseau de neurones peut être considéré comme un classificateur linéaire si l'on utilise une fonction d'activation linéaire. L'ajustement du poids en fonction d'un ensemble donné de X et Y est ce que l'on appelle l'apprentissage du réseau. X représente un ensemble de vecteurs p -dimensionnels et Y représente un ensemble de classes indiquant à quelle classe $y_i \in Y$ appartient le point $x_i \in X$. Habituellement, l'apprentissage est effectué en initialisant de manière aléatoire les poids du réseau et en utilisant l'algorithme de rétropropagation pour ajuster les poids dans le but de minimiser l'erreur de la sortie du réseau. Le processus peut être divisé en deux parties : la propagation et la mise à jour du poids.

Propagation : ce processus peut être répété plusieurs fois et une propagation progressive +rétrogressive est parfois considérée comme une seule époque. Avant d'exécuter la première époque, on initie aléatoirement les poids du réseau de neurones. Pour chaque propagation du réseau :

- Effectuer une propagation progressive, c'est-à-dire faire passer l'entrée via le réseau pour générer les sorties prévisionnelles du réseau.
- Effectuer une propagation rétrogressive, en commençant par la sortie en cours et en mesurant les erreurs, dénommés deltas (prédictions - valeurs actuelles), pour chaque poids.

Mises à jour des poids : pour chaque poids du réseau :

- Poids de chaque sortie du réseau est multiplié par le delta pour obtenir le gradient.
- Gradient est multiplié par le taux d'apprentissage puis il est soustrait du poids.

Le taux d'apprentissage peut être choisi arbitrairement et affectera la rapidité et la précision de l'apprentissage du modèle. Une grande valeur entraînera le modèle plus rapidement, mais une petite valeur le fera avec plus de précision.

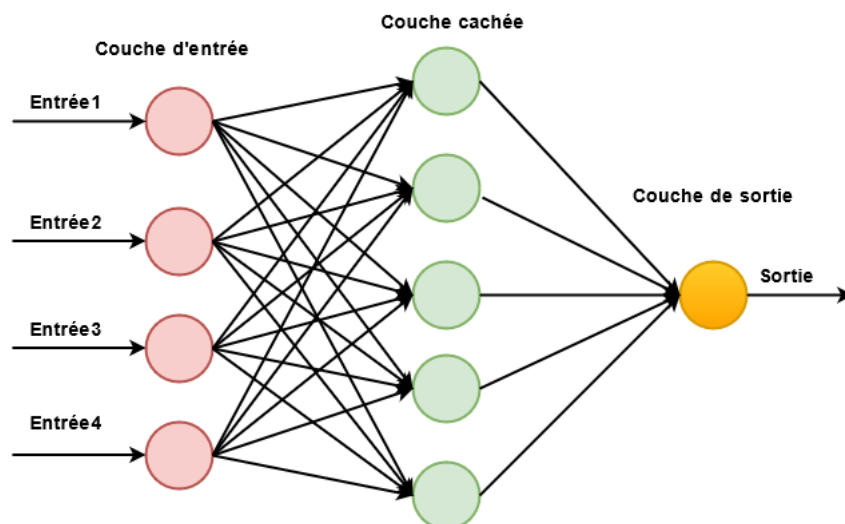


FIGURE 3.9 – Illustration de la structure d'un réseau de neurones.

3.8.5.1 Réseau de neurones à propagation avant

Un réseau neuronal multicouche est constitué d'un grand nombre d'unités (neurones) reliées entre elles selon un modèle de connexions (figure 3.10). Les unités d'un réseau sont généralement divisées en trois classes : les unités d'entrée, qui reçoivent les informations à traiter ; les unités de sortie, où se trouvent les résultats du traitement ; et les unités intermédiaires, appelées unités cachées [128]. Les RNPA (figure 3.10) permettent aux signaux de circuler dans un seul sens, de l'entrée à la sortie.

En général, il est difficile de déterminer correctement la taille de la couche cachée, car une sous-estimation du nombre de neurones peut entraîner une mauvaise approximation et une généralisation tandis qu'un nombre excessif de nœuds peuvent entraîner un suréquipement et rendre la recherche de l'optimum global plus difficile. On trouvera un excellent argument à ce sujet dans [129, 130] a également recherché le nombre minimum de neurones et le nombre d'instances nécessaires pour programmer une tâche donnée en RNPA.

Le RNPA dépend de trois aspects fondamentaux, à savoir les fonctions d'activation de l'unité, l'architecture du réseau et le poids de chaque connexion en entrée. Étant donné que les deux premiers aspects sont fixes, le comportement du RNPA est défini par les valeurs actuelles des poids. Les poids du réseau à entraîner sont d'abord fixés à des valeurs aléatoires, puis des instances de l'ensemble d'entraînement sont exposées de manière répétée au réseau. Les valeurs pour l'entrée d'une instance sont placées sur les unités d'entrée et la sortie du réseau est comparée à la sortie souhaitée pour cette instance. Ensuite, tous les poids du réseau sont légèrement ajustés dans le sens qui rapprocherait les valeurs de sortie du réseau à des valeurs de la sortie souhaitée. Il existe plusieurs algorithmes avec lesquels un réseau peut-être entraîné [131]. Cependant, l'algorithme d'apprentissage le plus connu et le plus utilisé pour estimer les valeurs des poids est l'algorithme de la rétropropagation du gradient. La règle générale pour la mise à jour des poids est la suivante :

$$\Delta W_{ij} = \eta \delta_j O_i \quad (3.10)$$

Où η est un nombre positif (appelé taux d'apprentissage), qui détermine la taille des pas dans la recherche de la descente de la pente. Une valeur élevée permet à la propagation rétrogressive de se déplacer plus rapidement vers la configuration de poids cible, mais elle augmente également la probabilité qu'elle n'atteigne jamais cette cible. O_i est la sortie calculée par le neurone i . $\delta_j = O_j(1 - O_j)(T_j - O_j)$ pour les neurones de sortie, où T_j est la sortie souhaitée pour le neurone j et $\delta_j = O_j(1 - O_j) \sum_k \delta_k W_{kj}$ pour les neurones internes (cachés).

Lors de la classification, le signal des unités d'entrée se propage à travers le réseau pour déterminer les valeurs d'activation de toutes les unités de sortie. Chaque unité d'entrée a une valeur d'activation qui représente une caractéristique externe au réseau. Ensuite, chaque unité d'entrée envoie sa valeur d'activation à chacune des unités cachées auxquelles elle est connectée. Chacune de ces unités cachées calcule sa propre valeur d'activation et ce signal est ensuite transmis aux unités de sortie. La valeur d'activation de chaque unité de réception est calculée selon une fonction d'activation simple. La fonction additionne les contributions de toutes les unités d'envoi, où la contribution d'une unité est définie comme le poids de la connexion entre les unités d'envoi et de réception multipliée par la valeur d'activation de l'unité d'envoi. Cette somme est généralement ensuite modifiée, par exemple en ajustant la somme d'activation à une valeur comprise entre 0 et 1 et/ou en fixant la valeur d'activation à zéro, à moins qu'un seuil ne soit atteint pour cette somme.

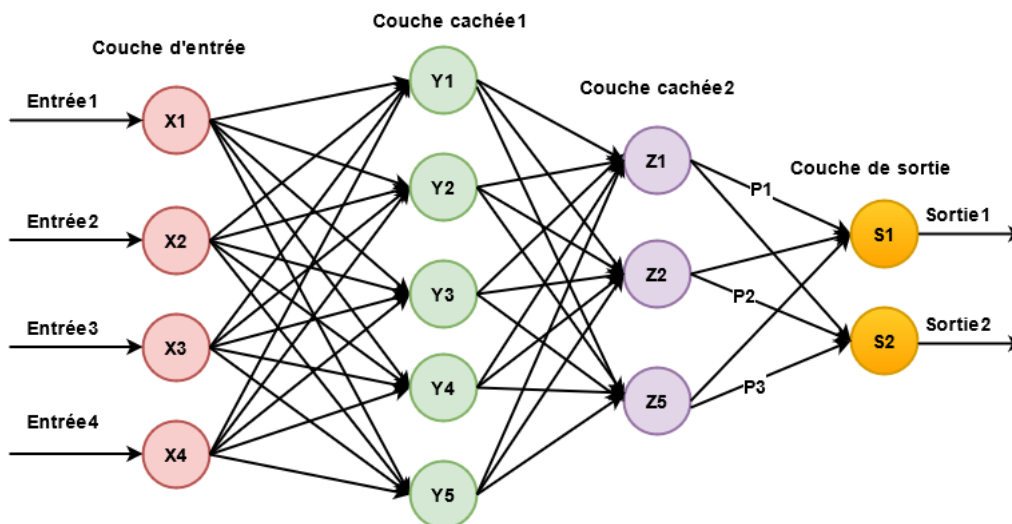


FIGURE 3.10 – Structure d'un réseau de neurones à propagation avant.

3.8.5.2 Réseau neuronal convolutif

La structure RNC a été proposée pour la première fois en 1988 par Fukushima [132], qui est l'un des réseaux d'apprentissage à convolution en profondeur les plus courants et les plus efficaces. Fukushima a conçu l'architecture de RNC basée sur l'approche LeNet à convolution. Au passé, RNC n'était utilisé que pour la reconnaissance de l'écri-

ture manuscrite et la reconnaissance d'images. Actuellement, cette architecture de réseau est également utilisée pour des tâches de classification de texte (y compris l'analyse des sentiments). Par conséquent, l'utilisation de RNC dans tous ces domaines présentés précédemment a obtenu de bons résultats en termes de taux de classification et de temps d'exécution selon multiples travaux de la littérature. Le secret de ces grands succès est la structure de RNC, qui est conçue pour devenir similaire au cortex visuel du chat. En effet, le cortex visuel du chat est composé d'un arrangement compliqué de neurones. Ces neurones sont chargés de couvrir de petites sous-zones de la zone visuelle, appelée zones réceptives. Ensuite, les zones réceptives sont carrelées pour détecter la zone visuelle globale [133]. Par conséquent, les zones réceptives sont considérées comme des filtres dans le modèle d'apprentissage en profondeur RNC. En résumé, l'objectif principal des RNC est d'innover une solution pour diminuer le nombre total de paramètres et construire un réseau neuronal plus profond avec moins de paramètres. Figure 3.11 illustre la structure globale de RNC, qui comprend trois couches fondamentales : la couche de convolution, la couche de pooling et la couche entièrement connectée

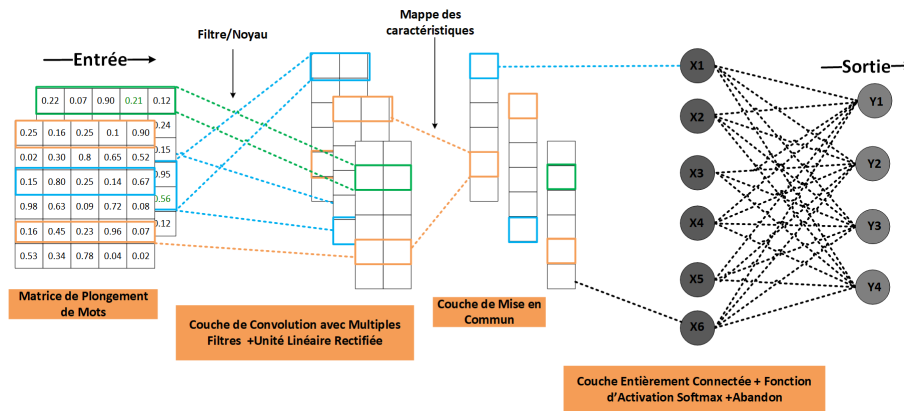


FIGURE 3.11 – Réseau neuronal convolutif.

Comme nous l'avons dit précédemment et comme précisé dans la figure 3.11, l'architecture de RNC est constituée de trois couches principales, contrairement aux réseaux neuronaux artificiels classiques. Ces couches sont la couche de convolution, la couche de pooling et la couche entièrement connectée, comme décrit ci-dessous :

Couche de convolution : est le bloc essentiel de RNC et est toujours la première couche dans la structure globale de RNC. L'objectif principal de cette couche est de détecter et de capturer les caractéristiques d'une matrice obtenue en appliquant l'une des méthodes plongement lexical de mots sur la phrase d'entrée donnée. La couche de convolution utilise un filtre glissant sur la matrice de plongement lexical et produit une caractéristique convoluée. Plusieurs filtres sont appliqués sur la matrice de plongement pour obtenir plusieurs cartes de caractéristiques. Ces cartes de caractéristiques obtenues sont activées (c'est-à-dire transformer les cartes de caractéristiques linéaires en cartes de caractéristiques non linéaires) à l'aide de la fonction d'activation d'unité linéaire rectifiée (ReLU) au niveau de la tâche intermédiaire qui relie la couche de convolution et la couche de pooling. Enfin, les cartes de caractéristiques non linéaires obtenues sont transmises à la couche de pooling. En résumé, la ReLU est la fonction d'activation la plus populaire utilisée avec la couche de convolution. Il calcule simplement à l'aide de la formule suivante :

$$f(y) = \max(0; y) \quad (3.11)$$

En substance, la fonction d'activation ReLU émet 0 si elle obtient une valeur négative en entrée, et si elle obtient une valeur positive en entrée, la ReLU émettra la même valeur positive [134] comme le montre la figure 3.12.

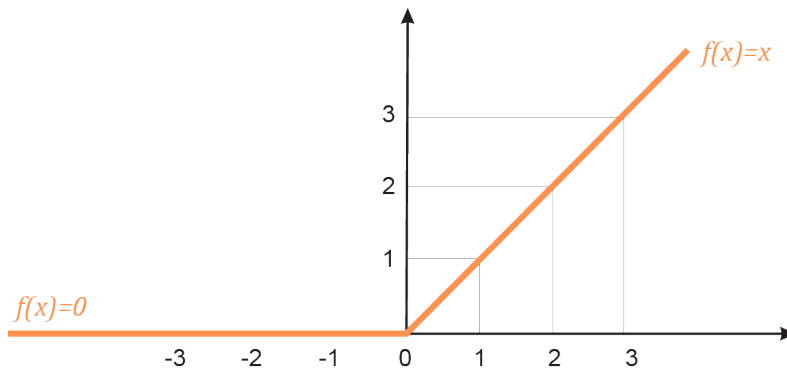


FIGURE 3.12 – Représentation graphique de la fonction d'activation ReLU.

Les avantages de la fonction ReLU sont sa capacité à surmonter le problème du gradient de disparition, sa convergence est plus rapide grâce à sa formule mathématique simple, et son temps d'exécution est relativement court contrairement à d'autres méthodes d'activation telles que tanh et sigmoïde.

Couche de pooling : Après la convolution de la matrice de plongement lexical avec plusieurs filtres dans la première étape (couche de convolution), la deuxième phase est l'application de la couche de pooling pour réduire la dimensionnalité des cartes de caractéristiques obtenues dans la première étape. De ce fait, le nombre total de paramètres RNC est réduit, le coût de calcul est diminué et le problème du surapprentissage est limité. Deux fonctions de pooling populaires sont les opérations de pooling moyenne et maximale. La méthode de *pooling moyenne* détermine la caractéristique de pooling comme la moyenne de toutes les valeurs dans la carte des caractéristiques convolutées. La méthode de pooling maximal sélectionne l'élément maximum dans la carte des caractéristiques convolutées comme caractéristique de pooling et rejette le reste. Ces deux opérations sont décrites dans la figure 3.13. Généralement, la couche de pooling transforme les cartes de caractéristiques convolutées en une seule colonne, qui est ensuite transmise à une couche entièrement connectée [135].

Couche entièrement connectée : est également appelée couche dense, qui sert à calculer la valeur de sortie de chaque valeur d'entrée à partir de la colonne unique obtenue d'une couche de pooling dans la phase précédente. Nous pouvons résumer sa fonctionnalité comme un processus linéaire dans lequel chaque entrée est liée à toutes les sorties par un poids différent [135]. La couche entièrement connectée calcule les valeurs de sortie en utilisant la formule (3.12) comme suit :

$$Sv = f(Wm * Cp + Bi) \quad (3.12)$$

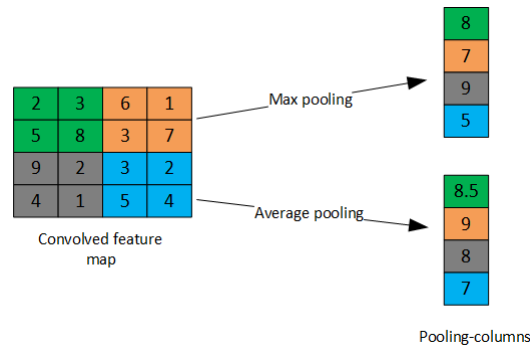


FIGURE 3.13 – Illustration des fonctions de pooling-moyenne et de pooling-maximale.

Où Sv est la valeur de sortie calculée, f est la méthode d'activation softmax, Wm est la matrice de poids utilisée, Cp est la colonne unique obtenue à partir de la couche de pooling, et Bi est le biais.

La fonction d'activation Softmax (ou une fonction exponentielle normalisée) est appliquée dans le processus d'apprentissage intermédiaire entre les couches entièrement connectées et la couche de sortie. Cette fonction d'activation convertit les valeurs numériques reçues de la couche entièrement connectée en valeurs probables, qui sont dans l'intervalle $[0,1]$, et la somme de ces valeurs probables est égale à 1, comme le montre la figure 3.14.

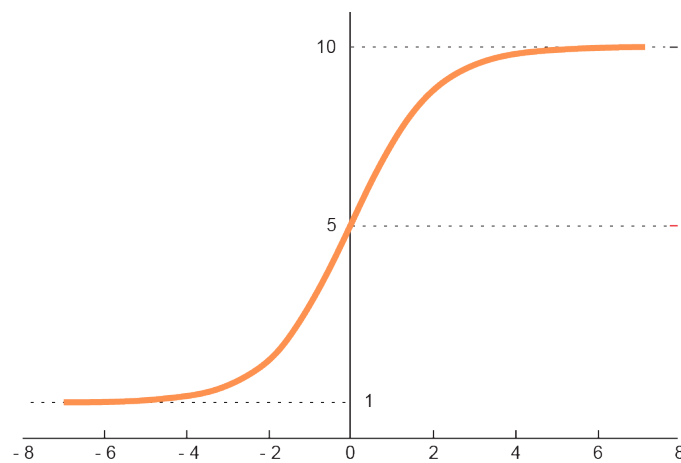


FIGURE 3.14 – Représentation graphique de la fonction d'activation du softmax.

Dans cette thèse, nous avons appliqué la fonction Softmax sur le vecteur reçu de i valeurs réelles à travers la dernière couche cachée du RNPA pour calculer les valeurs de sortie en utilisant la fonction d'activation Softmax qui est désignée comme suit :

$$f(y) = \frac{e_i^y}{\sum_{k=1}^K e_i^y} \quad (3.13)$$

3.8.6 Système flou de Mamdani

SFM est l'un des systèmes d'inférence floue (FIS) les plus populaires, qui sont également appelés systèmes flous à base de règles. L'objectif fondamental de ce type de systèmes est le processus de la prise de décision. Les SIFs sont considérés comme une nouvelle version des modèles classiques basés sur des règles. En outre, dans les SIFs, la valeur de la variable prend une valeur numérique comprise entre 0 et 1. En contraste, la valeur de la variable prend soit "0" soit "1" dans le modèle classique basé sur des règles. Par conséquent, les SIFs servent à convertir un problème de prise de décision en noir et blanc en un problème de prise de décision en gris. Le SFM consiste en un processus de fuzzification, une unité de base de connaissances (base de règles + base de données), un processus de décision et enfin un processus de défuzzification.

Processus de fuzzification : qui convertit l'ensemble croustillant en ensemble flou à l'aide d'une fonction d'appartenance triangulaire, trapézoïdale ou gaussienne. En d'autres termes, chaque valeur de l'ensemble net est transformée en valeur linguistique.

Unité de base de connaissances : la base de connaissances se compose de deux parties, à savoir une base de règles et une base de données, la base de règles comprenant un ensemble de règles flou SI-ALORS, et la base de données comprenant les paramètres, des informations sur la fonction d'appartenance employée et une définition simple de l'ensemble flou.

Unité de décision ou Processus d'inférence : il effectue les étapes d'inférence sur les règles floues SI-ALORS stockées dans la base de règles et obtient une sortie floue.

Processus de défuzzification : qui transforme les sorties floues du mécanisme d'inférence en une sortie nette en utilisant l'une de ces méthodes : la *Méthode du Centre de Gravité (CG)*, *Approche par Bisecteur de Zone (BZ)*, *Procédure du Premier du Maximum (PM)*, *Technique du Dernier du Maximum (DM)*, *Approche de la Moyenne du Maximum (MM)*, *Procédure Moyenne Pondérée (MP)*, et la *Méthode du Centre des Sommes (CS)*. L'architecture globale d'un SIF est illustrée dans la figure 3.15.

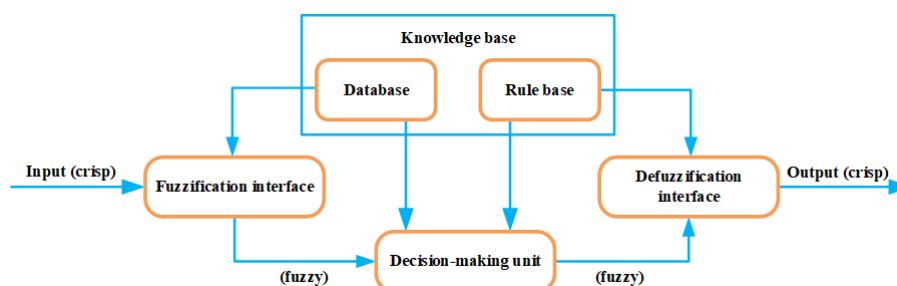


FIGURE 3.15 – Architecture globale d'un systèmes d'inférence floue.

3.9 Mesures d'évaluation pour la classification

Afin d'évaluer la performance d'une technique de classification, un certain nombre de mesures d'évaluation peuvent être utilisées. Dix mesures d'évaluation ont été utilisées pour apprécier la performance des processus de classification utilisés à des fins de synthèse, comme décrit dans cette thèse, à savoir : Taux positif vrai (TPR), Taux négatif vrai (TNR) ou spécificité, Taux positif faux (FPR), Taux négatif faux (FNR), Taux d'erreur (TR), Précision (PR), Taux de classification ou exactitude (TC), Statistique Kappa (KS), Score F1 (FS) et Temps d'exécution (TE). Ces mesures de performance sont calculées à l'aide de la matrice de confusion pour la classification binaire ou multiclasse, comme indiquées dans les figures 3.16 et 3.17.

Les abréviations FN (Faux Négatif), VP (Vrai Positif), VN (Vrai Négatif), et FP (Faux Positif) dans la matrice de confusion simple pour la classification binaire telle que décrite dans la figure 3.16 sont définies comme suit [136] :

- . **Vrai Positif (VP)** : Nombre de cas qui sont effectivement positifs et dont on prévoit qu'ils seront positifs
- . **Faux Négatif (FN)** : Nombre de cas qui sont effectivement positifs et dont on prévoit qu'ils seront négatifs
- . **Vrai Négatif (VN)** : Nombre de cas qui sont effectivement négatif et dont on prévoit qu'ils seront négatif
- . **Faux Positif (FP)** : Nombre de cas qui sont effectivement négatifs et dont on prévoit qu'ils seront positifs

		Predicted value	
		Positive	Negative
Actual value	Positive	TP	FP
	Negative	FN	TN

FIGURE 3.16 – Matrice de confusion pour un problème de classification binaire.

Les paramètres d'évaluation a savoir : Rappel, Spécificité, Taux de faux positifs, Taux de faux négatifs, Taux d'erreurs, Taux de précision, Taux de classification, Statistique Kappa et Score F1 sont calculés dans le cas d'une classification binaire à l'aide de la matrice de confusion décrite dans la figure 3.16 comme suit :

Rappel mesure la performance et l'efficacité d'un classificateur pour prédire le nombre de cas qui ont un label de classe positif. Cette mesure est calculée à l'aide de l'équation (3.14) , où VP est le nombre d'occurrences positif prévu et VP+FN est le nombre total d'occurrences positives dans l'ensemble de données.

$$Rappel = \frac{vp}{vp + fn} \quad (3.14)$$

Spécificité mesure l'efficacité d'un classificateur pour identifier le nombre de cas qui ont des étiquettes de classe négatives. Cette mesure est calculée à l'aide de l'équation (3.15). Où VN correspond au nombre d'échantillons qui ont des étiquettes de classe négatives, et VN+FP est le nombre total d'instances qui sont des étiquettes de classe négatives dans l'ensemble de données utilisé.

$$Spécificité = \frac{vn}{vn + fp} \quad (3.15)$$

Taux de faux positifs est le taux permettant de détecter l'inefficacité d'un classificateur et de mesurer le taux d'erreurs de classification en calculant le nombre de cas, qui sont en fait négatif mais que le classificateur a prédit pour être positif. Le taux de faux positifs est calculé à l'aide de l'équation suivante (3.16)

$$TauxdeFauxPositif = \frac{fp}{fp + vn} \quad (3.16)$$

Taux de faux négatif est le taux permettant de détecter l'inefficacité d'un classificateur et de mesurer le taux d'erreurs de classification en calculant le nombre de cas, qui sont en fait positifs mais que le classificateur a prédit pour être négatifs. Cette mesure d'évaluation est calculée à l'aide de l'équation (3.17)

$$TauxdeFauxNégatif = \frac{fn}{fn + vp} \quad (3.17)$$

La métrique du taux d'erreur sert à mesurer le taux d'erreurs de classification, c'est-à-dire que cette métrique calcule le nombre de cas erronée sur l'ensemble des cas de l'ensemble de données utilisé. Fondamentalement, son objectif est de mesurer la capacité du classificateur à limiter les fausses classifications. Le taux d'erreur est défini par l'équation (4.2) présente.

$$Error = \frac{fp + fn}{vp + fn + vn + fp} \quad (3.18)$$

Précision mesure le nombre d'échantillons récupérés en tant qu'étiquettes de classe positives qui sont, en fait, positives. Le taux de précision est utile pour évaluer les classificateurs fragiles, qui sont appliqués pour classer toutes les instances de l'ensemble de données utilisé. Cette métrique d'évaluation est déterminée comme le décrit l'équation (4.3).

$$Précision = \frac{vp}{vp + fp} \quad (3.19)$$

Taux de classification est une mesure globale permettant d'estimer l'efficacité et l'exactitude des classificateurs d'apprentissage. Taux de classification est calculé à l'aide d'un ensemble de tests qui est détaché de l'ensemble d'apprentissage. Ce taux est mesuré à l'aide de l'équation (4.4). Où vp+vn est tous les exemples classifiés correctement par le

classificateur et $Vp+fn +vn+Fp$ sont tous les instances dans l'ensemble de données

$$TauxdeClassification = \frac{vp + vn}{vp + fn + vn + fp} \quad (3.20)$$

Le Score-F1 ou la mesure F est la moyenne harmonique entre le taux de précision et le taux de rappel. La valeur du Score-F1 est comprise entre 0 et 1. Elle mesure la précision et la robustesse du classificateur utilisé. Si le Score-F1 augmente, la performance du classificateur utilisé sera meilleure. En d'autres termes, pour obtenir une performance extrêmement précise, la précision doit être supérieure et le rappel doit être inférieur. Cette mesure est calculée à l'aide de l'équation (3.21).

$$Score - F1 = \frac{2 * Précision * Rappel}{Précision + Rappel} \quad (3.21)$$

Statistique Kappa est un critère de performance qui compare une précision mesurée et une précision prévue (hasard). Elle est utilisée non seulement pour estimer un classificateur, mais aussi pour inspecter les classificateurs entre eux. Le statistique kappa est calculé à l'aide de l'équation (4.5).

$$StatistiqueKappa = \frac{P_0 - P_e}{1 - P_e} \quad (3.22)$$

Where : $P_0 = \frac{tp+tn}{100}$.
and $P_e = [\frac{tp+fn}{100} * \frac{tp+fp}{100}] + [\frac{fp+tn}{100} * \frac{fn+tn}{100}]$

Les paramètres d'évaluation a savoir : Rappel, Spécificité, Taux de faux positifs, Taux de faux négatifs, Taux d'erreurs, Taux de précision, Taux de classification, Statistique Kappa et Score F1 sont calculés dans le cas d'une classification multi-classes à l'aide de la matrice de confusion illustrée dans la figure 3.17. La première étape pour calculer ces métriques est de calculer les mesures VN, FN, VP, FP, comme décrit dans la figure 3.16 pour chaque classe dans la matrice de confusion multi-classe. Par exemple, pour la classe Positive, les valeurs de ces mesures sont déterminées comme suit : $VP = 5$; $VN = (4 + 1 + 6 + 8) = 19$; $FN = (7 + 3) = 10$; $FP = (2 + 9) = 11$. Dans le cas d'une classe négative, ces métriques seront $VP = 4$; $VN = (5 + 9 + 3 + 8) = 25$; $FN = (2 + 6) = 8$; $FP = (1 + 7) = 8$. et dans le cas de la classe Neurale, ces mesures sont calculées comme suit : $VP = 8$; $VN = (5 + 2 + 4 + 7) = 18$; $FN = (3 + 6) = 9$; $FP = (1 + 9) = 10$ après le calcul de ces métriques, nous calculons les métriques d'évaluation précédemment comme suit.

Le calcul du rappel pour la classification multi-classes est effectué selon l'équation (3.23). Où vp_i et le nombre d'instances prédites sont étiquetées classe i , i est l'indice de la classe, l est le nombre total d'étiquettes de classe, et $vp_i + fn_i$ est le nombre total d'instances étiquetées label de classe i dans l'ensemble de données utilisé.

$$Rappel = \frac{\sum_{i=1}^l \frac{vp_i}{vp_i + fn_i}}{l} \quad (3.23)$$

La métrique de spécificité est mesurée pour la classification multi-classes à l'aide de l'équa-

		Predicted value		
		Positive	Negative	Neural
Actual value	Positive	5	2	9
	Negative	7	4	1
	Neural	3	6	8

FIGURE 3.17 – Matrice de confusion pour un problème de classification multi-classes.

tion . Où vn_i est le nombre d'instances prédites qui ne sont pas étiquetées classe i , i est l'indice de la classe, l est le nombre total d'étiquettes de classe, et $vn_i + fp_i$ est le nombre total d'instances qui ne sont pas étiquetées classe i dans l'ensemble de données utilisé.

$$spécificité = \frac{\sum_{i=1}^l \frac{vn_i}{vn_i + fp_i}}{l} \quad (3.24)$$

La mesure du taux de faux positif est calculée comme décrit par l'équation (3.25). Où fp_i est le nombre d'instances qui ne sont pas réellement étiquetées classe i mais dont le classificateur a prédit qu'elles seraient étiquetées classe i , i est l'indice de la classe, l est le nombre total d'étiquettes de classe, et $vn_i + fp_i$ est le nombre total d'instances qui ne sont pas étiquetées classe i dans l'ensemble de données utilisé.

$$TauxdeFauxPositif = \frac{\sum_{i=1}^l \frac{fp_i}{fp_i + vn_i}}{l} \quad (3.25)$$

Le critère d'évaluation du taux de faux négatif est calculé à l'aide de l'équation (3.26). Où fn_i est le nombre d'instances qui sont effectivement étiquetées classe i mais le classificateur a prédit qu'elles ne l'étaient pas, i est l'indice de la classe, l est le nombre total d'étiquettes de classe, et $fn_i + vp_i$ est le nombre total d'instances effectivement étiquetées classe i dans l'ensemble de données utilisé.

$$TauxdeFauxNégatif = \frac{\sum_{i=1}^l \frac{fn_i}{fn_i + vp_i}}{l} \quad (3.26)$$

Le taux d'erreur pour la classification multi-classes est mesuré à l'aide de l'équation (3.27). Où $fp_i + fn_i$ est le nombre de toutes les instances qui sont prévues de manière incorrecte, i est l'indice de la classe, l est le nombre total d'étiquettes de classe, et $fn_i + vp_i + fp_i + vn_i$ est le nombre total d'instances dans l'ensemble de données utilisé.

$$Tauxd'erreur = \frac{\sum_{i=1}^l \frac{fp_i + fn_i}{vp_i + fn_i + vn_i + fp_i}}{l} \quad (3.27)$$

La mesure de précision est calculée dans le cas d'une classification multi-classes, comme

l'illustre l'équation (3.28). Où vp_i est le nombre d'instances qui sont effectivement étiquetées la classe i et prédites correctement par le classificateur utilisé, i est l'indice de la classe, l est le nombre total d'étiquettes de classe, et $vp_i + fp_i$ est le nombre total d'instances étiquetées la classe i dans l'ensemble de données utilisé.

$$Précision = \frac{\sum_{i=1}^l \frac{vp_i}{vp_i + fp_i}}{l} \quad (3.28)$$

La mesure du taux de classification dans le cas d'une classification multi-classes est calculée selon l'équation (3.29). Où $vp_i + vn_i$ est le nombre de toutes les instances qui sont prédites correctement, i est l'index de la classe, l est le nombre total d'étiquettes de classe, et $fn_i + vp_i + fp_i + vn_i$ est le nombre total d'instances dans l'ensemble de données utilisé.

$$TauxdeClassification = \frac{\sum_{i=1}^l \frac{tp_i + tn_i}{tp_i + fn_i + tn_i + fp_i}}{l} \quad (3.29)$$

Une autre mesure, le Score-F1, est utilisée pour intégrer la précision et le rappel dans une seule mesure. La valeur de cette mesure est comprise entre 0 et 1 comme nous l'avons présenté précédemment, et si le classificateur évalué classe correctement toutes les instances, cette mesure prendra la valeur 1. Le score-F1 est calculé à l'aide de l'équation (3.30) pour la classification multi-classes.

$$Score - F1 = \frac{\sum_{i=1}^l \frac{2 * Précision_i * Rappel_i}{Précision_i + Rappel_i}}{l} \quad (3.30)$$

La statistique Kappa est calculée à l'aide de l'équation (3.31) dans le cas d'une classification multi-classes.

$$StatistiqueKappa = \frac{\sum_{i=1}^l \frac{P_{0i} - P_{ei}}{1 - P_{ei}}}{l} \quad (3.31)$$

Où : $P_0 = \frac{tp + tn}{100}$.

et $P_e = \left[\frac{tp + fn}{100} * \frac{tp + fp}{100} \right] + \left[\frac{fp + tn}{100} * \frac{fn + tn}{100} \right]$

3.10 Conclusion

La fouille de données constitue l'étape clé du processus de découverte de connaissances, étape sur laquelle bien des acteurs se focalisent d'emblé au détriment des autres étapes qui sont toutes aussi importantes. Depuis son apparition, la fouille de données a connu un essor fulgurant. À la faveur du développement technologique et des besoins du marché, elle s'est trouvée utilisée dans bien des domaines (marketing, télécommunication, détection de fraude, domaine manufacturier, etc.). La fouille de données en elle-même consiste en l'utilisation des techniques et algorithmes afin d'extraire des connaissances implicites et potentiellement utiles, pouvant servir de support au processus de décision de l'entreprise. Son objectif principal est d'être soit prédictive, soit descriptive. Prédictive dans le sens de prédire la valeur future des variables étudiées et descriptive dans le sens de la production de modèle expliquant les données sous étude. Elle est composée d'un grand nombre de techniques et algorithmes que l'on peut caractériser selon plusieurs points de vue : supervisé vs non-supervisé, prédictif vs descriptif, et transparent vs opaques.

Ce chapitre est consacré à la présentation des notions fondamentales de la fouille de données, en particulier nous avons présenté les différents types de fouille de données à savoir les méthodes de découverte et les méthodes de vérification. Ensuite, nous avons défini les méthodes supervisées qui sont communément appelées méthodes d'apprentissage de prédiction. Puis, nous avons décrit la définition et tous les différents types de classification en détail à savoir la classification binaire et la classification multiclassées. En outre, nous avons présenté les différentes tâches de prétraitement à savoir "Supprimer numéros, hashtag, urls, et nom d'utilisateurs", "Éliminer ponctuations, espaces vides et les caractères spéciaux", "Mise en minuscules", "Remplacer les mots allongés", "Supprimer les mots vides", "Corriger les contractions", "Gérer les effets de négation", "Racinisation", "Lemmatisation", et "Tokenisation". Aussi ce chapitre décrit l'extraction de caractéristiques à savoir Sac de Mots, N-gramme, GloVe, Word2vec, FastText, et TF-IDF. De plus, nous avons présenté les différentes méthodes de sélection de caractéristiques à savoir gain d'information, rapport de gain d'information, indice de gini, et khi carré. Nous avons discuté également les différents algorithmes de classification à savoir Naïve bayésienne, Arbres de décision (C4.5, ID3, CART, et forêts d'arbres décisionnels), Méthode des k plus proches voisins, Machines à vecteurs de support, Réseau de neurones à propagation avant, Réseau de neurones convolutifs et le système flou de Mamdani. Also, les mesures d'évaluation pour la classification sont décrites dans ce chapitre. D'une manière générale, l'objectif principal de ce chapitre est de mieux comprendre la définition de la fouille de données, les différentes techniques de la fouille de données notamment les techniques de classification.

Dans le prochain chapitre, nous allons aborder et détailler notre première contribution dans cette thèse qui vise à améliorer l'arbre de décision ID3 et de l'utiliser pour classifier les sentiments exprimés dans les tweets.

Extraction d'opinion pour la classification des tweets à l'aide d'un classificateur ID3 amélioré et Hadoop

4.1 Introduction

La fouille d'opinion également connue sous le nom d'analyse de sentiments est un sujet de recherche intense dans le domaine du *traitement automatique du langage naturel* (NLP). Elle vise à classer, étudier, sélectionner et évaluer les opinions, les attitudes, les émotions et les réactions des textes publiés par les utilisateurs sur les plateformes de médias sociaux à l'égard d'entités telles qu'une organisation, un service, un individu, un produit, un sujet, un événement ou un problème [137]. Avec la diffusion de textes générés par les utilisateurs aux réseaux sociaux et les sites de microblogage tels que Youtube, Trip Advisor, Tiktok, Instagram, Twitter, Facebook, Amazon et Whatsapp, l'analyse des sentiments dans les sites web et les réseaux sociaux a acquis une popularité croissante auprès de nombreuses communautés de recherche scientifique et industrielle [138].

La fouille d'opinion est considérée comme étant effectuée sous trois aspects différents, à savoir l'aspect caractéristiques, l'aspect documents et l'aspect phrases [139]. Au niveau des caractéristiques, nous classons les phrases/documents comme négatifs, positifs ou neutres sur la base des mots d'opinions extraites de ces phrases/documents [140, 141]. L'aspect documents est une tâche de classification des sentiments qui s'efforce de classer le document entier en polarité négative, neutre ou positive [141]. L'aspect phrases est l'opération d'extraction de sentiments qui cherche à calculer la polarité sentimentale de la phrase analysée. En d'autres termes, la phrase sera classée comme négative, positive ou neutre en fonction de l'opinion exprimée dans cette phrase [142]. Dans cette proposition, la classification des sentiments au niveau de l'aspect phrases a été prise en considération.

La fouille d'opinion est employée dans plusieurs autres applications comme la prédiction des tendances ou des comportements futurs des clients [143], la proposition de nouvelles stratégies de marketing futures [144], l'amélioration des techniques d'apprentissage en ligne [145], et les pronostics boursiers [146]. L'une des applications les plus importantes de la fouille d'opinion est qu'elle fournit aux partis gouvernementaux ou aux partis politiques de nombreuses informations qui sont analysées pour deviner les chances

de leur victoire lors des prochaines élections politiques. Elle permet également de déterminer si les citoyens sont satisfaits de leurs politiques, comme cela a été le cas lors des élections américaines de 2020.

Twitter est le réseau social le plus commun en temps réel pour exprimer les opinions et les idées d'un individu ou d'un groupe d'individus sur un sujet spécifique par le biais de messages courts de 280 caractères appelés tweets. En raison de la limitation en termes de longueur d'un tweet, les utilisateurs ont tendance à utiliser d'argot, de langage informel, de nombreuses abréviations, des URL, des formes courtes et une utilisation intensive d'émoticônes, ainsi que des expressions spécifiques à Twitter comme des hashtags et des mentions d'utilisateurs [138], ce qui pose des défis considérables pour la classification des sentiments sur Twitter. Par conséquent, il est obligatoire d'appliquer des mécanismes intelligents pour capturer des connaissances utiles à partir des tweets. Dans cette proposition, nous utilisons une technique supervisée parallèle pour traiter les tweets de manière à surmonter les difficultés mentionnées ci-dessus.

Le nombre d'utilisateurs de Twitter a atteint 330 millions d'utilisateurs actifs mensuels, 145 millions d'utilisateurs actifs quotidiens en 2020, et un demi-milliard de tweets sont envoyés chaque jour, ce qui équivaut à 5787 tweets par seconde [147]. En raison de ces données massives générées chaque jour sur la plate-forme Twitter, les chercheurs scientifiques ont considéré Twitter comme une instance de Big Data [148]. Parce qu'il présente les mêmes caractéristiques de Big Data, à savoir (i) *Volume* : le nombre moyen de tweets pour annoncer un incident est d'au moins 1000000 ; (ii) *Variété* : chaque tweet est composé de différents termes (tels que des mots courts, d'argot, des abréviations, des URL et des émoticônes) ; (iii) *Vélocité* : les tweets sur un sujet sont extrêmement dynamiques, par exemple, les nouveaux tweets quotidien sur un événement représentent plus de 500 To sur Twitter. (iv) *Valeur* : la richesse des données cachées dans les tweets générés est un domaine de recherche brûlant pour les chercheurs scientifiques dans le domaine du Big Data et de l'analyse des sentiments, et également un outil solide pour les organisations et les gouvernements pour prendre des décisions ou des stratégies universelles. Dans notre travail, le cadre Hadoop est appliqué pour paralléliser notre contribution améliorée afin de traiter les données massives extraites à partir de Twitter.

La plateforme Twitter facilite les interactions entre les clients et les organisations ou institutions. La liberté d'utilisation du réseau social Twitter offre à ses utilisateurs la possibilité d'écrire des commentaires qui expriment leur opinion sur certaines situations, produits, événements et services. [149]. Ces commentaires sont exprimés principalement en fonction de l'expérience de l'utilisateur avec ces produits ou services qui peuvent être des jugements négatifs, positifs ou neutres sur les produits ou services. L'extraction des opinions des utilisateurs qui ont un impact négatif sur le produit ou le service à partir des commentaires exprimés sur la plateforme Twitter est une tâche essentielle qui aide les organisations propriétaires à améliorer leurs produits ou services, récoltant ainsi plus de profits [150]. Par conséquent, il est important d'évaluer les commentaires des utilisateurs recueillis à partir des plateformes de réseaux sociaux. La fouille d'opinion est un outil efficace pour déterminer la polarité (négative, neutre ou positive) des jugements des utilisateurs sur un service ou un produit via l'analyse des données des microblogages. Dans le cadre de la fouille d'opinion, les sous-opérations de la figure 4.1 sont effectuées en évaluant les commentaires des utilisateurs publiés sur les différentes plateformes de réseaux

sociaux.

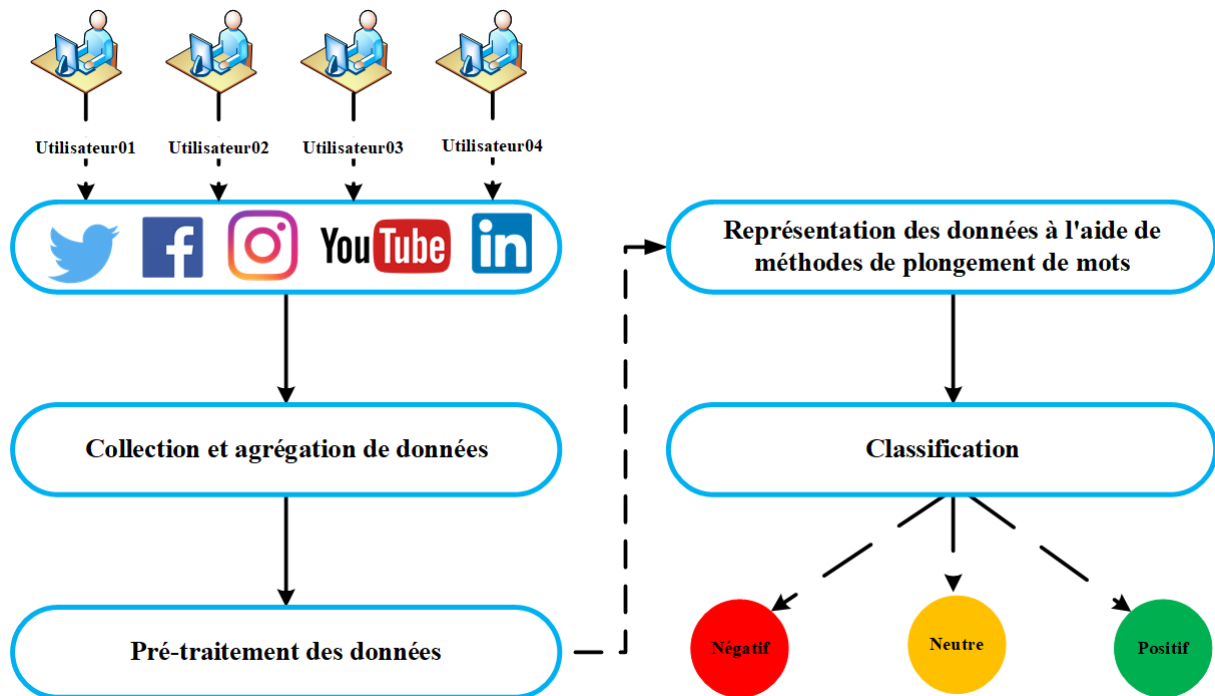


FIGURE 4.1 – Les étapes de base de la fouille d’opinion.

Ces sous-opérations visent à extraire des informations utiles à partir des données des médias sociaux qui indiquent la polarité de ces données analysées. Les techniques d’apprentissage automatique peuvent être utilisées pour révéler les connaissances précieuses qui sont cachées dans ces données bruyantes de médias sociaux créées quotidiennement. Il existe de nombreuses techniques d’apprentissage automatique qui facilitent l’apprentissage telles que *K-Plus Proches Voisins* (KPPV) (en anglais *k-Nearest Neighbors*, K-NN) [151], *Machine à Vecteurs de Support* (SVM) (en anglais *Support Vector Machine*,) [152], *ID3 arbre de décision* [153], *Régression Logistique* (LR) (en anglais *Logistic Regression*, LR) [154], *Bayésien Naïf* (NB) (en anglais *Naive Bayes*, NB) [155], ou *Forêt Aléatoire* (FA) (en anglais *Random Forest*, RF) [156], etc., Ces algorithmes ont été largement utilisés dans divers domaines tels que la banque [157], la détection de la cyberhaine [158], la bio-informatique [159], la détection du langage abusif [160], les médias sociaux [161], et l’identification de la cyberintimidation [162].

Les techniques d’apprentissage automatique se divisent en deux types, à savoir les algorithmes d’apprentissage supervisé et non supervisé [141]. L’apprentissage supervisé utilise un ensemble de données étiquetées pour construire un modèle de classification, qui est ensuite utilisé pour prévoir les étiquettes d’un ensemble de données de test non étiquetées. D’autre part, les algorithmes d’apprentissage non supervisé, comme le regroupement, visent à répartir les données non étiquetées en différents groupes en calculant la similarité. Les algorithmes d’apprentissage supervisé ont été largement utilisés pour la fouille d’opinion [163, 164]. Dans ce travail, nous utiliserons l’algorithme d’apprentissage supervisé ID3 amélioré, car il est exécuté uniquement sur des données étiquetées. Ainsi, notre travail décrit dans ce chapitre réalise la fouille d’opinion pour le contenu en langue Anglaise en utilisant notre arbre de décision ID3 amélioré mis en œuvre sous Hadoop.

Les principales propositions de notre contribution peuvent être résumées comme suit : comme nous l'avons dit précédemment, notre travail classe les tweets collectés en trois étiquettes de classe : négative, positive ou neutre. Les tweets sont généralement sous forme de texte non structuré, composé d'argot, d'abréviations, de mots courts, d'URL, de mots vides, etc. Donc, la première étape de notre travail consiste à appliquer des techniques de pré-traitement sur les tweets afin de supprimer les informations indésirables et bruyantes pour une analyse ultérieure. Après l'étape de pré-traitement, ces tweets seront passés par une opération de représentation dans laquelle les tweets sont transformés en valeurs numériques, qui prennent la forme de matrice. Ensuite, nous minimisons la dimensionnalité de la matrice obtenue dans la deuxième étape en employant plusieurs sélecteurs de caractéristiques. La matrice réduite dans la troisième étape sera l'entrée donnée de notre ID3 [165] amélioré pour la classification des tweets. Enfin, nous parallélisons toutes les étapes précédentes en utilisant l'écosystème Hadoop, avec son modèle de programmation MapReduce et son système de fichiers distribués HDFS. Divers paramètres sont ensuite utilisés pour évaluer les performances de notre proposition. Par conséquent, notre contribution a principalement consisté en sept aspects :

1. Les techniques de pré-traitement des données, comme l'opération de lemmatisation, le processus d'enracinement et la technique de négation, sont appliquées pour améliorer la qualité des tweets en éliminant le bruit et les données indésirables.
2. Les approches de représentation de données telles que *TF-IDF*, *FastText*, plongement de mots (*Word2vec*, *GloVe*), *Bag-Of-Words*, *N-gram* sont appliquées sur le jeu de données utilisé des tweets pour convertir les données textuelles (tweets) en données numériques.
3. Les sélecteurs de caractéristiques tels que *l'Indice de Gini*, le *Gain d'Information*, le *Khi-Carré* et le *Rapport de Gain* sont utilisés pour réduire la dimension élevée des caractéristiques.
4. Notre amélioration de l'algorithme ID3 est utilisée pour classer les tweets analysés comme étant de classe neutre, négative ou positive.
5. Le modèle que nous avons introduit est mis en œuvre de manière parallèle à l'aide du cadre Hadoop afin d'éviter le problème du temps d'exécution long et d'améliorer la capacité de notre contribution à traiter les jeux de données massifs.
6. Les performances de notre méthode proposée dans ce travail ont été comparés avec d'autres méthodes sélectionnées à partir de la littérature.
7. Notre modèle proposé surpasse les approches sélectionnées par une marge significative en termes de TC, TR, TE, TPV, TNF, TNV, TPF, PR, SK, et SF, qui sont expliqués précédemment dans la section 3.9 du chapitre 3.

Le reste de ce chapitre est construit comme suit : les précédentes recherches dans la littérature sont décrites dans la section 4.2. La section 4.3 présente notre approche que nous avons développée en détail. Dans la section 4.4, nous présentons la parallélisation de notre approche proposée en utilisant le cadre Hadoop. La configuration expérimentale et les résultats obtenus sont discutés dans la section 4.5. Enfin, la section 4.6 récapitule la méthode suggérée et fait des recommandations pour des travaux futurs.

4.2 Travaux connexes

Une discussion approfondie a été menée sur la fouille d'opinion en différentes langues par l'utilisation des algorithmes d'apprentissage automatique, dont certains sont présentés ci-dessous :

Voici quelques travaux qui emploient différentes techniques d'apprentissage automatique pour effectuer la fouille d'opinion en diverses langues comme dans les articles : Sharma *et al.* [164]; Patil *et al.* [166]; Shein *et al.* [167]; Gamallo *et al.* [168]; Anjaria *et al.* [169]; Duwairi *et al.* [170]; Soni *et al.* [171]; Ngoc *et al.* [172]. Les auteurs de l'article [166] ont développé une nouvelle technique hybride qui intègre la méthode de vectorisation TF-IDF avec l'algorithme SVM pour déterminer la polarité des données textuelles analysées. Ce travail a prouvé que le SVM pouvait extraire un espace de caractéristiques de haute dimension à partir de données textuelles, ce qui annule le besoin d'autres techniques de sélection de caractéristiques. La contribution suggérée dans l'article [167] intègre une ontologie formelle d'analyse de concepts et le SVM pour étiqueter les commentaires collectés sur certains logiciels comme étant négatifs, neutres ou positifs.

Il existe de nombreux travaux concernant l'algorithme de l'arbre de décision ID3 et leurs améliorations comme décrit dans les articles : Yu-xun *et al.* [173]; Chai *et al.* [174]; Elyassami *et al.* [175]; Zou *et al.* [176]; Srinivasan *et al.* [177]; Chen *et al.* [178]; Ding *et al.* [179]; Zhu *et al.* [180]; Kaewrod *et al.* [181]; Rajeshkanna *et al.* [182]. Dans l'article [175], les auteurs ont appliqué l'algorithme ID3 flou pour la prédiction du prix des logiciels. Leur contribution est mis en œuvre en combinant les concepts de base de l'algorithme ID3 et les principes de la théorie des ensembles flous, ce qui donne au modèle la capacité de traiter des données incertaines et ambiguës pour améliorer considérablement le taux de classification. Ainsi Zou *et al.* [176] ont appliqué l'algorithme d'arbre de décision ID3 pour construire le modèle de détection de la fraude. Leur utilisation d'ID3 a prouvé que cet algorithme pouvait très bien effectuer la classification des données et qu'il fournissait aux décideurs un ensemble de règles de décision. Srinivasan *et al.* [177] ont conçu la méthode de classification floue rapide pour obtenir de meilleures performances de classification. Ils ont également incorporé les avantages de l'arbre de décision ID3 et de l'algorithme SVM, pour améliorer la précision et obtenir un résultat de classification rapide. Dans ce travail [178], un ID3 amélioré est proposé. Cette nouvelle version de l'ID3 est basée sur une méthode originale de sélection des caractéristiques au lieu du gain d'information utilisé dans l'ID3 traditionnel. Cette nouvelle méthode de sélection des caractéristiques est calculée en utilisant l'importance de l'attribut, la fonction d'association, et en ajoutant le nombre de valeurs de chaque attribut. Ding *et al.* [179] ont conçu un nouvel ID3 amélioré, qui utilise comme mesure de séparation la théorie des ensembles rugueux au lieu du gain d'information dans l'ID3 traditionnel. Cet ID3 amélioré basé sur la théorie des ensembles rugueux devient inefficace dans le cas d'un ensemble de données avec des données manquantes. Zhu *et al.* [180] ont proposé un nouvel ID3 amélioré qui est basé sur la moyenne du gain d'information avec différents paramètres, avec une marge spécifique pour éviter le problème du biais des attributs à valeurs multiples. L'objectif principal du travail [181] est d'élaborer une nouvelle procédure pour alléger la rigueur de l'algorithme ID3 traditionnel en supprimant les exemples mineurs. Rajeshkanna *et al.* [182] ont mis en œuvre la technique ID3 en utilisant divers ensembles de données d'apprentissage UCI et

l'ont également évaluée à l'aide de différentes mesures statistiques.

Les articles de recherche les plus récents et les plus novateurs concernant la fouille d'opinion sont les suivants : Lakshmi *et al.* [183]; Guerreiro *et al.* [184]; Mehta *et al.* [185]; Zhang [186]; Lopez-Chau *et al.* [187]; AitAddi *et al.* [188]; Patel *et al.* [189]; et Wang *et al.* [190] comme présenté dans le tableau 4.1. Dans l'article [183], les auteurs ont proposé une nouvelle contribution qui vise à classer les commentaires en utilisant deux classificateurs et à déterminer lequel des deux est le plus performant. Ces deux classifieurs sont l'arbre de décision et le NB. Guerreiro *et al.* [184] ont conçu une nouvelle approche de fouille d'opinion, qui est appliquée à des commentaires collectés en ligne afin d'extraire les moteurs derrière chaque recommandation explicite. Dans l'article [185], l'auteur a mis en œuvre neuf algorithmes distincts qui sont : *Réseau de Neurones Récurrent à Longue Mémoire à Court Terme* (LSTM), NB, NLP, *Perceptron Multicouche* (MLP), Principe d'entropie maximale, *Réseau Neuronal Convolutif* (CNN), RF, XGBoost, ID3, SVM pour classer les tweets et comparer leur précision sur les données de prétraitement. Le résultat expérimental de cette étude comparative a prouvé que le CNN surpasse les autres approches appliquées en atteignant une précision de 79%. Zhang [186] a mis au point une nouvelle approche qui utilise le TF-IDF comme extracteur de caractéristiques et emploie le Khi carré et l'information mutuelle comme sélecteurs de caractéristiques. Les caractéristiques extraites sont ensuite introduites dans des classificateurs de LR, de SVM linéaire et de NB multinomiaux pour effectuer la classification des sentiments. Les auteurs Lopez-Chau *et al.* [187] ont analysé les ensembles de données collectées sur Twitter à propos du tremblement de terre du 19 septembre 2017, en appliquant des outils de fouille d'opinion et d'apprentissage automatique supervisé. Ils ont construit trois classificateurs pour découvrir le sentiment des tweets qui apparaissent sur le même sujet. Le résultat expérimental a prouvé que le SVM, et NB ont obtenu le meilleur taux de classification des émotions. L'auteur de [188] a proposé une méthode innovante basée sur une structure d'arbre binaire à trois niveaux pour la fouille d'opinion hiérarchique à partir des textes en Arabe multi-classes. Les résultats expérimentaux montrent que la méthode proposée obtient des améliorations considérables par rapport aux autres approches de la littérature. Dans l'article [189], les auteurs ont effectué une analyse des sentiments sur les données Twitter de la coupe du monde de football 2014 qui s'est tenue au Brésil afin d'extraire les émotions des gens partout dans le monde, en utilisant des algorithmes d'apprentissage automatique. Après l'application d'opérations de NLP comme la tokénisation des mots, la lemmatisation, etc., les classificateurs SVM, NB, CNN ont été utilisés pour extraire les émotions à partir de chaque tweet. Le résultat expérimental a prouvé que le NB obtient la meilleure précision qui égale à 88.17%. L'article [190] propose un nouveau modèle pour la fouille d'opinion au niveau caractéristique. Ce modèle utilise le paradigme de l'apprentissage automatique graduel pour effectuer un étiquetage automatique précis. Les résultats expérimentaux ont prouvé que la précision de la technique proposée est considérablement meilleure que celle des algorithmes non supervisés.

TABLE 4.1 – Articles de recherche les plus récents de la fouille d’opinion.

Auteurs	Fouille d’opinion	Langue	Algorithme	Jeu de données	TC	Année	Réf.
Lakshmi <i>et al.</i>	Oui	Anglais	NB+AD	IMDB commentaires sur de films qui contient 25250 commentaires.	65 %	2020	[183]
Guerreiro <i>et al.</i>	Oui	Anglais	RL+ CHAID	Yelp académique qui contient 1112708 commentaires.	63.2 %	2020	[184]
Mehta <i>et al.</i>	Oui	Anglais	NB+SVM	Produits électroniques contenant 1200 tweets.	90 %	2020	[185]
Zhang	Oui	Chinois	NB+ SVM+RL	film court V2 qui contient 480000 commentaires.	68 %	2020	[186]
López-Chau <i>et al.</i>	Oui	Mexicain	NB+SVM	Tremblement de terre du 19 septembre 2017	59 %	2020	[187]
AitAddi <i>et al.</i>	Oui	Arabe	CART	IMDB commentaires sur de films qui contient 35070 commentaires.	74 %	2020	[188]
Patel <i>et al.</i>	Oui	Anglais	NB+SVM+ CNN+FA	Coupe du monde 2014 qui contient 1415958 tweets	88.17 %	2020	[189]
Wang <i>et al.</i>	Oui	Anglais	le paradigme de l’apprentissage automatique graduel	Ensemble de données LAP16, RES16, LAP15 et RES	67.01 %	2020	[190]

4.3 Processus de la détection de polarité d'opinions

Dans les sections suivantes, nous allons présenter les raisons qui nous ont motivé à proposer et à faire évoluer notre démarche. En général, la structure fondamentale de la démarche proposée est constituée de cinq phases comme montré dans la figure 4.2 : la première phase est la collecte de données dans laquelle nous avons choisi deux jeux de données massives afin d'évaluer notre classificateur proposé. La deuxième phase, appelée prétraitement de données, qui sert à supprimer les données indésirables et bruitées afin d'améliorer la qualité de données. La troisième phase est la vectorisation de données, qui a pour but de convertir les tweets en données numériques. La quatrième phase est la sélection des caractéristiques, qui vise à réduire la dimensionnalité des caractéristiques extraites dans la phase précédente. La dernière phase est la phase de classification dans laquelle nous avons appliqué notre ID3 amélioré [165] afin de classer les tweets selon les catégories suivantes : positive, négative ou neutre.

4.3.1 Motivation

La fouille d'opinion est un domaine de recherche devenu populaire depuis une dizaine d'années et connaît un succès grandissant dû à l'abondance de données provenant de réseaux sociaux, notamment celles fournies par Twitter. Son objectif est d'analyser une grande quantité de données afin d'en déduire les différents sentiments qui y sont exprimés. Les sentiments extraits peuvent ensuite faire l'objet de statistiques sur le ressenti général d'une communauté. En outre, la fouille d'opinion s'efforce de concevoir des techniques d'intelligence computationnelles pour détecter, capturer et évaluer les idées et les opinions des gens envers une entité (produit, service, événement, etc.) et ses divers aspects. Ces opinions et ces idées ayant une valeur commerciale importante pour les entreprises et leurs clients. Par exemple, les commentaires écrits par les utilisateurs sur les produits aident les nouveaux utilisateurs à prendre des décisions, comme acheter ce nouveau produit ou non. Ainsi qu'ils sont extrêmement précieuse pour les grandes entreprises/organisations dans la supervision de leurs produits/événements/services, la consolidation des meilleures relations avec leurs clients, la conception et l'évolution de stratégies commerciales efficaces, l'amélioration et la conception de leurs produits/événements/services.

La diversité et la popularité croissantes de sources riches en opinions, comme les sites de publication d'avis de consommateurs et les blogs personnels apportent une importance significative à l'analyse de sentiment ce qui donne lieu à de nouvelles opportunités mais aussi à des défis inédits pour développer des nouvelles approches de classification d'opinions. Ce que nous a amené dans cette contribution à développer une approche innovante qui a pour objectif de réaliser une analyse exploratoire et visuelle des tweets. Notre approche proposée est basée sur la combinaison de plusieurs techniques d'apprentissage automatique. Afin d'atteindre un haut niveau d'efficacité pour la détection de polarité d'opinions en langue anglaise, il est important de passer par les phases suivantes : le prétraitement de données, l'extraction de caractéristiques, la sélection de caractéristiques, la classification de textes d'opinions et finalement la parallélisation de notre contribution sur plusieurs nœuds de traitement sous Hadoop.

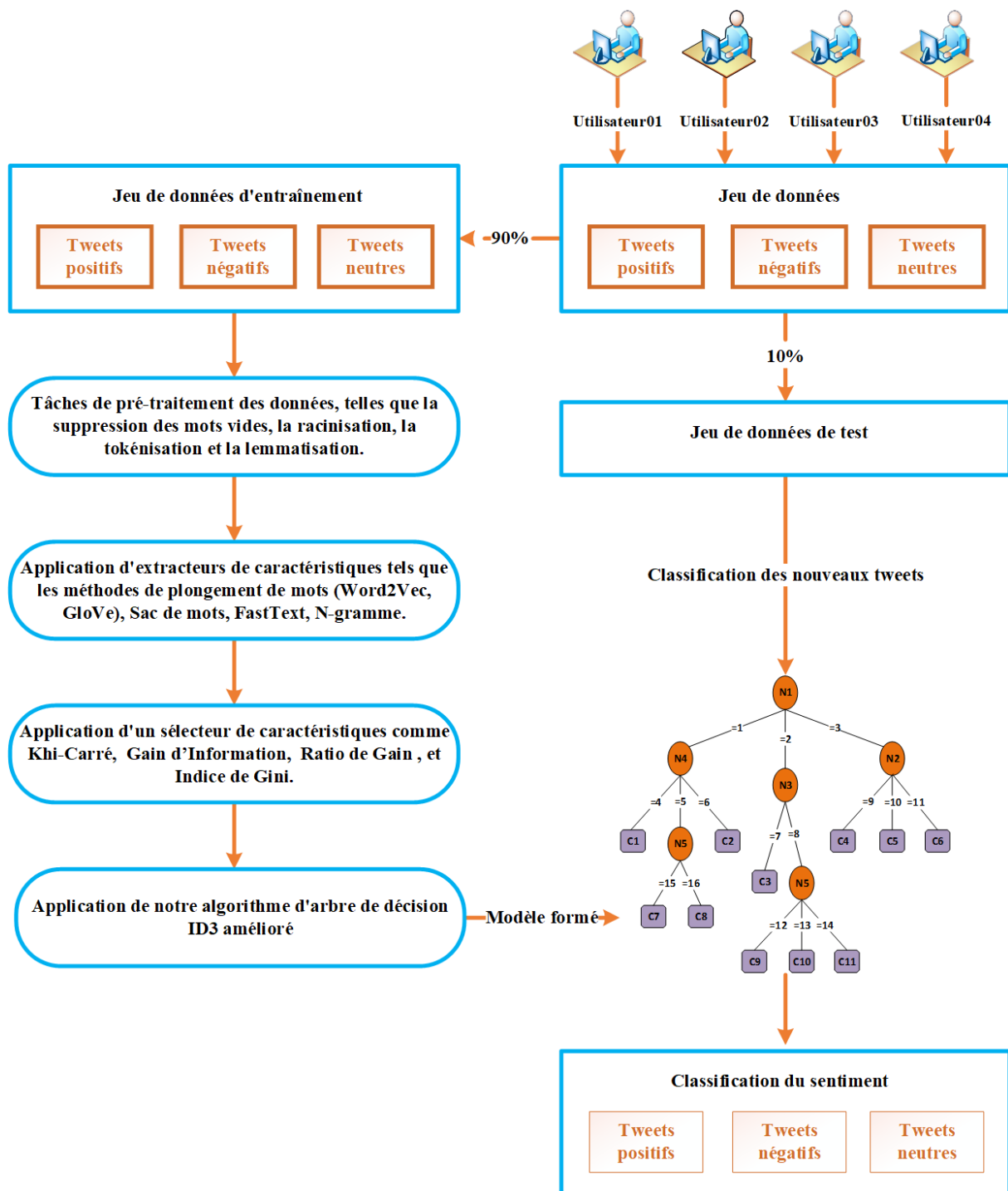


FIGURE 4.2 – Architecture de notre processus pour la détection de polarité d'opinions.

4.3.2 Matériels et méthodes

Nous présentons maintenant toute la méthodologie du pré-traitement des données, nous expliquons nos méthodes utilisées pour la vectorisation des textes et présentons nos techniques appliquées pour la sélection de caractéristiques.

4.3.2.1 Pré-traitement de données

La première étape de notre approche consiste à vérifier la qualité et la cohérence des données, à corriger les erreurs et les données erronées, à remplacer ou supprimer les valeurs manquantes, et à éliminer les valeurs extrêmes et excentrées. Cela se fait en appliquant sur les jeux de données les techniques de pré-traitement de données décrites dans la section 3.5 du chapitre 3. Par conséquent, le pré-traitement de données est l'une des étapes les plus importantes dans le processus de classification de textes, comme l'a prouvé les articles [191]- [192] dans lesquels les auteurs nous ont fourni une étude comparative qui vise à évaluer l'impact des étapes de pré-traitement des données sur la classification des textes en termes de précision et leurs résultats expérimentaux ont prouvé que le pré-traitement de données permet d'améliorer considérablement la performance de classification.

Après avoir observé les données, nous avons vu que les phrases contenaient des balises HTML, des mots-vides et toute la ponctuation. Nous avons donc commencé par éliminer le bruit pour normaliser nos phrases. Nous supprimons les balises HTML avec le module BeautifulSoup. Nous supprimons aussi tous les caractères qui ne sont pas des lettres et donc, supprimons toute la ponctuation des textes. Parce que les mots-vides, par définition, n'apportent pas d'information au texte, nous les éliminons aussi. Toutes les lettres sont également passées en minuscule. Finalement, nous racinisons tous les mots pour traiter chaque flexion d'un mot en un seul et même mot. Nous détaillons ci-dessous quelques étapes importantes du pré-traitement.

Pseudos : Les pseudos (e.g., @username) étant inutiles pour l'analyse de sentiments, nous remplaçons tous les @usernames par le texte *AT_USER* afin de pouvoir les supprimer par la suite.

Lettres répétées : Le langage utilisé sur Twitter est pour la plupart du temps familier. Il n'est donc pas rare que des mots soient écrits avec une lettre (ou plusieurs) qui se répète alors qu'elle ne le devrait pas. Par exemple le mot "nice" peut se trouver de la forme "niiiiiiiiiiice" sur Twitter. Dès qu'un mot contient des lettres identiques qui se répètent plus de deux fois, elles sont remplacées par seulement deux occurrences de cette même lettre ("niiiiiiiiiiice" devient "niice").

Hashtags : Les hashtags sur Twitter sont utilisés pour créer une connexion instantanée avec d'autres utilisateurs. Le mot qui suit le # est généralement un mot qui fournit beaucoup d'informations sur le sentiment de la phrase. Nous conservons donc ce mot, mais le caractère hashtag est supprimé.

Racinisation : Nous transformons toutes les flexions en leur racine. L'objectif est de réduire les formes dérivées d'un mot à une forme de base commune afin de pouvoir faciliter la correspondance entre les différents termes.

4.3.2.2 Extraction des caractéristiques

La majorité des algorithmes d'apprentissage automatique ne prennent pas comme entrée un texte brut, mais des vecteurs numériques. Pour cela il est nécessaire de trouver

une transformation représentative qui convertit le texte des tweets vers des vecteurs numériques. Dans la littérature les méthodes de transformation du texte vers des vecteurs numériques peuvent être divisées sur cinq approches principales : Word2Vec, GloVe [193], N-gram [194], FastText [195], Bag-Of-words [196], et TF-IDF qui sont décrites dans la section 3.6 du chapitre 3.

La première est une approche qui s'appuie sur deux réseaux de neurones artificiels : *skip-Gram* et *continuous bag-of-words*. La deuxième est basée sur l'idée de prendre le meilleur des informations globales (telles que celles utilisées pour les sacs-de-mots distributionnels) et locales (telles que celles utilisées pour les plongements lexicaux). La troisième consiste à donner la probabilité d'apparition du n -ième mot en fonction des $(n - 1)$ qui précèdent. La quatrième est une approche qui s'appuie sur la même architecture que *Skip-Gram* mais en calculant des représentations pour des n -grammes de caractères plutôt que pour des mots. La cinquième consiste à décrire un texte par les occurrences (fréquences) des mots qui le composent et la sixième est une approche purement statistique basée sur l'occurrence des termes comme TF et TF-IDF.

D'une part, notre but principal est d'appliquer tous les extracteurs de caractéristiques présentés précédemment et de les comparer pour déterminer la meilleure méthode d'extraction qui influence positivement le taux de classification de notre approche. Donc, pour souligner l'importance et clarifier l'effet de chaque extracteur de caractéristiques dans les résultats de la classification d'opinions en langue Anglaise ; nous avons effectué la deuxième expérimentation dans ce chapitre pour calculer le taux de classification et d'analyser les paramètres de chaque extracteur de caractéristiques. Les résultats de cette expérimentation sont décrits et discutés en détail dans la section «Évaluation de plusieurs extracteurs de caractéristiques»

4.3.2.3 Selection des caractéristiques

Lorsque l'espace vectoriel des caractéristiques s'étend de plus en plus en ajoutant de plus en plus de caractéristiques, il devient de plus en plus dispersé. Le fait que l'espace vectoriel des caractéristiques soit excessivement dispersé entraîne un sur-ajustement supplémentaire dans l'algorithme d'apprentissage automatique appliqué au jeu de données d'apprentissage ce qui rend cet algorithme inefficace pour classifier les instances inconnues dans le jeu de données de test. Ce phénomène est connu sous le nom de "fléau de la dimensionnalité" [197]. Une des solutions qui peut être mise en œuvre pour limiter les effets de ce phénomène est de réduire la dimension des caractéristiques avant de les traiter par la sélection de caractéristique les plus pertinents.

La sélection de caractéristique consiste à projeter l'ensemble de caractéristiques de l'espace original sur un nouvel espace de dimensions réduites. La sélection de caractéristique est une étape couramment utilisée dans l'apprentissage automatique, surtout face à un espace de caractéristiques de dimension très élevé. Les principales raisons motivant l'utilisation de sélection de caractéristique dans l'apprentissage sont les suivantes :

- Améliorer la performance de la prédiction, afin d'améliorer l'efficacité d'apprentissage.

- Fournir des prédicteurs plus rapides éventuellement en utilisant moins d'informations sur les données d'origine.
- Réduire la complexité des résultats produits et permettre une meilleure compréhension du processus de classification.

Autrement dit, la sélection de caractéristiques est définie comme un processus de recherche permettant de trouver un sous-ensemble "pertinent" de caractéristiques parmi celles de l'ensemble de départ. La notion de pertinence d'un sous-ensemble de caractéristiques dépend toujours des objectifs et des critères du système. En général, le problème de sélection de caractéristiques peut être défini par :

Soit $\mathcal{F} = \{f_1, f_2, \dots, f_N\}$ un ensemble de caractéristiques de taille N , ou N représente le nombre total de caractéristiques étudiées. Soit E_v une fonction qui permet d'évaluer un sous ensemble de caractéristiques. Nous supposons que la plus grande valeur de E_v soit obtenue pour le meilleur sous-ensemble de caractéristiques. L'objectif de la sélection est de trouver un sous ensemble F' ($F' \in \mathcal{F}$) de taille N' ($N' \in N$) tel que :

$$E_v(F') = \max_{Z \in \mathcal{F}} E_v(Z) \quad (4.1)$$

Où $|Z| = N'$ et N' est, soit un nombre prédéfini par l'utilisateur ou soit contrôlé par une des méthodes de sélection de sous-ensemble pertinent de caractéristiques.

D'autre part, les différentes méthodes de sélection de caractéristiques se différencient les unes des autres par le choix du critère mesurant la pertinence du sous-ensemble de caractéristiques. Certain de ces méthodes sont détaillées dans la section 3.7 du chapitre 3 et sont expérimentées dans la section "Analyse des sélecteurs de caractéristiques" pour préciser la meilleure méthode de sélection de caractéristique parmi les autres en termes de taux de classification.

4.3.3 Classification

La classification des sentiments est un raffinement de la détection d'opinions dans la mesure où elle permet de classifier le texte ayant une opinion sur un sujet en classes. Pour effectuer cette classification d'opinion, il existe deux approches : l'une basée sur le lexique, et l'autre sur l'apprentissage automatique. Dans notre travail, nous avons proposé une amélioration de l'algorithme d'apprentissage automatique ID3 et l'utilisée comme classificateur dans notre processus de détection de polarité d'opinions. Le choix de cet algorithme est dû à la minimisation de l'erreur de classement, à la facilité avec lequel les résultats peuvent être visualisés en arbre de décision, à la classification prédictive élevée présentée, et enfin à l'utilisation répandue des règles dans le domaine de la fouille d'opinions.

4.3.3.1 Algorithme ID3 amélioré

L'algorithme d'arbre de décision ID3 est un modèle d'apprentissage supervisé basé sur le calcul du gain d'information décrit par l'équation (3.4) pour créer un arbre de décision. Cet algorithme utilise le gain d'information pour choisir le meilleur attribut de fractionnement à chaque itération de l'algorithme. Le processus de calcul du gain d'information ne prend en compte que l'attribut en cours de test et l'attribut cible, et les autres attributs ne peuvent pas être utilisés pour mesurer l'importance de l'attribut en cours de test. En raison de ces problèmes, nous avons proposé une amélioration d'ID3 en se basant sur le gain d'information pondéré. Le gain d'information pondéré prend en compte la corrélation entre l'attribut en cours de test, l'attribut cible et les autres attributs à chaque itération du processus d'apprentissage. L'objectif essentiel de notre amélioration est de mesurer l'influence de tous les attributs sur le gain d'information d'attribut en cours de test. Plus précisément, le gain d'information pondéré est calculé à l'aide de l'attribut pondéré et de la fonction de corrélation pondérée.

4.3.3.2 Attribut pondéré

Soit $\mathcal{A} = \{A_1, A_2, \dots, A_n\}$ un ensemble de \mathcal{N} attributs conditionnés. Nous supposons que le nombre d'occurrences de l'attribut $A_k (k = 1, 2, \dots, N)$ est N_k . Par conséquent, la fréquence de A_k est définie comme suit :

$$FA_k = \frac{N_k}{N} \quad (4.2)$$

Ensuite, l'attribut pondéré A_k est calculé comme suit :

$$WA_k = \frac{FA_k}{\sum_{k=1}^N FA_k} \quad (4.3)$$

Selon la définition de la théorie des poids, la somme de tous les attributs pondérés satisfait à la condition de l'équation définie ci-dessous :

$$\sum_k WA_k = 1 \quad (4.4)$$

4.3.3.3 Fonction de corrélation pondérée

Soit $\mathcal{A} = \{A_1, A_2, \dots, A_n\}$ un ensemble de \mathcal{N} attributs conditionnés avec des plages de valeurs $\{R_1, R_2, \dots, R_n\}$, respectivement. Soit \mathcal{Y} un attribut de décision avec une plage de valeurs R_Y . $A_{k \in (k=1,2,\dots,N)}$ est l'un des attributs conditionnés de l'ensemble \mathcal{A} et possède \mathcal{V} valeurs, alors que $R_{k \in (k=1,2,\dots,N)} = a_1, a_2, \dots, a_V$. \mathcal{Y} est l'attribut de décision et a \mathcal{M} valeurs $R_Y = \{Y_1, Y_2, \dots, Y_M\}$. Par conséquent, la fonction de corrélation pondérée entre l'attribut conditionné A_k et l'attribut de décision \mathcal{Y} est définie comme suit :

$$CF(A_k, Y) = \frac{\sum_{i=1}^V |A_{ij}| - \sum_{j=2}^M |A_{ij}|}{\mathcal{V}} \quad (4.5)$$

Où \mathcal{V} est le nombre de valeurs de A_k , $|A_{ij}|$ est le nombre de cas où la i ème valeur de A_k appartient à la j ème catégorie de l'attribut de décision \mathcal{Y} . Ensuite, la fonction de corrélation pondérée de l'attribut A_k est calculée comme suit :

$$WCF(A_k, Y) = \frac{CF(A_k, Y)}{\sum_{k=1}^N CF(A_k, Y)} \quad (4.6)$$

Par conséquent, selon le critère de gain d'information et la définition de la théorie des poids, la mise en œuvre concrète de l'algorithme d'arbre de décision ID3 basé sur le gain d'information modifié-pondéré est décrite en détail comme suit :

4.4 Parallélisation de l'approche proposée

L'apparition de l'ère du Big Data pose un défi aux algorithmes classiques d'apprentissage automatique. Dans le domaine des technologies de l'information, le Big Data est un ensemble de jeux de données qui contiennent une quantité massive de données, ce qui rend leur traitement très compliqué à l'aide des périphériques de gestion de base de données populaires ou des applications de traitement de données conventionnelles. Le Big Data est généralement constitué de jeux de données dont la taille dépasse la capacité des dispositifs logiciels habituellement utilisés, qui sont incapables de capturer, de gérer ou de traiter ces données dans un délai d'exécution raisonnable et réalisable. Les défis liés aux Big Data concernent l'acquisition, le stockage, l'exploration, la distribution, le traitement et la visualisation. Par conséquent, l'efficacité temporelle et spatiale des algorithmes classiques d'apprentissage automatique diminue considérablement lors du traitement des Big Data. Pour remédier à ces difficultés dans cette proposition, nous avons travaillé sous le cadre Hadoop [198] comme le montre la figure 4.3 qui représente la mise en œuvre de notre proposition en utilisant Hadoop avec son système de fichiers distribués et son modèle de programmation MapReduce.

La figure 4.3 montre que Hadoop est un cadre utilisé pour stocker et traiter une quantité massive de données. Notre jeu de données est stocké sur cinq machines de calcul : quatre noeuds de calcul esclaves et un noeud de calcul maître. D'autre part, le traitement de données est effectué par le modèle de programmation MapReduce, qui fournit une sorte d'environnement de calcul parallèle distribué pour récupérer et traiter les données massives stockées dans le système HDFS. MapReduce divise le processus de calcul et d'apprentissage en étapes Map et Reduce, qui correspondent à la mise en œuvre d'une méthode Mapper() et d'une méthode Reducer(), respectivement. MapReduce divise le processus de calcul et d'apprentissage en étapes Map et Reduce, qui correspondent respectivement à la mise en œuvre d'une méthode Mapper() et d'une méthode Reducer(). Le processus MapReduce prend en entrée la partition du tweet sous la forme de paire $\langle \text{key}, \text{value} \rangle$, où la variable "key" représente le numéro de série de la partition en entrée, et la variable "value" décrit la valeur de données de la partition en entrée. Dans l'étape Map, MapReduce divise les données en partitions de taille égale et traite chaque partition sous la forme de pair $\langle \text{Key1}, \text{Value1} \rangle$ pour déterminer l'entrée formelle. Il applique la méthode Mapper() pour produire des résultats intermédiaires sous la forme de paire $\langle \text{Key2}, \text{value2} \rangle$, qui sont classés en fonction de la valeur des données. Les valeurs "Val2" dont le

Algorithm 2 NOTRE ALGORITHME AMÉLIORÉ ID3**Entrées:**

- . **Ensemble d'attributs** : $A = \{A_1, A_2, \dots, A_N\}$ être un ensemble de N attributs conditionnés avec des plages de valeurs $\{R_1, R_2, \dots, R_N\}$, respectivement. Et Y est un attribut de décision et a M valeurs $R_Y = \{Y_1, Y_2, \dots, Y_M\}$;
- . **Jeu d'échantillons** : $S = \{(x_i, y_i) \mid x_i \in R_1 \times R_2 \times \dots \times R_n, y_i \in R_Y\}$ est un échantillon tiré d'une distribution inconnue. Où x_i est relié à une sortie y_i ;
- . **Critères d'arrêt** : est le critère pour arrêter le processus d'apprentissage
 - 1 Toutes les instances de l'ensemble d'apprentissage appartiennent à une seule valeur de y ;
 - 2 L'ensemble d'attributs A est vide ou toutes les valeurs d'attributs de S sont les mêmes ;

Sorties: Arbre de décision **TREE****ArbreGénérer(S, A)** : Créer un nouvel arbre TREE avec un seul nœud racine ;si $Critèresd'arrêt(S) = 1$ marquer le noeud comme un noeud feuille de classe Y ;**retourner**sinon si $Critèresd'arrêt(S) = 2$ marquer TREE comme un noeud feuille avec la valeur la plus commune de Y dans S comme étiquette ;**retourner****sinon****pour** $CA_i \in A$ Calculez le gain d'information à l'aide de la formule suivante : $Gain(CA_{(Y,S)})[i] \leftarrow E_{DA}(Y) - E_{CA}(A_i, S)$;**fin pour** $sPoids \leftarrow 0$ **pour** chaque valeur d'attribut $V_i \in R_i$ $pPoids \leftarrow 1$ Calculez le poids de chaque attribut en utilisant l'ensemble d'apprentissage S_i de chaque valeur V_i de l'attribut A_i $Som \leftarrow 0$ **pour** chaque valeur d'attribut $A_j \in A \setminus \{A_i\}$ Calculez la fréquence à l'aide de la formule : $CF_{(A_j,Y)}[j] \leftarrow \frac{\sum_{V=1}^{|V_i|} \|A_{vy}\| - \sum_{y=2}^M \|A_{vy}\|}{V_j}$ $Som \leftarrow Som + CF_{(A_j,Y)}[j]$ **fin pour****pour** chaque valeur d'attribut $A_j \in A \setminus \{A_i\}$ Calculez le poids à l'aide de la formule : $WCF_{(A_j,Y)}[j] \leftarrow \frac{CF_{(A_j,Y)}[j]}{Som}$ $pPoids \leftarrow pPoids \times WCF_{(A_j,Y)}[j]$ **fin pour** $sPoids \leftarrow sPoids + \frac{|S_i|}{|S|} \times pPoids$ **fin pour** $Gain(CA_{(Y,S)})[i] \leftarrow Gain(CA_{(Y,S)})[i] \times sPoids$

Trouvez le meilleur attribut de fractionnement $A_{meilleur}$ ayant le gain d'information modifié pondéré maximum $Gain(CA_{(Y,S)})$
 $A_{meilleur} \leftarrow \text{argmax}_A \text{Gain}(CA_{(Y,S)})$
 Attachez $A_{meilleur}$ dans TREE
pour chaque valeur d'attribut $v \in A_{meilleur}$ Générez une branche pour le nœud, de sorte que S_v représente un sous-ensemble des échantillons de S dont l'attribut $A_{meilleur}$ est v
if $S_v = \text{vide}$ **then**
 Marquez le nœud de branche comme un nœud feuille, et sa classe est marquée comme la classe avec le plus grand nombre d'échantillons dans S
return
else
 Récursion de **ArbreGénérer**($S_v, A \setminus \{A_{meilleur}\}$) continue
end if

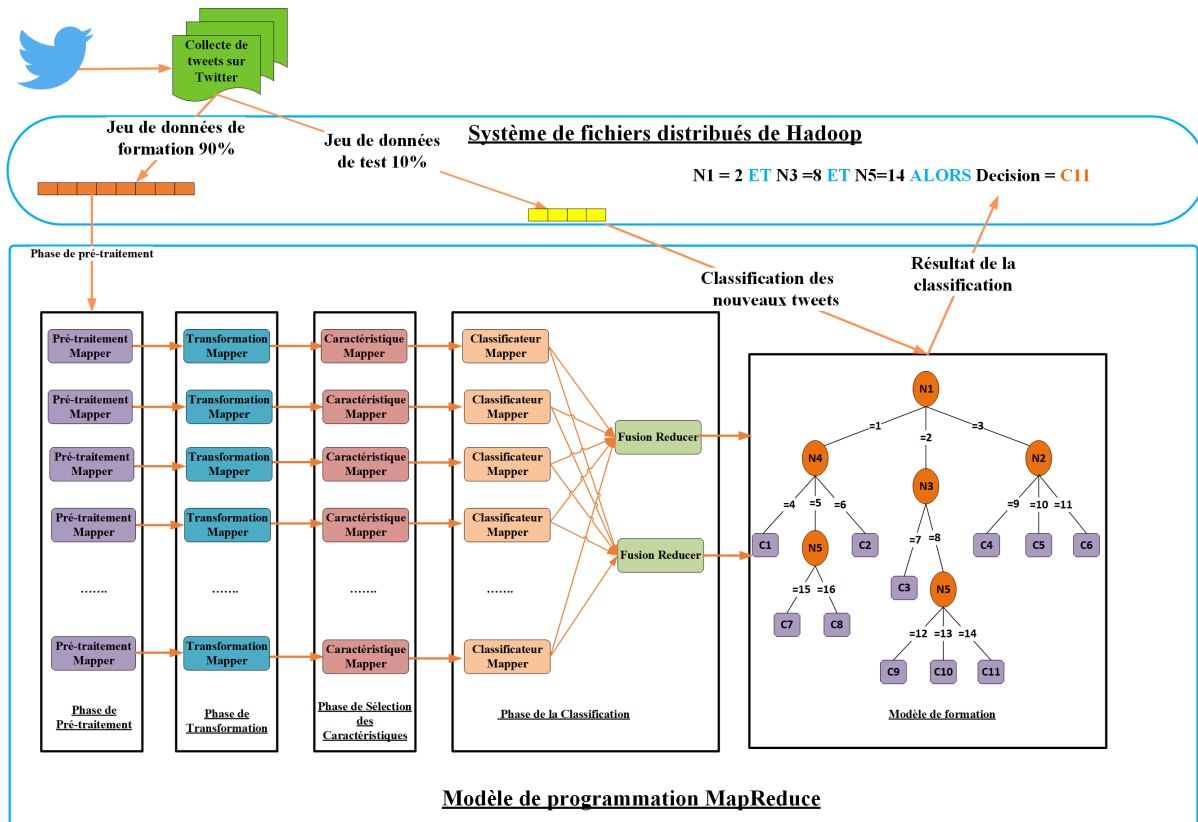


FIGURE 4.3 – Les étapes de base de la fouille d'opinion sur les tweets sous Hadoop

numéro de série "key2" est le même sont fusionnées pour créer une nouvelle liste $\langle \text{Key2}, \text{list}(\text{Val2}) \rangle$ et par la suite, elles sont rassemblées selon le numéro de série "Key2" pour démarrer l'exécution des tâches Reduce. Dans l'étape de Reduce, les résultats des travaux de Map sont combinés et organisés sous la forme $\langle \text{Key2}, \text{list}(\text{value2}) \rangle$ qui est utilisée comme entrée, et la méthode de $\text{Reducer}()$ est exécutée pour obtenir les paires $\langle \text{Key3}, \text{value3} \rangle$, qui est la sortie à stocker dans les systèmes de fichiers distribués Hadoop.

4.5 Expérimentations et résultats

Les sections précédentes ont été consacrées à l'introduction générale, aux travaux connexes, à la méthode hybride proposée. Comme nous l'avons dit, le classifieur que nous avons proposé se compose de cinq étapes. Dans l'étape de collecte de données, nous avons sélectionné les jeux de données **Sentiment140** et **COVID -19_Sentiments**. Dans cette section, nous menons cinq expériences pour démontrer la justesse et l'efficacité du classificateur que nous proposons par rapport aux autres méthodes de la littérature. Et pour évaluer son efficacité et ses performances, nous avons sélectionné dix critères d'évaluation tels que *TC*, *TR*, *TE*, *TPV*, *TNF*, *TNV*, *TPF*, *PR*, *SK*, et *SF*.

4.5.1 Influence de pré-traitement des données

Dans cette première expérience, nous avons analysé les performances de toutes les techniques de pré-traitement de données appliquées en matière de taille du jeu de données et de temps d'exécution avant et après la mise en œuvre du cadre Hadoop. Par conséquent, l'objectif principal d'appliquer les techniques de pré-traitement est de supprimer le bruit, d'améliorer la qualité de données et de diminuer la taille du jeu de données. Les figures 4.4 et 4.5 illustrent le résultat de cette expérience. En conséquence, le traitement de données permet d'obtenir des performances de classification élevées dans l'évaluation et l'analyse de données lorsqu'un des algorithmes d'apprentissage supervisé est entraîné. De plus, la taille du jeu de données est réduite après l'application de chaque tâche de pré-traitement des données. Les tableaux 4.2 et 4.3 montrent la réduction de la taille du fichier du jeu de données après l'application de chaque tâche de pré-traitement des données sur les jeux de données Sentiment140 et COVID-19_Sentiments.

Comme le montre la figure 4.4, nous constatons que l'application de la tâche de pré-traitement "*Supprimez les noms d'utilisateurs, les numéros, les hashtags et les URLs*" réduit la taille du jeu de données Sentiment140 de 5.29% et prend un temps d'exécution égal à 0.29 seconde. De même, la tâche de pré-traitement "*Supprimez la ponctuation, les espaces blancs et les caractères spéciaux*" diminue la taille du jeu de données Sentiment140 de 9.05 % et consomme un temps d'exécution égal à 0.49 seconde. La tâche de pré-traitement "*Mise en minuscule*" minimise la taille du jeu de données Sentiment140 de 8.62% et son temps d'exécution est égal à 0.46 seconde. En ce qui concerne la tâche de pré-traitement "*Remplacez les mots allongés*", elle réduit la taille du jeu de données Sentiment140 de 5.18% et elle prend le temps d'exécution de 0.27 seconde. Pour la tâche de pré-traitement "*Supprimez les mots vides*" abaisse la taille du jeu de données Sentiment140 de 11.84 % et consomme 0.62 seconde en temps d'exécution. Et la tâche de pré-traitement "*Lemmatisation*" réduit la taille du jeu de données Sentiment140 de 3.07%, et son temps d'exécution est égal à 0.16 seconde. Enfin, la tâche de pré-traitement "*Tokenisation*" augmente la taille du jeu de données Sentiment140 de 18.87 %, et elle consomme un temps d'exécution égal à 0.80 seconde.

Selon le tableau 4.2, nous observons que la taille du jeu de données (TJD) Sentiment140 avant l'application des tâches de prétraitement des données est égale à 283234.71 KB, et après l'application de toutes les tâches de prétraitement des données utilisées, à l'except-

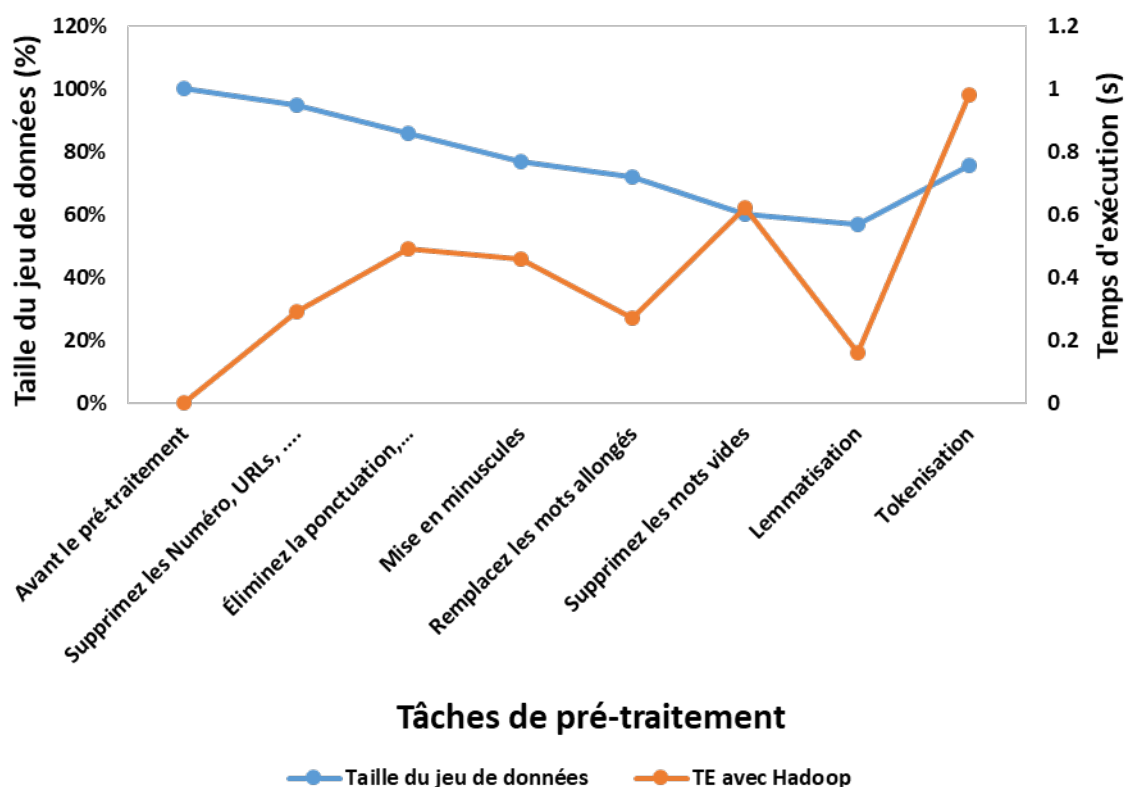


FIGURE 4.4 – Taille du jeu de données et temps d'exécution de chaque tâche de pré-traitement appliquée au Sentiment140.

tion de la tâche "*Tokenisation*", la taille du jeu de données est devenue 161302.17 KB. Par conséquent, la taille du jeu de données est réduite de 43.05%, ce qui représente le pourcentage de données bruyantes et indésirables. Après l'application de la tâche de pré-traitement "*Tokenisation*", la taille du jeu de données augmente à 214748.55KB parce que cette technique sert à diviser chaque tweet d'entrée en un ensemble de jetons, augmentant la taille du jeu de données. En outre, le temps d'exécution consommé après l'application de toutes les tâches de pré-traitement des données est égal à 16.45 seconde, mais la mise en œuvre du cadre Hadoop réduit cette valeur à 3.27 seconde.

D'après la figure 4.5, nous remarquons que l'application de la technique de pré-traitement "*Supprimer les noms d'utilisateur, les numéros, les hashtags et les URL*" réduit la taille du jeu de données COVID-19_Sentiments de 9.11 % et consomme un temps d'exécution égal à 0.096 seconde. En outre, la technique de pré-traitement des données "*Supprimez les espaces blancs, la ponctuation et les caractères spéciaux*" réduit la taille du jeu de données COVID-19_Sentiments de 11.47 %, et son temps d'exécution est égal à 0.12 seconde. La tâche "*Mise en minuscule*" minimise la taille du jeu de données COVID-19_Sentiments de 7.39 %, et elle prend le temps d'exécution de 0.077 seconde. Ainsi, la tâche de pré-traitement des données "*Remplacez les mots allongés*" réduit la taille du jeu de données COVID-19_Sentiments de 2.88 %, et son temps d'exécution est de 0.03 seconde. Pour la tâche "*Supprimez les mots vides*" fait diminuer la taille du jeu de données COVID-19_Sentiments de 10.47 %, et elle consomme un temps d'exécution égal à 0.11 seconde. La tâche de pré-traitement des données "*Lemmatization*" réduit la taille du jeu de

TABLE 4.2 – Analyse de la taille du jeu de données (TJD) et du temps d'exécution des données avec Hadoop (TEA) et sans Hadoop (TES) dans le cas de Sentiment140.

Techniques	TJD (KB)	TJD (%)	TEA (s)	TES (s)
Avant le pré-traitement	283234.71	100 %	—	—
Supprimez les noms d'utilisateurs, les numéros, les hashtags et les URLs	268251.59	94.71 %	1.49	0.29
Supprimez la ponctuation, les espaces blancs et les caractères spéciaux	242618.85	85.66 %	2.45	0.49
Mise en minuscule	218204.02	77.04 %	2.33	0.46
Remplacez les mots allongés	203532.46	71.86 %	1.38	0.27
Supprimez les mots vides	169997.47	60.02 %	3.08	0.62
Lemmatisation	161302.17	56.95 %	0.80	0.16
Tokenisation	214748.55	75.82 %	4.92	0.98

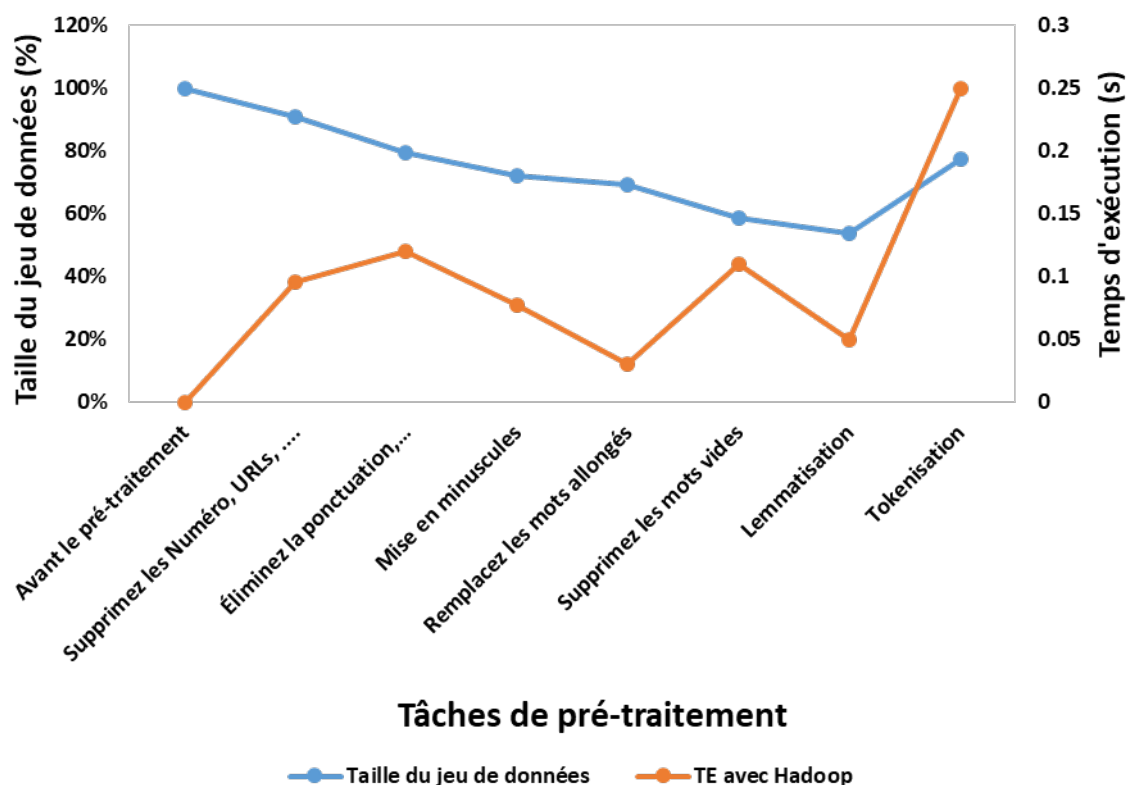


FIGURE 4.5 – Taille du jeu de données et temps d'exécution de chaque tâche de pré-traitement appliquée au COVID-19_Sentiments.

données COVID-19_Sentiments de 4.77 %, et son temps d'exécution est de 0.05 seconde. Enfin, la tâche de pré-traitement des données "*Tokenisation*" augmente la taille du jeu de données COVID-19_Sentiments de 23.54 %, et elle consomme un temps d'exécution égal à 0.25 seconde.

D'après le tableau 4.3, nous remarquons que la taille du jeu de données COVID-19_Sentiments avant l'application des tâches de pré-traitement des données est égale à 112935.94 KB, et après l'application de toutes les tâches de pré-traitement des données utilisées sauf la "*Tokenisation*", la taille du jeu de données est devenue 60883.76 KB. Par conséquent, la taille du jeu de données est réduite de 46.09 %, ce qui représente le pourcentage de données bruyantes et indésirables. Après l'application de la tâche de pré-traitement "*Tokenisation*", la taille du jeu de données COVID-19_Sentiments augmente à 87468.88 KB parce que cette technique sert à diviser chaque phrase d'entrée en un ensemble de jetons, ce qui augmente la taille du jeu de données COVID-19_Sentiments. En outre, le temps consommé après l'application de toutes les tâches de pré-traitement des données sur le jeu de données COVID-19_Sentiments est égal à 3.67 secondes, mais l'utilisation du cadre Hadoop abaisse cette valeur à 0.733 seconde.

TABLE 4.3 – Analyse de la taille du jeu de données (TJD) et du temps d'exécution des données avec Hadoop (TEA) et sans Hadoop (TES) dans le cas de COVID-19_Sentiments.

Techniques	TJD (KB)	TJD (%)	TEA (s)	TES (s)
Avant le pré-traitement	112935.94	100 %	—	—
Supprimez les noms d'utilisateurs, les numéros, les hashtags et les URLs	102647.45	90.89 %	0.49	0.096
Supprimez la ponctuation, les espaces blancs et les caractères spéciaux	89693.72	79.42 %	0.61	0.12
Mise en minuscule	81347.75	72.03	0.39	0.077
Remplacez les mots allongés	78095.20	69.15 %	0.15	0.03
Supprimez les mots vides	66270.81	58.68 %	0.54	0.11
Lemmatisation	60883.76	53.91 %	0.26	0.05
Tokenisation	87468.88	77.45 %	1.23	0.25

Une autre expérience est réalisée pour évaluer l'efficacité de toutes les tâches de pré-traitement des données utilisées sur les jeux de données COVID-19_Sentiments et Sentiment140 en calculant le taux d'erreur avec et sans utilisation des tâches de pré-traitement des données. Le tableau 4.4 présente les résultats expérimentaux du calcul du taux d'erreur (TE %) sans et avec l'application des approches de pré-traitement des données sur les deux jeux de données COVID-19_Sentiments et Sentiment140.

D'après les résultats expérimentaux, tels qu'ils sont présentés dans le tableau 4.4, nous avons conclu que les tâches de prétraitement des données minimisent l'TE. En effet, le

TABLE 4.4 – TE sans et avec les techniques de pré-traitement des données.

Nom du jeu de données	TE sans pré-traitement	TE avec pré-traitement
Sentiment140	39.81	13.47
COVID-19_Sentiments	30.12	11.18

TE dans le cas du jeu de données Sentiment140 diminue de 39.59 % à 13.47 %, et il diminue de 30.04 % à 11.18 % dans le cas du jeu de données COVID-19_Sentiments. Par conséquent, il est recommandé d'appliquer les techniques de pré-traitement des données avant d'appliquer tout algorithme d'apprentissage automatique sur le jeu de données utilisé.

4.5.2 Évaluation des extracteurs de caractéristiques

Cette deuxième expérience vise à déterminer l'extracteur de caractéristiques le plus efficace en termes de taux de classification parmi les extracteurs de caractéristiques utilisés dans ce travail, notamment N-gramme, Sac de Mots, TF-IDF, GloVe, Word2Vec, Fast-Text. Comme nous l'avons dit précédemment, ces extracteurs de caractéristiques servent à convertir les tweets d'entrée en un ensemble de caractéristiques numériques. La figure 4.6 présente le taux de classification obtenu après l'extraction d'opinions en fonction de la longueur des n-grammes de caractères à partir du jeu de données Sentiment140 sous le cadre Hadoop.

D'après la figure 4.6, nous constatons que le N-gramme = 5 a obtenu les meilleures performances de classification que les autres N-grammes de caractères (N-gramme=2, N-gramme=3, N-gramme=4). Puisque le taux de classification sur le jeu de données Sentiment140 atteint par le N-gramme = 5 est égal à 31.80%, 34.15%, et 35.49% respectivement dans le cas des trois algorithmes d'apprentissage automatique utilisés C4.5, ID3, et notre ID3 amélioré.

La figure 4.7 décrit le taux de classification obtenu pour les tâches de classification des sentiments en fonction de la longueur des n-grammes de caractères appliqués sur le jeu de données COVID-19 Sentiments sous le cadre Hadoop. Comme l'illustre la figure 4.7, nous remarquons également dans cette expérience que le N-gramme = 5 a atteint une haute performance de classification sur le jeu de données COVID-19 Sentiments par rapport aux autres longueurs des N-grammes de caractères et qui est égale à 51.76%, 54.41% et 54.99% respectivement dans le cas des trois algorithmes d'apprentissage automatique utilisés C4.5, ID3, et notre ID3 amélioré respectivement.

En comparant les résultats obtenus dans les figures 4.6 et 4.7, nous déduisons que la méthode de représentation N-gramme utilisée est inefficace dans le cas des Big Data car elle a obtenu un taux de classification inférieur à 36 % dans le cas du grand jeu de données Sentiment140. Ainsi que dans les deux expériences, nous remarquons que le taux de classification augmente lorsque l'on augmente la longueur des N-grammes de caractères.

La figure 4.8 présente le taux de classification obtenu après l'application des tâches de classification des sentiments sur les jeux de données Sentiment140 et COVID-19 Senti-

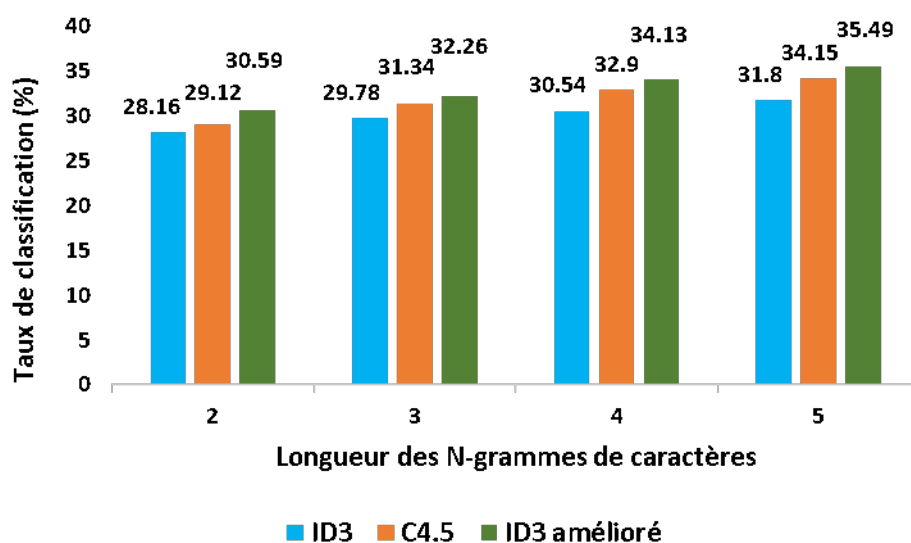


FIGURE 4.6 – TC de l'extraction d'opinions en appliquant N-gram sur Sentiment140 sous Hadoop.

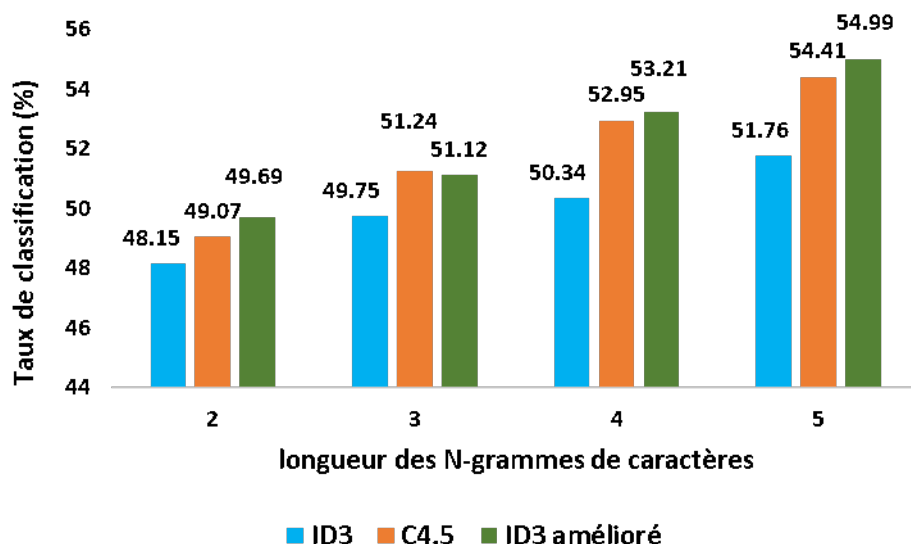


FIGURE 4.7 – TC de l'extraction d'opinions en appliquant N-gram sur COVID-19 Sentiments sous Hadoop.

ments en utilisant la méthode d'extraction par Sac de Mots et le cadre Hadoop. Comme le montre la figure 4.8, nous constatons que l'application du Sac de Mots sur le jeu de données COVID-19 Sentiments est plus efficace que son application sur le jeu de données Sentiment140 car le jeu de données Sentiment140 contient une grande quantité de données. Cependant, la méthode d'extraction par sac de mots a obtenu un taux de classification de 66.32 % après sa mise en œuvre sur le jeu de données COVID-19 Sentiments, et un taux de classification de 37.53 % après sa mise en œuvre sur le jeu de données Sentiment140. Par conséquent, cette expérience a prouvé que la méthode d'extraction par Sac de Mots n'est pas évolutive. En outre, la méthode d'extraction par Sac de Mots est plus efficace que la méthode d'extraction par N-gramme avec N-grammes = 5.

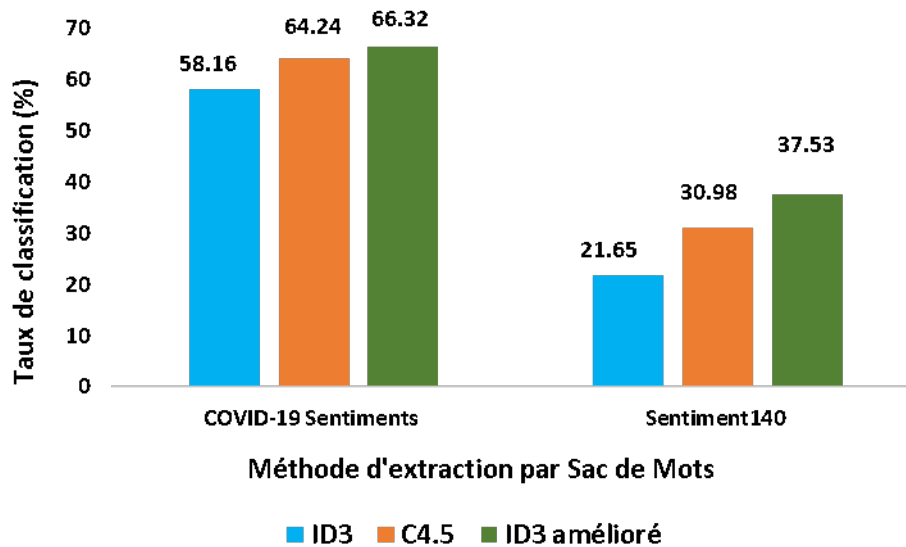


FIGURE 4.8 – TC en appliquant l’extracteur Sac de Mots sur COVID-19 Sentiments et Sentiment140 sous Hadoop.

La figure 4.9 présente le taux de classification obtenu par la mise en œuvre de l’extracteur TF-IDF sur les jeux de données Sentiment140 et COVID-19 Sentiments en utilisant le cadre Hadoop. Comme l’illustre la figure 4.9, nous remarquons que l’extracteur TF-IDF a obtenu les meilleures performances sur les jeux de données COVID-19 Sentiments et Sentiment140. En effet, il a obtenu un taux de classification de 74.67 % après son application au jeu de données COVID-19 Sentiments, et un taux de classification de 71.73 % après son application au jeu de données Sentiment140. Ensuite, nous remarquons que l’extracteur TF-IDF atteint un taux de classification approximatif (74.67 % ; 71.73 %) lorsqu’il est appliqué aux deux ensembles de données employés. Par conséquent, cet extracteur TF-IDF est évolutif par rapport aux méthodes d’extraction par N-grammes et par Sac de Mots.

La figure 4.10 montre le taux de classification obtenu pour les tâches de classification des sentiments après l’application de la méthode d’extraction GloVe sur les jeux de données COVID-19_Sentiments et Sentiment140 sous le cadre Hadoop. Comme présenté dans la figure 4.10, nous déduisons que l’extracteur GloVe a atteint de bonnes performances de classification sur les jeux de données COVID-19_Sentiments et Sentiment140. Puisqu’il a atteint un taux de classification égal à 76.35 % après avoir été appliqué sur le jeu de données COVID-19_Sentiments, et il a atteint un taux de classification égal à 70.91% après avoir été appliqué sur le jeu de données Sentiment140. Ensuite, nous remarquons que l’extracteur GloVe a atteint un taux de précision approximatif (76.35% ; 70.91%) lorsqu’il a été appliqué sur les deux ensembles de données utilisés. Par conséquent, l’extracteur GloVe est évolutif et son taux de classification est approximativement égal au taux de classification obtenu (74.67% ; 71.73%) après l’application de l’extracteur TF-IDF sur les deux jeux de données employés.

La figure 4.11 présente le taux de classification obtenu pour les tâches de classification de sentiments en appliquant la méthode d’extraction Word2vec sur les jeux de données COVID-19_Sentiments et Sentiment140 en utilisant le cadre Hadoop. Comme le

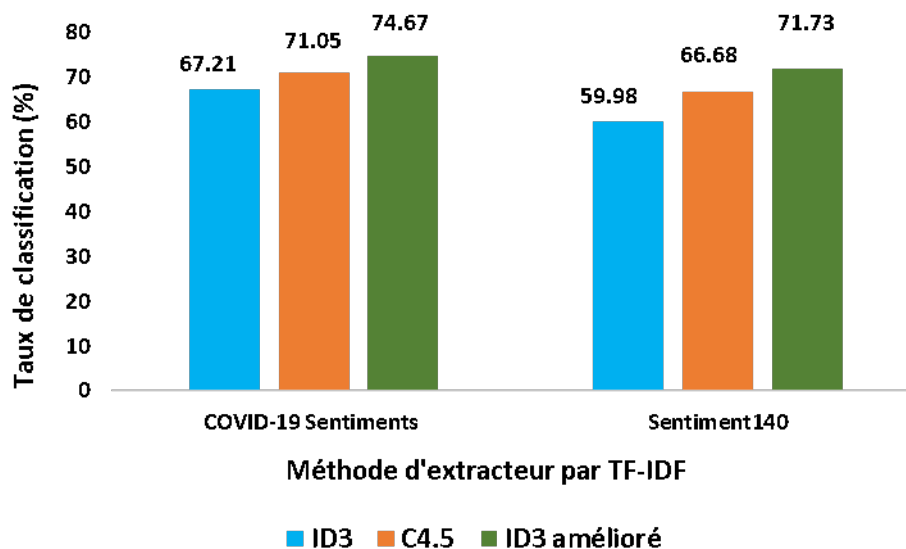


FIGURE 4.9 – TC en appliquant l'extracteur TF-IDF sur Sentiment140 et COVID-19 Sentiments sous Hadoop

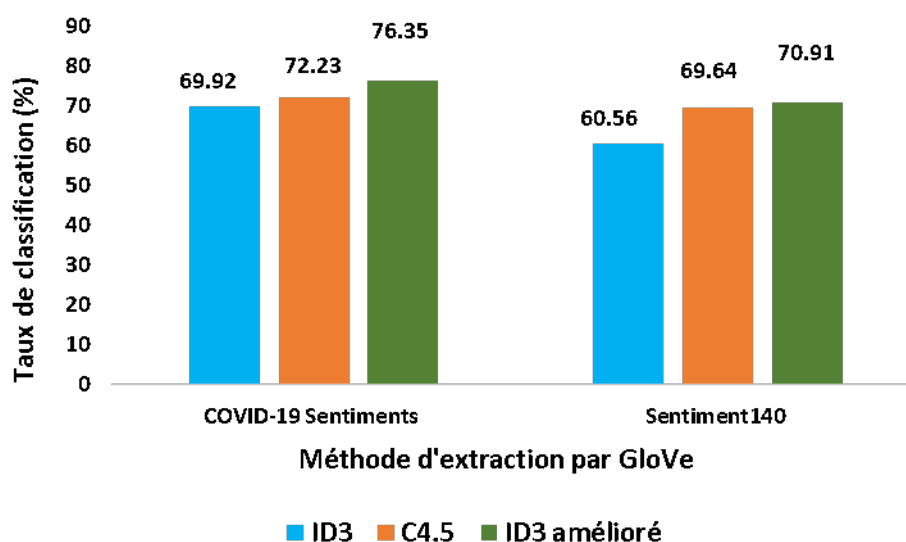


FIGURE 4.10 – TC en appliquant l'extracteur GloVe sur les deux jeux de données sous Hadoop

montre la figure 4.11, nous remarquons que la méthode d'extraction Word2vec est très efficace sur un grand jeu de données. En effet, elle atteint 80.65% lorsqu'elle est appliquée au jeu de données Sentiment140 et 81.72% lorsqu'elle est appliquée au jeu de données COVID-19_Sentiments. Par conséquent, l'extracteur Word2vec est évolutif, et il surpasse les extracteurs N-grammes, Sac de Mots, TF-IDF et GloVe en termes de taux de classification. Jusqu'à présent, la méthode d'extraction la plus efficace est Word2vec. La prochaine expérience vise à comparer la méthode d'extraction Word2vec avec la méthode méthode d'extraction FastText et à trouver la plus efficace d'entre elles.

La figure 4.12 montre le taux de classification obtenu pour les tâches de classification

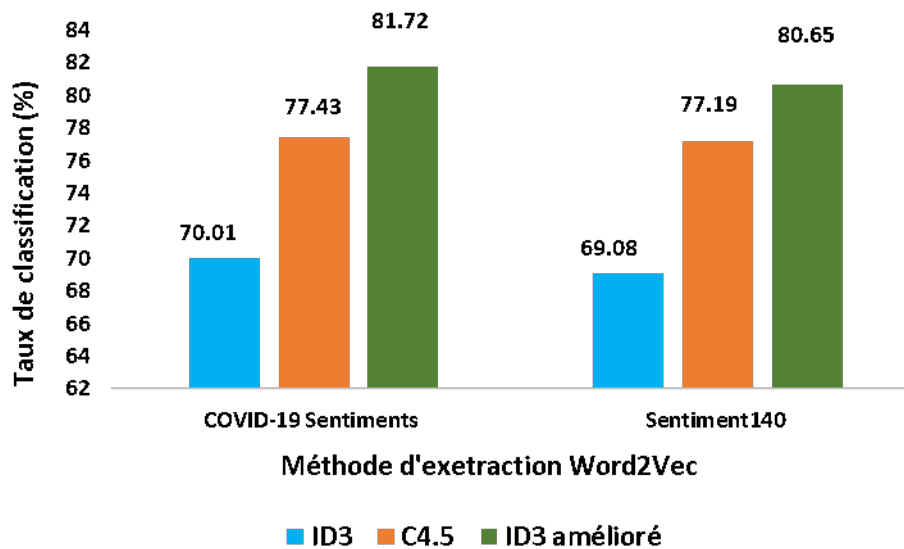


FIGURE 4.11 – TC en appliquant l’extracteur Word2Vec sur Sentiment140 et COVID-19-Sentiments sous Hadoop.

des sentiments en appliquant la méthode d’extraction FastText sur les jeux de données COVID-19_Sentiments et Sentiment140 sous le cadre Hadoop. Comme on peut le voir sur la figure 4.12, on observe que la méthode d’extraction FastText donne un bon taux de classification sur les deux jeux de données utilisés par rapport à toutes les autres méthodes d’extraction utilisées (N-grammes, Sac de Mots, TF-IDF, GloVe, et Word2Vec). Puisqu’elle a obtenu 86.53 % lorsqu’elle a été appliqué sur le jeu de données Sentiment140 et 88.82% lorsqu’elle a été appliqué sur COVID-19_Sentiments. D’après toutes les études comparatives réalisées dans cette sous-section 4.5.2, nous déduisons que la méthode d’extraction FastText surpasse toutes les autres méthodes. Ainsi, dans la suite de ce travail, nous utiliserons uniquement le FastText comme méthode de représentation des données.

4.5.3 Analyse des sélecteurs de caractéristiques

La phase qui suit la phase d’extraction des caractéristiques est la phase de sélection des caractéristiques, dans laquelle nous avons appliqué de nombreuses techniques, notamment le Khi-Carré (1), le Ratio de Gain (2), le Gain d’Information (3), et l’Indice de Gini (4). Par conséquent, cette expérience vise à trouver le meilleur sélecteur de caractéristiques parmi toutes les méthodes de sélection utilisées en termes de taux de classification. Le tableau 4.5 décrit le taux de classification obtenu en appliquant ID3, C4.5 et le ID3 amélioré sur les jeux de données Sentiment140 et COVID-19_Sentiments en utilisant différentes techniques de sélection de caractéristiques présentées précédemment. Dans le tableau 4.5, nous remarquons que le Gain d’Information utilisé comme sélecteur de caractéristiques surpasse les autres sélecteurs de caractéristiques en termes de taux de classification puisqu’il a atteint un taux de classification égal à 86.53% en appliquant notre ID3 amélioré sur le jeu de données Sentiment140 et un taux de classification égal à 88.82% en appliquant l’ID3 amélioré sur le jeu de données COVID-19_Sentiments. Par conséquent, dans la suite de ce travail, nous utiliserons uniquement le Gain d’Information comme sélecteur

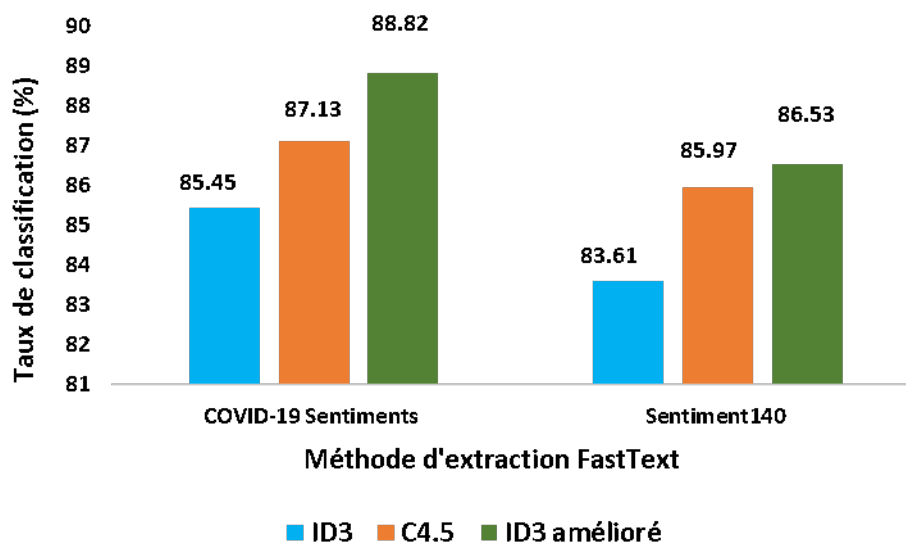


FIGURE 4.12 – TC en appliquant l’extracteur FastText sur Sentiment140 et COVID-19-Sentiments sous Hadoop.

de caractéristiques.

TABLE 4.5 – TC en appliquant ID3, C4.5, et notre ID3 amélioré (ID3A) avec différents sélecteurs (1,2,3,4) sur les deux jeux de données.

	Sentiment140			COVID-19_Sentiments		
	ID3	C4.5	ID3A	ID3	C4.5	ID3A
1	67.14	70.91	73.35	76.45	77.95	79.20
2	75.70	77.26	80.16	84.67	86.51	88.13
3	83.61	85.97	86.53	85.45	87.13	88.82
4	57.02	59.41	60.62	67.59	72.32	74.70

Après de multiples expériences, nous déduisons que dans la phase de représentation des données, la technique **FastText** surpasse toutes les autres méthodes (N-grammes, Sac de Mots, TF-IDF, GloVe, Word2Vec). Elle a atteint un taux de classification égal à 88.82% lorsqu’elle est appliquée au jeu de données COVID-19_Sentiments et un taux de classification égal à 86.53% lorsqu’elle est appliquée au jeu de données Sentiment140. Ensuite, dans la phase de sélection des caractéristiques, les résultats empiriques ont prouvé que le **Gain d’Information** utilisé comme méthode de sélection des caractéristiques surpasse tous les autres sélecteurs de caractéristiques (Indice de Gini, Ratio de Gain et Khi-carré.) en termes de taux de classification puisqu’il a atteint un taux de classification égal à 86.53% lorsqu’il est appliqué sur Sentiment140 et un taux de classification égal à 88.82% lorsqu’il est appliqué sur COVID-19_Sentiments. Enfin, nous avons appliqué le ID3 amélioré [165] comme classificateur. La figure 4.13 illustre la structure finale de la méthode hybride que nous proposons.

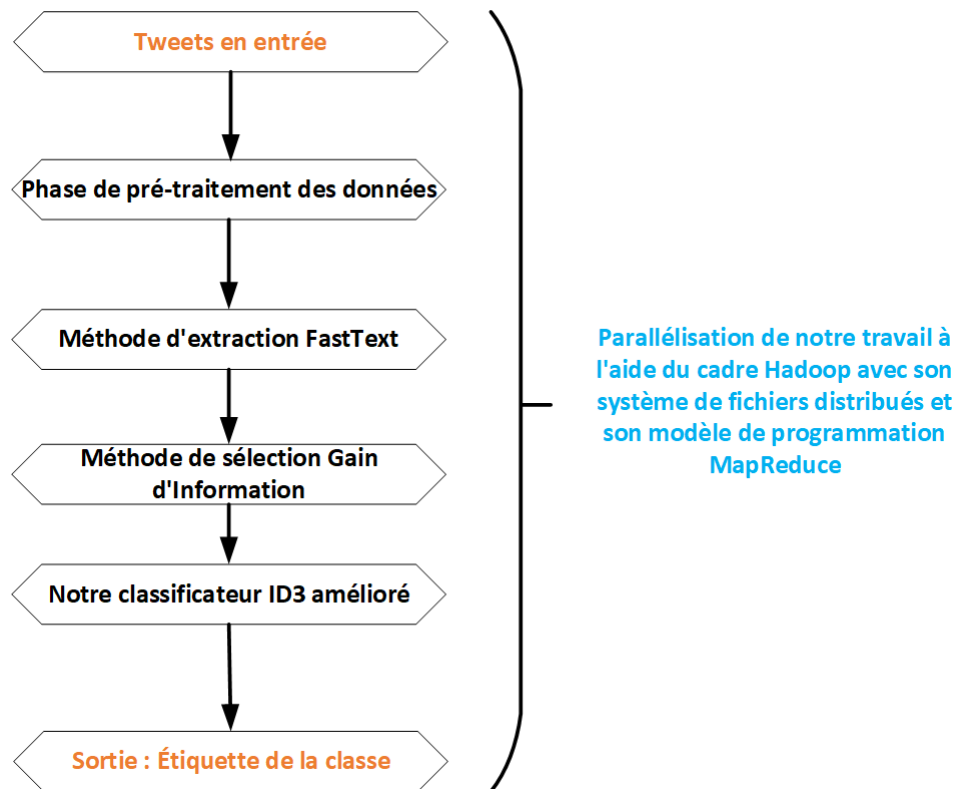


FIGURE 4.13 – Structure finale de la méthode hybride proposée sous le cadre Hadoop.

4.5.4 Évaluation de performances de notre approche

Dans cette expérience, nous allons présenter les résultats expérimentaux de notre méthode hybride proposée. Ces résultats empiriques sont obtenus en pratiquant notre classificateur proposé et d'autres méthodes telles que ID3, C4.5, Soni *et al.* [171], Ngoc *et al.* [172], Lakshmi *et al.* [183], AitAddi *et al.* [188], Patel *et al.* [189], et Wang *et al.* [190] sur les jeux de données choisis Sentiment140 et COVID-19_Sentiments. Pour démontrer laquelle de ces méthodes est la plus efficace et a les meilleures performances, nous mesurons dix paramètres d'évaluation TC , TR , TE , TPV , TNF , TNV , TPF , PR , SK , et SF comme expliqué précédemment. Notre classificateur sera exécuté en parallèle, sous Hadoop avec le système de fichiers distribué Hadoop et le paradigme de programmation MapReduce. Notre cluster Hadoop comprend quatre nœuds esclaves et un nœud maître.

La figure 4.14, illustre le résultat expérimental de la quatrième expérience. Comme on peut le voir sur la figure 4.14, nous remarquons que notre classificateur proposé surpasse les autres approches mises en œuvre en termes de taux de classification. Il a atteint un TC plus élevé égal à 86.53% et 88.82% lorsqu'il a été appliqué aux jeux de données Sentiment140 et COVID-19_Sentiments, respectivement. Et la méthode AitAddi *et al.* [188] a un TC plus faible égal à 43.02 %, et 31.08% lorsqu'il a été appliqué aux jeux de données Sentiment140 et COVID-19_Sentiments.

La figure 4.15, illustre le résultat expérimental obtenu pour le TR en appliquant notre classificateur suggéré et les approches précédemment choisies sur les jeux de données Sentiment140 et COVID-19_Sentiments. La figure 4.15 montre que notre classificateur

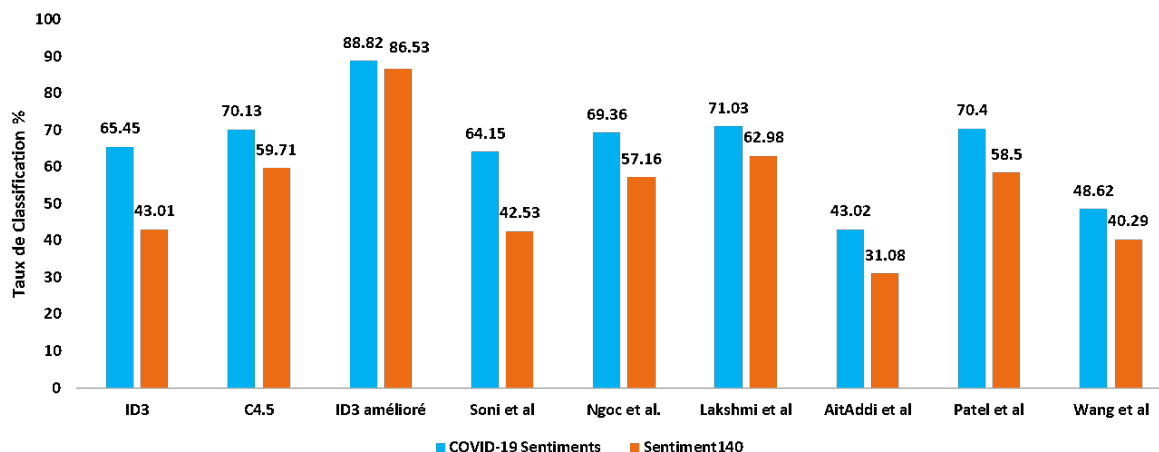


FIGURE 4.14 – TC en appliquant notre classificateur et d'autres approches.

proposé a un TR plus faible par rapport aux autres approches mises en œuvre. Et si nous comparons notre classificateur avec AitAddi *et al.* [?] qui a un TR plus élevée, nous constatons que notre classificateur réduit le TR de 56.98 %, et 68.92 % (AitAddi *et al.* [188]) à 11.18 %, et 13.47 % (notre ID3 amélioré) pour les jeux de données Sentiment140, et COVID-19_Sentiments respectivement.

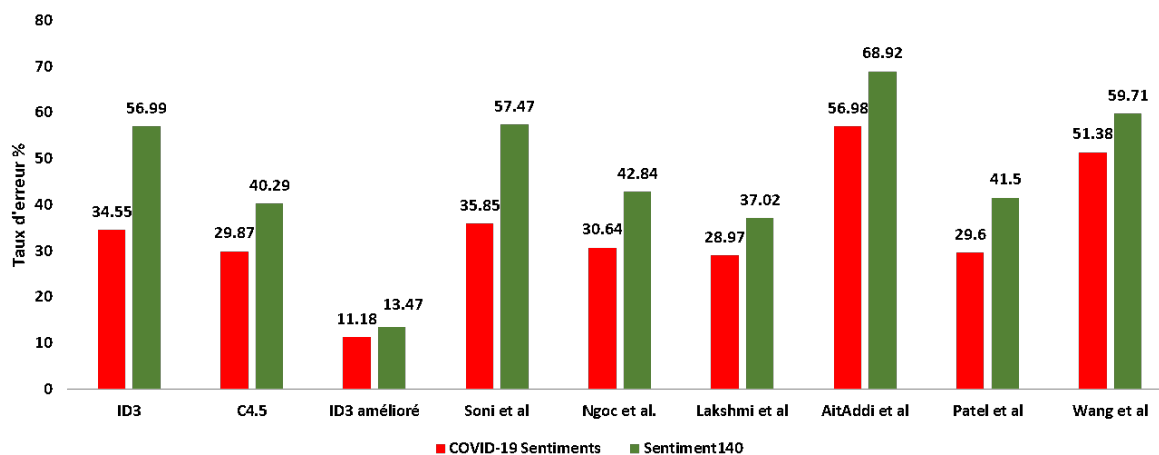


FIGURE 4.15 – TR en appliquant notre classificateur et d'autres approches.

Une autre expérience est menée pour examiner le temps d'exécution entre notre classificateur et les autres méthodes choisies. Sans oublier que notre classificateur est exécuté en mode parallèle sur cinq ordinateurs en utilisant le framework Hadoop. La figure 4.16 montre que notre modèle proposé a un TE plus faible par rapport aux autres méthodes mises en œuvre. Et si nous comparons notre classificateur avec la méthode Patel *et al.* [189], nous remarquons que notre classificateur réduit le TE de 5402.15 secondes (approche Patel *et al.* [189]) à 45.21 secondes (notre ID3 amélioré) pour le jeu de données Sentiment140 et de 842.91 secondes (approche Patel *et al.* [189]) à 15.95 secondes (notre ID3 amélioré) pour le jeu de données COVID-19_Sentiments. La comparaison entre ID3 et notre classificateur confirme que l'implémentation de notre classificateur en utilisant le cluster Hadoop de cinq noeuds de calcul est un outil plus efficace pour réduire le temps

d'exécution.

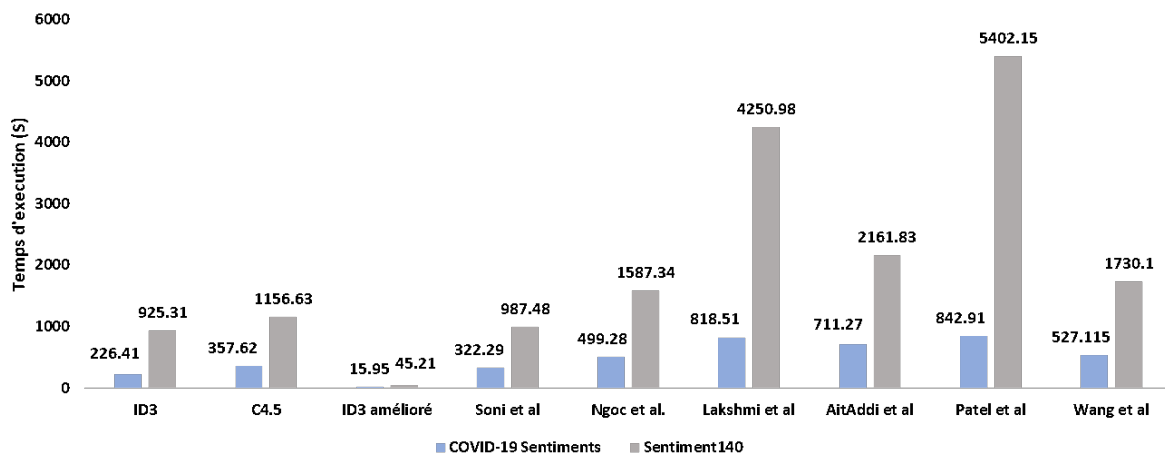


FIGURE 4.16 – TE en appliquant notre classifieur et d'autres approches.

Pour mieux prouver la performance de notre classifieur, nous avons utilisé d'autres mesures d'évaluation telles que TPV , TNF , TNV , TPF , PR , SK , et SF . Le tableau 4.6 montre les résultats expérimentaux obtenus et nous déduisons que notre classifieur surpasse tous les autres classificateurs appliqués aux deux jeux de données utilisés COVID-19_Sentiments et Sentiment140. De plus notre classifieur a obtenu des valeurs plus élevées au niveau de toutes les mesures d'évaluation calculées.

4.5.5 Examen de notre approche en termes de complexité, convergence et stabilité

L'objectif principal de cette dernière expérience est de comparer notre classifieur proposé avec Soni *et al.* [171], Ngoc *et al.* [172], Lakshmi *et al.* [183], AitAddi *et al.* [188], Patel *et al.* [189], et Wang *et al.* [190], et de trouver le classifieur le plus efficace parmi tous les classificateurs évalués en termes de stabilité, de complexité et de convergence.

4.5.5.1 Complexité

Par définition, le taux de complexité d'un classifieur est un critère pour mesurer l'espace employé et le temps consommé par un classifieur. Dans cette expérience, nous avons mesuré la complexité spatiale et la complexité temporelle de notre classifieur suggéré, Soni *et al.* [171], Ngoc *et al.* [172], Lakshmi *et al.* [183], AitAddi *et al.* [188], Patel *et al.* [189], et Wang *et al.* [190] approches. En résumé, le tableau 4.7 décrit les résultats obtenus en termes de complexité spatiale après avoir calculé la taille des instructions, et la taille des paramètres du classifieur proposé et des autres classificateurs implémentés.

Comme le montre le tableau 4.7, nous remarquons que le classifieur proposé a effectué plusieurs instructions qui occupent une taille mémoire égale à (29.106 M, 12.60 M) respectivement pour les jeux de données Sentiment140 et COVID-19_Sentiments. La

TABLE 4.6 – Résultats expérimentaux en termes de TPV , TNF , TNV , TPF , PR , SK , et SF

	TPV	TNF	TNV	TPF	PR	SK	SF
Notre classificateur	85.72	14.28	86.51	13.49	86.67	87.69	85.54
Soni <i>et al.</i> [171]	64.57	35.43	59.12	40.88	60.76	61.89	60.48
Ngoc <i>et al.</i> [172]	69.08	30.92	67.15	32.85	66.51	65.70	66.84
Lakshmi <i>et al.</i> [183]	71.46	28.54	70.89	29.11	72.32	73.51	70.54
AitAddi <i>et al.</i> [188]	43.54	56.46	44.62	55.38	42.27	43.25	42.92
Patel <i>et al.</i> [189]	71.92	28.08	70.52	29.48	72.69	70.56	71.96
Wang <i>et al.</i> [190]	49.11	50.89	48.17	51.83	47.92	49.43	48.69
Notre classificateur	81.41	18.59	82.33	17.67	83.04	84.58	83.87
Soni <i>et al.</i> [171]	43.56	56.44	42.71	57.29	40.19	41.79	42.64
Ngoc <i>et al.</i> [172]	56.24	43.76	57.19	42.81	56.92	54.61	55.24
Lakshmi <i>et al.</i> [183]	63.78	36.22	64.50	35.5	63.29	62.26	64.73
AitAddi <i>et al.</i> [188]	32.91	67.09	33.13	66.87	32.92	34.51	33.85
Patel <i>et al.</i> [189]	58.27	41.73	59.62	40.38	57.92	58.41	57.86
Wang <i>et al.</i> [190]	40.91	59.09	43.78	56.22	41.92	40.54	42.81

TABLE 4.7 – Complexité spatiale de notre classificateur et d'autres approches.

Nom du jeu de données	Techniques	N. instructions (M)	N. paramètres (M)
COVID-19_Sentiments	Notre classificateur	12.60	9.57
	Soni <i>et al.</i> [171]	13.51	11.03
	Ngoc <i>et al.</i> [172]	12.95	10.23
	Lakshmi <i>et al.</i> [183]	16.73	14.36
	AitAddi <i>et al.</i> [188]	15.41	12.98
	Patel <i>et al.</i> [189]	19.62	17.19
	Wang <i>et al.</i> [190]	17.25	15.48
	Sentiment140	Notre classificateur	29.106
Soni <i>et al.</i> [171]		32.71	26.58
Ngoc <i>et al.</i> [172]		31.21	24.65
Lakshmi <i>et al.</i> [183]		38.64	32.88
AitAddi <i>et al.</i> [188]		35.59	29.72
Patel <i>et al.</i> [189]		45.32	39.36
Wang <i>et al.</i> [190]		39.84	35.45

taille des paramètres du classificateur conçu est égale à (20.192 M, 9.57 M) respectivement pour les jeux de données Sentiment140 et COVID-19_Sentiments. Comme l'indique le résultat expérimental, le classificateur que nous proposons nécessite une complexité de calcul dans l'espace plus faible si on le compare à celui de Soni *et al.* [171], Ngoc *et al.* [172], Lakshmi *et al.* [183], AitAddi *et al.* [188], Patel *et al.* [189], et Wang *et al.* [190] méthodes.

Le tableau 4.8 présente les résultats de complexité temporelle obtenus après avoir mesuré les temps de formation et de test de notre classificateur proposé et d'autres classificateurs sélectionnés. Comme est illustré dans ce tableau, nous remarquons que le classificateur suggéré a consommé un temps d'apprentissage égal à 33.90 secondes, et 11.26 secondes respectivement pour les jeux de données Sentiment140 et COVID-19_Sentiments. De plus, le classificateur que nous proposons a consommé un temps de test égal à 11.30 secondes et 3.98 secondes respectivement pour les jeux de données Sentiment140 et COVID-19_Sentiments. Comme le montrent les résultats expérimentaux obtenus, le classificateur que nous proposons est beaucoup moins complexe en termes de temps de calcul si on le compare à celui de Soni *et al.* [171], Ngoc *et al.* [172], Lakshmi *et al.* [183], AitAddi *et al.* [188], Patel *et al.* [189], et Wang *et al.* [190] approches.

4.5.5.2 Convergence

Le classificateur proposé sera démontré s'il est convergent ou non convergent en trouvant un nombre particulier de tours d'entraînement dans lequel le classificateur proposé satisfait la condition décrite dans l'équation 4.7. Cette équation définit la condition de la convergence :

$$E_{rp} - E_{rc} \geq T_{re} \quad (4.7)$$

TABLE 4.8 – Complexité temporelle de notre classificateur et d'autres approches.

Nom du jeu de données	Techniques	Temps de formation (s)	Temps de test (s)
COVID-19_Sentiments	Notre classificateur	11.96	3.98
	Soni <i>et al.</i> [171]	241.71	80.57
	Ngoc <i>et al.</i> [172]	374.46	124.82
	Lakshmi <i>et al.</i> [183]	613.88	204.62
	AitAddi <i>et al.</i> [188]	533.45	177.81
	Patel <i>et al.</i> [189]	632.18	210.72
	Wang <i>et al.</i> [190]	395.33	131.77
Sentiment140	Notre classificateur	33.90	11.30
	Soni <i>et al.</i> [171]	740.61	246.87
	Ngoc <i>et al.</i> [172]	1190.50	396.83
	Lakshmi <i>et al.</i> [183]	3250.98	1062.74
	AitAddi <i>et al.</i> [188]	1621.37	540.45
	Patel <i>et al.</i> [189]	4051.61	1350.53
	Wang <i>et al.</i> [190]	1297.57	432.52

Où E_{rp} est l'erreur moyenne du classificateur du tour d'apprentissage précédent, E_{rc} est l'erreur moyenne du classificateur du tour d'apprentissage actuel, et T_{re} est la valeur seuil qui définit la valeur du taux de convergence. Après avoir effectué plusieurs expériences, nous avons fixé ce taux seuil à 0.0001. Par conséquent, l'erreur moyenne du classificateur suggéré est mesurée à l'aide de l'équation suivante :

$$E = \frac{1}{2} \times \frac{\sum_{j=1}^S \sum_{i=1}^C (y - y_{label})^2}{S} \quad (4.8)$$

Où S est le nombre total d'instances dans l'ensemble de données utilisé, C est le nombre total d'étiquettes de classe (dans notre cas, il y a trois étiquettes de classe qui sont négatives, neutres et positives), y est la décision de classification souhaitée en sortie, et y_{label} la décision de classification obtenue en sortie. Si l'équation (4.7) est vérifiée, notre classificateur proposé peut être considéré comme convergent, et l'algorithme est exécuté jusqu'à ce que l'erreur moyenne du classificateur satisfasse la condition. Dans le cas contraire, notre classificateur proposé ne converge pas.

La figure 4.17 illustre le taux de convergence de notre classificateur suggéré lorsqu'il est appliqué aux jeux de données Sentiment140 et COVID-19_Sentiments. Comme le montre la figure 4.17, nous percevons que le classificateur proposé a convergé vers la valeur seuil de 0.0001 après que l'algorithme de notre classificateur ait atteint 90 et 270 tours lorsqu'il a été appliqué au jeu de données COVID-19_Sentiments et Sentiment140, respectivement.

Le tableau 4.9 présente la valeur du taux de convergence de notre classificateur proposé, Soni *et al.* [171], Ngoc *et al.* [172], Lakshmi *et al.* [183], AitAddi *et al.* [188], Patel *et al.* [189], et Wang *et al.* [190] approches. Comme on peut le voir dans le tableau 4.9, nous concluons que notre classificateur proposé converge très rapidement.

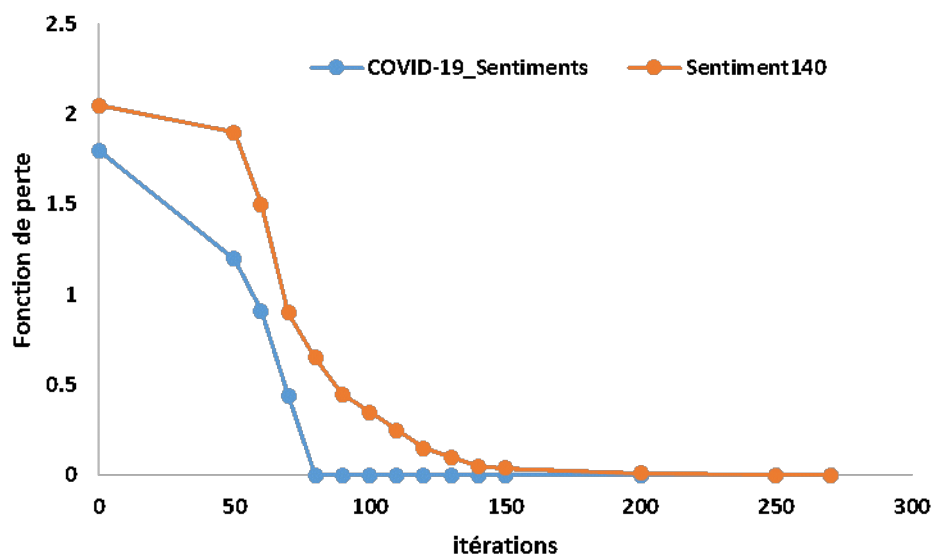


FIGURE 4.17 – Taux de convergence en appliquant notre classificateur sur COVID-19-Sentiments et Sentiment140.

TABLE 4.9 – Taux de convergence de notre classificateur et d’autres approches.

Jeux de données	Algorithmes	Itérations
COVID-19_Sentiments	Notre classificateur	90
	Soni <i>et al.</i> [171]	150
	Ngoc <i>et al.</i> [172]	102
	Lakshmi <i>et al.</i> [183]	215
	AitAddi <i>et al.</i> [188]	184
	Patel <i>et al.</i> [189]	426
	Wang <i>et al.</i> [190]	357
Sentiment140	Notre classificateur	270
	Soni <i>et al.</i> [171]	461
	Ngoc <i>et al.</i> [172]	317
	Lakshmi <i>et al.</i> [183]	645
	AitAddi <i>et al.</i> [188]	552
	Patel <i>et al.</i> [189]	1278
	Wang <i>et al.</i> [190]	1071

4.5.5.3 Stabilité

L’objectif principal de cette étape est de déterminer le classificateur le plus stable parmi tous les classificateurs mis en œuvre. Le tableau 4.10 présente l’écart type moyen et le taux de classification moyen (TCM) obtenus pour notre classifieur par rapport à Soni *et al.* [171], Ngoc *et al.* [172], Lakshmi *et al.* [183], AitAddi *et al.* [188], Patel *et al.* [189], et Wang *et al.* [190] sur les cinq validations croisées des deux jeux de données utilisés dans cette contribution.

Comme l’illustre le tableau 4.10, nous remarquons que notre classifieur est plus stable que les autres classifieurs car il a obtenu un taux de classification moyen plus élevé (88.82

TABLE 4.10 – Stabilité de notre classificateur et d'autres approches.

Jeux de données	Algorithmes	TCM (%)	ETM (%)
COVID-19_Sentiments	Notre classificateur	88.82	0.12
	Soni <i>et al.</i> [171]	64.15	1.95
	Ngoc <i>et al.</i> [172]	69.36	1.06
	Lakshmi <i>et al.</i> [183]	71.03	0.91
	AitAddi <i>et al.</i> [188]	43.02	4.08
	Patel <i>et al.</i> [189]	70.40	0.98
	Wang <i>et al.</i> [190]	48.62	3.56
Sentiment140	Notre classificateur	86.53	0.26
	Soni <i>et al.</i> [171]	42.53	3.45
	Ngoc <i>et al.</i> [172]	57.16	3.68
	Lakshmi <i>et al.</i> [183]	62.98	2.31
	AitAddi <i>et al.</i> [188]	31.08	7.82
	Patel <i>et al.</i> [189]	58.50	2.55
	Wang <i>et al.</i> [190]	40.29	4.48

%, et 86.53 %) avec un écart-type moyen très faible (0.12 %, et 0.26 %) lorsqu'il a été appliqué aux jeux de données COVID-19_Sentiments et Sentiment140, respectivement.

4.6 Conclusions

En conclusion, l'objectif de cette étude est d'augmenter la performance de classification de la fouille d'opinion en incorporant les points forts des techniques de pré-traitement des données pour améliorer la qualité des données en supprimant les données indésirables et bruyantes, les méthodes d'extraction des caractéristiques pour convertir les données textuelles en valeurs numériques et extraire les caractéristiques les plus pertinentes, les sélecteurs de caractéristiques pour réduire la haute dimensionnalité des caractéristiques extraites dans l'étape précédente et sélectionner les caractéristiques les plus intéressantes et notre algorithme amélioré d'arbre de décision ID3 pour classer le tweet d'entrée et améliorer la précision de la classification. L'architecture universelle de l'approche suggérée dans ce chapitre est composée de cinq étapes, à savoir l'étape de collecte des données, la phase de prétraitement des données, l'étape de représentation des données, les méthodes de sélection des caractéristiques, l'algorithme d'arbre de décision ID3 amélioré et l'utilisation d'Hadoop pour paralléliser la méthode hybride proposée.

Amélioration de l'algorithme d'arbre de décision C4.5 basé sur le MapReduce et le système de règles floues

5.1 Introduction

La technique de classification est la plus importante technique dans le domaine de la fouille de données et elle est très largement utilisée dans divers domaines. La classification est une fonction de la fouille de données qui attribue les éléments d'une collection à des catégories ou des classes de décision. En général, le principe dominant de la classification est de prévoir avec précision la décision ou la classe cible pour chaque élément de l'ensemble de données en utilisant le modèle construit [199]. Les données historiques d'un projet de classification sont typiquement divisées en deux ensembles de données : l'un pour la construction du modèle, l'autre pour tester le modèle. L'algorithme de classification le plus utilisé est l'algorithme de l'arbre de décision [200].

L'algorithme d'arbre de décision C4.5 est un arbre orienté composé d'un nœud racine, ainsi que de nœuds de décision et d'autres nœuds internes. Afin de construire un arbre de décision, le processus est le suivant : Étant donné un ensemble de données de formation, appliquez une fonction de mesure sur tous les attributs disponibles, trouvez le meilleur attribut de partitionnement en fonction du résultat obtenu par le calcul de la fonction de mesure, une fois le meilleur attribut déterminé, l'ensemble de données est divisé en de nombreuses partitions en fonction des plages de valeurs ou du nombre de valeurs associées au meilleur attribut. Au sein de chaque partition, si tous les échantillons appartiennent à une seule classe, l'algorithme s'arrête [201, 202]. Sinon, la procédure de partitionnement est exécutée de manière récursive jusqu'à ce que chaque partition appartienne à une seule classe, ou qu'il ne reste aucun attribut. Dans ce domaine de la recherche scientifique, toutes les recherches portent sur le problème de trouver les meilleurs critères de partitionnement de l'algorithme de l'arbre de décision afin de construire de petits arbres précis et de réduire le temps d'exécution pour un ensemble de données donné [199].

L'une des excellentes caractéristiques de l'arbre de décision est qu'il n'exige pas beaucoup de connaissances de base dans la procédure d'apprentissage puisque l'ensemble des données de formation peut donner l'expression par l'attribut qui est la conclusion du mo-

dèle [201]. Ensuite, utilisez l'algorithme pour l'apprentissage. Les algorithmes d'arbre de décision ont les avantages suivants : (1) la structure de l'algorithme est simple, facile à comprendre ; (2) l'algorithme a une précision prédictive élevée. Mais aujourd'hui, les algorithmes traditionnels des arbres de décision ont rencontré de nombreux défis en raison de la croissance plus rapide des données. Premièrement, comme la quantité de données est devenue extrêmement massive, le processus de construction d'un modèle d'arbre de décision peut prendre beaucoup de temps. Deuxièmement, plusieurs calculs ont été déplacés vers le stockage externe car la capacité de stockage de la mémoire est limitée. Augmentez donc le coût des entrées/sorties. Dans notre travail, pour surmonter ces défis, nous utilisons le grand cadre de données Hadoop avec sa composante modèle de calcul MapReduce et le système de fichiers distribués HDFS.

Actuellement, les données massives (Big Data) sont la capacité d'extraire des modèles ou des informations utiles à partir de données à grande échelle [203]. Pour traiter cette énorme quantité de données à l'aide d'un seul ordinateur, il est inefficace en temps réel. Pour résoudre ce problème, le cadre de traitement des données massives est déployé sur des ordinateurs en clusters avec une plate-forme de calcul haute performance, et les tâches de fouille de données sont déployées sur ce cluster d'ordinateurs en exécutant le cadre parallèle de données de haut niveau Hadoop. Apache Hadoop est un logiciel à code source ouvert qui facilite efficacement l'écriture d'applications distribuées. Il contient deux composants, le système de fichiers distribués HDFS, et le modèle de programmation MapReduce.

Les systèmes flous peuvent être définis comme des systèmes qui utilisent la théorie des ensembles flous proposée par le professeur L.A. Zadeh [204] pour représenter au moins une de ses variables. La théorie des ensembles flous permet la représentation et le traitement d'informations imprécises et incertaines, qui sont abondantes dans le monde réel. En fait, la plupart des approches informatiques disponibles ne peuvent pas traiter directement les informations avec imprécision et incertitude, ce qui fait des systèmes flous une alternative intéressante pour travailler avec des domaines présentant de telles caractéristiques. Les systèmes flous basés sur des règles sont un type particulier du système flou qui utilise un mécanisme de raisonnement basé sur le raisonnement approximatif qui a la capacité d'exprimer l'ambiguïté et la subjectivité présentes dans le raisonnement humain. La règle basée sur des systèmes flous stocke des connaissances représentées par des règles [205]. Un système flou se compose d'une base de connaissances et d'un mécanisme d'inférence. La base de connaissances contient une base de règles floues et une base de données floues. La base de règles floues contient les règles qui forment le cœur du système. Ces règles sont construites à partir des ensembles flous définissant les attributs du système qui sont stockés dans la base de données floue. La base de données floue et la base de règles floues sont utilisées par le mécanisme d'inférence pour classer de nouveaux exemples [205].

Dans ce chapitre, nous proposons une nouvelle approche pour classer les données, en utilisant les notions de logique floue, d'algorithme d'arbre de décision C4.5 basé sur le gain d'informations floues, et de logiciel à code source ouvert Hadoop. La première étape consiste à fuzzifier les données à classer (transformer l'ensemble net en ensemble flou) en utilisant les méthodes de fuzzification (fonction d'appartenance de forme trapézoïdale ou fonction d'appartenance de forme triangulaire) et les stocker dans le HDFS. Une fois les données stockées dans le HDFS, nous parallélisons les instructions de l'algorithme flou C4.5 appliqué sur les données en utilisant le modèle de programmation MapReduce. Nous

pouvons en déduire que le but de notre nouvelle méthode est de fuzzifier l'algorithme C4.5 afin de traiter les données incertaines et imprécises, et afin de classer un énorme jeu de données en utilisant cet algorithme fuzzifié sans avoir le problème du temps d'exécution, nous parallélisons notre méthode sous le cadre Hadoop.

Le reste de ce chapitre est organisé comme suit : La section 5.2 présente les travaux antérieurs. Dans la section 5.3.1 nous décrivons la motivation de notre travail. Dans la section 4 nous présentons notre méthodologie de recherche. Ensuite, dans la section 5 nous donnons les résultats des expériences et des comparaisons effectuées pour démontrer la précision de notre modèle proposée. Finalement, nous concluons ce chapitre dans la section 6.

5.2 Travaux connexes

Plusieurs articles de recherche dans la littérature poursuivent pour étudier, interpréter et identifier les problèmes de classification à l'aide de méthodes de logique floue, et leurs applications dans divers domaines. La logique floue (LF) est l'une des techniques informatiques douces qui a joué un rôle crucial dans la construction de modèles de classification hybrides ces dernières années. LF suggéré par le professeur L.A. Zadeh [204] explique la manière de représenter la pensée et la perception humaines en particulier dans divers domaines tels que la fouille de données, l'abstraction de l'information, l'apprentissage automatique, la reconnaissance de formes, le traitement du langage naturel et d'autres domaines qui résolvent les problèmes d'incertitude. Ces problèmes ambigus et d'incertitude peuvent être résolus par différentes méthodes de fuzzification qui sont appliquées pour transformer l'ensemble net d'entrée en ensemble flou.

Ducange et al [206] proposent un modèle de classification associative, floue, distribuée, et efficace basé sur le modèle de programmation MapReduce. La première étape de leur approche vise à extraire un ensemble de règles de classification d'associations floues en utilisant l'extension floue de l'algorithme d'apprentissage FP-Growth, puis ils élaguent l'ensemble de règles obtenu en utilisant des outils d'élagage tels que fuzzysuppConfL, minFuzzysupp, et minFuzzyConf. L'objectif de ce processus d'élagage est de réduire les règles bruitant et redondantes générées dans la première phase de l'approche proposée. Ils ont mis en œuvre leur travail sous le cadre Hadoop, ils étudient également l'évolutivité de leur travail en réalisant de nombreuses expériences sur un énorme ensemble de données réel.

Les auteurs d'article de recherche [207] ont proposé un système flou qui peut extraire les principaux aspects des opinions touristiques, puis classer ces aspects extraits dans la catégorie positive ou négative ; ils utilisent des algorithmes basés sur la logique floue dans les deux phases : la classification des aspects et l'extraction des aspects. Ils ont évalué cinq algorithmes courants basés sur la logique floue, à savoir FURIA, FLR, FNN, VQNN et FRNN afin de choisir le meilleur. Selon le résultat présenté, l'algorithme FURIA a donné de bons résultats par rapport aux autres algorithmes d'apprentissage flou avec une précision de 90,12% sur l'ensemble des données du restaurant et le classificateur FLR a obtenu un meilleur résultat avec une précision de 86,02% sur l'ensemble des données

de l'hôtel. En général, leur travail se déroule en quatre phases, la collecte des données, le prétraitement des données, l'extraction des règles floues et l'étape de classification à l'aide d'algorithmes de logique floue.

Abdul-Jaleel et al. [208] ont proposé une approche combinant l'algorithme génétique et la théorie de la logique floue pour résoudre la question de la classification des textes en fonction du degré d'appartenance. Les entrées de leur application de classification proposée sont un ensemble de caractéristiques obtenues à partir d'un tweet et le résultat de ce système de classification est la classe (négative, neutre, positive) à laquelle le tweet lui appartient. Les résultats obtenus grâce au système proposé sont comparés à la technique de la logique floue et à la technique de recherche par mots-clés. Cette comparaison est basée sur les deux taux, qui sont le taux de correction et le taux incrémental. Dans le taux incrémental, leur système de classification est plus efficace que ces techniques (recherche par mot-clé et logique floue), où le nombre de tweets extraits en utilisant l'approche proposée, est de 160 tweets contre 98 et 141 en utilisant les autres techniques. En outre, le système de classification proposé a obtenu un meilleur résultat avec un taux de correction de 98.75% par rapport à un taux de correction de 97.9% et de 95.7% obtenu par d'autres techniques.

Les auteurs de [209] présentent une méthodologie hybride pour classer le sol à l'aide des cartes de couleurs de sol Munsell. Dans leur approche proposée, ils résolvent le problème de la classification des sols en combinant des systèmes de logique floue et des réseaux de neurones artificiels. Melin et al. [210] développent une nouvelle approche pour l'adaptation des paramètres dynamiques dans l'optimisation des essaims de particules (OSP), où la OSP est une métaheuristique inspirée des comportements sociaux. Dans ce travail, les auteurs ont également utilisé la logique floue afin d'améliorer la variété et la convergence de l'essaim dans l'OSP. Les résultats de l'expérience prouvent que leur proposition a donné de bons résultats en termes de performance de la OSP. Les auteurs de l'article [211] présentent un nouvel algorithme de regroupement appelé C-Means possibilistes flous (CMPF). Cet algorithme proposé est basé sur la technique de la logique floue de type-2. L'objectif de ce travail est d'améliorer les performances du CMPF. Plusieurs simulations ont été réalisées en appliquant l'algorithme de C-Means flou de type-2 à intervalle et le CMPF sur 6 jeux de données bien connus. Les auteurs de ces articles de recherche [212–214] ont proposé des nouvelles techniques d'apprentissages automatiques pour résoudre le problème de la classification dans certains domaines tels que la reconnaissance des formes et la classification des maladies liées au diabète.

Les auteurs de [215] présentent un travail qui associe à la fois les parties prenantes et les décideurs de l'entreprise afin de choisir le meilleur fournisseur. Dans leur travail, les auteurs convertissent l'ensemble des opinions extraites en un ensemble flou et doux, puis combinent l'ensemble flou et doux obtenu avec la théorie de l'approximation grossière. Les attributs dans ce travail sont représentés par des termes linguistiques. Pour évaluer l'efficacité et la performance de leur méthode proposée, les auteurs ont présenté une étude de cas utilisant leur technique améliorée. En outre, de nombreux travaux dans la littérature ont exploité la possibilité de combiner la théorie des ensembles flous avec les algorithmes des arbres de décision pour traiter les données d'incertitude. Et ces algorithmes d'arbre de décision flous ont été utilisés avec succès dans plusieurs domaines tels que les applications industrielles, la prise de décision, l'apprentissage automatique, l'ingénierie des connaissances et la fouille

de données. Dans cette section, je vais décrire certains de ces travaux de recherche.

Les auteurs de l'article [216] ont proposé une nouvelle méthode basée sur la logique floue pour la classification multi-étiquettes. Le nouvel algorithme utilise une entropie floue généralisée, et des étiquettes globales agrégées pour sélectionner le meilleur attribut pour la croissance de l'arbre. Les raisons adoptées par les auteurs pour améliorer cette nouvelle décision floue sont deux raisons : Premièrement, l'interopérabilité intrinsèque des systèmes flous donne une certaine anticipation ou explication sur la classification. Ce qui est une caractéristique très importante dans plusieurs découvertes de connaissances et tâches de fouille de données. Deuxièmement, la nouvelle méthode présente plusieurs degrés d'ambiguïté parmi les limites des étiquettes, qui ne peuvent être correctement découvertes par les classificateurs classiques.

Un autre travail qui utilise des ensembles flous dans l'arbre de décision est celui présenté dans [217] ; dans cet article, les auteurs ont introduit une approche consistant à utiliser des estimations d'informations cumulées pour l'induction d'arbres de décision flous. Ils ont proposé un nouveau type d'arbre de décision flou appelé arbre ordonné. Cet arbre est utilisé pour traiter les attributs de manière parallèle avec des coûts différents. L'arbre non ordonné diffère de l'arbre de décision flou ordonné en ce qui concerne la manière de tester les attributs. Dans l'arbre ordonné, l'ordre de l'attribut testé n'est pas lié aux résultats des tests précédents, donc ils peuvent examiner les attributs suivants de manière parallèle. Cela permet de réduire les coûts des attributs testés.

Suryawanshi et Thakore [218] ont proposé une méthode qui intègre la théorie des ensembles flous à l'algorithme de l'arbre de décision ID3. Cet article se concentre essentiellement sur la méthode de classification de la fouille des données pour reconnaître la classe d'un attribut en utilisant l'algorithme de l'arbre de décision ID3, puis il ajoute le principe de fuzzification afin d'améliorer les performances d'ID3. Les auteurs de l'article [219] présentent une approche hybride, qui combine la sélection d'échantillons basée sur l'ambiguïté maximale et l'induction d'arbre de décision flou. Cet article présente une nouvelle technique de sélection d'échantillons, c'est-à-dire la sélection d'échantillons basée sur l'ambiguïté maximale dans l'induction d'arbre de décision floue. Les résultats expérimentaux montrent que la capacité de généralisation de l'arbre utilisant cette nouvelle méthode de sélection est plus performante que celle basée sur la technique de sélection aléatoire.

5.3 Première approche hybride

5.3.1 Motivation

L'idée de la théorie de la logique floue vise à analyser les données recueillies dans différents domaines d'une manière qui est similaire aux sentiments des êtres humains [220], contrairement à la stratégie d'analyse traditionnelle. La sortie d'un système flou est obtenue grâce à l'application des fonctions d'appartenance sur les entrées et les sorties, ce processus est appelé processus de fuzzification. Une entrée claire sera transformée en les différents membres des fonctions d'appartenance connexes fondées sur sa valeur. En

outre, la sortie du système de la logique floue est dérivée de ses appartenances aux diverses fonctions d'appartenance, qui peuvent être traitées comme un ensemble d'entrées [221].

Les idées de la logique floue sont souvent utilisées dans notre vie quotidienne et personne n'y prête même attention. Par exemple, pour répondre à quelques questions dans certaines enquêtes, la personne pouvait toujours répondre par «Non satisfait» ou «Entièrement satisfait», qui sont également des réponses vagues, ambiguës et floues. Précisément dans quelle mesure une personne est-elle satisfaite ou mécontente de certains produits ou services pour ses enquêtes? Ces réponses ambiguës ne peuvent être créées que par des êtres humains, mais pas par des machines [220]. Est-il possible pour une machine de répondre immédiatement à ces questions d'enquête comme les êtres humains l'ont fait? C'est définitivement impossible. Les machines ne peuvent comprendre que «FALSE» ou «TRUE» et «0» ou «1». Ces informations sont appelées des données précises et peut-être traité par tous les ordinateurs. Est-il possible que l'être humain aide les machines à traiter ces données vagues? Si tel est le cas, comment les machines et les ordinateurs peuvent-ils traiter ces données ambiguës? Oui, inspiré par les sentiments de l'être humain, le professeur L.A. Zadeh a proposé la logique floue qui peut aider les ordinateurs à traiter ces données vagues/ambiguës comme le font les êtres humains [204].

La logique floue est considérée comme une extension de la logique classique. En d'autres termes, la valeur de vérité prend un nombre réel de l'intervalle $[0, 1]$ dans la logique floue plutôt qu'une valeur binaire «0» ou «1» dans la logique classique. L'objectif principal de la théorie de la logique floue est de convertir un problème blanc et noir en un problème gris [204]. Dans les définitions de la théorie des ensembles, la logique classique ou déterministe considère l'ensemble des éléments comme un ensemble croustillant, ce qui signifie que le degré d'appartenance de chaque élément à un ensemble est égal à 1, c'est-à-dire que l'élément appartient entièrement à l'ensemble. Contrairement à la logique floue considère l'ensemble des éléments comme un ensemble flou, ce qui indique que le degré d'appartenance de chaque élément dans un ensemble flou est compris entre 0 et 1, c'est-à-dire que l'élément appartient partiellement à l'ensemble. Le degré d'appartenance est calculé par une fonction d'appartenance spécifique telle qu'une fonction d'appartenance triangulaire, une fonction d'appartenance gaussienne et une fonction d'appartenance trapézoïdale [222].

En général, les caractéristiques d'un processus d'apprentissage peuvent être divisées en deux catégories, à savoir les caractéristiques à valeur continue et les caractéristiques à valeur discrète. La première catégorie est considérée comme des concepts nominaux tandis que la seconde, comme des nombres réels. L'algorithme d'arbre de décision C4.5 suppose que toutes les valeurs de caractéristiques sont nominales. Par conséquent, les attributs à valeur continue doivent être discrétisés avant que le C4.5 mesure le critère de partitionnement. il existe plusieurs manières de discrétiser, mais la méthode efficace est le partitionnement binaire qui indique qu'une caractéristique à valeur continue est discrétisée au début du processus de l'algorithme d'apprentissage en divisant sa plage en deux intervalles [223]. Le partitionnement binaire est généralement effectué en sélectionnant la valeur de seuil qui diminue la mesure d'impureté (rapport de gain C4,5) utilisée comme critère de partitionnement [224]. Une fois que la valeur de seuil T est déterminée pour l'attribut à valeur continue A , les instances de l'ensemble d'apprentissage avec $A \leq T$ sont affectées à la branche du nœud gauche, tandis que les instances de l'ensemble d'apprentissage avec $A > T$ sont affectées à la branche du nœud droit.

C4.5 gère les caractéristiques à valeur continue en mettant les nombres réels dans deux intervalles différents en utilisant la technique de partitionnement binaire, chaque intervalle est utilisé comme un jugement de condition par le nœud actuel vers le nœud suivant. Dans la littérature, plusieurs travaux de recherche [217, 218, 223, 224] critiquent cette façon de traiter la caractéristique à valeur continue et la considèrent comme un biais de jugement. Motivés par l'efficacité et l'avantage des techniques de la logique floue pour résoudre le problème du biais de jugement dans plusieurs problèmes, nous avons proposé une nouvelle version de C4.5 en représentant les caractéristiques à valeur continue en utilisant des termes linguistiques flous au lieu de la technique de partitionnement binaire. Dans la section 5.3.2, je décrirai comment nous utilisons la technique de la logique floue avec l'algorithme C4.5 pour traiter les caractéristiques à valeur continue.

Du point de vue de certains articles de recherche, le système flou basé sur des règles (SFBR) est le domaine le plus important de la théorie des ensembles flous. Ce type de système est considéré comme une extension des systèmes traditionnels basés sur des règles, prenant en compte les règles SI ALORS dont le bloc conséquent et le bloc antécédent sont constitués de termes de la logique floue, au lieu de ceux de la logique traditionnelle. Comme indiqué dans [222], SFBR peut augmenter le taux d'interprétabilité des algorithmes d'apprentissage pour la classification des textes par rapport aux modèles de calcul. En général, le SFBR est un type particulier de systèmes experts, qui sont généralement composés d'un ensemble de règles floues. Chaque règle est un ensemble de termes linguistiques, que l'on appelle conditions ou antécédents. Dans la littérature, il existe trois types communs de RBFS, à savoir Sugeno, Tsukamoto et Mamdani [225]. Mamdani et Sugeno sont des systèmes flous utilisés dans les cas de problème de régression, et Tsukamoto est un système flou qui est généralement utilisé pour les problèmes de classification. Tsukamoto se compose de trois phases, qui sont la fuzzification, l'inférence et la défuzzification. Dans l'étape de fuzzification, le Tsukamoto utilise l'une des trois fonctions de fuzzification populaires, telles que la fonction d'appartenance triangulaire, la fonction d'appartenance gaussienne, et la fonction d'appartenance trapézoïdale [222], le mécanisme d'inférence est basé sur des connaissances d'experts, et dans l'étape de défuzzification, et dans l'étape de défuzzification, l'une des fonctions les plus populaires est utilisée comme la fonction d'appartenance maximale, la fonction de centroïde et la fonction de moyenne pondérée. Similaire au modèle de Tsukamoto, la méthode que nous proposons comprend trois phases, telles que l'étape de fuzzification effectuée en utilisant une fonction d'appartenance triangulaire, l'étape d'inférence réalisée en appliquant l'algorithme flou C4.5 [226] à l'ensemble de données fuzzifiées, et dans la dernière phase, nous avons appliqué les méthodes de raisonnement classique et général sur les règles floues extraites pour classer les nouvelles instances.

5.3.2 Méthodologie de recherche

Notre proposition vise à résoudre l'un des problèmes rencontrés par l'algorithme de l'arbre de décision C4.5 en utilisant des techniques de la logique floue. Le problème est de savoir comment C4.5 gère les attributs à valeur continue. En général, le C4.5 classique utilise le partitionnement binaire pour traiter les attributs à valeur continue, comme expliqué dans la section 5.3.1. Après de nombreuses expériences, analyses et études menées

par les chercheurs, il s'avère que la technique du partitionnement binaire n'est pas plus efficace et ils la considèrent comme un biais de jugement. Trouver un autre moyen de surmonter ce biais de jugement dans le processus d'apprentissage du C4.5 est la première phase de notre proposition. Après des études approfondies de la théorie de la logique floue, nous avons déduit que cette théorie est plus efficace pour résoudre le problème du biais de jugement dans plusieurs problèmes présentés dans la section 5.2. Par conséquent, nous avons décidé dans la première phase de notre proposition de fuzzifier l'ensemble de données en utilisant les techniques de fuzzification. Cette étape nous permet d'améliorer l'arbre de décision C4.5, au lieu du processus de discrétisé utilisant la technique de partitionnement binaire pour l'attribut à valeur continue, nous remplaçons la valeur continue d'un tel attribut par le terme linguistique avec le degré d'appartenance le plus élevé.

Dans la deuxième phase de notre approche, nous proposons un nouveau système flou basé sur des règles pour gérer les données d'incertitude et d'imprécision dans le processus de classification. Ce système se compose de trois étapes telles que l'étape de fuzzification présentée précédemment dans la première phase de notre approche, la phase d'inférence et la phase de classification. La phase d'inférence est la composante qui extrait l'ensemble des règles floues à partir de l'ensemble de données floues selon l'application de l'algorithme C4.5 qui est flou et parallèle sur les données floues. La phase de classification vise à classer les nouvelles instances en utilisant deux méthodes qui sont la méthode de raisonnement classique et la méthode de raisonnement générale. Par conséquent, l'intégration de la logique floue (en utilisant le terme linguistique flou pour représenter les caractéristiques à valeur continue) et du système flou basé sur des règles (conçu par l'algorithme C4.5 qui est flou et parallèle) peut faire apparaître les règles sous une forme extrêmement identique au langage naturel et peut donc rendre les connaissances générées par les règles plus interprétables et compréhensibles.

Pour plus de détails, dans cette section, nous allons présenter les différentes étapes de notre travail et décrire la méthodologie de notre système hybride. L'objectif de notre système hybride proposé est d'améliorer l'algorithme de l'arbre de décision C4.5 en utilisant la logique floue et de proposer un nouveau système à base de règles floues en utilisant notre C4.5 amélioré, afin de gérer l'incertitude et l'imprécision des données. La classification est faite en utilisant l'algorithme C4.5 flou, le système à base de règles floues et le cadre Hadoop, qui parallélise les tâches de classification entre cinq machines ; un nœud maître et quatre nœuds esclaves, en utilisant son système de fichiers distribués (HDFS) pour stocker l'ensemble de données à classifier et l'ensemble de données classifiées (le résultat de la classification), et le modèle de programmation MapReduce pour le processus et le développement de notre travail. Nous pouvons résumer notre travail dans les étapes suivantes :

- Stockez l'ensemble de données dans le système de fichiers distribué Hadoop.
- Appliquez la méthode de fuzzification pour fuzzifier l'ensemble de données en ensemble flou, dans ce travail, nous utilisons la méthode de fuzzification appelée fonction d'appartenance triangulaire (FA).
- Une fois le processus de fuzzification terminé, nous stockons l'ensemble des données fuzzifiées dans le système de fichiers HDFS.
- Après l'implémentation et l'exécution de notre C4.5 flou parallèle, l'arbre de décision flou parallèle est créé, et nous utilisons cet arbre de décision résultant pour

- déduire les règles floues (qui sont appelées règles d'inférence dans le système flou).
- Après le mécanisme d'inférence, nous utilisons deux méthodes qui sont la méthode raisonnement classique et la méthode de raisonnement générale pour classer les nouveaux exemples.
 - Enfin, stockez le résultat de la classification dans le système de fichiers HDFS.

La figure 5.1 présente l'organigramme de notre algorithme amélioré.

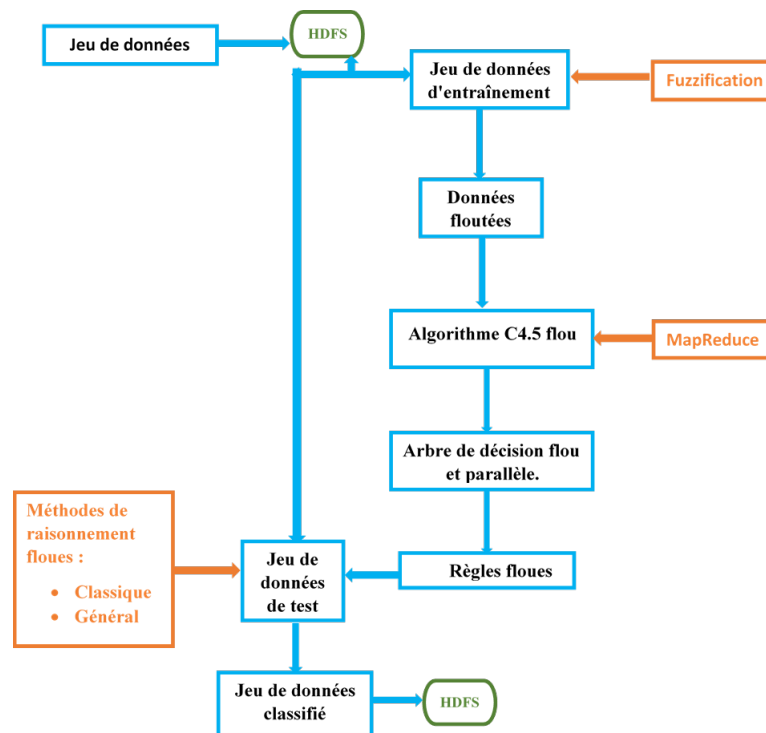


FIGURE 5.1 – Organigramme de notre algorithme amélioré.

5.3.2.1 Méthodes de fuzzification

Comme le montre la figure 5.1 et comme présenté précédemment, la première étape de notre méthode proposée consiste à stocker les données dans un système distribué HDFS, une fois le stockage effectué, nous divisons l'ensemble de données en deux sous-ensembles : l'ensemble de données d'apprentissage et l'ensemble de données de test en utilisant une stratégie de validation croisée (Cross-Validation). L'étape suivante consiste à fuzzifier l'ensemble des données d'apprentissage en utilisant la fonction d'appartenance (FA). L'objectif de cette étape est de prendre l'entrée nette et de calculer le degré auquel l'entrée nette appartient à chacun des ensembles flous appropriés (termes linguistiques). Dans notre cas, les entrées nettes sont les valeurs prises par chaque attribut dans notre ensemble de données et nous utilisons le FA triangulaire pour déterminer le degré d'appartenance. L'algorithme 3 illustre les étapes de la fuzzification de l'ensemble de données d'apprentissage.

Algorithm 3 Fuzzification de l'attribut continu

Input : Un ensemble de données donné décrit par des attributs \mathbf{m} et des exemples \mathbf{n} , et la base de données floue prédéfinie.

Output : Ensemble de données floues.

for $i \leftarrow 1$ **to** m **do**

for $j \leftarrow 1$ **to** n **do**

si L'attribut A est continu **alors**

for $k \leftarrow 1$ **to** *Le nombre total de termes linguistiques de l'attribut A* **do**

 Calculez le degré d'appartenance FA de la valeur d'entrée de l'attribut A, instance b , dans l'ensemble flou définissant le x terme linguistique de l'attribut A

end for

 Remplacez la valeur continue de l'attribut A, instance b , par la valeur linguistique ayant le degré d'appartenance le plus élevé.

end for

end for

return *Ensemble de données floues*

Notre algorithme de fuzzification prend en entrée l'ensemble des données d'apprentissage décrit par m attributs et n exemples ainsi que la base de données floue prédéfinie qui contient l'ensemble des termes linguistiques. La première étape de notre algorithme consiste à vérifier si l'attribut est continu, puis et pour chaque valeur linguistique, nous calculons le degré d'appartenance de la valeur d'entrée de l'attribut. Ensuite, nous remplaçons la valeur continue de l'attribut par le terme linguistique ayant le degré d'appartenance le plus élevé. Pour calculer le degré d'appartenance, nous utilisons le FA triangulaire, qui est déterminé par trois paramètres a , b et c est défini par l'équation (5.1).

$$f(x) = \begin{cases} 0 & \text{si } x \leq a \\ \frac{x-a}{b-a} & \text{si } a \leq x \leq b \\ \frac{c-x}{c-b} & \text{si } b \leq x \leq c \\ 0 & \text{si } c \leq x \end{cases} \quad (5.1)$$

Afin de calculer la fonction d'appartenance pour chaque valeur linguistique, il est nécessaire de déterminer les valeurs des paramètres scalaires a , b et c . Dans notre cas, nous calculons ces paramètres en utilisant la valeur maximale (Max) et minimale (Min) de chaque attribut dans tous les exemples de l'ensemble de données d'apprentissage. La première étape consiste à déterminer pour chaque attribut les valeurs max et min, puis nous calculons la moyenne de ces deux valeurs. Ensuite, nous déterminons la valeur des paramètres scalaires comme suit $a = \min$, $b = \text{moyenne}$ et $c = \max$.

Une fois que tous les attributs continus de notre ensemble de données de formation sont flous, l'étape suivante consiste à définir les règles floues. Et pour réaliser cette étape, nous appliquons l'algorithme de l'arbre de décision C4.5 basé sur le gain d'information floue, et qui est exécuté de manière parallèle en utilisant le modèle de programmation Map Reduce.

5.3.2.2 Algorithme d'arbre de décision C4.5 parallèle et flou

L'étape suivante est celle de la définition de la base de règles. Pour cela, nous appliquons l'algorithme d'arbre de décision C4.5 flou parallèle à l'ensemble des données d'apprentissage floues. L'approche que nous proposons intègre le principe de la logique floue, l'arbre de décision et le cadre Hadoop.

L'algorithme d'arbre de décision C4.5 est un arbre orienté composé d'un noeud racine, ainsi que de noeuds de décision et d'autres noeuds internes. Afin de construire un arbre de décision, le processus est le suivant : étant donné un ensemble de données d'apprentissage, appliquer une fonction de mesure sur tous les attributs disponibles, trouver le meilleur attribut de partitionnement en fonction du résultat obtenu par le calcul de la fonction de mesure, une fois le meilleur attribut est déterminé. L'ensemble de données est divisé en de nombreuses partitions en fonction des plages de valeurs ou du nombre de valeurs associées au meilleur attribut. Dans chaque partition, si tous les échantillons appartiennent à une seule classe, l'algorithme s'arrête. Sinon, la procédure de partitionnement est exécutée de manière récursive jusqu'à ce que chaque partition appartienne à une seule classe, ou qu'il ne reste plus aucun attribut.

D'autre part, le C4.5 flou intègre des arbres de décision avec un raisonnement ambigu par la logique floue pour gérer les incertitudes de langage. C4.5 flou utilise des termes linguistiques flous pour désigner les conditions de division des noeuds et autoriser les instances à suivre simultanément diverses branches avec différents degrés d'appartenance compris entre $[0,1]$. La construction de l'arbre de décision C4.5 flou est identique à celle du C4.5 classique, à la différence que, dans le processus d'apprentissage pour choisir le meilleur attribut de partitionnement. Le C4.5 classique calcule le rapport de gain d'information basé sur la probabilité des exemples ordinaires, tandis que le C4.5 flou calcule le rapport de gain d'information en utilisant la probabilité des degrés d'appartenance des exemples.

Pour calculer le rapport de gain d'information flou, on sait que, dans chaque ensemble de données, un attribut peut prendre plusieurs valeurs. Et avec la logique floue, ces valeurs sont exprimées en termes linguistiques (ensemble flou). Chaque ensemble flou est décrit par un FA. Soit X est l'ensemble des instances, $A^{(k)}$ avec $k = \{1, 2, \dots, n\}$ est l'ensemble d'attributs qui a des valeurs floues décrites par l'ensemble flou $A_i^{(k)}$ avec $i = \{1, 2, \dots, m\}$, $M_{A_i^{(k)}}$ est le FA de l'ensemble flou $A_i^{(k)}$, et les exemples d'apprentissage doivent être classés en classes floues décrites par des ensembles flous Y_j avec $j = \{1, 2, \dots, c\}$. Soit M_{Y_j} dénote le FA de l'ensemble flou Y_j . Le degré de classe (DC) du i ème ensemble flou du k ème attribut $A_i^{(k)}$ par rapport à la j ème classe floue Y_j est défini par l'équation (5.2).

$$DC_{A_i^{(k)}}(Y_j) = \frac{\sum_{x \in X_j} M_{A_i^{(k)}}(x^{(k)})}{\sum_{x \in X} M_{A_i^{(k)}}(x^{(k)})} \quad (5.2)$$

Où : X_j sont tous les membres de l'ensemble des instances d'apprentissage qui possèdent le k ème attribut dans le sens où ils tombent dans le support de l'ensemble flou $A_i^{(k)}$, et ils appartiennent également à la classe Y_j , X sont tous les membres de l'ensemble

des instances d'apprentissage qui possèdent le k ième attribut dans le sens où ils tombent dans le support de l'ensemble flou $A_i^{(k)}$. $x^{(k)}$ est la valeur du k ième attribut de l'instance x , et $M_{A_i^{(k)}}(x^{(k)})$ est le degré d'appartenance de la valeur du k ième attribut de l'instance x représenté par l'ensemble flou $A_i^{(k)}$. Ainsi, l'entropie floue (EF) du i ème ensemble flou du k ième attribut $A_i^{(k)}$ est défini par l'équation (5.3).

$$EF_{A_i^{(k)}} = - \sum_{j=1}^c DC_{A_i^{(k)}}(Y_j) \log DC_{A_i^{(k)}}(Y_j) \quad (5.3)$$

Où $DC_{A_i^{(k)}}(Y_j)$ est le degré de classe (DC) du i ème ensemble flou du k ième attribut $A_i^{(k)}$ par rapport à la j ième classe floue Y_j qui est calculée en utilisant l'équation (5.2). En outre, l'entropie floue (EF) du k ième attribut $A^{(k)}$ est définie comme une somme pondérée du $EF_{A_i^{(k)}}$ décrits par l'équation (5.4).

$$EF_{A^{(k)}} = \sum_{i=1}^m \frac{\sum_{x \in X} M_{A_i^{(k)}}(x^{(k)})}{\sum_{l=1}^m \sum_{x \in X} M_{A_l^{(k)}}(x^{(k)})} \cdot EF_{A_i^{(k)}} \quad (5.4)$$

Où X représente tous les membres de l'ensemble des instances de formation qui possèdent le k ième attribut dans le sens où ils tombent dans le support de l'ensemble flou $A_i^{(k)}$, m est le nombre total de l'ensemble flou $A_i^{(k)}$, $M_{A_i^{(k)}}(x^{(k)})$ est le degré d'appartenance de la valeur du k ième attribut de l'instance x représenté par l'ensemble flou $A_i^{(k)}$, et $EF_{A_i^{(k)}}$ est l'entropie floue (EF) du i ème ensemble flou du k ième attribut $A_i^{(k)}$ calculé à l'aide de l'équation (5.3). D'autre part, le degré de classe (DC) des instances de formation par rapport à la j ième classe floue Y_j est défini par l'équation (5.5).

$$DC(Y_j) = \frac{\sum_{x \in X_j} M_{Y_j}(x)}{\sum_{x \in X} M_{Y_j}(x)} \quad (5.5)$$

Où X représente tous les membres de l'ensemble des instances de formation, X_j représente tous les membres de l'ensemble des instances de formation qui appartiennent à la classe Y_j et $M_{Y_j}(x)$ représente le degré d'appartenance à la classe j représenté par l'ensemble flou Y_j dans l'instance x . L'entropie floue (EF) des instances de formation est définie par l'équation (5.6).

$$EF = - \sum_{j=1}^c DC(Y_j) \log DC(Y_j) \quad (5.6)$$

Où $DC(Y_j)$ est le degré de classe (DC) des instances de formation par rapport à la j ème classe floue Y_j calculée à l'aide de l'équation (5.5). Par conséquent, le gain d'information floue (GIF) du k ième attribut par rapport à un ensemble d'instances de formation est finalement défini par l'équation (5.7).

$$GIF_{A^{(k)}} = EF - EF_{A^{(k)}} \quad (5.7)$$

Par conséquent, le gain d'information floue (GIF) est la différence entre l'entropie floue (EF) des instances de formation calculée à l'aide de l'équation (5.6) et l'entropie floue ($EF_{A^{(k)}}$) du k ème attribut calculé à l'aide de l'équation (5.4). L'information de split $IS_{A^{(k)}}$ du k ème attribut est défini par l'équation (5.8).

$$IS_{A^{(k)}} = \sum_{i=1}^m - \left(\frac{\sum_{x \in X} M_{A_i^{(k)}}(x^{(k)})}{\sum_{l=1}^m \sum_{x \in X} M_{A_l^{(k)}}(x^{(k)})} \right) \log \left(\frac{\sum_{x \in X} M_{A_i^{(k)}}(x^{(k)})}{\sum_{l=1}^m \sum_{x \in X} M_{A_l^{(k)}}(x^{(k)})} \right) \quad (5.8)$$

Où X est tous les membres de l'ensemble des instances de formation qui possèdent le k ème attribut dans le sens où ils tombent dans le support de l'ensemble flou $A_i^{(k)}$, m est le nombre total de l'ensemble flou $A_i^{(k)}$, et $M_{A_i^{(k)}}(x^{(k)})$ est le degré d'appartenance de la valeur du k ème attribut de l'instance x représenté par l'ensemble flou $A_i^{(k)}$. Par conséquent, le rapport de gain d'information floue (RGIF) du k ème attribut est défini par l'équation (5.9).

$$RGIF_{A^{(k)}} = \frac{GIF_{A^{(k)}}}{IS_{A^{(k)}}} \quad (5.9)$$

Où $GIF_{A^{(k)}}$ est le gain d'information floue (GIF) du k ème attribut par rapport à un ensemble d'instances de formation qui est calculé en utilisant l'équation (5.7), et $IS_{A^{(k)}}$ est l'information split calculée en utilisant l'équation (5.8).

La construction d'un arbre de décision est un processus répétitif, c'est-à-dire elle se base sur l'algorithme classique qui dépensait beaucoup de ressources seulement pour une petite quantité de données, sans parler d'une énorme quantité de données. Pour remédier à ce problème, nous avons utilisé la méthode de programmation parallèle. L'arbre de décision flou C4.5 est également généré par le processus itératif. Dans le cas de grandes quantités de données, il est difficile d'atteindre l'objectif de la classification en utilisant le C4.5 flou avec un seul nœud. En particulier, le calcul du rapport de gain d'information floue dans le processus de construction de l'arbre de décision flou est le processus le plus long et il utilise beaucoup de ressources. Dans notre travail, pour traiter ce problème, nous appliquons le modèle de programmation MapReduce, qui parallélise les tâches de classification entre cinq machines ; un nœud maître et quatre nœuds esclaves. L'algorithme 4 illustre les étapes de notre algorithme d'arbre de décision parallèle flou C4.5.

5.3.2.3 Règles floues

Nous créons la base de règles en transformant d'abord l'ensemble de données d'apprentissage en données floues en utilisant la méthode de fuzzification (FA triangulaire). Ensuite, nous appliquons l'algorithme parallèle flou C4.5 à l'ensemble de données floues pour produire un arbre de décision flou. Enfin, nous extrayons la base de règles à partir de

Algorithm 4 Notre Algorithme MapReduce Flou C4.5

Input : X est un ensemble de données de formation floues, et A est un ensemble d'attributs

Output : ArbreFlou : Arbre de décision flou.

GénérationArbre(S,A) : Créer un nouvel arbre ArbreFlou avec un seul nœud racine

Définir la tâche effectuée par Hadoop

Définir SelectTaskMapper comme la classe Mapper

Définir selectTaskReducer comme la classe Reducer

Ajuster la taille du bloc de HDFS jusqu'à ce que l'ensemble de données X peut être divisé en S sous-ensembles $X_j = \{j = 1, 2, \dots, S\}$

Dans le jème SelectTaskMapper

Input : $X_j = \{x_1, x_2, \dots, x_r\}$ avec x_r est la r-ème instance avec l'attribut n $A_k = \{1, 2, \dots, n\}$

Output : (clé, valeur) = $(A_k, Ratio_k(A_k, X_j))$

for $k \leftarrow 1$ **to** n **Attribut do**

Calculer $DC_{A_i^{(k)}}(Y_j)$ en utilisant l'équation (5.2).

Calculer $EF_{A_i^{(k)}}$ en utilisant l'équation (5.3).

Calculer $EF_{A^{(k)}}$ en utilisant l'équation (5.4).

Calculer $DC(Y_j)$ en utilisant l'équation (5.5).

Calculer EF en utilisant l'équation (5.6).

Calculer $GIF_{A^{(k)}}$ en utilisant l'équation (5.7).

Calculer $IS_{A^{(k)}}$ en utilisant l'équation (5.8).

Calculer le rapport de gain d'information floue $GIF_{A^{(k)}}$ en utilisant l'équation (5.9).

$Ratio_k(A_k, X_j) = GIF_{A^{(k)}}$

SortieMapper : (clé, valeur) = $(A_k, Ratio_k(A_k, X_j))$

end for

Dans le jème SelectTaskReducer

Input : (clé, valeur) = $(A_k, list([Ratio_k(A_k, X_j)], (j = 1, \dots, S)))$

Output : (clé, valeur) = $(A_{k^*}, Ratio_{k^*}(X))$

for $k \leftarrow 1$ **to** n **Attribut do**

$Ratio_k(X) = \sum_{j=1}^m Ratio_k(A_k, X_j)$.

end for

Trouver le meilleur attribut de partitionnement A_{k^*} qui a le rapport de gain d'information floue maximum $A_{k^*} = argmax_A \{Ratio_k(X)\}_{k=1}^n$

SortieReducer : (clé, valeur) = $(A_{k^*}, Ratio_{k^*}(X))$

Attacher A_{k^*} à ArbreFlou;

for valeurs d'attribut $v \in A_{k^*}$ **do**

Générer une branche pour le nœud, de sorte que X_v représente un sous-ensemble des échantillons de X dont la valeur d'attribut A_{k^*} est v **if** $S_v = vide$ **then**

Marquer le nœud de branche comme un nœud de feuille, et sa classe est marquée comme la classe ayant le plus grand nombre d'échantillons dans X **return**

else

Récursion de **GénérationArbre**($X_v, A \setminus \{A_{k^*}\}$)

continue

end if

end for

return ArbreFlou

l'arbre de décision flou produit. La base de règles contient les règles floues qui doivent être utilisées pour prendre des décisions. Le processus de génération de ces règles est généralement basé sur certaines approches telles que les réseaux neuronaux, les arbres de décision (utilisés dans notre travail), les algorithmes génétiques ou d'autres méthodes empiriques. Cependant, dans certaines situations, la base de règles peut être produite en utilisant l'intuition et l'expérience personnelle. Les règles sont parmi les premières techniques utilisés pour représenter la connaissance. En fait, les règles sont encore largement utilisées parce qu'elles permettent d'exprimer clairement des directives et des stratégies, ainsi que de capturer les connaissances des experts humains. Les règles ont également l'avantage de leur format linguistique, qui est facilement compréhensible. Les règles floues sont un moyen facile de formuler des connaissances vagues. En général, les règles floues ont la forme suivante :

SI (antécédent) **ALORS** (conséquent)

Une règle est composée de deux parties principales : un bloc antécédent (entre SI et ALORS) et un bloc conséquent (suivant ALORS). Dans notre travail, les règles sont générées de chaque chemin à partir de la racine à un nœud feuille de l'arbre de décision produit. La figure 5.2 montre un exemple d'arbre de décision flou produit par l'application de l'arbre de décision C4.5 qui est parallèle et flou à un ensemble de données d'apprentissage flou caractérisé par deux classes (Y_1, Y_2) et six attributs. Et chaque attribut a 5 ensembles flous.

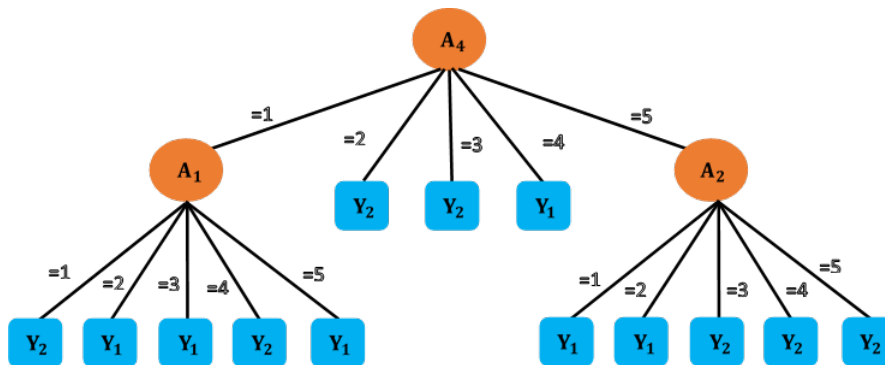


FIGURE 5.2 – Un exemple d'arbre de décision flou

A partir de l'arbre de décision flou illustré par la figure 5.2, nous pouvons déduire l'ensemble de règles floues. Le nombre de règles correspondra au nombre de chemins possibles à partir de la racine aux nœuds feuilles. D'après la figure 5.2, le nombre de chemins est treize donc le nombre de règles sera également treize comme décrit ci-dessous :

- Règle1** : **SI** A_4 est $v_{4,1}$ **ET** A_1 est $v_{1,1}$ **ALORS** C est Y_2
Règle2 : **SI** A_4 est $v_{4,1}$ **ET** A_1 est $v_{1,2}$ **ALORS** C est Y_1
Règle3 : **SI** A_4 est $v_{4,1}$ **ET** A_1 est $v_{1,3}$ **ALORS** C est Y_1
Règle4 : **SI** A_4 est $v_{4,1}$ **ET** A_1 est $v_{1,4}$ **ALORS** C est Y_2
Règle5 : **SI** A_4 est $v_{4,1}$ **ET** A_1 est $v_{1,5}$ **ALORS** C est Y_1
Règle6 : **SI** A_4 est $v_{4,2}$ **ALORS** C est Y_2

Règle7 : SI A_4 est $v_{4,3}$ ALORS C est Y_2

Règle8 : SI A_4 est $v_{4,4}$ ALORS C est Y_1

Règle9 : SI A_4 est $v_{4,4}$ ET A_2 est $v_{2,1}$ ALORS C est Y_1

Règle10 : SI A_4 est $v_{4,5}$ ET A_2 est $v_{2,2}$ ALORS C est Y_1

Règle11 : SI A_4 est $v_{4,5}$ ET A_2 est $v_{2,3}$ ALORS C est Y_2

Règle12 : SI A_4 est $v_{4,5}$ ET A_2 est $v_{2,4}$ ALORS C est Y_2

Règle13 : SI A_4 est $v_{4,5}$ ET A_2 est $v_{2,5}$ ALORS C est Y_2

5.3.2.4 Méthodes de raisonnement floues

Après l'étape d'extraction des règles floues, l'étape du test de notre modèle d'apprentissage sera ensuite entamée. C'est-à-dire que nous utilisons notre arbre de décision généré pour classer la nouvelle entrée, par l'application de deux mécanismes d'inférence. La méthode de raisonnement flou générale et la méthode de raisonnement flou classique, qui sont largement utilisées dans la littérature.

5.3.2.5 Méthode de raisonnement floue classique

De nombreux systèmes de classification floue utilisent la méthode de raisonnement flou classique (MRFC) [227], qui choisit la règle avec le degré de compatibilité le plus élevé pour classer la nouvelle instance donnée. Soit $e_p = \{a_{p1}, a_{p2}, \dots, a_{pm}\}$ une nouvelle instance à classer et $\{R_1, R_2, \dots, R_s\}$ un ensemble de s règles de classification floues. Soit $M_i(a_{pi}), i = 1, \dots, m$ est le degré d'appartenance de la valeur de l'attribut. Le MRFC applique les étapes suivantes pour classer une nouvelle instance :

- Calculer le degré de compatibilité entre l'exemple e_p et chaque règle R_k pour $k = 1, \dots, s$ et la t-norm \mathbf{t} (t-norm = maximum ou t-norm = minimum), est donné par :

$$\text{compat}(e_p, R_k) = \mathbf{t}[M_1(a_{p1}), M_2(a_{p2}), \dots, M_m(a_{pm})] \quad (5.10)$$

- Trouver la règle, R_{kmax} qui est la règle ayant le degré de compatibilité le plus élevé avec l'instance, c'est-à-dire

$$\text{compat}(e_p, R_{kmax}) = \max[\text{compat}(e_p, R_k)], k = 1, \dots, s \quad (5.11)$$

- Attribuer la classe c_j à l'instance e_p , où c_j la classe prédite par la règle R_{kmax} trouvée dans l'étape précédente.

Par exemple, nous devons classer $e_p = \{a, b, c, ?\}$ une nouvelle instance dont la classe est inconnue "?". Où a, b et c sont des ensembles flous. Soit $M(a) = 0.65$, $M(b) = 0.32$ et $M(c) = 0.82$ sont respectivement les degrés d'appartenance de a, b et c. Et nous avons deux règles telles que :

- R_1 : SI A est a_1 ET B est b_1 ET C est c_1 ALORS D est Y_1 . Avec $M(a_1) = 0.52$, $M(b_1) = 0.21$ et $M(c_1) = 0.92$
- R_2 : SI A est a_2 ET B est b_2 ET C est c_2 ALORS D est Y_2 . Avec $M(a_2) = 0.13$,

$$M(b_2) = 0.85 \text{ et } M(c_2) = 0.63.$$

étape 1 : Calculer le degré de correspondance entre l'instance d'entrée (a,b,c) et chaque terme de la règle $(a_1, a_2, b_1, b_2, c_1, c_2)$, puis nous utiliserons ces degrés calculés pour calculer le degré de compatibilité pour chaque règle.

$$\left. \begin{array}{l} 1. d(a, a_1) = \min(M(a), M(a_1)) = \min(0.65, 0.52) = 0.52 \\ 2. d(b, b_1) = \min(M(b), M(b_1)) = \min(0.32, 0.21) = 0.21 \\ 3. d(c, c_1) = \min(M(c), M(c_1)) = \min(0.82, 0.92) = 0.82 \\ 4. d(a, a_2) = \min(M(a), M(a_2)) = \min(0.65, 0.13) = 0.13 \\ 5. d(b, b_2) = \min(M(b), M(b_2)) = \min(0.32, 0.85) = 0.32 \\ 6. d(c, c_2) = \min(M(c), M(c_2)) = \min(0.82, 0.63) = 0.63 \end{array} \right\} = \min(0.52, 0.21, 0.82) = 0.21$$

$$\left. \begin{array}{l} 4. d(a, a_2) = \min(M(a), M(a_2)) = \min(0.65, 0.13) = 0.13 \\ 5. d(b, b_2) = \min(M(b), M(b_2)) = \min(0.32, 0.85) = 0.32 \\ 6. d(c, c_2) = \min(M(c), M(c_2)) = \min(0.82, 0.63) = 0.63 \end{array} \right\} = \min(0.13, 0.32, 0.63) = 0.13$$

Par conséquent, le degré de compatibilité entre l'exemple e_p et la règle R_1 est égal à $\text{compat}(e_p, R_1) = 0.21$, et $\text{compat}(e_p, R_2) = 0.13$ est le degré de compatibilité entre l'exemple e_p et la règle R_2 . Pour calculer le degré de compatibilité, nous avons utilisé $t\text{-norm} = \text{minimum}$ parce que nous avons le **ET** dans les règles, et dans le cas où nous avons le **OU** dans les règles, nous devons utiliser $t\text{-norm} = \text{maximum}$.

étape 2 : Trouver la règle R_{max} comme étant la règle ayant le degré de compatibilité le plus élevé avec l'instance e_p , c'est-à-dire :

$$\text{compat}(e_p, R_{kmax}) = \max[\text{compat}(e_p, R_1), \text{compat}(e_p, R_2)] = \max(0.21, 0.13) = 0.21.$$

Par conséquent, la règle ayant le degré de compatibilité le plus élevé est la règle R_1 : **SI** A est a_1 **ET** B est b_1 **ET** C est c_1 **ALORS** D est Y_1 .

étape 3 : Affecter la classe Y_1 à l'instance $e_p = \{a, b, c, Y_1\}$, où Y_1 est la classe prédite par la règle R_1 : **SI** A est a_1 **ET** B est b_1 **ET** C est c_1 **ALORS** D est Y_1 trouvée comme R_{max} dans l'étape précédente.

La figure 5.3 illustre graphiquement le MRFC. Le degré de compatibilité de la nouvelle instance d'entrée est calculé par rapport à toutes les règles floues s , et parce que la classe c_j de la règle R_{kmax} a le degré de compatibilité le plus élevé, elle est attribuée à l'exemple d'entrée.

5.3.2.6 Méthode générale de raisonnement flou

La méthode générale de raisonnement flou (MGRF) [227] suit les étapes indiquées ci-dessous pour classer un exemple donné e_p :

- Calculer le degré de compatibilité entre l'exemple e_p et chaque règle R_k pour $k = 1, \dots, s$ et une t-norme \mathbf{t} , donnée par :

$$\text{compat}(e_p, R_k) = \mathbf{t}[M_1(a_{p1}), M_2(a_{p2}), \dots, M_m(a_{pm})] \quad (5.12)$$

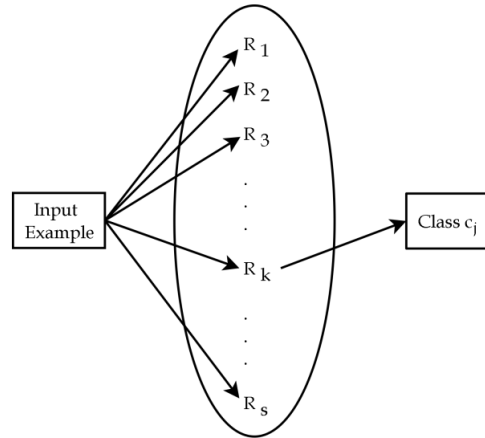


FIGURE 5.3 – Méthode classique de raisonnement flou.

- Pour chaque classe, calculer la valeur de classification $class_c$. $class_c$ est défini comme l'agrégation du degré de compatibilité, calculé à l'étape précédente, de toutes les règles avec la classe c_j et représente le degré de compatibilité de l'instance avec toutes les règles dont la classe prédite est c_j , donné par :
- $$class_{c_j} = \mathbf{f}\{compat(e_p, R_k) | c_j \text{ est la classe de } R_k\}$$
- Où \mathbf{f} est un opérateur d'agrégation.

Par exemple, nous devons classer $e_p = \{a, b, c, ?\}$ une nouvelle instance dont la classe est inconnue "?". Où a , b et c sont des ensembles flous. Soit $M(a) = 0,65$, $M(b) = 0,32$ et $M(c) = 0,82$ sont les degrés d'appartenance de a , b et c respectivement. Et nous avons quatre règles telles que :

- R_1 : **SI** A est a_1 **ET** B est b_1 **ET** C est c_1 **ALORS** D est Y_1 . Avec $M(a_1) = 0.52$, $M(b_1) = 0.21$ et $M(c_1) = 0.92$
- R_2 : **SI** A est a_2 **ET** B est b_2 **ET** C est c_2 **ALORS** D est Y_2 . Avec $M(a_2) = 0.13$, $M(b_2) = 0.85$ et $M(c_2) = 0.63$.
- R_3 : **SI** A est a_3 **ET** B est b_3 **ET** C est c_3 **ALORS** D est Y_1 . Avec $M(a_3) = 0.19$, $M(b_3) = 0.97$ et $M(c_3) = 0.38$
- R_4 : **SI** A est a_4 **ET** B est b_4 **ET** C est c_4 **ALORS** D est Y_2 . Avec $M(a_4) = 0.75$, $M(b_4) = 0.53$ et $M(c_4) = 0.20$.

étape 1 : Calculer le degré de correspondance entre l'instance d'entrée (a, b, c) et chaque terme de la règle ($a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4, c_1, c_2, c_3, c_4$), et ensuite nous utiliserons ces degrés calculés pour calculer le degré de compatibilité de chaque règle.

$$\begin{array}{l}
1. d(a,a_1) = \min(M(a),M(a_1)) = \min(0.65,0.52) = 0.52 \\
2. d(b,b_1) = \min(M(b),M(b_1)) = \min(0.32,0.21) = 0.21 \\
3. d(c,c_1) = \min(M(c),M(c_1)) = \min(0.82,0.92) = 0.82 \\
4. d(a,a_2) = \min(M(a),M(a_2)) = \min(0.65,0.13) = 0.13 \\
5. d(b,b_2) = \min(M(b),M(b_2)) = \min(0.32,0.85) = 0.32 \\
6. d(c,c_2) = \min(M(c),M(c_2)) = \min(0.82,0.63) = 0.63 \\
7. d(a,a_3) = \min(M(a),M(a_3)) = \min(0.65,0.19) = 0.19 \\
8. d(b,b_3) = \min(M(b),M(b_3)) = \min(0.32,0.97) = 0.32 \\
9. d(c,c_3) = \min(M(c),M(c_3)) = \min(0.82,0.38) = 0.38 \\
10. d(a,a_4) = \min(M(a),M(a_4)) = \min(0.65,0.75) = 0.65 \\
11. d(b,b_4) = \min(M(b),M(b_4)) = \min(0.32,0.53) = 0.32 \\
12. d(c,c_4) = \min(M(c),M(c_4)) = \min(0.82,0.20) = 0.20
\end{array}
\left. \begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \right\} = \min(0.52,0.21,0.82) = 0.21 \\
\left. \begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array} \right\} = \min(0.13,0.32,0.63) = 0.13 \\
\left. \begin{array}{l} \\ \\ \\ \\ \end{array} \right\} = \min(0.19,0.32,0.38) = 0.19 \\
\left. \begin{array}{l} \\ \\ \end{array} \right\} = \min(0.65,0.32,0.20) = 0.20$$

Par conséquent, le degré de compatibilité entre l'exemple e_p et chaque règle R_1, R_2, R_3 et R_4 , est égal à : $\text{compat}(e_p, R_1) = 0.21, \text{compat}(e_p, R_2) = 0.13, \text{compat}(e_p, R_3) = 0.19$ et $\text{compat}(e_p, R_4) = 0.20$ respectivement.

étape 2 : Pour chaque classe, calculer la valeur de classification $class_c$. Dans notre exemple, nous avons deux classes Y_1 et Y_2 .

$$class_{Y_1} = f\{\text{compat}(e_p, R_k) | Y_1 = \text{compat}(e_p, R_1) + \text{compat}(e_p, R_3) = 0.21 + 0.19 = 0.40$$

$$class_{Y_2} = f\{\text{compat}(e_p, R_k) | Y_2 = \text{compat}(e_p, R_2) + \text{compat}(e_p, R_4) = 0.13 + 0.20 = 0.33$$

étape 3 : Affecter la classe Y_1 à l'instance $e_p = \{a, b, c, Y_1\}$, où Y_1 est la classe ayant la somme la plus élevée ($class_{Y_1} = 0.40$) trouvée dans l'étape précédente.

La figure 5.4 décrit graphiquement le GFRM. Le degré de compatibilité de la nouvelle instance d'entrée est calculé par rapport à toutes les règles floues s , et parce que la classe c_j est la classe qui a obtenu le plus grand degré de classification parmi toutes les classes, elle a été attribuée à l'exemple d'entrée.

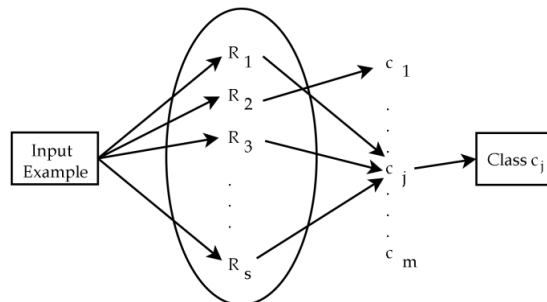


FIGURE 5.4 – Méthode générale de raisonnement flou.

5.4 Deuxième approche hybride

Dans cette section, nous présentons une approche hybride qui consiste à améliorer la performance de classification dans les méthodologies conventionnelles de classification au niveau de la phrase Anglaise basées sur l'arbre de décision C4.5 en appliquant le réseau neuronal convolutif comme extracteur de caractéristiques, en utilisant le système basé sur des règles floues pour traiter l'imprécision et l'incertitude qui existent dans le sentiment exprimé par l'homme au niveau de chaque tweet et en mettant en œuvre l'approche proposée de manière parallèle avec le cluster Hadoop.

Structurellement, le modèle hybride que nous avons proposé se compose de six étapes : dans la première étape, nous avons sélectionné deux grands jeux de données, à savoir Sentiment140 et COVID-19_Sentiments contenant les tweets collectés sur la plateforme Twitter. Dans la deuxième phase, nous avons appliqué les techniques de pré-traitement des données nécessaires pour supprimer toutes les données bruyantes et indésirables dans chaque tweet et préparer le tweet pour une analyse appropriée. Dans la troisième étape, nous avons implémenté la méthode de vectorisation FastText pour convertir chaque tweet en un vecteur numérique. Dans ce travail, nous avons utilisé FastText en raison du résultat expérimental de notre chapitre 4.5.2 dans la section dans la section 5.1, qui a prouvé que FastText a une meilleure performance que les autres méthodes de vectorisation dans le domaine de l'analyse des sentiments. Dans la quatrième étape, nous avons utilisé le réseau neuronal convolutif pour extraire les caractéristiques les plus pertinentes à l'aide de la couche de convolution et réduire la dimensionnalité des caractéristiques extraites à l'aide de la couche de mise en commun. Dans la cinquième phase, nous avons fuzzifié la sortie de la couche entièrement connectée en utilisant la fonction d'appartenance gaussienne, et nous avons choisi cette fonction d'appartenance parmi les fonctions d'appartenance triangulaire et trapézoïdale en raison des résultats expérimentaux présentés dans notre chapitre 6 dans la section 6.6.2, qui démontrent que la fonction d'appartenance gaussienne a obtenu de meilleures performances que les autres. Après avoir fuzzifié les données, nous avons appliqué C4.5 basé sur le gain d'information flou pour construire l'arbre de décision flou tel que décrit dans la première approche hybride 5.3.2.2. Enfin, nous avons utilisé l'approche générale de raisonnement flou pour classifier les nouveaux tweets en utilisant la base de règles floues extraite de l'arbre flou créé. Nous avons décidé d'appliquer la méthode de raisonnement flou général au lieu d'utiliser la méthode de raisonnement flou classique en raison des résultats expérimentaux obtenus par la mise en œuvre de la première approche hybride 5.3 qui confirment que la méthode de raisonnement flou général donne une meilleure performance que la méthode de raisonnement flou classique. Dans la sixième phase, nous avons mis en œuvre notre modèle hybride proposé sur un cluster Hadoop composé de cinq machines : un nœud de calcul maître et quatre nœuds de calcul esclaves.

5.4.1 Motivation

Nous proposons dans cette deuxième approche un nouveau modèle hybride pour effectuer l'analyse des sentiments des tweets collectés. Le modèle hybride proposé combine

les points forts des techniques de pré-traitement des données pour enlever les données indésirables et bruyantes, de la méthode FastText de plongement de mots pour représenter chaque tweet par un vecteur numérique, du réseau neuronal convolutif pour extraire et sélectionner automatiquement les caractéristiques pertinentes, de la fonction d'appartenance gaussienne pour fuzzifier les données afin de traiter des données incertaines et vagues, de l'algorithme d'arbre de décision C4.5 basé sur le gain d'information flou pour créer un arbre de décision flou et générer une base de règles floues, de la méthode générale de raisonnement flou pour classifier les nouveaux tweets et du cadre Hadoop pour mettre en œuvre notre approche de manière parallèle et ainsi améliorer les performances de notre approche proposée.

La première étape de notre démarche est l'acquisition des données. À ce stade, nous avons téléchargé deux grands jeux de données de Twitter. Le premier jeu de données est nommé **Sentiment140**, qui est obtenu à partir du lien <https://www.kaggle.com/kazanov/sentiment140>. Le second jeu de données s'appelle **COVID-19_Sentiments**, et se trouve sur le lien <https://www.kaggle.com/abhaydhiman/covid19-sentiments>. Ces jeux de données contiennent une quantité critique de données indésirables et bruitées, donc pour améliorer la qualité des données dans les deux jeux de données, nous avons appliqué plusieurs techniques de pré-traitement des données comme deuxième étape de notre contribution qui est appelée purification des données. Dans la phase de purification des données, nous avons appliqué plusieurs techniques telles que :

- Elargir les abréviations.
- Remplacer l'argot par le mot correct à partir du dictionnaire.
- Corriger les fautes d'orthographe à l'aide du dictionnaire WordNet.
- Supprimer les noms d'utilisateur, les chiffres, les espaces blancs, les hashtags, les caractères spéciaux, les URLs et la ponctuation.
- Remplacer les mots allongés.
- Enlever les mots vides.
- Mise en minuscules.
- Stemming.
- Lemmatisation.
- Tokenisation.

Conséquemment, toutes les techniques de pré-traitement des données influencent positivement la classification des sentiments, comme présenté dans le chapitre 4 dans la section 4.5.1. Dans laquelle, nous avons fourni une étude analysée efficace pour évaluer l'impact des techniques de pré-traitement des données sur la classification des sentiments en termes de taux de classification. Les résultats obtenus à partir des expériences réalisées dans ce chapitre ont confirmé que l'application de chaque technique de pré-traitement des données a une amélioration significative sur la performance de l'analyse des sentiments. De plus, l'objectif de ce document de recherche [228] est de juger la performance des méthodes de pré-traitement des données sur les problèmes de classification. Les auteurs de cet article utilisent trois jeux de données différents pour évaluer chaque technique de pré-traitement. Les résultats expérimentaux prouvent que l'impact de chaque méthode de pré-traitement est le même dans tous les jeux de données, et que la mise en œuvre de chaque technique de pré-traitement améliore la performance de classification. Les auteurs de l'article [229] identifient l'influence des techniques de pré-traitement des données

sur le taux de classification de l'algorithme d'apprentissage automatique BN. L'influence des techniques de purification des données sur le taux de classification de l'algorithme BN est mesurée en comparant le résultat expérimental avec l'application de techniques de pré-traitement et sans application de techniques de pré-traitement. Les résultats empiriques démontrent que l'intégration de l'algorithme BN avec toutes les techniques de pré-traitement peut augmenter le taux de classification. Par conséquent, nous avons motivé par le résultat précis donné concernant l'application des techniques de pré-traitement des données, et nous avons mis en œuvre toutes les étapes de purification des données introduites ci-dessus dans cette contribution.

Après la phase de purification des données, l'étape suivante est la vectorisation des données qui vise à représenter chaque tweet par un vecteur numérique. Il existe plusieurs méthodes de vectorisation à savoir *TF-IDF*, *FastText*, *Word2vec*, *GloVe*, *Bag-Of-Words*, *N-gram*. Selon l'étude comparative réalisée dans le chapitre 4, les algorithmes *N-gram*, *Bag-Of-Words*, *TF-IDF*, *GloVe*, *Word2vec* et *FastText* ont atteint un taux de classification égale à 51.76%, 64.24%, 71.05%, 72.23%, 77.43%, 87.13%, respectivement. Ces résultats expérimentaux sont obtenus lorsque nous avons appliqué l'algorithme d'arbre de décision C4.5 sur le jeu de données **COVID-19_Sentiments**. Par conséquent, nous avons motivé par la bonne performance obtenue en utilisant la méthode de plongement de mots FastText. En conséquence, dans cette contribution, nous avons utilisé FastText comme méthode de vectorisation des données. L'étape suivante de notre contribution consiste à appliquer le réseau neuronal convolutif pour effectuer les opérations d'extraction et de sélection des caractéristiques. Le réseau neuronal convolutif utilisé dans cette contribution est composé de trois couches : La couche de convolution vise à extraire des caractéristiques pertinentes et convoluées en appliquant plusieurs filtres sur la matrice de plongement de mots obtenue dans la phase précédente. La couche de mise en commun applique l'opérations de mise en commun maximale (Max-Pooling) sur les caractéristiques convoluées produites dans la couche de convolution pour réduire la dimensionnalité des caractéristiques extraites. Enfin, la couche entièrement connectée convertit les sorties de la couche de mise en commun en sorties linéaires à l'aide de la fonction d'activation softmax.

Après la phase du réseau neuronal convolutif, l'étape suivante est la phase de fuzzification, qui vise à fuzzifier les sorties de la couche entièrement connectée en appliquant une fonction d'appartenance à chaque neurone entièrement connecté. En d'autres termes, le processus de fuzzification vise à convertir la valeur nette de chaque neurone en valeur floue en mesurant le degré d'appartenance à l'aide de l'une des fonctions d'appartenance les plus populaires, à savoir la fonction d'appartenance trapézoïdale, la fonction d'appartenance triangulaire et la fonction d'appartenance gaussienne. Conformément à l'expérience réalisée dans le chapitre 4 pour évaluer les performances des fonctions d'appartenance trapézoïdale, triangulaire et gaussienne, nous avons décidé d'utiliser la fonction d'appartenance gaussienne dans cette contribution car elle atteint un bon taux de classification égale à 94.87 % par rapport à la fonction d'appartenance trapézoïdale qui atteint un taux de classification égale à 91.21 % et à la fonction d'appartenance triangulaire qui donne un taux de classification égale à 90.14 %. La phase suivante consiste à appliquer C4.5 sur la base du ratio de gain d'information flou pour créer l'arbre de décision flou et utiliser cet arbre flou obtenu pour générer la base de règles floues. Le C4.5 flou calcule le ratio de gain d'information de chaque caractéristique en calculant la probabilité des degrés d'appartenance. Enfin, nous avons appliqué la méthode générale de raisonnement flou sur la base de

règles floues pour classifier les nouveaux tweets comme étant neutres, positifs ou négatifs. Nous avons choisi d'appliquer la méthode générale de raisonnement flou plutôt que la méthode classique de raisonnement flou en raison du résultat expérimental obtenu par la mise en œuvre de la première approche hybride 5.3 qui a prouvé que la méthode générale de raisonnement flou donne un meilleur taux de classification (86.56 %) que la méthode classique de raisonnement flou qui atteint un taux de classification égale à 63.96%.

Enfin, nous avons mis en œuvre notre modèle hybride proposé en utilisant le cadre Hadoop avec son modèle de programmation MapReduce qui vise à faciliter les tâches de calcul en les divisant en sous-tâches distribuées sur un ensemble de machines et son système de fichiers distribués Hadoop pour stocker le grand ensemble de données sur plusieurs nœuds de calcul de manière redondante afin d'éviter les pannes de nœuds et les problèmes de goulots d'étranglement. Notre cluster Hadoop est composé de cinq machines de calcul : un nœud de calcul maître qui est responsable de la surveillance du stockage des données dans le système de fichiers distribué Hadoop, du partitionnement des données d'entrée en blocs, de la gestion de la communication inter-machine requise, de la planification de l'exécution du programme sur un ensemble de machines esclaves et de l'exécution du calcul parallèle effectué sur les données stockées à l'aide de MapReduce. Le nœud de calcul maître possède trois nœuds, qui sont JobTracker, NameNode, Secondary NameNode. Quatre nœuds de calcul esclaves, chaque esclave est responsable du stockage d'un bloc de données et de l'exécution de calculs sur ce bloc stocké. Ainsi, ce nœud esclave exécute à la fois un service TaskTracker et un service DataNode pour communiquer avec le nœud de calcul maître du cluster.

En résumé, l'objectif de notre contribution est d'améliorer la performance de la classification des sentiments en intégrant les avantages des méthodes de pré-traitement des données pour améliorer la qualité des données en supprimant les données indésirables et incohérentes, le point fort de la méthode de vectorisation des données FastText pour transformer chaque tweet en un vecteur numérique, l'avantage de réseau neuronal convolutif pour extraire des caractéristiques pertinentes et complètes en appliquant l'opération de convolution et pour sélectionner les caractéristique les plus pertinentes en appliquant l'opération de mise en commun maximale pour diminuer la haute dimensionnalité des caractéristiques convolutives obtenues, la puissance de la fonction d'appartenance gaussienne dans la fuzzification des sorties de la couche entièrement connectée pour traiter les données incertaines et ambiguës, la performance du C4.5 basé sur le ration de gain d'information floue pour créer un arbre de décision flou et extraire la base de règles qui contient un ensemble de règles floues, la capacité de la méthode générale de raisonnement flou à classifier les nouveaux tweets en fonction des règles floues extraites à l'étape précédente, et le cadre Hadoop pour mettre en œuvre notre approche hybride de manière parallèle. Comme l'illustre la figure 5.5, la structure globale de la deuxième approche hybride développée dans ce chapitre est composée de huit étapes : l'acquisition des données, la purification des données, la vectorisation des données, l'extraction et la sélection des caractéristiques à l'aide d'un réseau neuronal convolutif, la fuzzification des données, la construction d'un modèle d'apprentissage à l'aide d'un arbre de décision C4.5 flou, le test de ce modèle d'apprentissage créé à l'aide d'une méthode générale de raisonnement flou, et enfin, la parallélisation de notre approche hybride à l'aide du cadre Hadoop.

dans cette contribution, les phases d'acquisition et de purification des données sont les

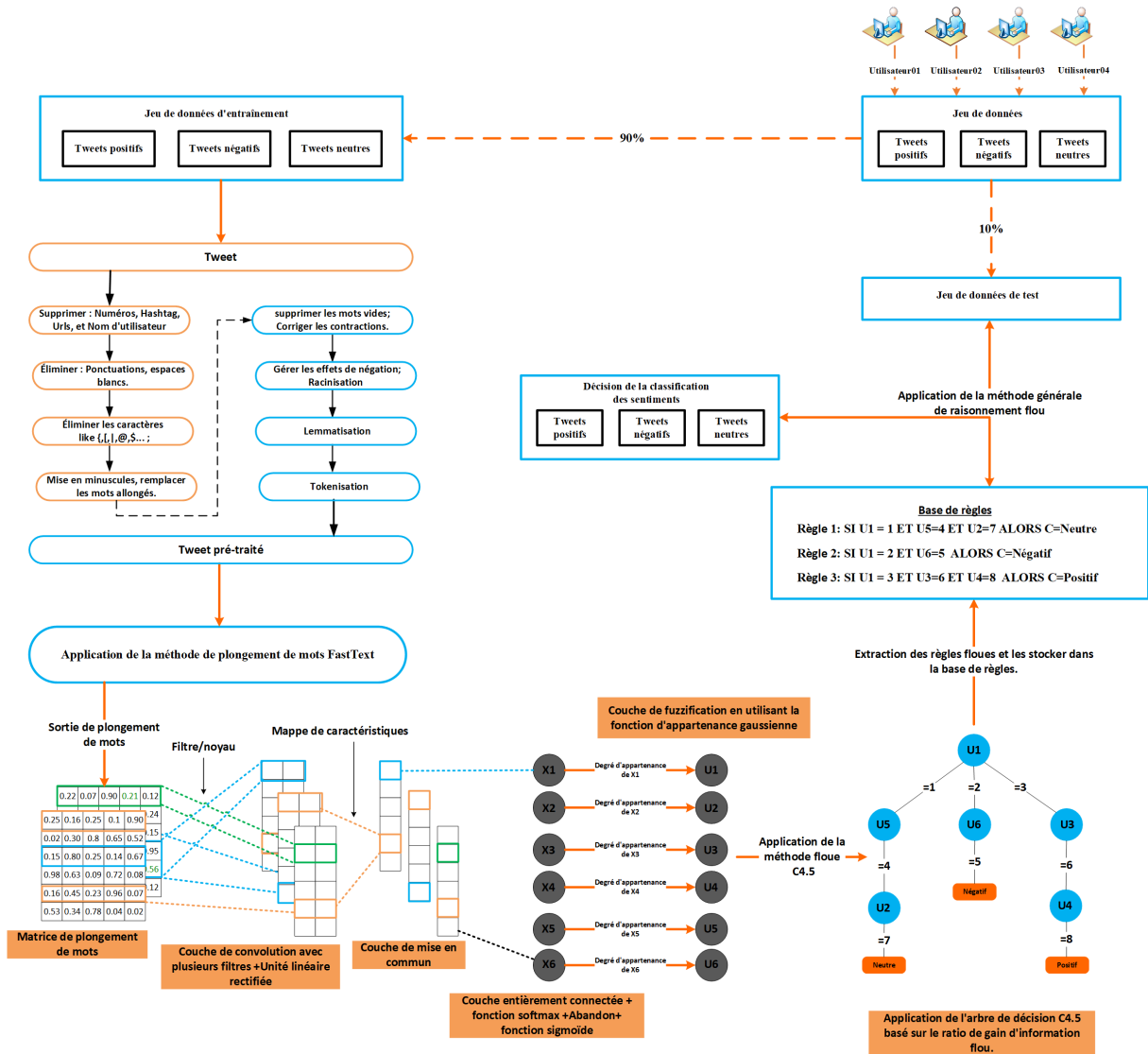


FIGURE 5.5 – Structure générale de la deuxième approche hybride.

mêmes que dans la contribution du chapitre précédent 4. Après la phase de pré-traitement des données, l'étape suivante est la vectorisation des données, qui convertit le tweet d'entrée en un vecteur numérique. Dans cette contribution, la méthode d'incorporation de mots FastText est mise en œuvre car elle a obtenu de bonne performance de classification par rapport à d'autres approches comme décrit dans le chapitre 4. Dans cette contribution, les sorties de la phase de pré-traitement des données seront l'entrée de la phase de vectorisation des données.

5.4.2 Vectorisation des données

L'objectif de cette démarche est de reconnaître les polarités de sentiment exprimées dans les tweets collectés à partir de la plateforme Twitter. Par conséquent, les tweets pré-traités doivent être exprimés en langage machine pour une analyse, un traitement et une classification ultérieurs. Dans cette phase, nous avons effectué un apprentissage non su-

pervisé en utilisant la méthode de vectorisation des données FastText qui convertit chaque mot avec $n\text{-gramme}=2$ en un vecteur de basse dimension. FastText intègre des notions les plus populaires proposées ces dernières années par les communautés de l'apprentissage automatique et du traitement du langage naturel. Celles-ci impliquent de vectoriser chaque phrase en utilisant un sac de caractères $n\text{-grammes}$ ou en employant un sac de mots et de les traiter en utilisant le modèle de Continuous Bag-Of-Words ou l'approche Skip-Gram pour obtenir son vecteur de basse dimension comme illustré dans la figure 5.6. FastText applique également une fonction softmax hiérarchique qui profite de la distribution déséquilibrée des classes pour accélérer le calcul. Ces diverses théories sont utilisées pour deux tâches distinctes : la classification efficace des données et la vectorisation des mots.

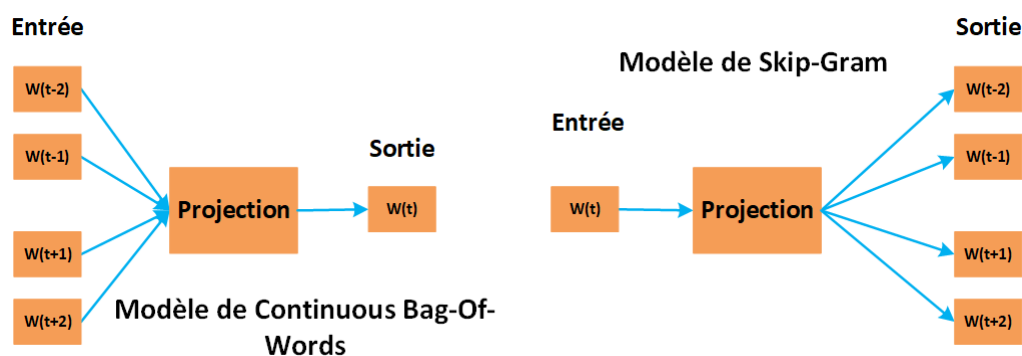


FIGURE 5.6 – Modèle de Continuous Bag-Of-Words et l'approche Skip-Gram.

Basé sur l'étude comparative réalisée dans [230] pour comparer le modèle Continuous Bag-Of-Words et l'approche Skip-Gram, les résultats expérimentaux de cette étude ont prouvé que l'approche Skip-Gram est plus performant que le modèle Continuous Bag-Of-Words pour représenter les mots mais qu'il nécessite un temps d'apprentissage important. Mais, nous avons utilisé le modèle de Skip-Gram dans notre travail parce que nous avons résolu son inconvénient concernant le temps en utilisant Hadoop.

Par exemple, la première étape pour vectoriser le mot "Cheery" consiste à appliquer l'approche $n\text{-gramme}$ sur le mot "Cheery" où $n\text{-gramme}=2$. Donc, le résultat de la première étape sera (C, Ce, ee, er, ry, y) où les parenthèses désignent le début et la fin du mot vectorisé. La deuxième étape consiste à appliquer le modèle de Skip-Gram sur le sac de $n\text{-grammes}$ de caractères obtenu pour calculer le vecteur de plongement de mots.

Comme nous l'avons dit précédemment, il existe plusieurs méthodes de vectorisation de mots telles que $N\text{-gram}$, $Bag\text{-Of}\text{-Words}$, $TF\text{-IDF}$, $GloVe$, $Word2vec$ et $FastText$ qui ont atteint respectivement un taux de classification égale à 51.76 %, 64.24 %, 71.05 %, 72.23 %, 77.43 %, 87.13 %, selon l'étude comparative réalisée dans le chapitre 4. Par conséquent, nous pouvons expliquer l'excellente performance obtenue par FastText par rapport aux autres méthodes de plongement de mots par la capacité de FastText à capturer le sens des mots plus abrégés et à aider l'opération de vectorisation des mots à détecter les préfixes et les suffixes du sac de $n\text{-grammes}$ à vectoriser. En outre, la procédure FastText a une grande capacité à créer des vecteurs pour n'importe quels mots, même les mots non reconnus, les mots mal orthographiés, les mots hors-dictionnaire et la concaténation de mots, par rapport aux autres méthodes de vectorisation. Bien que Word2Vec et GloVe

aient également utilisé le modèle de Skip-Gram pour représenter les mots, la structure globale de ces méthodes ne parvient pas à produire le vecteur de plongement de basse dimension d'un quelconque mot. Contrairement au schéma FastText qui a la capacité de produire le vecteur de plongement de basse dimension de n'importe quel mot. Ce sont les avantages de cette méthode par rapport aux autres stratégies de plongement des mots.

Après avoir appliqué FastText comme phase de vectorisation des données de notre contribution, dans laquelle nous convertissons les tweets en une matrice de vecteurs de plongement de basse dimension. L'étape suivante est l'extraction et la sélection des caractéristiques à l'aide du réseau neuronal convolutif, comme décrit dans la sous-section suivante.

5.4.3 Extraction et sélection des caractéristiques

L'extraction et la sélection des caractéristiques les plus pertinentes sont considérées comme une phase essentielle dans différentes utilisations du TAL et de nombreux travaux de recherche ont été réalisés pour générer des caractéristiques robustes, complètes et appropriées. Après, nous avons étudié différents travaux de recherche [231–234] dans la littérature. Nous avons déduit que les travaux récents ont accordé beaucoup d'attention à la technique d'extraction automatique de caractéristiques adoptée par les modèles d'apprentissage en profondeur au lieu de l'extraction manuelle de caractéristiques adoptée par les algorithmes d'apprentissage automatique traditionnels. En d'autres termes, les réseaux neuronaux en profondeur ont la capacité d'extraire et de sélectionner les caractéristiques les plus pertinentes pendant le processus d'apprentissage sans nécessiter d'intervention humaine, contrairement aux algorithmes d'apprentissage automatique classiques qui ont besoin de caractéristiques prédéfinies avant de commencer le processus d'apprentissage. Il existe plusieurs modèles d'apprentissage en profondeur dans la littérature. Le réseau neuronal convolutif est reconnu comme l'un des modèles d'apprentissage en profondeur les plus populaires, qui a été développé pour donner une représentation appropriée de l'entrée. Selon son architecture globale, le réseau neuronal convolutif peut être considéré comme un bon choix pour extraire et sélectionner les caractéristiques les plus appropriées dans ce travail.

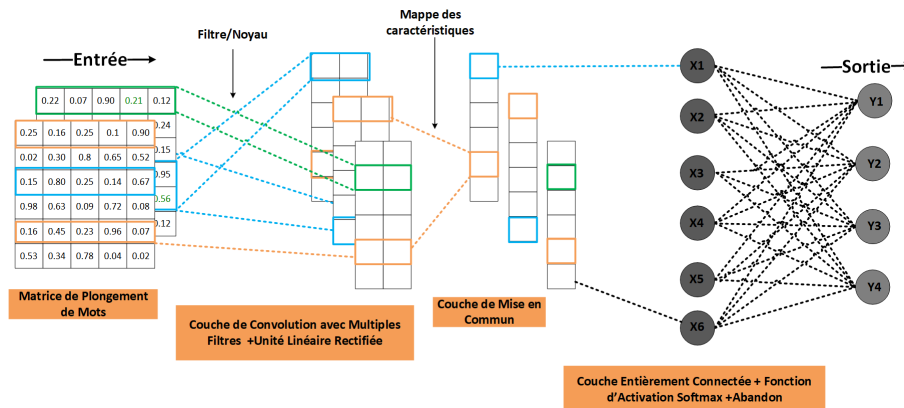


FIGURE 5.7 – Extraction et sélection de caractéristiques à l'aide d'un réseau neuronal convolutif.

Comme illustré dans la figure 5.7, l'architecture du simple réseau neuronal convolutif comprend trois couches essentielles. Ces couches sont les suivantes : La couche de convolution est utilisée pour extraire un ensemble de caractéristiques à partir de la matrice de plongement de mots obtenue dans la phase de vectorisation des données. La couche de mise en commun est utilisée pour sélectionner les caractéristiques les plus pertinentes et les plus appropriées parmi l'ensemble des caractéristiques extraites au niveau de la couche de convolution. Une couche entièrement connectée convertit les sorties de la couche de mise en commun en sorties linéaires en utilisant la fonction d'activation softmax.

Couche de Convolution : est la couche primaire de la structure globale du réseau neuronal convolutif. Cette première couche identifie et extrait les caractéristiques les plus pertinentes à partir de la matrice de plongement de mots E construite dans la phase de vectorisation des données. La couche de convolution consiste en plusieurs opérations de convolution. À chaque opération de convolution, un filtre glissant (S) est mis en œuvre sur chaque fenêtre de matrice de plongement de mots (EW) prélevée à partir du matrice E , et une mappe de caractéristiques est produite comme résultat. Par conséquent, des opérations multiples de convolution indiquant des filtres multiples avec une taille de fenêtre variable sont mises en œuvre sur la matrice E , et des mappes de caractéristiques multiples sont fournies comme sorties de ces opérations de convolution. Nous supposons que le $EW = [v_1; v_2; \dots; v_n]$ avec $v_i \in R^m$, une mappe de caractéristiques M_i est construite en appliquant S_i sur un fenêtre EW de taille $V_i : i+x-1$ en utilisant l'équation suivante :

$$M_i = \mathbf{ReLU}(S_i.V_{i:i+x-1} + a) \quad (5.13)$$

Où ReLU est une fonction d'activation non linéaire, $a \in R$ est le biais appliqué et x est la dimension du filtre employé S . Par conséquent, une mappe de caractéristiques $M_0 = [F_0, F_1, \dots, F_{i+x-1}]$ est construite par la mise en œuvre de (5.13) sur toutes les fenêtres sélectionnées EW à partir de la matrice E . Plusieurs filtres $S_{i:1 \rightarrow z}$ sont exercés pour créer un ensemble de mappes de caractéristiques $MF_{i:1 \rightarrow z}$. L'ensemble produit de mappes de caractéristiques $MF_{i:1 \rightarrow z}$ est activé, c'est-à-dire que l'ensemble linéaire de mappes de caractéristiques est converti en un ensemble non linéaire de mappes de caractéristiques par l'application de l'unité linéaire rectifiée $ReLU$ sur l'ensemble linéaire de mappes de caractéristiques. Par rapport aux fonctions d'activation Tanh et Sigmoid, les points forts de la fonction d'activation ReLU sont sa capacité à surmonter le problème de gradient en voie de disparition. Sa convergence est plus rapide en raison de son équation mathématique simpliste, et son temps d'exécution est court. Le $ReLU$ est calculé à l'aide de l'équation suivante (5.14) :

$$ReLU(y) = \max(0; y) \quad (5.14)$$

Après avoir appliqué toutes les opérations de la couche de convolution pour extraire les caractéristiques les plus appropriées à partir de la matrice de plongement de mots obtenue dans la phase de vectorisation des données, nous avons obtenu un ensemble de mappes de caractéristiques non linéaires. La phase suivante consiste à transmettre ces ensembles obtenus de mappes de caractéristiques non linéaires à la couche de mise en commun pour sélectionner les caractéristiques les plus pertinentes et réduire la dimensionnalité élevée

des caractéristiques.

Couche de Mise en Commun : Après avoir mis en œuvre la couche de convolution pour extraire les caractéristiques appropriées en appliquant plusieurs filtres glissants sur la matrice de plongement de mots produite dans la phase de vectorisation des données, l'étape suivante consiste à mettre en œuvre la couche de mise en commun pour sélectionner les caractéristiques les plus pertinentes en réduisant la dimension des mappes de caractéristiques extraites dans la phase de convolution. La couche de mise en commun applique une opération de mise en commun *maximale* ou de mise en commun *moyenne* sur l'ensemble des mappes de caractéristiques extraites pour sélectionner les caractéristiques pertinentes. La fonction de mise en commun moyenne calcule la moyenne de toutes les caractéristiques de la mappe convoluée et considère le résultat comme caractéristique poolée. La fonction de mise en commun maximale considère que la caractéristique poolée a la valeur maximale dans la mappe de caractéristiques convoluée et élimine le reste. Ces opérations de mise en commun moyenne et de mise en commun maximale sont illustrées dans la figure 5.8.

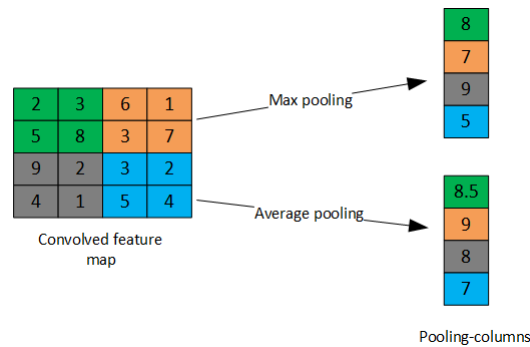


FIGURE 5.8 – Illustration des fonctions de mise en commun moyenne et de mise en commun maximale.

Dans cette proposition, nous avons employé la fonction de mise en commun *maximale*. L'opération de mise en commun *maximale* est mise en œuvre sur chaque mappe de caractéristiques M_i et sélectionne la caractéristique ayant la valeur maximale dans la mappe de caractéristiques comme caractéristique poolée $pf = \max[F_i]$. Cette opération produit un ensemble de caractéristiques poolées dont la taille est égale au nombre de mappes de caractéristiques N dans l'entrée de la couche de mise en commun. Par conséquent, l'opération de mise en commun *maximale* miniaturise la taille de chaque carte de caractéristiques, et sa sortie prend la forme d'une colonne. Le nombre total de colonnes en sortie sera égal au nombre total de mappes de caractéristiques en entrée. La miniaturisation obtenue en appliquant la fonction de mise en commun *maximale* dépend de la taille de la dimension du noyau de mise en commun *maximale*. En général, la couche de mise en commun convertit chaque mappe de caractéristiques convoluées en une seule colonne. Son objectif principal est de sélectionner et de préserver la caractéristique optimale importante en compressant les données et en diminuant la taille de la vectorisation. L'ensemble obtenu de caractéristiques optimales est ensuite transmis à la couche dense, également appelée couche entièrement connectée.

Couche Entièrement Connectée ou Couche Dense : est appliquée dans cette contribution pour convertir les sorties de la couche de mise en commun en sorties linéaires à

l'aide de la fonction d'activation softmax. Nous pouvons résumer sa fonctionnalité comme un processus linéaire dans lequel chaque entrée est liée à toutes les sorties par un poids différent. La couche dense transforme les mappes de caractéristiques de mise en commun en sortie linéaire en utilisant l'équation suivante (5.15) :

$$Lo = \text{soft}(Wc * Pf + A) \quad (5.15)$$

Où Lo est la valeur linéaire calculée, Wc est la matrice de poids appliquée, Pf est la mappe de caractéristiques poolée générée par la couche de mise en commun, A est le biais appliqué et soft est la fonction d'activation softmax, qui est définie comme dans l'équation (5.16) :

$$\text{soft}(y) = \frac{e_j^n}{\sum_{i=1}^I e_j^n} \quad (5.16)$$

Où n est la valeur du neurone d'entrée, e_j^n est la méthode exponentielle standard de la valeur du neurone d'entrée, et I est le nombre total de catégories dans le jeu de données utilisé.

Après l'application du réseau neuronal convolutif pour extraire et sélectionner les caractéristiques les plus appropriées dans cette contribution. La phase suivante est l'application de la fonction d'appartenance gaussienne pour flouter les caractéristiques obtenues dans la phase du réseau neuronal convolutif. Nous avons appliqué l'approche de fuzzification afin de fournir à notre approche hybride la capacité de traiter des données incertaines et vagues, puis d'améliorer le taux de classification de notre approche suggérée.

5.4.4 Phase de fuzzification des données

Après avoir extrait et sélectionné les caractéristiques les plus pertinentes et les plus cohérentes en appliquant le réseau neuronal convolutif. L'étape suivante est la phase de fuzzification des données qui vise à fuzzifier l'ensemble des caractéristiques obtenues dans la phase précédente. L'objectif principal de cette phase est de flouter les caractéristiques afin de leur appliquer l'arbre de décision C4.5 flou et de donner à notre modèle la capacité de traiter des données incertaines et vagues.

Dans cet approche, nous appliquons la fonction de fuzzification pour transformer les valeurs des neurones de la couche dense en un ensemble de valeurs floues en mesurant le degré d'appartenance de chaque valeur de neurone à l'aide de la fonction d'appartenance gaussienne expliquée dans le chapitre 3. Nous avons choisi la fonction d'appartenance gaussienne au lieu des fonctions d'appartenance triangulaire ou trapézoïdale en raison du résultat expérimental présenté dans le chapitre 4, qui a prouvé que la fonction d'appartenance gaussienne atteint un bon taux de classification égal à 94.87% par rapport à la fonction d'appartenance trapézoïdale qui atteint un taux de classification égal à 91.21% et à la fonction d'appartenance triangulaire qui donne un taux de classification égal à 90.14%. L'étape suivante est la mise en œuvre de l'algorithme de l'arbre de décision C4.5 flou, qui sont basés sur le ratio de gain d'information floue.

5.4.5 Arbre de décision C4.5 floue

Notre classificateur dans cette contribution combine le principe de l'arbre de décision C4.5 et la théorie des ensembles flous, comme décrit dans la première approche hybride proposée 5.3.2.2. Après avoir créé l'arbre de décision flou en appliquant le C4.5 basé sur le ratio de gain d'information flou, nous avons extrait toutes les règles floues possibles à partir de l'arbre de décision flou construit et nous les avons stockées dans la base de règles. L'étape suivante de cette deuxième approche proposée est l'application de la méthode générale de raisonnement flou sur la base de règles obtenue pour classifier les nouveaux tweets.

5.4.6 Méthode générale de raisonnement flou

Nous avons choisi dans cet approche d'appliquer l'approche générale de raisonnement flou pour classifier les nouveaux tweets d'entrée au lieu d'appliquer l'approche classique de raisonnement flou. Ce choix est fait sur le résultat expérimental introduit de la première approche hybride proposée 5.3 qui a prouvé que l'approche générale de raisonnement flou donne un meilleur taux de classification égal à 86.56% que la méthode classique de raisonnement flou qui atteint un taux de classification égal à 63.96%. Après cette étape, nous présentons la manière dont nous avons utilisé Hadoop.

5.4.7 Parallélisation de la deuxième approche sous Hadoop

Pour surmonter les difficultés cruciales auxquelles sont confrontées les données volumineuses, nous avons mis en œuvre le cadre Hadoop, comme le montre la figure 5.9 qui décrit la mise en œuvre de notre modèle hybride proposé sur un cluster Hadoop de cinq machines : un noeud maître et quatre noeuds esclaves. Notre jeu de données est

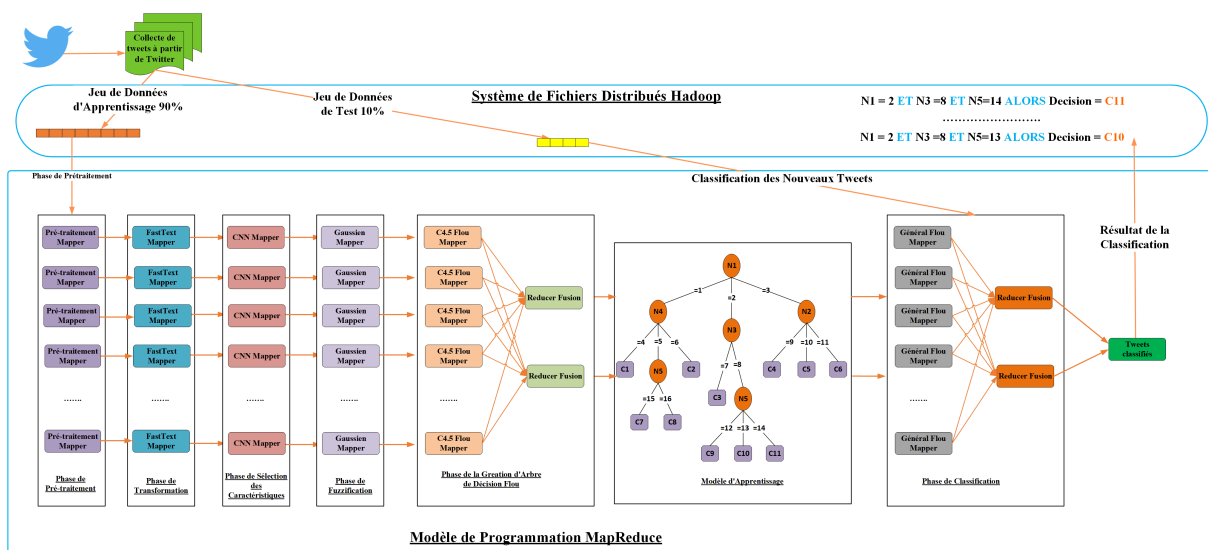


FIGURE 5.9 – Parallélisation de notre deuxième approche en utilisant Hadoop.

divisé en jeux de données d'apprentissage et de test enregistrés de manière distribuée sur cinq ordinateurs à l'aide du HDFS avec son NameNode et ses DataNodes. Le processus d'apprentissage des données est également réalisé de manière parallèle en utilisant MapReduce avec son JobTracker et ses Tasks-Trackers. Dans la première tâche MapReduce, l'ensemble de données d'apprentissage est divisé en blocs. Chaque bloc est transmis à chaque fonction Mapper qui lui applique les techniques de pré-traitement des données, le plongement de mots FastText pour convertir le bloc en un vecteur numérique, puis l'application d'un réseau neuronal convolutif pour extraire et sélectionner les caractéristiques les plus pertinentes, ensuite l'utilisation d'une fonction d'appartenance gaussienne pour flouter les caractéristiques sélectionnées, enfin l'application d'un C4.5 flou pour créer l'arbre de décision flou de ce bloc de données. Au niveau de chaque nœud Mapper, nous obtenons un petit arbre de décision flou. Nous avons utilisé deux nœuds reducers pour agréger les résultats obtenus des nœuds Mappers. Le résultat de sortie des deux nœuds Reducers est l'arbre de décision flou complet. Dans la deuxième tâche MapReduce, nous avons exécuté l'arbre de décision flou généré dans la première tâche MapReduce sur le jeu de données de test en utilisant cinq nœuds Mappers. Les résultats de la classification sont stockés dans le HDFS. L'algorithme 5 décrit notre modèle hybride parallèle qui nous proposons dans cette contribution pour classifier les nouveaux tweets.

5.5 Expériences de simulation et analyse

Dans notre approche, nous avons divisé l'ensemble de données en deux sous-ensembles (ensemble de données d'apprentissage et ensemble de données de test) en utilisant une stratégie de validation croisée décuplée et nous l'avons stocké dans le HDFS. Nous avons ensuite utilisé la méthode de fuzzification, en particulier la FA triangulaire, pour fuzzifier l'ensemble de données de formation. Après avoir appliqué notre algorithme proposé (arbre de décision C4.5 qui est parallèle et flou) aux données floues, nous avons obtenu un arbre de décision flou. De plus, nous avons utilisé l'arbre de décision flou généré pour extraire un ensemble de règles. Enfin, nous avons appliqué les deux méthodes de raisonnement flou sur l'ensemble de règles pour classer l'ensemble de données de test et stocker les données classifiées dans le HDFS.

5.5.1 Ensembles de données expérimentales

Pour évaluer la performance de notre approche (logique floue+ MapReduce+ C4.5) par rapport à d'autres méthodes comme ID3, C4.5, MapReduce+C4.5, logique floue+C4.5, Damanik et al. [235], Cherfi et al. [236], et Lee [237], nous avons considéré trois ensembles de données sélectionnés dans la base de données UCI (Machine Learning repository) [221] à savoir "PAMAP2 Physical Activity Monitoring", "Gas sensor array under dynamic gas mixtures" et "Record Linkage Comparison Patterns" avec un nombre d'instances allant de 3850505 à 5749132 comme décrit dans le tableau 5.1. Et pour évaluer son efficacité, nous avons choisi dix paramètres d'évaluation présentés dans le chapitre 3 à savoir Taux Positif Vrai (TPV), Taux Négatif Vrai (TNV) ou spécificité, Taux Positif Faux (TPF), Taux Négatif Faux (TNF), Taux d'Erreur (TR), Précision (PR), Taux de Classification

Algorithm 5 Algorithme de notre deuxième approche hybride et parallèle

Input : Jeu de données d'apprentissage de tweets E

Output : ArbreFlou

Définir le travail d'Hadoop

Définir `SelectTaskMapper` comme la classe de Mapper

Définir `selectTaskReducer` comme la classe de Reducer.

Ajustez la taille des blocs de HDFS jusqu'à ce que le jeu de données E puisse être divisé en S blocs $E_j = \{j = 1, 2, \dots, S\}$

for $j \leftarrow 1$ **to** S *bloc* **do**

Dans le j -ième SelectTaskMapper

 — Si (un mot dans le j -ième donnée $bloc_j$ pas en Anglais) Alors
 Traduire(mot en anglais)

 — $T = \text{pré-traitement-données}(bloc_j)$

 — $F = \text{FastText}(T)$

 — $C = \text{RéseauNeuronalConvolutif}(F)$

 — $D = \text{FonctionAppartenanceGaussienne}(C)$

 — $Arbre_j = \text{Algorithme1}(D)$

end for

for $j \leftarrow 1$ **to** n *Reducer* **do**

Dans le j -ième SelectTaskReducer

$ArbreFlou = \sum_{j=1}^S Arbre_j$.

end for

Input : Jeu de données de test des tweets E , et ArbreFlou

Output : Résultat de la classification (RC)

for $j \leftarrow 1$ **to** C *bloc* **do**

Dans le j -ième SelectTaskMapper

$RC_j = \text{RaisonnementGénéralFlou}(bloc_j, ArbreFlou)$

end for

for $j \leftarrow 1$ **to** n *Reducer* **do**

Dans le j -ième SelectTaskReducer

$RC = \sum_{j=1}^C RC_j$.

end for

return RC à stocker dans HDFS comme résultat de la classification.

ou exactitude (TC), Statistique Kappa (SK), Score F1 (SF) et Temps d'Exécution (TE).

TABLE 5.1 – Propriétés des ensembles de données.

N.	Nombre d'instances	Nombre d'attributs
1	3850505	52
2	4178504	19
3	5749132	12

5.5.2 Résultats et discussions

La figure 5.10 montre le résultat de la précision de la classification après l'application de logique floue+C4.5 en utilisant la méthode de raisonnement général (approche 1) et logique floue+C4.5 en utilisant la méthode de raisonnement classique (approche 2) sur trois ensembles de données sélectionnés numéro 1, 2 et 3.

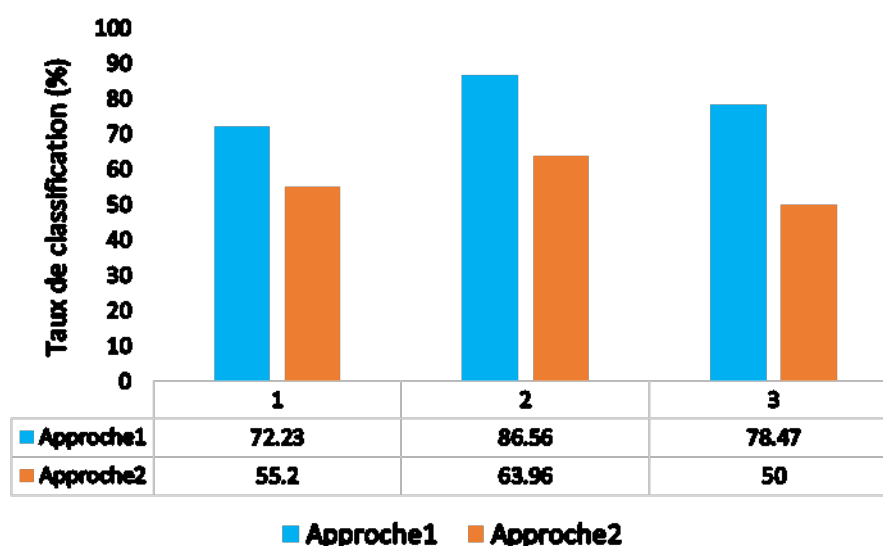


FIGURE 5.10 – TC en mettant en œuvre l'approche 1 et l'approche 2.

En analysant la figure 5.10, nous remarquons que l'approche 1 est plus performante que l'approche 2 dans tous les ensembles de données sélectionnés avec un taux de précision égal à 72.23%, 86.56% et 78.47% respectivement pour les ensembles de données numéros 1, 2 et 3. Autrement dit, la méthode de raisonnement général est plus efficace dans la classification des nouvelles instances que la méthode de raisonnement classique. Dans la suite de ce travail, nous utiliserons donc la méthode de raisonnement général.

Une autre expérience est faite pour comparer la méthode classique C4.5 et l'approche1, et pour démontrer si l'application de la logique floue influence le résultat de la classification. Comme la première expérience, nous avons appliqué les deux approches sur les trois ensembles de données choisis. La figure 5.11 montre le résultat du taux de classification en utilisant les deux algorithmes : flou et classique.

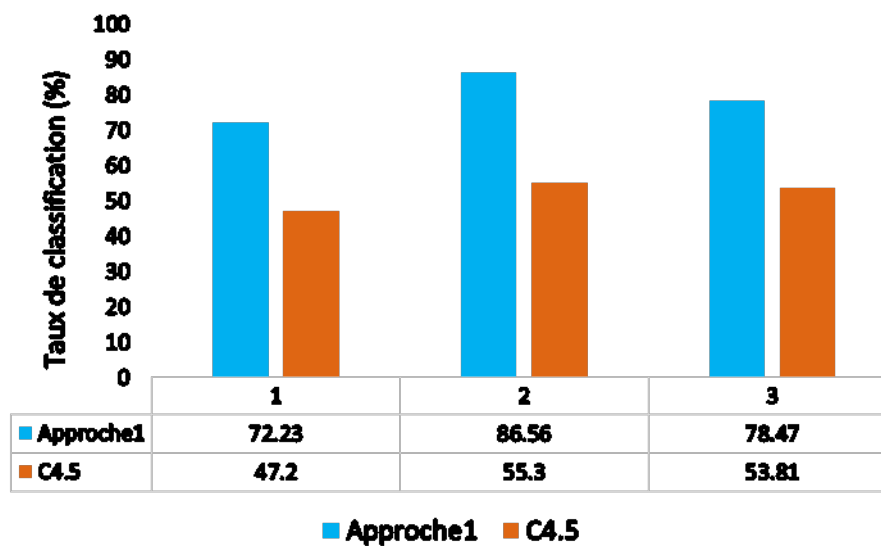


FIGURE 5.11 – TC en mettant en œuvre l’approche 1 et C4.5.

D’après la figure 5.11, nous déduisons que l’application de la logique floue avec l’algorithme C4.5 améliore les performances de la classification. C’est-à-dire qu’elle augmente le taux de précision de 25.03%, 31.26% et 24.66% respectivement pour les ensembles de données numéros 1, 2 et 3, par rapport à C4.5 classique.

Encore pour valider et évaluer notre travail, nous avons sélectionné trois ensembles de données contenant une énorme quantité de données, tels que l’ensemble de données n.1 a 3850505 instances, l’ensemble de données n.2 a 4178504 instances et l’ensemble de données n.3 a 5749132 instances. L’application du C4.5 classique prend beaucoup de temps, qui peut varier d’une heure à 2,5 heures selon la taille de l’ensemble de données utilisé. Pour remédier à ce problème, nous utilisons le modèle de programmation MapReduce, qui partage le travail sur cinq machines (quatre nœuds esclaves et un nœud maître). La figure 5.12 montre le temps d’exécution des algorithmes C4.5 et C4.5+MapReduce.

Nous remarquons, d’après la figure 4.12, que l’algorithme C4.5 + MapReduce a mis moins de temps d’exécution que l’algorithme C4.5, ce qui rend l’application de MapReduce sur C4.5 plus efficace. Par exemple, le temps de consommation en appliquant le C4.5 sur l’ensemble de données n.1 est de 3685 secondes, par contre, le temps utilisé en exécutant le C4.5 + MapReduce sur le même ensemble de données est de 500 secondes. Nous remarquons que l’algorithme C4.5+MapReduce réduit de temps d’exécution de 7.37 fois par rapport à C4.5. Cette réduction est due à l’utilisation de cinq machines (quatre nœuds esclaves et un nœud maître) dans l’exécution de l’algorithme C4.5 + MapReduce. De plus, l’algorithme C4.5+MapReduce réduit respectivement de temps d’exécution pour les ensembles de données n.2 et n.3 de 7.81 et 9.43 fois, par rapport à C4.5.

En résumé, la première expérience montre que la logique floue + C4.5 basée sur la méthode de raisonnement général surpasse la logique floue + C4.5 basée sur la méthode de raisonnement classique, donc dans la suite de notre travail, nous adoptons la méthode puissante pour classer les de nouvelles instances. Aussi, à partir de la deuxième expérience, telle que décrits dans la figure 5.11, nous déduisons que l’application de la logique floue sur

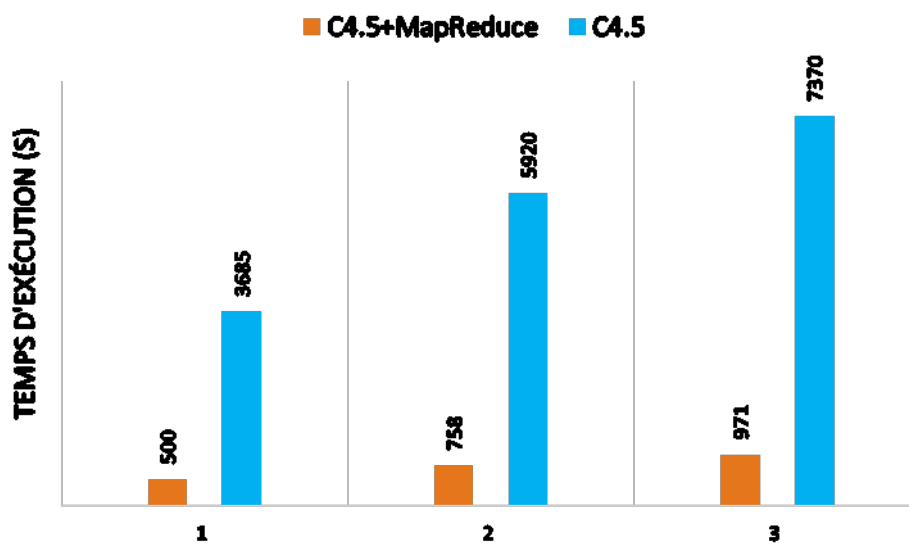


FIGURE 5.12 – TE en mettant en œuvre C4.5+MapReduce et C4.5

C4.5 nous permet d'améliorer la précision de classification de l'algorithme C4.5 classique. Par conséquent, dans notre travail, nous appliquerons la logique floue. Enfin, à partir de la troisième expérience, comme l'illustre la figure 5.12, nous remarquons que l'utilisation du modèle de programmation MapReduce diminue le temps d'exécution utilisé par C4.5 sur une énorme quantité de données. En conséquence, dans la suite de notre travail, nous évaluerons les performances de notre approche "C4.5+Logique floue+MapReduce".

La figure 5.13 illustre le résultat obtenu pour le taux de classification et le taux d'erreur en utilisant notre approche proposée (C4.5+LogiqueFloue+MapReduce), et nous comparons le résultat obtenu avec d'autres méthodes comme ID3, C4.5 et C4.5+LogiqueFloue. Les figures 5.13.a, 5.13.b, 5.13.c montrent les résultats obtenus par l'application de toutes les approches sur le jeu de données respectivement n.1, n.2 et n.3.

Selon la figure 5.13, la première remarque est que notre algorithme proposé (C4.5 + LogiqueFloue + MapReduce) surpasse les autres algorithmes en termes de taux de classification et de taux d'erreur. Et comme le montre la figure 5.13.a, si nous comparons notre approche avec C4.5, nous remarquons que notre approche augmente le taux de classification de 63.42% (C4.5) à 91.62% (notre méthode) et réduit le taux d'erreur de 36.58%(C4.5) à 8.38%(notre approche) pour le jeu de données numéro 1. La deuxième remarque est que l'intégration de MapReduce et de la logique floue avec C4.5 améliore les performances de la classification. La troisième remarque est que l'approche proposée est évolutive. En effet, comme nous l'avons dit précédemment, nous avons évalué notre travail en utilisant trois jeux de données, le jeu de données n.2 (4178504 instances) contient plus d'instances que le jeu de données n.1 (3850505 instances) et le jeu de données n.3 (5749132 instances) est également plus important que le jeu de données n.2. Le but principal de cette variation du nombre d'instances est de tester l'évolutivité de notre approche. Comme nous le voyons, dans la figure 5.13.a qui représente le jeu de données n.1, le taux de classification est de 91.62% pour notre méthode, 77.16% pour C4.5+LogiqueFloue, 63.42% pour C4.5 et 57.61% pour ID3. Comme le montre la figure 5.13.b qui illustre le résultat obtenu par l'application de toutes les approches sur le jeu de données n.2, le taux de classification est

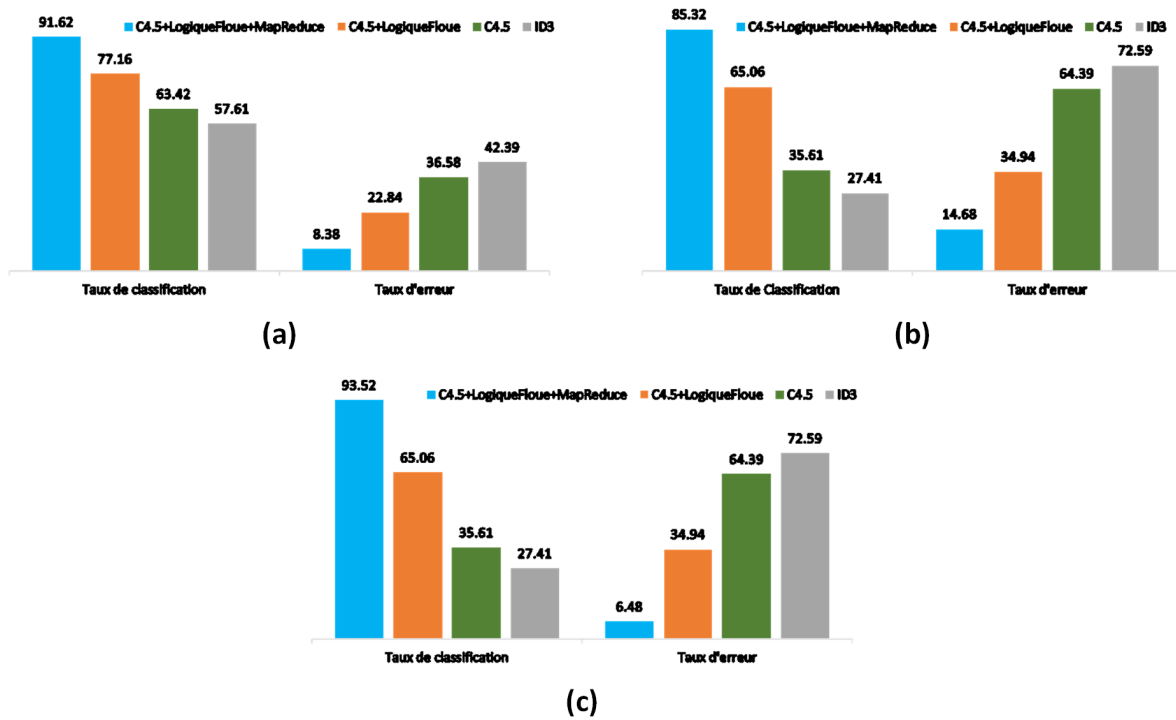


FIGURE 5.13 – TC et TE dans les cas (a) jeu de données n.1, (b) jeu de données n.2, (c) jeu de données n.3

de 89.32% pour notre procédure, 70.06% pour C4.5+LogiqueFloue, 55.61% pour C4.5 et 47.41% pour ID3. Aussi pour le jeu de données n.3 (figure 5.13.c), le taux de classification est de 93.52% pour notre approche, 65.06% pour C4.5+LogiqueFloue, 35.61% pour C4.5 et 27.61% pour ID3. Et comme le C4.5+LogiqueFloue n'est pas évolutif, nous pouvons en déduire que cette évolutivité de notre approches est due au modèle de programmation MapReduce.

Pour démontrer l'efficacité de notre approche, nous avons calculé d'autres paramètres d'évaluation tels que TPV, TNF, TNV, TPF, PR, SK, et SF, le tableau 5.2 montre le résultat obtenu.

Selon le tableau 5.2, notre approche (C4.5+logique floue+MapReduce) surpasse les autres algorithmes dans tous les ensembles de données (1,2,3) et au niveau du TPV (92.03%, 89.19%, 92.13%), du TNF (7.97%, 10.81%, 7.87%), TNV (89.71%, 87.61%, 89.56%), TPF (10.29%, 12.39%, 10.44%), PR (91.45%, 75.52%, 90.18%), SK (89.96%, 88.49%, 85.4%) et SF(88.24%, 80.74%, 79.05%).

Pour comparer le temps d'exécution entre notre approche et d'autres techniques, nous avons lancé une autre expérience. Cette expérience consiste à appliquer toutes les approches sur les trois ensembles de données sélectionnés. Sans oublier que notre approche est implémentée de manière parallèle sur cinq machines utilisant Hadoop.

La figure 5.14, montre que notre approche a un temps de mise en œuvre plus court dans les trois ensembles de données (a,b,c). Par rapport à ID3, notre approche diminue le temps d'exécution de 4007s à 556s pour l'ensemble de données n.1, de 7320s à 798s

TABLE 5.2 – Le résultat obtenu de TPV, TNF, TNV, TPF, PR, SK, et SF.

		TPV	TNF	TNV	TPF	PR	SK	SF
JdD n.1	Notre Approche	92.03	7.97	89.71	10.29	91.45	89.96	88.24
	C4.5+LogiqueFloue	79.19	20.81	78.06	21.94	72.85	68.67	76.49
	C4.5	64.53	35.47	70.11	29.89	63.25	59.46	67.20
	ID3	58.14	41.86	50.89	49.20	42.72	50.52	62.42
JdD n.2	Notre Approche	89.19	10.81	87.61	12.39	75.52	88.49	80.74
	C4.5+LogiqueFloue	69.23	30.77	60.57	39.42	70.45	58.18	67.92
	C4.5	56.31	43.69	49.82	50.18	59.11	45.68	57.38
	ID3	37.61	62.39	31.36	68.64	40.43	37.02	39.92
JdD n.3	Notre Approche	92.13	7.87	89.56	10.44	90.18	85.4	79.05
	C4.5+LogiqueFloue	64.15	35.85	60.33	39.67	59.45	62.26	61.05
	C4.5	36.81	63.19	35.09	64.91	30.29	33.67	35.70
	ID3	26.48	73.52	30.00	69.98	29.46	31.52	32.20

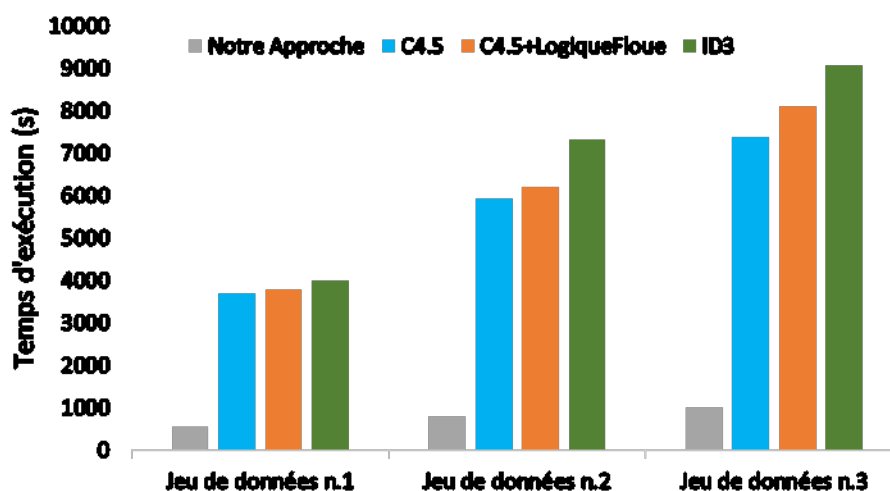


FIGURE 5.14 – TE en secondes de notre approche et d'autres techniques

pour l'ensemble de données n.2 et de 9080s à 1010s pour l'ensemble de données n.3, ce qui démontre que l'utilisation de la parallélisation est une bonne idée. Une autre remarque que nous avons déduite lorsque nous avons mis en œuvre notre travail sur le cluster Hadoop, le temps d'exécution diminue avec l'augmentation du nombre de nœuds dans le cluster Hadoop.

5.5.3 Étude comparative

Pour évaluer les résultats obtenus en appliquant notre méthode proposée, nous comparons notre approche avec d'autres techniques de la littérature. Tel que ; un « optimisation de l'arbre de décision génère par un algorithme C4.5 en utilisant un algorithme génétique » proposé par Damanik et al. [235], Cette approche intègre l'arbre de décision et

l'algorithme génétique pour améliorer les performances du C4.5 afin de générer des règles efficaces. un «algorithme d'arbre de décision C4.5 très rapide» proposé par Cherfi et al. [236], cette approche utilise la moyenne et la médiane arithmétiques pour améliorer une faiblesse rapportée de l'algorithme C4.5 lorsqu'il traite les attributs continus, et un "AUC4.5 : algorithme d'arbre de décision C4.5 basé sur l'AUC pour la classification des données déséquilibrées" proposé par Lee [237]. Cette approche présente une modification de l'algorithme C4.5, qui examine la différence de la SSC (surface sous la courbe ROC) pour choisir le meilleur attribut de partitionnement. La figure 5.15 illustre les résultats obtenus. D'après la figure 5.15, nous remarquons que notre approche basée sur l'arbre de

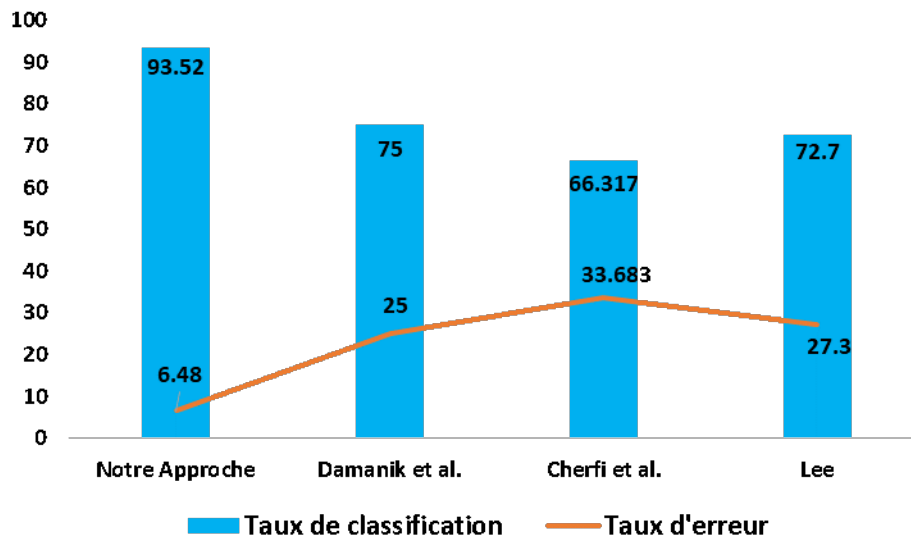


FIGURE 5.15 – Résultats obtenus par notre comparaison

décision, la logique floue et le cadre Hadoop surpasse les autres méthodes (Damanik et al. [235], Cherfi et al. [236], et Lee [237]) avec un taux de classification égal à 93.52% et un taux d'erreur égal à 6.48%. Cette efficacité et cet avantage de notre méthode proposée sont dus à l'utilisation de la théorie de la logique floue, de la méthode du raisonnement général et du cadre de Hadoop.

5.5.4 Première expérience

Dans la première expérience, nous avons mesuré les TC et TR de notre deuxième approche hybride et des quatre autres approches sélectionnées dans la littérature. La figure 5.16, illustre les résultats expérimentaux obtenus en termes de TC par la mise en œuvre de notre deuxième approche hybride, les classifieurs sélectionnés à savoir Xu *et al.* [238], Liao *et al.* [239], Maheswari *et al.* [240], et notre approche dans le chapitre 4 sur les jeux de données Sentiment140 et COVID-19_Sentiments afin de prouver l'efficacité de notre deuxième approche hybride proposé en comparant ses performances atteintes aux performances expérimentales obtenues avec les autres classifieurs sélectionnés.

Comme illustré dans la figure 5.16, nous constatons que l'approche [238] a atteint un TC plus bas 51.44 %, et 33.09 % dans le cas des jeux de données COVID-19_Senti-

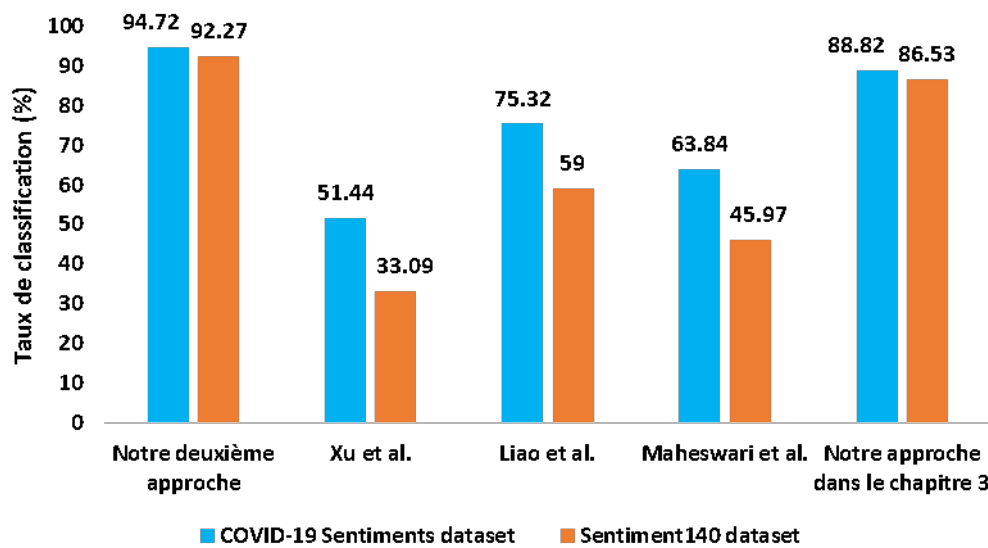


FIGURE 5.16 – TC en appliquant la deuxième approche hybride et les autres approches.

ments et Sentiment140, respectivement par rapport aux autres classifieurs parce que les auteurs de cette approche [238] n'accordent pas une grande importance aux tâches de pré-traitement des données. De plus, ils appliquent TF-IDF comme méthode de plongement de mots, et comme nous le prouvons dans l'étude comparative réalisée dans le chapitre 4, la méthode IF-IDF a un TC plus bas (71.05 %) comparé au TC atteint par GloVe (72.23 %), Word2Vec (77.43 %), et FastText (87.13 %). Par conséquent, les auteurs ne parviennent pas à choisir la méthode de plongement de mots la plus efficace. Ils ont ensuite appliqué l'algorithme continu de BN sur la matrice de plongement des mots obtenue sans appliquer au préalable un extracteur et un sélecteur de caractéristiques pour extraire les caractéristiques pertinentes à partir de la matrice de plongement des mots, ce qui affecte négativement les performances de leur classificateur. La méthode [239] a obtenu un TC égal à 63.84 %, et 45.97 % dans le cas des jeux de données COVID-19_Sentiments et Sentiment140, respectivement. Ses performances sont meilleures que celles de la méthode proposée dans [238], car elle a appliqué certaines techniques de pré-traitement, et utilisé le dictionnaire de mots AFINN pour extraire les mots d'opinion, enfin appliqué le SFM comme classificateur, qui est très efficace pour traiter les données inhérentes et ambiguës. Mais cette approche [239] ait des performances inférieures à celles des autres approches évaluées, elle n'applique pas toutes les techniques de pré-traitement nécessaires pour améliorer la qualité des données et le dictionnaire de mots AFINN a une capacité limitée pour détecter toutes les caractéristiques pertinentes. La méthode [240] a obtenu un TC égal à 75.32 %, et 59 % dans le cas des jeux de données COVID-19_Sentiments et Sentiment140, respectivement, ce qui représente une bonne performance par rapport aux classifieurs [239], et [238]. Parce que cette méthode a amélioré la qualité de ses données en appliquant plusieurs tâches de pré-traitement, puis a utilisé l'encodage one-hot pour représenter les données pré-traitées, et enfin a appliqué le réseau neuronal convolutif comme classificateur. En général, ses performances par rapport aux classifieurs [239], et [238] sont dues aux tâches de pré-traitement et au réseau neuronal convolutif. En outre, ce classificateur [240] a un TC plus bas comparé à notre deuxième approche hybride proposée et à notre approche dans le chapitre 4, parce qu'il a appliqué la méthode de plongement des

mots par l'encodage one-hot dans la phase de représentation, qui est moins précise que la méthode de plongement des mots FastText utilisée dans le travail 4 et qu'il n'applique pas la théorie de la logique floue pour traiter les données inhérentes et incertaines comme nous l'avons fait dans la deuxième approche hybride que nous proposons. Notre approche dans le chapitre 4 a atteint un TC égal à 88.82 %, 86.53 % dans le cas des jeux de données COVID-19_Sentiments et Sentiment140, respectivement. Cette approche a une meilleure performance comparée à toutes les approches précédemment étudiées parce qu'elle consiste en une phase de pré-traitement des données, une phase de représentation des données, une phase d'extraction des caractéristiques, une phase de sélection des caractéristiques et enfin une phase de classification. A chaque phase, elle applique une étude comparative entre plusieurs techniques pour choisir la technique la plus efficace pour effectuer la tâche. Par exemple, nous avons appliqué dans cette approche toutes les tâches de pré-traitement possibles. Nous avons également expérimenté en comparant les sélecteurs N-gramme, Sac de Mots, IF-IDF, GloVe, Word2Vec, et FastText pour sélectionner le plus efficace et l'utiliser dans la phase de représentation des données. Par exemple, nous avons appliqué dans cette approche toutes les tâches de pré-traitement possibles. Nous avons également expérimenté en comparant les sélecteurs N-gramme, Sac de Mots, IF-IDF, GloVe, Word2Vec, et FastText pour sélectionner le plus efficace et l'utiliser dans la phase de représentation des données. Les résultats expérimentaux montrent que les méthodes N-gramme, Sac de mots, IF-IDF, GloVe, Word2Vec et FastText atteignent un TC égal à 51.76 %, 64.24 %, 71.05 %, 72.23 %, 77.43 % et 87.13 %, respectivement. Nous avons donc choisi d'utiliser la méthode FastText comme méthode de vectorisation en raison de ses performances. Nous avons ensuite réalisé la deuxième expérience pour évaluer certains sélecteurs de caractéristiques, et le résultat expérimental montre que le Gain d'Information a atteint un TC de 86.53 %, le ratio de gain a obtenu un TC de 80.16 %, le khi-carré a atteint un TC de 73.35 %, et l'indice de Gini a obtenu un TC de 60.62 %. Selon le résultat expérimental de la deuxième expérience, nous avons décidé d'utiliser le Gain d'Information dans la phase de sélection des caractéristiques. En outre, nous avons appliqué notre ID3 amélioré basé sur le ratio de gain d'information pondéré comme classificateur. Enfin, nous avons mis en œuvre notre approche en utilisant le cadre Hadoop. Mais cette approche 4 a un TC plus bas comparé à l'approche hybride que nous proposons dans ce travail. Notre deuxième approche hybride proposée a obtenu un meilleur TC que tous les classificateurs évalués précédemment, qui est égal à 94.72 % et 92.27 % dans le cas des jeux de données COVID-19_Sentiments et Sentiment140, respectivement. Notre approche hybride se compose de cinq étapes : la phase de pré-traitement des données en appliquant toutes les tâches de pré-traitement nécessaires, la phase de représentation des données en utilisant la méthode de plongement de mots FastText basée sur l'étude comparative réalisée dans le chapitre 4 qui a prouvé que FastText a atteint un TC plus élevée égale à 87. 13% par rapport aux méthodes de plongement de mots les plus populaires, la phase d'extraction et de sélection des caractéristiques à l'aide d'un réseau neuronal convolutif basé sur le travail effectué dans notre article [241] qui a démontré que le réseau neuronal convolutif a la capacité d'extraire et de sélectionner efficacement les caractéristiques les plus pertinentes et les plus complètes à partir de la matrice de plongement des mots, la phase de classification des données en utilisant la fonction d'appartenance gaussienne pour la fuzzification des caractéristiques extraites basée sur l'étude réalisée dans notre article [241] qui a confirmé que la fonction d'appartenance gaussienne atteint un bon TC égal à 94.87% par rapport à la fonction d'appartenance trapézoïdale qui atteint un TC égal à 91.21% et la fonction

d'appartenance triangulaire qui donne un TC égal à 90.14%, en appliquant également le C4.5 basé sur le ratio de gain d'information flou pour créer l'arbre de décision flou en fonction du bon résultat expérimental obtenu dans la première approche hybride proposée 5.3, puis en appliquant la méthode de raisonnement flou général sur l'arbre de décision flou obtenue après l'application du C4.5 flou pour classifier des nouveaux tweets selon le résultat expérimental obtenu dans la première approche hybride proposée 5.3 qui a prouvé que l'approche de raisonnement flou général donne un meilleur TC égal à 86.56 % que la méthode de raisonnement flou classique qui atteint un TC égal à 63.96 %, et enfin la phase de parallélisation de notre approche à l'aide du cadre Hadoop. Par conséquent, les bonnes performances obtenues par notre deuxième approche hybride proposée par rapport à notre approche 4 sont dues à l'utilisation du réseau neuronal convolutif, qui a une plus grande capacité à extraire et à sélectionner avec précision les caractéristiques les plus pertinentes, et à la combinaison de la théorie de la logique floue et de l'arbre de décision pour traiter les données incertaines et ambiguës afin d'améliorer le processus de classification.

La figure 5.17, affiche les résultats expérimentaux atteints en termes de taux d'erreur après la mise en œuvre de notre deuxième approche hybride proposée, les classificateurs Xu *et al.* [238], Liao *et al.* [239], Maheswari *et al.* [240], et notre approche proposée dans le chapitre 6 sur les jeux de données Sentiment140 et COVID-19_Sentiments.

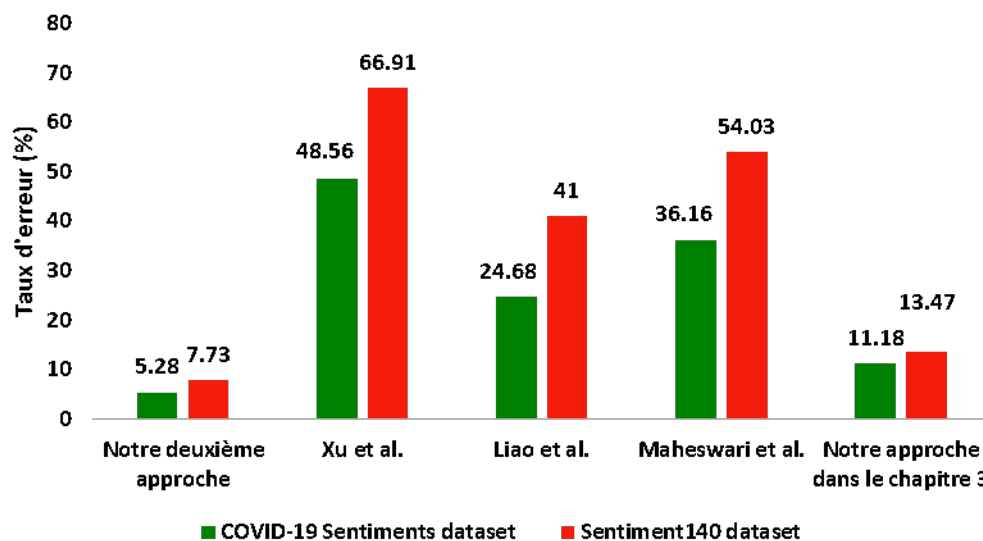


FIGURE 5.17 – TR en appliquant la deuxième approche et les autres approches.

Comme présenté dans la figure 5.17, nous remarquons que notre deuxième approche hybride proposée a atteint un TR égal à 5.28 % et 7.73 % dans le cas des jeux de données COVID-19_Sentiments et Sentiment140, respectivement. L'approche Xu *et al.* [?] a obtenu un TR égal à 48.56 %, et 66.91 % dans le cas des jeux de données COVID-19_Sentiments et Sentiment140, respectivement. La méthode Liao *et al.* [239] a atteint un TR égal à 24.68 %, et 41 % dans le cas des jeux de données COVID-19_Sentiments et Sentiment140, respectivement. Le modèle de Maheswari *et al.* [240] a un TR égal à 36.16 %, 54.03 % dans le cas des jeux de données COVID-19_Sentiments et Sentiment140, respectivement. Et le classificateur de Es-sabery *et al.* [4] a obtenu un TR égal à 11.18

%, et 13.47 % dans le cas des jeux de données COVID-19_Sentiments et Sentiment140, respectivement. D'après ces résultats expérimentaux, nous remarquons que la deuxième approche hybride que nous proposons a un TR plus faible que toutes les autres approches évaluées, et que l'approche de Xu *et al.* [238] a un TR plus élevé par rapport à toutes les autres méthodes évaluées. Par conséquent, les meilleures performances atteintes par notre deuxième approche hybride proposée sont dues aux étapes suivies dans le processus d'apprentissage de l'analyse des sentiments et à la technique sélectionnée pour être mise en œuvre à chaque étape. En général, notre deuxième approche hybride proposée a accordé une grande importance à chaque étape du processus d'analyse des sentiments en choisissant la méthode la plus efficace pour effectuer les tâches nécessaires à chaque étape.

5.5.5 Deuxième expérience

La deuxième expérience est réalisée pour calculer le TPV, TNF, TNV, TPF, PR, SK, et SF de notre deuxième approche hybride proposée et des quatre autres approches sélectionnées dans la littérature afin de prouver l'efficacité de notre deuxième approche hybride proposée. Le tableau 5.3 indique les résultats expérimentaux en termes de TPV, TNF, TNV, TPF, PR, SK, et SF obtenus par notre deuxième approche hybride proposée et par les classificateurs Xu *et al.* [238], Liao *et al.* [239], Maheswari *et al.* [240], et Es-sabery *et al.* 4.

TABLE 5.3 – Résultats expérimentaux en termes de TPV, TNF, TNV, TPF, PR, SK, et SF.

		TPV	TNF	TNV	TPF	PR	SK	SF
COVID-19_Sentiments	Notre approach	94.72	5.28	95.74	4.26	94.56	95.12	94.63
	Xu <i>et al.</i> [238]	60.75	39.25	61.04	38.96	61.29	59.43	61.01
	Liao <i>et al.</i> [239]	74.19	25.81	73.64	26.36	75.10	74.89	74.64
	Maheswari <i>et al.</i> [240]	62.79	37.21	63.51	36.49	64.07	61.42	63.42
	Es-sabery <i>et al.</i> 4	85.72	14.28	86.51	13.49	86.67	87.69	85.54
Sentiment140	Notre approach	91.83	8.17	90.58	9.42	92.31	91.76	92.06
	Xu <i>et al.</i> [238]	41.02	58.98	40.69	59.31	39.96	40.15	40.48
	Liao <i>et al.</i> [239]	58.97	41.03	59.02	40.98	58.99	58.75	58.97
	Maheswari <i>et al.</i> [240]	44.98	55.02	45.34	54.66	46	44.87	45.48
	Es-sabery <i>et al.</i> 4	81.41	18.59	82.33	17.67	83.04	84.58	83.87

Comme nous le voyons dans le tableau 5.3, notre approche hybride proposée surpasse toutes les autres méthodes en termes de TPV, TNF, TNV, TPF, PR, SK, et SF. Notre approche a obtenu un TPV égal à 94.72 % et 91.83 % sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement, qui mesure la proportion de tweets positifs que notre approche suggéré prédit avec précision. Notre modèle a également un TNF plus bas égal à 5.28 % et 8.17 % sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement, qui mesure le pourcentage de tweets positifs dans le jeu de données que notre approche suggéré prédit incorrectement. En outre, notre approche proposée a atteint un TNV égal à 95.74 % et 90.58 % sur les jeux de données COVID-19_Sentiments et Sentiment140, ce qui indique la proportion de tweets négatifs que notre approche développée classe avec précision. En ce qui concerne le pourcentage TPF de tweets négatifs que notre approche classe incorrectement, il est très bas par rapport aux

autres approches, qui s'élève à 4.26 %, et 9.42 % sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement. La précision PR mesure la proximité des critères d'évaluation les uns par rapport aux autres, et notre approche présente un taux de proximité plus élevé par rapport à toutes les autres approches évaluées qui est égale à 94,56 %, 92,31 % sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement. De plus, notre méthode a un taux SK plus élevé qui est égal à 95.12 % et 91.76 % sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement. Ce critère d'évaluation permet de mesurer le taux de proximité entre le TC observé et le TC attendu. Finalement, la métrique SF mesure la précision et la robustesse de notre modèle suggéré qui est égal à 94.63 % et 92.06 % sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement. Après avoir analysé toutes les mesures d'évaluation calculées, nous avons déduit que notre approche proposée est plus puissante et plus efficace que toutes les autres approches examinées.

5.5.6 Troisième expérience

Cette troisième expérience est réalisée pour mesurer le TE consommé par notre modèle hybride proposé et les approches sélectionnées dans la littérature à savoir Xu *et al.* [238], Liao *et al.* [239], Maheswari *et al.* [240], et Es-sabery *et al.* 4 pour entraîner les jeux de données COVID-19_Sentiments et Sentiment140. La figure 5.18, montre les résultats expérimentaux obtenus en termes de TE par la mise en œuvre de notre méthode hybride suggérée et les autres approches Xu *et al.* [238], Liao *et al.* [239], Maheswari *et al.* [240], et Es-sabery *et al.* 4 sur les jeux de données Sentiment140 et COVID-19_Sentiments.

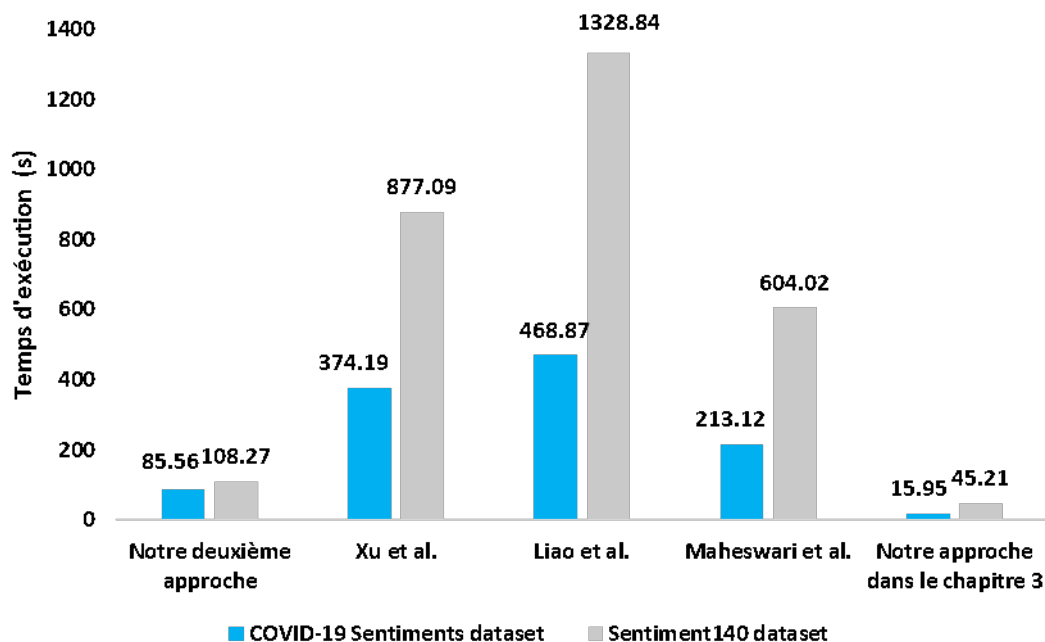


FIGURE 5.18 – TE consommé par l'approche hybride et les autres méthodes.

Comme présenté dans la figure 5.18, nous constatons que notre modèle hybride développé a consommé un temps égal à 58.56 s, et 108.27 s sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement. Xu *et al.* [238] a consommé un temps égal

à 374.19 s, 877.09 s sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement. Liao *et al.* [239] a dépensé un temps de 468.87 s, et 1328.84 s sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement. Maheswari *et al.* [240] a un temps d'exécution égal à 213.12 s, 604.02 s sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement. Notre classificateur proposé dans le chapitre 4 a consommé un temps égal à 15.95 s, 45.21 s sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement. D'après le résultat expérimental présenté, nous avons déduit que notre classificateur proposé dans le chapitre 4 a un temps d'exécution plus faible que le nôtre et que toutes les autres approches évaluées. De plus, notre modèle proposé a un temps d'exécution plus faible que toutes les autres approches examinées, à l'exception notre classificateur proposé dans le chapitre 4. Comme nous l'avons expliqué précédemment, notre approche proposée et notre classificateur dans le chapitre 4 ont été implémentées en utilisant le cluster Hadoop composé de cinq machines : un nœud de calcul maître et quatre nœuds esclaves. La différence entre les deux approches au niveau du temps d'exécution est due au nombre des instructions implémentées dans le programme de chaque approche. Où le nombre d'instructions de notre approche hybride suggérée est plus élevé que le nombre d'instructions de notre classificateur dans le chapitre 4, parce que notre approche proposée a intégré plusieurs algorithmes tels que les techniques de pré-traitement, l'algorithme FastText, le modèle de réseau neuronal convolutif (qui a un nombre d'instructions plus élevé), la fonction d'appartenance gaussienne, l'arbre de décision C4.5 basé sur le rapport de gain d'information flou, et la méthode générale de raisonnement flou. Contrairement à notre classificateur proposé dans le chapitre 4 qui utilise seulement les techniques de prétraitement, FastText, le gain d'information (a un petit nombre d'instructions), et notre ID3 amélioré basé sur le gain d'information pondéré (a un petit nombre d'instructions par rapport au C4.5 basé sur le ratio de gain d'information flou). Pour réduire le temps d'exécution consommé par notre deuxième approche hybride proposée, nous avons augmenté de manière incrémentielle le nombre de nœuds de calcul dans le cluster Hadoop, comme illustré sur la figure 5.19.

La figure 5.19 décrit les résultats expérimentaux après la mise en œuvre de notre approche hybride proposée sur plusieurs nœuds de calcul dans le cluster Hadoop.

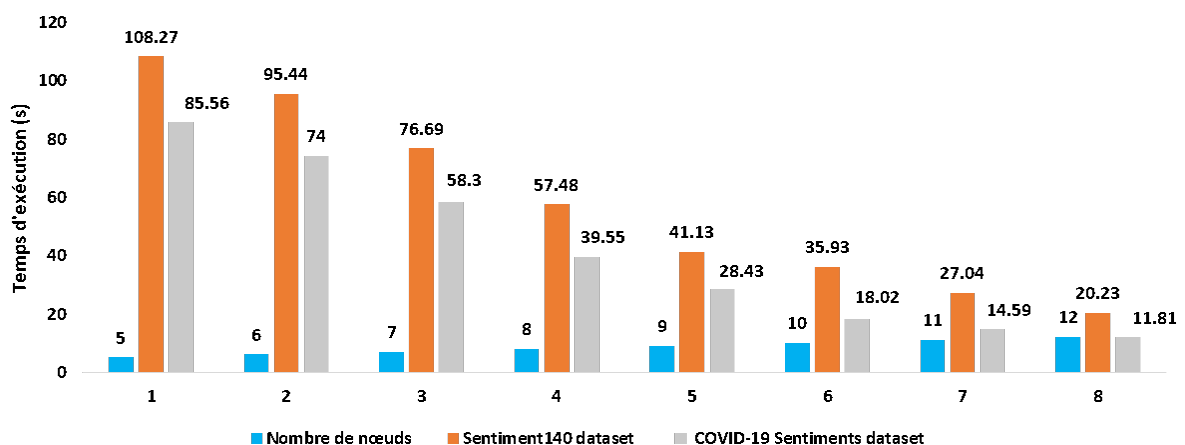


FIGURE 5.19 – TE en mettant en œuvre l'approche hybride sur un nombre différent de nœuds de Hadoop.

TABLE 5.4 – Complexité spatiale de l’approche hybride et les autres approches.

Nom du jeu de données	Techniques	N. instructions (M)	N. paramètres (M)
COVID-19_Sentiments	Notre approche	26.01	18.75
	Xu <i>et al.</i> [238]	9.79	5.34
	Liao <i>et al.</i> [239]	17.62	15.98
	Maheswari <i>et al.</i> [240]	14.47	10.63
	Es-sabery <i>et al.</i> 4	12.60	9.57
Sentiment140	Notre approche	39.80	25.06
	Xu <i>et al.</i> [238]	19.87	10.94
	Liao <i>et al.</i> [239]	25.76	23.41
	Maheswari <i>et al.</i> [240]	21.37	16.57
	Es-sabery <i>et al.</i> 4	29.106	20.192

Par conséquent, comme illustré dans la figure 5.19, nous observons que le temps d’exécution consommé par notre modèle hybride proposé a diminué de 108.27 s et 85.56 s sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement en utilisant cinq machines de calcul à 20.23 s, et 11.81 s sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement en utilisant douze nœuds de calcul. Après cette troisième expérience, nous déduisons que notre approche hybride proposée a un temps d’exécution plus bas que toutes les autres approches examinées.

5.5.7 Quatrième expérience

Dans la quatrième expérience, nous avons évalué l’efficacité de notre approche hybride développée et des approches sélectionnées dans la littérature, à savoir Xu *et al.* [238], Liao *et al.* [239], Maheswari *et al.* [240], et Es-sabery *et al.* 4 pour entraîner les deux jeux de données COVID-19_Sentiments et Sentiment140, respectivement, en mesurant la complexité spatiale et temporelle. Le tableau 5.4 présente les résultats expérimentaux en termes de la complexité spatiale après avoir calculé l’espace mémoire occupé par l’allocation des paramètres et l’implémentation des instructions de notre approche hybride développé, et les approches à savoir Xu *et al.* [238], Liao *et al.* [239], Maheswari *et al.* [240], et Es-sabery *et al.* 4.

Comme montré dans le tableau 5.4, nous observons que notre modèle hybride suggéré a effectué de nombreuses instructions qui occupent un espace mémoire égal à 39.80 M et 26.01 M dans le cas des jeux de données Sentiment140 et COVID-19_Sentiments, respectivement. En outre, l’espace mémoire alloué par les paramètres de notre approche est égal à 25.06 M et 18.75 M dans le cas des jeux de données Sentiment140 et COVID-19_Sentiments, respectivement. Comme l’indiquent les résultats expérimentaux, notre modèle hybride proposé occupe un espace mémoire bien plus élevé par rapport aux autres approches. Parce que notre approche met en œuvre de nombreuses techniques pour effectuer l’analyse des sentiments, comme l’application précise de toutes les tâches de pré-traitement nécessaires, la mise en œuvre de FastText, l’utilisation du réseau neuronal convolutif, l’application de la fonction d’appartenance gaussienne, et l’implémentation du C4.5 basé sur le ratio de gain d’information, enfin l’utilisation de la méthode de raisonnement flou générale. Contrairement aux autres approches qui ont mis en œuvre au plus trois techniques

pour effectuer la fouille d'opinion, ces approches ont obtenu des performances plus faibles en termes de TC, TR, TE, TPV, TNF, TNV, TPF, PR, SK, et SF par rapport à notre modèle hybride proposé.

Le tableau 5.5 présente les résultats expérimentaux obtenus concernant la complexité temporelle après avoir calculé le temps d'apprentissage et de test consommé par notre modèle hybride proposé et les autres approches évaluées.

TABLE 5.5 – Complexité temporelle de l'approche hybride et les autres approches.

Nom du jeu de données	Techniques	Temps d'apprentissage (s)	temps de test (s)
COVID-19_Sentiments	Notre approche	9.62	2.181
	Xu <i>et al.</i> [238]	336.77	37.42
	Liao <i>et al.</i> [239]	421.98	46.88
	Maheswari <i>et al.</i> [240]	191.80	21.31
	Es-sabery <i>et al.</i> 4	11.96	3.98
Sentiment140	Notre approche	16.21	4.023
	Xu <i>et al.</i> [238]	789.38	87.71
	Liao <i>et al.</i> [239]	1189.32	139.53
	Maheswari <i>et al.</i> [240]	542.98	61.04
	Es-sabery <i>et al.</i> 4	33.90	11.30

Comme le montre le Tableau 5.5, nous remarquons que notre modèle hybride proposé a consommé un temps d'apprentissage égal à 16.21 s et 9.62 s dans le cas des jeux de données Sentiment140 et COVID-19_Sentiments, respectivement. En outre, notre modèle hybride proposé a dépensé un temps de test égal à 4.023 s, et 2.181 s dans le cas des jeux de données Sentiment140 et COVID-19_Sentiments, respectivement. Comme le montrent les résultats pratiques obtenus, notre modèle hybride est moins complexe en termes de temps que les autres approches. Cette bonne performance en termes de complexité temporelle obtenue par notre modèle hybride est due à l'utilisation du cluster Hadoop qui se compose de douze nœuds de calcul : un nœud de calcul maître et onze nœuds de calcul esclaves, comme décrit dans la figure 5.19.

5.5.8 Cinquième expérience

Dans la cinquième expérience, nous avons évalué l'efficacité de notre méthode hybride proposée et des approches sélectionnées dans la littérature, à savoir Xu *et al.* [238], Liao *et al.* [239], Maheswari *et al.* [240], et Es-sabery *et al.* 4 pour entraîner sur les deux jeux de données COVID-19_Sentiments et Sentiment140 en démontrant la convergence de chaque approche évaluée à l'aide de l'équation 5.17 afin de déterminer le nombre d'itérations lorsque la méthode analysée a vérifié la condition décrite ci-dessous dans l'équation suivante 5.17.

$$Error_{ratep} - Error_{ratec} \geq T_{value} \quad (5.17)$$

Où $Error_{ratep}$ représente le taux d'erreur moyen atteint par l'approche évaluée à l'itération précédente du processus d'apprentissage, $Error_{ratec}$ mesure le taux d'erreur moyen

de la méthode évaluée à l'itération actuelle du processus d'apprentissage, et T_{value} est le taux seuil qui a initié le taux de convergence. Après avoir effectué de nombreuses expériences analysées, nous avons fixé cette valeur seuil à 0.0001. Ainsi, le taux d'erreur moyen de chaque méthode analysée est estimé en appliquant l'équation suivante :

$$E = \frac{1}{2} \times \frac{\sum_{j=1}^I \sum_{i=1}^D (z - z_{label})^2}{I} \quad (5.18)$$

Où I représente le nombre total d'instances stockées dans le jeu de données entraîné, D est le nombre total d'étiquettes de caractéristiques de décision dans le jeu de données utilisé, z est l'étiquette de caractéristique de décision attendue et requise à la sortie du processus de classification, et z_{label} représente l'étiquette obtenue de la caractéristique de décision à la sortie du processus de classification. Supposons que la formule définie dans l'équation (5.17) soit satisfaite. Dans ce cas, nous disons que la méthode entraînée converge, et l'algorithme est exécuté jusqu'à ce que le taux d'erreur moyen de la méthode entraînée atteigne la condition. Sinon, nous disons que la méthode entraînée n'est pas convergée.

La figure 5.20 montre le taux de convergence de notre modèle hybride proposé lorsqu'il a été exécuté sur les jeux de données Sentiment140 et COVID-19_Sentiments. Comme le présente la figure 5.20, nous remarquons que notre modèle hybride proposé a convergé vers la valeur seuil de 0.0001 après que l'algorithme de notre modèle hybride suggéré soit arrivé à 75 et 254 itérations lorsqu'il a été mis en pratique sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement.

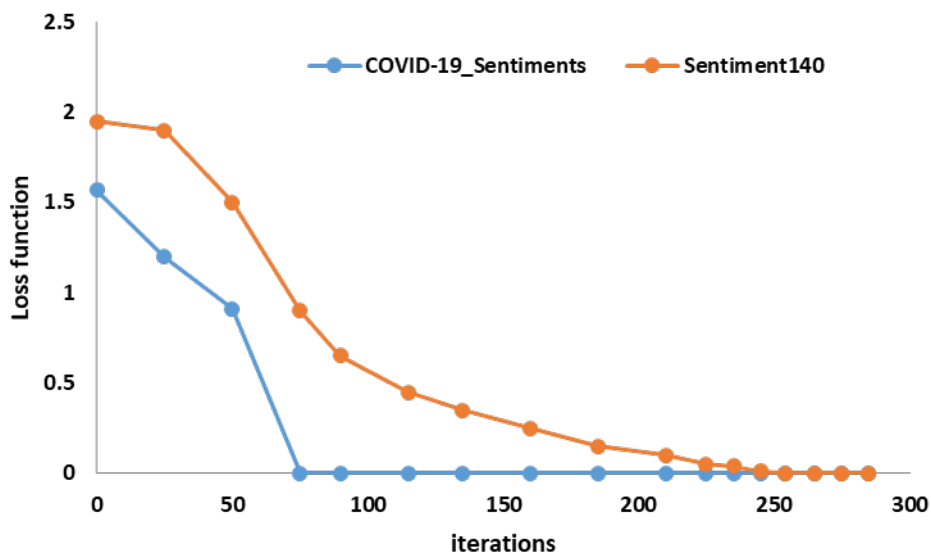


FIGURE 5.20 – Taux de convergence de l'approche hybride appliquée sur les deux jeux de données.

Le tableau 5.6 illustre le cycle de convergence de notre modèle hybride proposé et des autres approches. Comme décrit dans le tableau 5.6, nous déduisons que notre modèle hybride proposé converge très rapidement par rapport aux autres approches analysées car notre approche a un taux d'erreur plus faible par rapport aux autres approches évaluées.

TABLE 5.6 – Cycle de convergence de l’approche hybride et les autres approches.

Jeux de données	Algorithmes	Itérations
COVID-19_Sentiments	Notre approche	75
	Xu <i>et al.</i> [238]	675
	Liao <i>et al.</i> [239]	351
	Maheswari <i>et al.</i> [240]	514
	Es-sabery <i>et al.</i> 4	90
Sentiment140	Notre approche	254
	Xu <i>et al.</i> [238]	2198
	Liao <i>et al.</i> [239]	1347
	Maheswari <i>et al.</i> [240]	1775
	Es-sabery <i>et al.</i> 4	270

5.5.9 Sixième expérience

Dans cette dernière expérience, nous avons évalué les performances de notre modèle hybride proposé, et de différentes approches : Xu *et al.* [238], Liao *et al.* [239], Maheswari *et al.* [240], et Es-sabery *et al.* 4 en mesurant l’écart type moyen (ETM) de chaque approche sur cinq validations croisées différentes de du jeu de données donné. L’objectif principal de cette expérience est de déterminer l’approche la plus stable parmi toutes les approches examinées. Le tableau 5.7 présente les écart type moyen et le taux de classification moyen (TCM) de notre approche hybride suggérée par rapport aux autres approches sur les cinq validations croisées des deux jeux de données utilisés dans cette proposition.

TABLE 5.7 – Stabilité de l’approche hybride et les autres approches.

Jeux de données	Algorithmes	TCM (%)	ETM (%)
COVID-19_Sentiments	Notre approche	94.72	0.09
	Xu <i>et al.</i> [238]	51.44	1.67
	Liao <i>et al.</i> [239]	75.32	0.54
	Maheswari <i>et al.</i> [240]	63.84	1.12
	Es-sabery <i>et al.</i> 4	88.82	0.12
Sentiment140	Notre approche	92.27	0.18
	Xu <i>et al.</i> [238]	33.09	2.34
	Liao <i>et al.</i> [239]	59	0.98
	Maheswari <i>et al.</i> [240]	45.97	2.03
	Es-sabery <i>et al.</i> 4	86.53	0.26

Comme reporté dans le tableau 5.7, nous déduisons que notre approach hybride suggéré est plus stable comparé à Xu *et al.* [238], Liao *et al.* [239], Maheswari *et al.* [240], et Es-sabery *et al.* 4 sur cinq validations croisées différentes, car il a atteint un taux de classification moyen plus élevé de 94.72 % et de 92.27 % avec un écart type moyen plus faible de 0.09 % et de 0.18 % lorsqu’il a été mis en œuvre sur les jeux de données COVID-19_Sentiments et Sentiment140, respectivement.

5.6 Conclusion

Dans ce chapitre, nous avons tout d'abord amélioré l'algorithme de l'arbre de décision C4.5 au niveau du traitement avec des attributs à valeur continue. Cette amélioration est réalisée en utilisant la logique floue. Deuxièmement, nous avons proposé un nouveau modèle flou basé sur des règles, qui comprend trois phases, telles que la phase de fuzzification, la phase d'inférence et la phase de classification. Ce système proposé est prouvé par plusieurs expériences pour résoudre les problèmes de classification des données dans la fouille des données. Dans un premier temps, ce système applique la méthode de fuzzification pour déterminer le degré d'appartenance de chaque valeur d'attribut, et remplace la valeur continue de l'attribut par le terme linguistique qui a le degré d'appartenance le plus élevé. Cette phase initiale est réalisée pour faire face à l'incertitude et à l'imprécision des données. Dans l'étape suivante, l'algorithme C4.5 flou parallèle est appliqué pour construire l'arbre de décision flou, puis pour extraire l'ensemble des règles floues. Enfin, la méthode de raisonnement général est appliquée à l'ensemble des règles floues pour classer les nouvelles instances, et ensuite pour évaluer l'efficacité de notre modèle proposé.

Notre prochain travail est l'utilisation d'un modèle d'apprentissage en profondeur au lieu de l'algorithme d'arbre de décision C4.5 flou pour classifier les nouveaux tweets, et la recherche d'extracteurs de caractéristiques et de méthodes de sélection de caractéristiques pour comparer leur efficacité à le réseau neuronal convolutif qui vise à extraire et à sélectionner la caractéristique dans notre deuxième approche hybride proposée. Utilisation d'un système flou Mamdani pour traiter les données incertaines et vagues au lieu du modèle basé sur des règles floues utilisé dans cette contribution.

Classification des tweets Basée sur un classificateur d'apprentissage en profondeur parallèle et floue

6.1 Introduction

Les plateformes de réseaux sociaux comme Instagram, Twitter, Youtube, LinkedIn et Facebook ont été considérées comme essentielles et indispensables dans nos activités quotidiennes. Chaque jour, des milliards d'utilisateurs des réseaux sociaux diffusent des milliards des publications personnelles ou professionnelles [242]. Par exemple, les spécialistes du marketing utilisent les réseaux sociaux pour diffuser des publications professionnelles qui s'efforcent de présenter, promouvoir, annoncer et commercialiser leurs produits, services, événements et noms de marques. D'autre part, les clients interagissent avec les publications des spécialistes du marketing en exprimant leurs sentiments, opinions, idées, attitudes à propos des produits ou des services présentés [243]. En outre, les spécialistes du marketing recueillent les réactions des clients, les étudient et les analysent à l'aide de l'outil d'analyse des sentiments. L'objectif principal de ces opérations est d'améliorer la qualité de leurs produits et services, d'enrichir leurs offres en y ajoutant d'autres privilèges et d'améliorer les performances de leur marque [242], [243].

L'analyse des sentiments (AS) joue un rôle important dans *l'informatique décisionnelle* (ID). L'ID permet d'obtenir des réponses à des questions telles que «Pourquoi les ventes de produits sont-elles très faibles?», «Les besoins des clients sont-ils pleinement satisfaits par l'utilisation de nos produits?», «Qu'est-ce qu'ils ont le plus aimé dans nos services?», «Qu'est-ce qu'ils n'aimaient pas le plus?», «Nos clients sont-ils satisfaits d'utiliser nos produits/services ou ont-ils besoin de plus?» [244]. À savoir les méthodes de traitement automatique du langage naturel (TALN) pour prétraiter les données. Les méthodes *d'Apprentissage Automatique* (AA) pour classifier les sentiments. Les méthodes *d'Apprentissage en Profondeur* (AP) qui sont utilisées pour extraire automatiquement des caractéristiques à partir du texte et pour effectuer la classification des sentiments. Et les méthodes de vectorisation qui sont utilisés pour représenter du texte par des valeurs numériques.

La AS est également appelée "extraction d'opinion" et s'efforce à reconnaître les humeurs ou les émotions des gens à propos une entité telle que des événements, des produits, des individus, des services ou des sujets. À l'heure actuelle, la SA a été principalement

considérée comme une tâche de classification des sentiments dans le contexte du AA, c'est-à-dire que chaque sentiment exprimé dans le texte donné est classé comme positif, neutre ou négatif. Les techniques basées sur les données, impliquant des méthodes de AA et AP, sont considérées comme une solution précise et efficace pour effectuer la tâche de classification des sentiments. L'application des méthodes de AA a prouvé qu'elles sont un outil puissant pour classer les sentiments exprimés dans le texte donné. En particulier, les méthodes SVM, KNN, RF, LR, NB et DT, etc. qui sont largement utilisés avec une grande précision dans de vastes domaines d'application qui incluent l'analyse des sentiments, tels que la détection de la cyberhaine [245], critiques de films et de produits [246], [247], détection de langage abusif [248], identification de la cyberharcèlement ou de la cyberintimidation [249] et les réseaux sociaux [250]. En plus des algorithmes AA classiques présentés précédemment, il existe également des algorithmes AP tels que CNN, FFNN, LSTM, et RNN, qui sont actuellement préférés pour la classification des sentiments.

Au fil des âges, plusieurs méthodes ont été proposées pour fournir aux utilisateurs un processus efficace de classification des sentiments. Ces méthodes ont évolué à partir des approches basées sur le dictionnaire aux techniques de AA et actuellement aux modèles AP. Selon les études et analyses comparatives antérieures qui sont effectuées sur la AS en utilisant à la fois le AA et le AP. L'AA est plus efficace que AP dans les problèmes de classification des sentiments en raison de données massives [251]. La figure 6.1 prouve que la précision des algorithmes AA classiques est meilleure pour des données de petites tailles. Lorsque le volume de données dépasse une certaine taille, la précision des algorithmes de AA classiques devient constante. En revanche, la précision des algorithmes AP augmente en fonction de l'augmentation du volume des données.

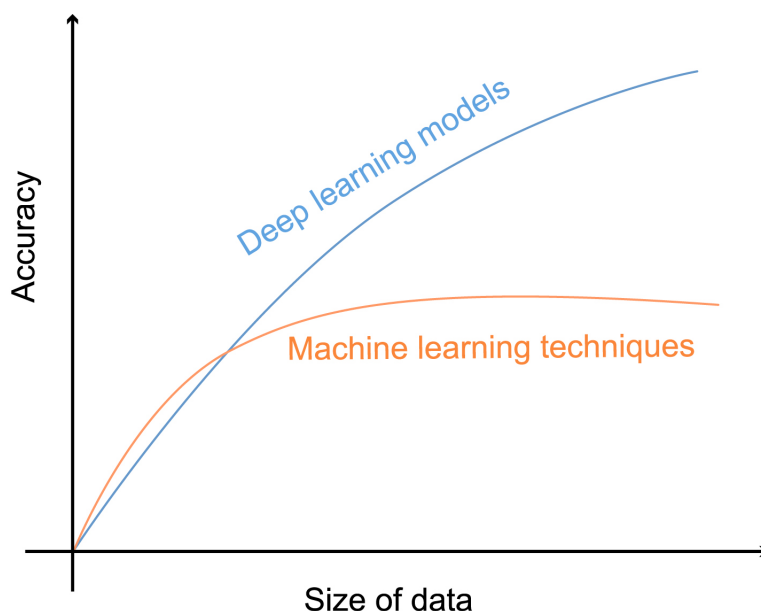


FIGURE 6.1 – Précision des approches AA en comparaison avec la précision des modèles AP par rapport à la taille des données.

Une différence principale entre les approches AA et les modèles AP réside dans la manière dont les caractéristiques sont extraites. Comme on le sait, la précision d'une

approche AA ou AP est extrêmement dépendante d'un bon processus d'extraction de caractéristiques à partir de données utilisées. Les techniques classiques de AA utilisent des caractéristiques d'ingénierie artisanales en employant de nombreuses méthodes d'extraction de caractéristiques, et appliquent de ce fait les méthodes d'apprentissage. Ce processus prend beaucoup de temps et permet d'extraire des caractéristiques incomplètes. En revanche, les caractéristiques sont extraites automatiquement dans le cas de AP, qui est le point fort des modèles AP par rapport aux techniques AA. En raison de l'augmentation accélérée du volume des données à cette époque et des bonnes performances des modèles DL sur la taille massive des données, nous avons décidé de combiner le Réseau Neuronal Convolutif (CNN) et le Réseau de Neurones à Propagation Avant (FFNN) afin de mettre en place un processus automatique plus efficace pour extraire les caractéristiques à partir de l'ensemble de données utilisées dans ce travail. Mais bien que l'utilisation des modèles AP les plus avancés, il y a une ambiguïté enracinée dans la NLP qui a besoin d'autres solutions.

D'après plusieurs études [2], [3], la théorie de la logique floue est considérée comme la solution la plus efficace pour traiter les données imprécises. La logique floue fournit un moyen éligible pour gérer les problèmes linguistiques au niveau des données floues. Zadeh et al. [3] discutent dans leurs travaux qu'aucun autre technique ne résout ces problèmes linguistiques. En conséquence, aux avantages introduits dans [3] sur la théorie de la logique floue dans le domaine des données floues, nous décidons d'employer cette théorie dans notre travail pour traiter l'incertitude inhérente aux données des médias sociaux.

Afin d'améliorer le taux de classification de la classification au niveau des phrases et de surmonter au mieux les lacunes discutées précédemment, nous introduisons un nouveau *Classificateur d'Apprentissage en Profondeur Flou* (CAPF) pour reconnaître la polarité (négatif, Positif, neutre) du sentiment exprimé dans la phrase analysée. Notre méthodologie dans ce chapitre comprend principalement cinq parties : la première partie concerne les étapes de prétraitement des données afin de réduire les données bruyantes et d'améliorer la qualité des données, la deuxième partie est l'application de méthodes de prolongement lexical de mots pour convertir les données textuelles en données numériques ; la troisième partie est l'application du modèle AP hybride proposé qui combine CNN et FFNN afin de construire un processus automatique efficace pour extraire les caractéristiques des données non structurées collectées et calculer à la fois un *Score Sentimental Positif* (SSP) et un *Score Sentimental Négatif* (SSN), et la quatrième partie est l'application de MFS qui est utilisée comme un classificateur flou pour classer les résultats des deux modèles d'apprentissage en profondeur (CNN+FFNN) utilisés en trois classes : Neutre, Négatif et Positif. Enfin, nous utilisons le cadre Hadoop pour paralléliser notre CAPF afin de surmonter le problème du long temps d'exécution. En outre, pour prouver la performance de notre CAPF suggéré, une étude comparative expérimentale entre notre modèle et certains autres modèles tirés de la littérature est réalisée. Les résultats pratiques ont prouvé que notre CAPF est plus performant que les autres modèles en termes Taux Positif Vrai (TPV), Taux Négatif Vrai (TNV) ou spécificité, Taux Positif Faux (TPF), Taux Négatif Faux (TNF), Taux d'Erreur (TR), Précision (PR), Taux de Classification ou exactitude (TC), Statistique Kappa (SK), Score F1 (SF), Temps d'Exécution (TE), Complexité, Convergence et Stabilité. La contribution de notre travail est fondamentalement incarnée en six aspects :

- Techniques de prétraitement des données sont utilisées pour améliorer la qualité des données textuelles de l'ensemble de données utilisé en supprimant les données de bruit existantes.
- Les méthodes de prolongement lexical de mots telles que Word2Vec, GloVe et FastText sont appliqués à l'ensemble de données utilisé pour transformer les textes en données numériques.
- Méthodes CNN et FFNN avec différents paramètres sont employés pour calculer le SSN et le SSP
- La théorie de la logique floue (MFS) est appliquée comme classificateur flou sur les résultats de l'étape précédente (SSN et SSP) afin de classer les phrases de l'ensemble de données utilisé en trois étiquettes : négatif, neutre et Positif. Une précision élevée a été obtenue parce que le CAPF suggéré suscite automatiquement des caractéristiques plus précises et diverses entités à partir de l'ensemble de données utilisé.
- Le cadre Hadoop est utilisé pour mettre en parallèle notre CAPF introduit afin d'éviter le problème du long délai d'exécution.
- Multiples expériences sont menées pour prouver l'efficacité de notre CAPF introduit, et ce CAPF est comparé à plusieurs classificateurs sélectionnés à partir de la littérature. Les résultats expérimentaux indiquent que le modèle suggéré atteignait un meilleur taux de classification que les autres classificateurs sur l'ensemble de données utilisé.

Le reste de ce chapitre est organisé comme suit : la section 2 présente en détail quelques travaux similaires sélectionnés dans la littérature. La section 3 décrit notre CAPF suggéré. La section 4 présente les résultats expérimentaux et les résultats de l'étude comparative visant à démontrer l'efficacité du classificateur proposé, et la section 6 présente la conclusion et le travail future.

6.2 Travaux connexes

L'objectif du processus d'AS est d'extraire la polarité émotionnelle incluse dans la phrase analysée. Pour cette raison, de nombreux chercheurs ont proposé diverses approches de classification des sentiments basées sur des méthodes AA, AP ou hybrides. Cette section présente des modèles récemment publiés dans la littérature. Les auteurs de l'article [252] ont proposé une nouvelle approche qui combine le CNN multi-échelle et le RRMCLT, afin d'augmenter le taux de classification et de réduire le taux d'erreur de l'AS. Ils traitent le texte des critiques de divers produits de manière différente, tout en préservant les caractéristiques communes entre chaque texte de critique, et intègrent les caractéristiques globales extraites et les caractéristiques locales fondées à partir de texte de critique afin d'améliorer l'efficacité du processus de classification. Leur classificateur suggéré est basé sur deux phases principales : premièrement, un système de d'apprentissage multi-tâches est appliqué pour détecter les caractéristiques privées et partagées à partir des données critiques de différents types de produits. La deuxième phase est la représentation de la phrase à l'aide des méthodes de prolongement lexical de mots [253]. Une étude comparative est menée par les auteurs afin de comparer leur modèle proposé avec d'autres approches telles que RL, FAD, NB, KPPV, MVS, XGBoost et l'amplification de gradient d'AD.

Les résultats expérimentaux de l'étude comparative prouvent que l'efficacité de l'AS de leur approche suggérée est plus précise que celle d'autres méthodes sélectionnées dans la littérature, avec une précision égale à 86.25%.

Lan, Hao, Xia, Qian, et Li [254] ont introduit un nouveau système d'apprentissage en profondeur appelée SRCLA, qui combine le processus d'attention inter-couches et le RNR résiduel empilé afin d'extraire plus des caractéristiques linguistiques puis de l'utiliser pour la tâche l'AS. L'objectif principal du SRCLA est de construire un RNR résiduel empilé pour identifier et filtrer différents types de caractéristiques sémantiques, puis d'utiliser le nouveau système d'attention inter-couches afin d'améliorer la tâche de filtrage. Basé sur le SRCLA proposé, il est possible d'identifier davantage de caractéristiques linguistiques et donc d'améliorer l'efficacité de la classification des sentiments. Les auteurs de ce travail ont appliqué les deux modèles RNRLC bidirectionnel empilé, et leur modèle SRCLA sur quatre ensembles de données TREC, SST-1, MR, et SST-2. Les résultats expérimentaux ont prouvé que SRCLA atteint une amélioration de 3.0% sur l'ensemble de données SST-1, un raffinement de 2.0% sur l'ensemble de données SST-2, une amélioration de 2.5% sur l'ensemble de données MR et un raffinement de 1.5% sur l'ensemble de données TREC, par rapport au RNRLC bidirectionnel empilé.

Lin, Li, Yang, Xu, et Lin [255] ont développé un nouveau modèle afin d'améliorer l'efficacité de l'AS. Le classificateur proposé intègre la capacité du RNRLC bidirectionnel à sélectionner des séries locales de caractéristiques et la puissance de l'attention multi-têtes pour identifier des caractéristiques approfondies. Ils utilisent une étude comparative pour améliorer le modèle proposé et augmenter ses performances. Les auteurs de l'article [256] ont proposé une approche hybride qui intègre les modèles AP avec les méthodes AA pour obtenir une meilleure classification des sentiments. Le modèle proposé est un modèle bilingue de classification des sentiments qui est appliqué aux ensembles de données en langues turque et chinoise. Cette méthode proposée combine RNR, RNRLC, NB, MVS et le plongement lexical de mots. Leurs résultats expérimentaux ont prouvé que la précision de l'approche proposée pouvait atteindre 89% et qu'elle était plus performante que toute autre approche AA ou AP individuellement. Dans l'article [257], les tweets sur les énergies renouvelables ont été classifiés comme étant positifs, neutres ou négatifs en appliquant cinq algorithmes d'apprentissage automatique différents, à savoir MVS, KPPV, BN, Ada-Boost et Bagging. Les auteurs de ce travail choisissent la fonction de gain d'information et CfsSubsetEvaluation [257] pour extraire les caractéristiques à partir des jeux de données utilisés. Ils implémentent leur approche proposée en employant les outils WEKA et R-Studio. Les résultats expérimentaux montrent que l'algorithme MVS surpasse les autres algorithmes d'apprentissage automatique lorsqu'il est incorporé avec la fonction CfsSubsetEvaluation.

Alec *et al.* [258] ont développé un nouveau classificateur d'apprentissage en profondeur pour la classification des sentiments par la combinaison de RNRLC avec CNN au niveau du noyau. Leur incorporation des schémas RNRLC et CNN a produit un nouveau modèle hybride d'apprentissage en profondeur avec une précision élevée lorsqu'ils l'ont appliqué au jeu de données de films sur internet. En outre, les auteurs présentent dans cet article de multiples variantes du modèle hybride qu'ils ont proposé afin de démontrer leurs tentatives d'améliorer la précision tout en réduisant la suradaptation. Ils ont effectué de nombreuses expériences avec différentes méthodes de régularisation, tailles de noyau et architectures

de réseau, afin de concevoir cinq modèles très performants à comparer avec d'autres approches de la littérature. Ces modèles ont obtenu 89 % de précision lorsqu'ils ont été utilisés pour prédire le score de polarité des commentaires à partir de jeux de données de films sur Internet.

Xing *et al.* [259] ont appliqué une nouvelle architecture de réseau neuronal pour l'analyse des sentiments de jeu de données de commentaires sur un marché. Ils ont combiné la méthode de clustering évolutif avec le modèle d'apprentissage en profondeur RNRLC. Les résultats expérimentaux de l'application de la méthodologie proposée sur des phrases provenant de StockTwits prouvent que le cadre suggéré atteint une bonne précision par rapport aux autres méthodes existantes de la littérature. Dans [260], les auteurs proposent une nouvelle méthodologie pour l'analyse des sentiments. Cette méthodologie combine le CNN, le plongement de mots, le RNR, et les lexiques de polarité. En outre, le système proposé est composé de trois CNNs afin d'extraire des caractéristiques de haut niveau à partir de représentations bruyantes du plongement de mots. Le résultat de ces trois CNNs est agrégé pour l'utiliser comme une variable d'entrée d'un PML entièrement connecté. Les résultats expérimentaux obtenus par leur classificateur étaient plus compétitifs dans plusieurs sous-tâches.

La logique floue est proposée pour traiter les données incertaines et imprécises. La force de la logique floue est sa ressemblance avec le raisonnement humain et le langage naturel. L'utilisation de la logique floue dans l'analyse des sentiments a fourni une meilleure précision de classification dans plusieurs travaux de la littérature. Par conséquent, l'efficacité de la logique floue dans le domaine de l'analyse des sentiments est basée sur la méthodologie adoptée dans le problème de la classification par les systèmes flous populaires. Pour cela, nous avons trouvé plusieurs travaux dans la littérature utilisant les systèmes flous pour résoudre les problèmes de classification des sentiments.

Wu *et al.* [261] ont proposé une méthode d'analyse des sentiments basée sur la logique floue. Le modèle qu'ils ont suggéré se résume à cinq étapes, à savoir la phase de collecte des données, l'étiquetage et l'examen manuels des données, le pré-traitement des données, la quatrième phase qui se sert pour extraire les caractéristiques essentielles, et la dernière étape est l'application du classificateur flou. Le classificateur flou se compose de trois parties, à savoir le processus de fuzzification, les règles floues SI-ALORS et le processus de défuzzification. Les auteurs de ce travail ont comparé l'approche qu'ils proposent avec l'approche de recherche par mots-clés. Le résultat expérimental démontre que leur méthode basée sur la logique floue a atteint de meilleures performances que la technique de recherche par mots-clés en termes de taux de correction et de tweets extraits.

Dans l'article [262], les auteurs ont développé un nouveau modèle basé sur des règles floues pour les problèmes d'analyse des sentiments. Les nouveautés de leur contribution sont i) la méthode proposée est non supervisée et peut être appliquée à n'importe quel ensemble de données et à n'importe quel dictionnaire. ii) la création de neuf règles floues pour classer les tweets. Ils ont mis en œuvre leur approche proposée en employant trois dictionnaires différents : AFINN, VADER et SentiWordNet. La méthode suggérée a été testée sur neuf jeux de données Twitter. Les résultats expérimentaux ont montré que l'approche qui utilise le dictionnaire VADER prend le moins de temps en exécution que le processus qui utilise le dictionnaire SentiWordNet. Pour les mesures de rappel et de

précision, les méthodes qui utilisent les règles floues ont obtenu de meilleures performances que celle qui utilisent les dictionnaires AFINN, VADER et SentiWordNet.

Abdul-Jaleel *et al.* [263] ont introduit un nouveau modèle pour résoudre le problème de l'analyse des sentiments. Ce modèle combine un algorithme génétique avec la théorie de la logique floue. Les entrées de ce classificateur proposé sont une collection de caractéristiques extraites à partir d'une phrase de sentiment, et le résultat de ce modèle de classification est la décision de classification pour la phrase de sentiment classifiée. Ils ont comparé le système de classification qu'ils ont proposé avec la méthode de recherche par mots-clés en calculant à la fois la précision et le taux incrémental. En termes de précision, le modèle introduit a donné de meilleurs résultats que la méthode de recherche par mots-clés, où la précision du système proposé est égale à (98,75%), mais la précision de cette méthode est égale à (95,7%). Pour le taux incrémental, l'approche proposée est capable d'extraire des phrases de sentiment plus que l'approche de recherche par mots-clés.

Motivés par les points forts des modèles d'apprentissage en profondeur et des techniques de logique floue dans le domaine de l'analyse des sentiments, nous développons dans le présent travail un nouveau classificateur d'apprentissage en profondeur parallèle et flou (CAPF), qui intègre fondamentalement les réseaux d'apprentissage en profondeur CNN+FFNN avec le système de logique floue MFS et le cadre Hadoop.

6.3 Phases de la méthode proposée

Dans les sections suivantes, nous allons discuter des motivations qui nous ont poussés à développer ce travail. L'architecture de notre modèle hybride est composée de la phase de collecte des données, des étapes de pré-traitement des données pour réduire les données bruyantes, des méthodes du plongement de mots pour transformer les données textuelles en données numériques, du CNN pour extraire automatiquement les caractéristiques essentielles, du FFNN pour calculer les valeurs SSN et SSP, et du MFS pour classifier leur entrée comme étant négatif, Positif ou neutre.

6.3.1 Motivation

Comme mentionné dans la section d'introduction 6.1, notre proposition s'efforce d'améliorer la performance de l'analyse des sentiments. Le but primordial de cette contribution est de classifier chaque tweet dans le jeu de données utilisé comme étant une classe Positif, négatif ou neutre avec une excellente performance et efficacité en termes de dix critères d'évaluation, qui sont présentés dans la section 3.9 du chapitre 3, et également en termes de convergence, de stabilité et de complexité.

Dans la littérature, de nombreuses catégories d'approches ont été appliquées pour effectuer la classification des sentiments. Parmi ces techniques, nous trouvons des modèles d'apprentissage en profondeur, des méthodes d'apprentissage automatique et des procédures basées sur des dictionnaires. Les performances des méthodes d'apprentissage automatique et des approches basées sur les dictionnaires sont inférieures à celles des mo-

dèles d'apprentissage en profondeur si nous les appliquons à un énorme jeu de données. Les études démontrent que les méthodes classiques d'apprentissage automatique et les techniques basées sur les dictionnaires sont meilleures pour des données de petite taille.

Dans ce travail, nous avons appliqué le modèle d'apprentissage en profondeur CNN comme extracteur automatique de caractéristiques. En raison du fait que le CNN possède une plus grande capacité à détecter et à extraire des caractéristiques pertinentes à différents niveaux locaux identiques à ceux d'un cerveau humain et par rapport aux modèles d'apprentissage en profondeur conventionnels qui ne peuvent pas effectuer l'apprentissage de caractéristiques, un autre avantage du CNN est le partage de poids, qui rend le CNN plus précis et efficace en termes de complexité et de mémoire utilisée que les réseaux neuronaux traditionnels. En outre, le CNN est caractérisé par une structure optimisée pour le traitement des textes et des images, la capacité d'extraire les caractéristiques abstraites, l'absorption des variations de forme par l'application de la couche de pooling, Le nombre de paramètres est moindre, et le taux de gradient décroissant est plus faible par comparaison avec les réseaux neuronaux classiques.

Nous avons également utilisé FFNN pour exploiter les sorties du CNN et calculer deux valeurs sentimentales SSN et SSP. Le SSP présente le pourcentage de mots d'opinion positifs existants dans le tweet sentimental donné, et le SSN présente le pourcentage de mots d'opinion négatifs existants dans le même tweet. En général, le FFNN est l'un des réseaux neuronaux les plus efficaces, qui est actuellement préféré pour les tâches de régression et de classification. La structure du FFNN est composée d'une couche d'entrée entièrement connectée, d'une ou plusieurs couches cachées et d'une couche de sortie. Par conséquent, le nombre variable de couches cachées permet de traiter des fonctions de plus en plus complexes. Ce type de réseau neuronal peut traiter les données par lui-même et générer des sorties qui ne sont pas limitées aux variables d'entrée qui lui sont fournies. En outre, il peut effectuer plusieurs opérations en parallèle sans que cela n'ait d'influence négative sur les performances du modèle de réseau neuronal. En outre, la régularisation du décrochage est appliquée au niveau de la couche entièrement connectée pour surmonter le surajustement du réseau et améliorer l'erreur de généralisation. En résumé, nous avons incorporé CNN et FFNN comme troisième étape de notre travail pour traiter les données non structurées collectées à partir des réseaux de médias sociaux et calculer les deux valeurs SSP et SSN comme sorties de notre modèle d'apprentissage en profondeur (CNN+FFNN).

En raison du bruit considérable et de l'imprécision imprévisible des données des médias sociaux, la notion de données ambiguës et incertaines suscite l'attention de nombreux chercheurs. Une telle imprécision représente un grand défi pour la capacité à mettre en œuvre et à classifier les données des médias sociaux. Tout d'abord, la capacité de symboliser les données des médias sociaux est limitée car les variables réagissent de manière incertaine. Deuxièmement, les modèles d'apprentissage en profondeur CNN et FFNN ne sont pas toujours performants lors du traitement des données des médias sociaux, qui sont perturbées par le bruit. La théorie de la logique floue a été appliquée pour surmonter les lacunes de l'apprentissage en profondeur et améliorer les performances de la classification des sentiments. Par comparaison avec les représentations logiques classiques, la représentation logique floue construit un ensemble de règles floues SI-ALORS pour éliminer les incertitudes dans les données des médias sociaux et atteint une plus grande précision à

la fois dans la symbolisation des données et dans la résistance au traitement des données bruyantes. Motivés par les points forts de la théorie de la logique floue, dans la quatrième étape de notre travail, nous avons employé le MFS comme classificateur flou. Simultanément, les variables d'entrée de le MFS sont les valeurs SSN et SSP, et la variable de sortie est l'étiquette de classe (Positif, Neutre, Négatif).

En résumé, l'essence de ce travail est d'augmenter l'efficacité de la classification des sentiments en intégrant la capacité du MFS à traiter les données incertaines et vagues et la capacité des deux modèles d'apprentissage en profondeur CNN et FFNN à détecter et capturer automatiquement les caractéristiques pertinentes à partir du jeu de données utilisé. Comme le schématise la figure 6.2, la structure globale de notre CAPF se compose de six phases, à savoir la collecte des données, le pré-traitement des données, le plongement de mots, le CNN, le FFNN et le MFS.

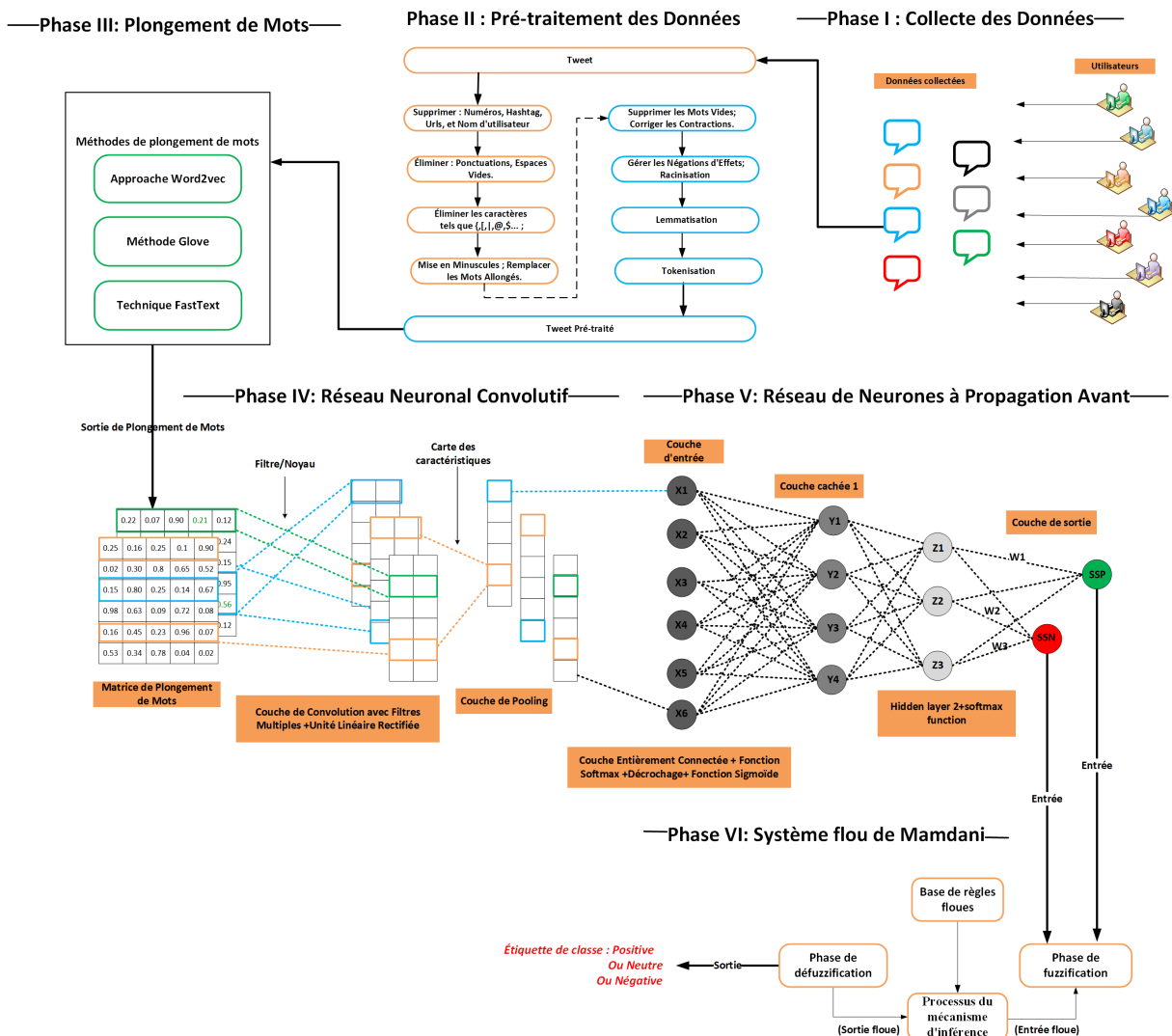


FIGURE 6.2 – Structure globale de notre classificateur d'apprentissage en profondeur parallèle et flou.

6.3.2 Pré-traitement et représentation des données

La première phase de notre proposition CAPF est l'étape de collecte de données dans laquelle nous avons sélectionné les jeux de données **Sentiment140** et **COVID-19_Sentiments**. Après la phase de collecte des données, la phase suivante est la phase de pré-traitement des données dans laquelle nous avons appliqué plusieurs tâches de pré-traitement. En général, les tâches de pré-traitement sont considérées comme la première phase de la tâche de classification des textes, et le choix de techniques de pré-traitement correctes et efficaces peut améliorer les performances de classification. L'objectif principal des tâches de pré-traitement des données est de préparer, normaliser, supprimer et nettoyer les données bruitées du jeu de données à classifier. Les données bruyantes sont celles qui ne contiennent aucune information utile pour la classification des sentiments. Les techniques de pré-traitement transforment les données bruitées des caractéristiques de haute dimension aux caractéristiques de basse dimension pour obtenir autant de données utiles précises que possible à partir du jeu de données employé. La phase de pré-traitement des données peut consister en de multiples techniques en fonction du problème de classification et de la situation. Dans ce travail, notre problème de classification est la classification des sentiments des données collectées à partir de Twitter. Twitter permet à ses utilisateurs de poster uniquement des messages de 140 caractères. En raison de cette règle restrictive, les utilisateurs de Twitter ont utilisé l'argot, les abréviations, les points d'exclamation, les liens, les répétitions, les signes de ponctuation pour affirmer leurs attitudes et leurs émotions dans un court tweet. En outre, les utilisateurs de Twitter sont vulnérables aux fautes d'orthographe et de typographie. Dans notre travail, il n'est pas essentiel d'inclure toutes les expressions du tweet dans le processus d'apprentissage, et plusieurs d'entre elles devraient être supprimées, normalisées, nettoyées ou remplacées par d'autres. Par conséquent, il est nécessaire d'appliquer des techniques de pré-traitement aux données, car leur élimination du bruit est un facteur crucial pour augmenter l'efficacité de la classification des sentiments. Les tâches de pré-traitement suivies dans cette contribution sont les suivantes : "*Supprimer les noms d'utilisateur, les numéros, les hashtags et les URL*", "*Supprimez les espaces blancs, la ponctuation et les caractères spéciaux*", "*Mise en minuscule*", "*Remplacez les mots allongés*", "*Supprimez les mots vides*", "*Racinisation*", "*Lemmatization*", "*Tokenisation*", "*Corriger les contractions*", "*Gérer les négations d'effets*" qui sont expliqués en détail dans la section 3.5 du chapitre 3.

Ensuite, l'étape de représentation des données suit l'étape de pré-traitement des données, qui consiste à convertir les données d'entrée textuelles en un ensemble de caractéristiques numériques. Comme on le sait, le modèle d'apprentissage en profondeur CNN ne peut traiter que les données numériques, et pour que notre contribution traite des données textuelles obtenues après la phase de pré-traitement des données à l'aide de modèles d'apprentissage en profondeur, ces données textuelles doivent être transformées en données numériques. Cette opération est appelée vectorisation ou représentation, qui est l'un des problèmes les plus critiques en TAL. Les approches telles que Word2Vec [264], GloVe [193], TF-IDF [265], Sac de Mots [196], FastText [195], N-grammes [194] sont les principales techniques de plongement de mots comme nous l'avons expliqué également dans la section 3.6 du chapitre 3. Selon l'étude comparative réalisée dans la section 4.5.2 du chapitre 4 sur les méthodes de plongement de mots, les résultats expérimentaux ont prouvé que les méthodes les plus efficaces dans le cas des jeux de données massifs sont

GloVe, Word2vec, et FastText, qui sont introduites respectivement par Stanford, Google, et Facebook. Par conséquent – dans ce travail – après la phase de pré-traitement des données, l'étape suivante est la représentation des données à l'aide des techniques Word2Vec et GloVe et FastText.

L'étape suivante de notre travail après l'étape de pré-traitement des données est la vectorisation des données en utilisant les techniques Word2Vec, GloVe, ou FastText. Dans le processus de pré-traitement des données, chaque phrase d'entrée est divisée en un ensemble de mots (Word2vec et GloVe) ou de caractères n-grammes (FastText). Ensuite, les méthodes de plongement de mots prennent cet ensemble de mots ou de caractères n-grammes comme entrées. En d'autres termes, ces méthodes prennent en entrée les vecteurs à un-coup qui représentent les mots ou les caractères n-grammes de la phrase. Nous symbolisons une phrase par $S = [W_{v1}; W_{v2}; \dots; W_{vn}]$, où W_{vi} est le vecteur de mots ou de caractères, qui est le vecteur à un-coup. La dimension du "vecteur à un-coup" est égale au nombre de caractères ou de mots dans la phrase de pré-traitement, dans ce cas, elle est égale à N . Une méthode pré-entraînée a multiplié sa matrice de poids (W_m) sur la matrice S et on obtient une représentation matricielle de basse dimension $M_r = [x_1; x_2; \dots; x_n]$ avec $x_i \in R^m$. Cette opération peut s'écrire comme suit :

$$M_r = W_m \cdot S \quad (6.1)$$

Où $W_m \in R^{m \cdot n}$ indique la matrice de poids, $M_r \in R^{m \cdot n}$ symbolise la représentation matricielle de basse dimension de la phrase, et S est la matrice à un-coup. Après la phase du plongement de mots, qui vise à transformer les données textuelles en données numériques. Elle prend un texte prétraité en entrée et produit une matrice de plongement. L'étape suivante est l'application du CNN, comme décrit dans la sous-section suivante.

6.3.3 Réseau neuronal convolutif

Après la phase du plongement de mots, le système proposé est entraîné à utiliser le modèle d'apprentissage en profondeur CNN, qui est formé de quatre couches. La première couche est appelée **Couche de plongement** ou **Couche d'entrée**, qui demande une matrice du plongement de mots en entrée, c'est-à-dire un ensemble de représentations vectorielles de la phrase analysée comme expliqué précédemment, où chaque vecteur $v_w \in R^{1 \cdot d_i}$ représente soit un caractère, soit un mot, selon la méthode du plongement de mots utilisée. Où d_i est la dimension du vecteur, et elle doit être inférieure à la taille du vocabulaire dans le dictionnaire de plongement. Dans notre travail, Word2vec, GloVe, et FastText ont été utilisés, qui sont éligibles pour découvrir les propriétés sémantiques et syntaxiques des caractères et des mots dans le jeu de données utilisé. Dans la première expérience de ce travail, nous l'avons effectué avec les méthodes du plongement de mots utilisées sous Hadoop, ces méthodes parallélisées ont été pré-entraînées sur 90 % du jeu de données utilisé. Après cette opération, nous obtenons un modèle pré-entraîné employé pour mapper chaque mot ou caractère sur sa propre représentation vectorielle. Nous avons ensuite calculé le taux d'erreur de ces méthodes du plongement de mots en utilisant les équations (4.2) et (3.27). Le taux d'erreur calculé indique que FastText est la méthode du plonge-

ment de mots la plus efficace. En fonction de ces résultats expérimentaux, nous utiliserons la méthode du plongement de mots FastText dans le reste de cette contribution. Donc, l'ensemble de vecteurs à haute dimension est calculé pour chaque caractère n-gramme en calculant la probabilité de softmax pour chaque caractère n-gramme en utilisant (3.13). La dimension de la représentation vectorielle produite est égale au nombre de nœuds neuronaux cachés dans la couche cachée Skip-Gram. Le nombre de nœuds neuronaux cachés a été fixé à 200. La taille de chaque tweet est paddée avec un vecteur de zéros. Le padding vise à garantir que tous les tweets du jeu de données utilisé ont la même dimension. Tous les vecteurs de représentation obtenus sont les lignes de la matrice de plongement E_m constituée de tous les caractères n-grammes du dictionnaire D . Ces caractères n-grammes sont notés avec les indices $1...|D|$ pour rechercher rapidement la représentation vectorielle du caractère n-gramme dans E_m . Ensuite, pour chaque tweet comportant t caractères n-grammes, une matrice de plongement $M = V_{nc1}; V_{nc2}; \dots; V_{nci}; \dots; V_{nc|t|}$ a été construite. Où V_{nci} est la représentation vectorielle du i th caractère n-gramme. Par conséquent, M est transmis à la couche de convolution.

La deuxième couche est appelée **Couche de convolution**, qui est appliquée à la matrice du plongement de mots M obtenue dans la couche précédente. En d'autres termes, chaque opération de convolution comprend une matrice de filtre (F), qui est appliquée à chaque fenêtre de caractère n-gramme (CW) dans la matrice du plongement de mots M , et une mappe de caractéristiques est générée comme résultat. Par conséquent, la couche de convolution est constituée de plusieurs opérations de convolution. Ainsi, plusieurs filtres avec des fenêtres de tailles variables sont appliqués à M , et un ensemble de mappes de caractéristiques est créé. Nous avons le $CW = [x_1; x_2; \dots; x_n]$ avec $x_i \in R^m$, une caractéristique k_i est produite à partir d'une CW dont la taille est $X_i : i + v - 1$ en utilisant la formule suivante :

$$k_i = \mathbf{ReLU}(F_i \cdot X_{i:i+l-1} + b) \quad (6.2)$$

Où ReLU est une fonction d'activation non linéaire décrite dans (3.11), $b \in R$ est le biais utilisé et l est la longueur du filtre utilisé F . Par conséquent, une mappe de caractéristiques $= [k_0, k_1, \dots, k_{i+l-1}]$ est obtenue par l'application de la formule décrite dans l'équation (6.2) sur toutes les fenêtres possibles CW de la matrice du plongement de mots M . Multiples filtres $F_{i:1 \rightarrow h}$ sont appliqués pour produire multiples mappes de caractéristiques $FM_{j:1 \rightarrow h}$. La figure 6.3 illustre un exemple simple d'application d'un filtre f sur un caractère n-gramme du mot "Fuzzy" pour calculer la mappe de caractéristiques basée sur (6.2).

En résumé, la couche de convolution prend une matrice de plongement M en entrée et produit un ensemble de mappes de caractéristiques en sortie. Elle est connue pour son efficacité et sa capacité à extraire automatiquement les caractéristiques locales. Ensuite, la troisième couche est **Couche max-pooling** qui est appliquée sur chaque mappe de caractéristiques dans l'ensemble $FM_{j:1 \rightarrow h}$ et extrait la valeur maximale de la mappe de caractéristiques $pv = \max[k_i]$. Dans cette couche, le nombre de ses sorties (L) sera similaire au nombre de ses mappes de caractéristiques d'entrée (L). Conformément à la couche de max-pooling, la taille de chaque dimension des mappes de caractéristiques d'entrée sera miniaturisée, et les sorties seront un ensemble de colonnes, où le nombre de

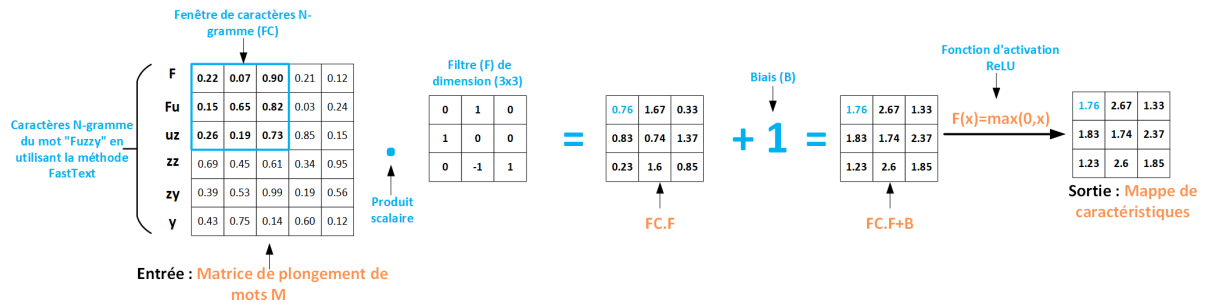


FIGURE 6.3 – Exemple d’application d’un filtre sur un caractère n-gramme du mot "fuzzy".

ces colonnes est égal au nombre de mappes de caractéristiques entrées. La miniaturisation appliquée par l’opération de max-pooling dépend de la taille de la dimension du noyau de max-pooling. La figure 6.4 présente un exemple d’une opération de max-pooling où nous avons utilisé 1x3 comme noyau de max-pooling.

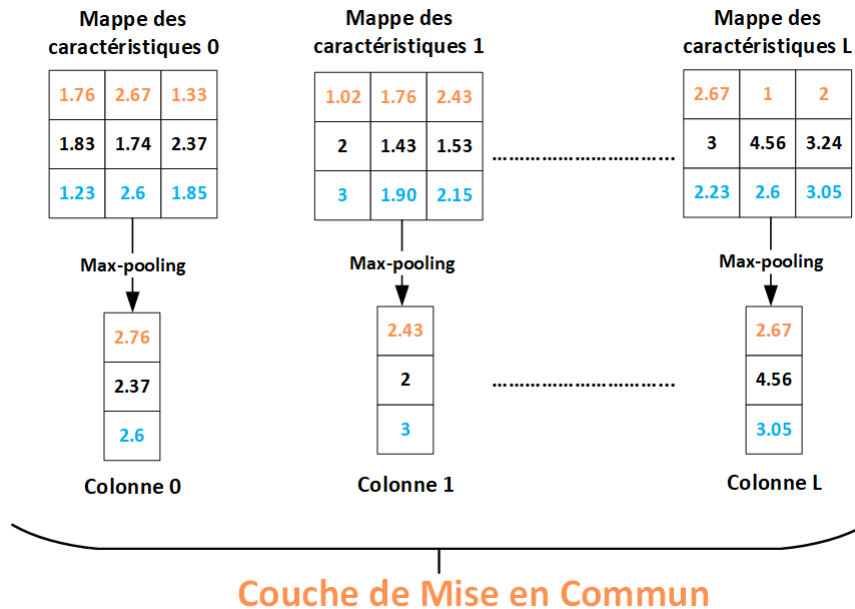


FIGURE 6.4 – Exemple d’application de la couche de pooling.

D’après la figure 6.4, nous constatons qu’une seule colonne de valeur est obtenue à la suite de l’unique opération de max-pooling. Par conséquent, cette opération vise à trouver et à sauvegarder la caractéristique essentielle et optimale en agrégeant les données et en diminuant la taille de la représentation. Enfin, la quatrième couche est la **Couche entièrement connectée**, qui est le point commun entre CNN et FFNN. En outre, la valeur de chaque neurone entièrement connecté est calculée dans la phase CNN à l’aide de la formule suivante (6.3).

$$V_n = \mathbf{f}(W_{connecteur} * C_{pooling} + B) \tag{6.3}$$

Où V_n est la valeur calculée du neurone, f est la fonction d’activation ReLU, $W_{connecteur}$ est le poids des connecteurs qui reposent sur la couche de pooling avec la couche entiè-

rement connectée, $C_{pooling}$ est l'ensemble des colonnes poolées (ou mappes de caractéristiques poolées) dans la couche précédente (couche de pooling), et B est le biais utilisé. L'algorithme suivant 6 résume toutes les étapes du CNN.

Le succès du modèle d'apprentissage en profondeur CNN est dû aux trois facteurs suivants : connectivité dispersée, poids partagé et représentation équivariante. Ainsi, le CNN diffère des réseaux neuronaux classiques, où la connexion entre le neurone d'entrée et de sortie est déterminée en multipliant la valeur du neurone par le poids du connecteur plus la valeur du biais, ce qui entraîne une charge de calcul. Alors que le CNN évite ce type de charge de calcul en se basant sur une connectivité dispersée, c'est-à-dire que la taille des noyaux est réduite pour être plus petite que la dimension des entrées en utilisant la couche de pooling. Ces noyaux sont considérés dans le reste du processus d'apprentissage CNN comme l'ensemble du texte/de l'image entré(e). Le paramètre de partage des poids permet d'augmenter l'efficacité de l'apprentissage en diminuant le nombre de paramètres de poids en cours d'apprentissage. L'idée principale derrière cette opération est que, au lieu d'entraîner plusieurs ensembles de paramètres de poids à chaque neurone comme dans le réseau neuronal classique, le CNN n'entraîne qu'un seul ensemble d'eux, ce qui donne une bonne performance en termes de taux de classification et de temps d'exécution. Les paramètres partagés du poids ont également donné au modèle d'apprentissage en profondeur CNN une nouvelle propriété appelée la représentation équivariante. Grâce à ces trois facteurs, le CNN nécessite moins de paramètres de poids que les autres modèles de réseaux neuronaux, ce qui minimise la taille de la mémoire utilisée et améliore l'efficacité du CNN.

En général, le modèle d'apprentissage profond que nous proposons (CNN+FFNN) est divisé en deux parties : la première partie consiste à appliquer le CNN sur la matrice du plongement de mots M obtenue lors de la phase précédente du plongement des mots pour capturer et extraire les caractéristiques les plus importantes. Dans la deuxième partie, nous utilisons un réseau d'apprentissage en profondeur FFNN pour calculer SSN et SSP. Le FFNN reçoit en entrée les sorties du CNN et génère les deux valeurs SSN et SSP. La sous-section suivante présente en détail le FFNN appliqué dans ce travail.

6.3.4 Réseau de neurones à propagation avant

Après la phase CNN, l'étape suivante est le FFNN. L'objectif principal de cette phase est de prendre l'ensemble des caractéristiques obtenues dans la phase CNN et de calculer les deux valeurs, qui sont les scores sentimentaux négatifs et positifs. Notre version simple du FFNN se compose de quatre couches : la couche d'entrée entièrement connectée, deux couches sigmoïdes cachées et la couche de sortie softmax. La couche entièrement connectée d'entrée est la même couche entièrement connectée du modèle d'apprentissage en profondeur CNN. Donc, cette couche est constituée de plusieurs neurones qui représentent les caractéristiques extraites lors de la phase CNN précédente. Tous les nœuds de neurones de la couche entièrement connectée sont liés à tous les nœuds de neurones de la première couche cachée via les connexions avec différents poids, qui sont ajustables. La valeur du neurone caché est calculée à l'aide de l'équation suivante (6.4) :

Algorithm 6 Réseau neuronal convolutif

Input : Une matrice du plongement de mots donnée M est décrite par R lignes et C colonnes, et le SF est l'ensemble des filtres de taille variable.

Output : Ensemble de caractéristiques.

==Opérations de calcul de la couche de convolution.==

```

for  $a \leftarrow 1$  to  $R$  do
  for  $b \leftarrow 1$  to  $C$  do
    for  $c \leftarrow 1$  to  $k$  do
      for  $d \leftarrow 1$  to  $k$  do
        som = 0
        for  $v \leftarrow 1$  to  $f$  do
          for  $w \leftarrow 1$  to  $f$  do
            som = som +  $F[v][w] * M[ss*(c-1)+v][ss*(d-1)+w]$ ; Où  $ss$  est le pas
            de décalage
          end for
        end for
         $FM[a][c][d] = FM[a][c][d] + som$ ; Où  $FM$  est la matrice de la mappe des
        caractéristiques.
        si  $b == C$  alors
           $FM[a][c][d] = F(FM[a][c][d] + B)$ ; Où  $B$  est le biais utilisé, et  $f$  est la fonc-
          tion d'activation ReLU.
           $FM[a][c][d] = \max(0; (FM[a][c][d] + B))$ ;
        end for
      end for
    end for
  end for

```

end for

==Opérations de calcul de la couche de pooling.==

valeur-maximale = 0;

valeur-moyenne = 0;

for $a \leftarrow 1$ **to** R **do**

for $c \leftarrow 1$ **to** C **do**

$y = 0$;

for $d \leftarrow 1$ **to** k **do**

$x = 0$;

valeur-maximale = $\max(\text{valeur}, FM[a][c][d])$; Lorsque l'opération utilisée est le maximale-pooling.

valeur-moyenne = $+ FM[a][c][d]$; Lorsque l'opération utilisée est le moyenne-pooling.

end for

$C_{pooling}[a][y][x] = \text{valeur-maximale}$; Lorsque l'opération utilisée est le maximale-pooling.

$C_{pooling}[a][y][x] = \text{valeur-moyenne} / (r * c)$; Lorsque l'opération utilisée est le moyenne-pooling.

$x++$;

end for

$y++$;

end for

==Opérations de calcul de la couche entièrement connectée==

```

for  $a \leftarrow 1$  to  $R$  do
  vartemp=0; for  $c \leftarrow 1$  to  $C$  do
    for  $d \leftarrow 1$  to  $k$  do
      vartemp = vartemp+  $W_{connecteur}[a][c][d] * C_{pooling}[a][y][x]$ ; où  $C_{miseencommun}$ 
      est la mappe de caractéristiques poolée et  $W_{connecteur}$  est le poids du connecteur.
    end for
  end for
   $Y[a][c][d] = vartemp$ ;
end for
return Ensemble de caractéristiques  $Y[a]$ 

```

$$X_h = \sigma\left(\sum_{i=1}^m W_i * X_i\right) = \frac{1}{1 + e^{(\sum_{i=1}^m W_i * X_i)}} \quad (6.4)$$

Où X_h est la valeur du neurone caché, W_i est le poids du connecteur i , X_i est la valeur de chaque neurone dans la couche entièrement connectée, et σ est la fonction d'activation sigmoïde, qui est calculée à l'aide de l'équation suivante (6.4) :

$$\sigma = \frac{1}{1 + e^x} \quad (6.5)$$

Où e^x est la fonction exponentielle standard de la valeur d'entrée x .

La fonction d'activation sigmoïde est appliquée aux réseaux neuronaux en tant que fonction d'activation et est également connue sous le nom de fonction d'écrasement, c'est-à-dire que cette fonction garantit que la sortie du neurone aura une valeur dans l'intervalle $[0,1]$. En pratique, la fonction d'activation sigmoïde utilisée au niveau de la couche cachée est représentée par le graphique de la figure 6.5 :

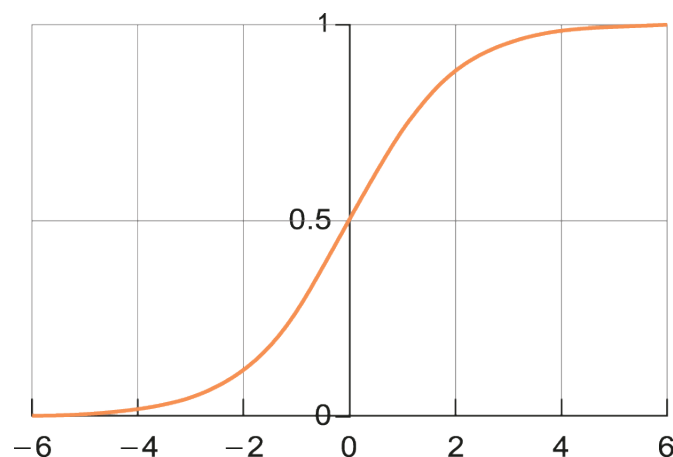


FIGURE 6.5 – Représentation graphique de la fonction d'activation sigmoïde.

La deuxième couche cachée du FFNN a la même fonction que la première couche cachée. Elle utilise également la méthode d'activation sigmoïde pour calculer la valeur de

ses nœuds de neurones cachés. La différence entre les deux couches cachées réside dans le nombre de nœuds de neurones cachés. La deuxième couche cachée comporte moins de nœuds de neurones cachés que la première. À chaque niveau de couche cachée, une fonction de pondération est appliquée aux entrées et les transmet via une fonction d'activation en tant que sortie. En outre, les couches cachées peuvent changer en fonction des poids qui leur sont associés. Dans ce travail, nous utilisons ces deux couches cachées comme fonctions d'écrasement car les sorties prévues de ce modèle sont des degrés de probabilité, c'est-à-dire que la valeur de sortie sera dans l'intervalle $[0,1]$.

La dernière couche de notre FFNN utilisé est la couche de sortie softmax. Les entrées introduites dans cette couche sont les résultats de la deuxième couche cachée multipliés par les poids des connexions, et le résultat passe par la fonction d'activation softmax au niveau des nœuds des neurones de la couche de sortie. Cette couche de sortie produit deux valeurs, qui sont SSN et SSP. La valeur des deux est comprise entre 0 non inclus et 1 non inclus. Par conséquent, l'algorithme suivant 7 décrit toutes les étapes du FFNN.

Dans ce travail, nous avons également utilisé l'opération de décrochage, qui indique le décrochage d'un certain ensemble de neurones cachés et visibles dans notre FFNN afin d'éviter le problème de surajustement. C'est-à-dire que ces neurones qui abandonnent aléatoirement ne sont pas considérés pendant la phase d'apprentissage dans la propagation vers l'avant ou vers l'arrière, comme le montre la figure 6.6. À chaque tour de la phase d'apprentissage, chaque neurone est soit inactif (abandon) dans l'architecture totale du FFNN avec le degré de probabilité $1-p$, ou soit actif avec le degré de probabilité p . Une question se pose : pourquoi abandonnons-nous certains ensembles de neurones dans toutes les couches du réseau ? L'objectif principal de l'opération de décrochage est de prévenir le surajustement résultant de la co-dépendance établie par les neurones entre eux à chaque tour pendant la phase d'apprentissage. En bref, le décrochage est une méthode de régularisation dans les FFNN qui permet d'éliminer l'apprentissage interdépendante entre les nœuds.

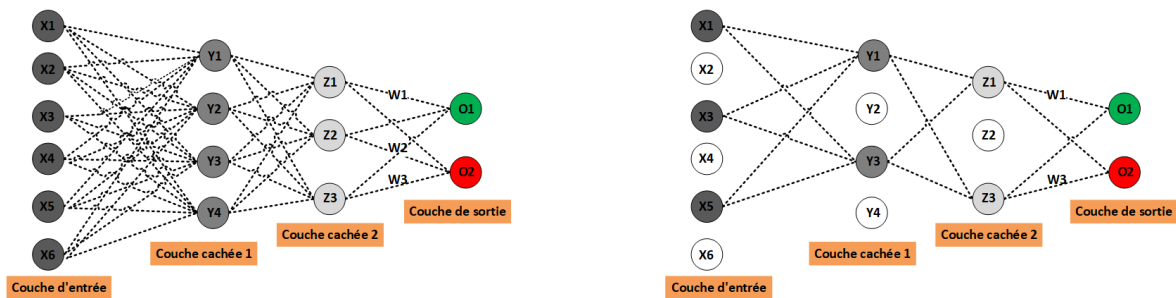


Figure a: Réseau de neurones à propagation avant sans abandon.

Figure b: Réseau de neurones à propagation avant avec abandon.

FIGURE 6.6 – Réseau de neurones à propagation avant avec et sans l'opération de décrochage.

6.3.5 Système flou de Mamdani

Après notre phase d'apprentissage en profondeur (CNN+FFNN), l'étape suivante est la classification à l'aide du classificateur flou. Les deux sorties SSN et SSP du modèle

Algorithm 7 Notre Réseau de Neurones à Propagation Avant

Input : Un ensemble donné de mappes de caractéristiques poolées $C_{pooling}$ décrit par \mathbf{R} lignes et \mathbf{C} colonnes, et $b = 1$ est le biais utilisé

Output : Les deux valeurs SSN et SSP.

Initialiser aléatoirement les poids du réseau neuronal à l'aide de l'équation suivante

$W_i^k = U_d[-\frac{1}{\sqrt{n^{k-1}}}; \frac{1}{\sqrt{n^{k-1}}}]$; Où U_d est la distribution uniforme continue, n^{k-1} est le nombre de neurones dans la $(k-1)$ ème couche, et i est le i ème connecteur.

do

==Opérations de calcul de la couche d'entrée entièrement connectée==

for $a \leftarrow 1$ **to** R **do**

$vartemp=0$; **for** $b \leftarrow 1$ **to** C **do**

for $c \leftarrow 1$ **to** k **do**

$vartemp = vartemp + W_{connecteur}[a][b][c] * C_{pooling}[a][b][c]$; Où $C_{pooling}$ est la mappe de caractéristiques de pooling et $W_{connecteur}$ est le poids du connecteur.

end for

end for

$Y[a][b][c] = vartemp$;

end for

==Opérations de calcul de la première couche cachée==

for $a \leftarrow 1$ **to** R **do**

$vartemp=0$; **for** $b \leftarrow 1$ **to** C **do**

for $c \leftarrow 1$ **to** k **do**

$vartemp = vartemp + W_{connecteur}[a][b][c] * Y[a][b][c]$; Où Y est l'ensemble des caractéristiques extraites par le CNN dans la phase précédente, et $W_{connecteur}$ est le poids du connecteur.

end for

end for

$fh[a][b][c] = \sigma(vartemp + B) = \frac{1}{1+e^{vartemp+B}}$; Où σ est la fonction d'activation sigmoïde.

end for

==Opérations de calcul de la deuxième couche cachée==

for $a \leftarrow 1$ **to** R **do**

$vartemp=0$; **for** $b \leftarrow 1$ **to** C **do**

for $c \leftarrow 1$ **to** k **do**

$vartemp = vartemp + W_{connecteur}[a][b][c] * fh[a][b][c]$; Où fh est la sortie de la première couche cachée, et $W_{connecteur}$ est le poids du connecteur

end for

end for

$sh[a][b][c] = \sigma(vartemp + B) = \frac{1}{1+e^{vartemp+B}}$

end for

while $lf < 0.000001$

==Opérations de calcul de la couche de sortie de softmax==

```

for  $a \leftarrow 1$  to  $R$  do
  vartemp=0; for  $b \leftarrow 1$  to  $C$  do
    for  $c \leftarrow 1$  to  $k$  do
      vartemp = vartemp+Wconnecteur[a][b][c] * fh[a][b][c]; Où  $fh$  est la sortie de la
      première couche cachée, et  $W_{connecteur}$  est le poids du connecteur.
    end for
  end for
  sortie[a][b][c] =  $f(vartemp + B) = \frac{e^{vartemp+B}}{\sum_{i=1}^t e^{vartemp+B}}$ 
end for
oo0 = sortie[0][0][0];
oo1 = sortie[0][0][1];

```

Par conséquent, l'adaptation de ce réseau neuronal peut être effectuée en réduisant (optimisant) la fonction de perte du réseau neuronal lf. La fonction de perte est donnée par l'équation suivante :

$lf(w(i)) = \frac{1}{N_c} * \sum_{i=1}^{N_c} \sum_{j=1}^{N_{on}} (ro - oo_j)^2$; Où $lf(w(i))$ est le taux d'erreur au i ème tour, $w(i)$ sont les poids réels des connecteurs au i ème tour; ro est la sortie requise du noeud de neurone; oo_j est la valeur obtenue du j ème noeud de neurone de sortie; N_{on} est le nombre de noeuds de neurones de sortie et N_c est le nombre de connecteurs. $SSN = oo_0$;

$SSP = oo_1$;

return Les deux valeurs SSN et SSP

d'apprentissage en profondeur (CNN+FFNN) seront les entrées de notre classificateur flou. Les sentiments exprimés par les humains sur un sujet particulier sont vagues et imprécis. Il est donc difficile de déterminer si leur opinion est négatif, neutre ou Positif. Notre modèle d'apprentissage en profondeur proposé (CNN+FFNN) est puissant pour extraire les caractéristiques à partir du jeu de données utilisé, mais il est impuissant à traiter les données vagues et ambiguës. Pour faire notre proposition plus précise et plus efficace, nous avons appliqué la théorie des ensembles flous sur les sorties du modèle d'apprentissage en profondeur proposé (CNN+FFNN), principalement, nous avons utilisé le MFS comme classificateur flou.

Le MFS est construite en utilisant la théorie des ensembles flous introduite par Zadeh [204]. L'objectif principal de cette théorie est de traiter les concepts imprécis et vagues comme le fait le cerveau humain. Selon de nombreux travaux dans la littérature, cette théorie a prouvé son efficacité à traiter les données ambiguës, et sa capacité à traiter les données comme le cerveau humain. Par conséquent, cette théorie est largement appliquée pour résoudre des problèmes de la vie réelle qui ne peuvent être résolus et traités par l'application de la théorie des ensembles classique. Basé sur la théorie des ensembles flous, de multiples systèmes flous sont proposés. Nous trouvons parmi eux le système flou de Mamdani, Tsukamoto et Sugeno [225]. Ces deux derniers systèmes sont appliqués dans le problème de la régression, mais le Mamdani est utilisé dans le problème de la classification. Parce que notre travail sert à résoudre le problème de la classification, nous avons utilisé le Mamdani. Le MFS comprend trois phases principales, à savoir le processus de fuzzification, le processus d'inférence et le processus de défuzzification comme l'illustre

la figure 6.7.

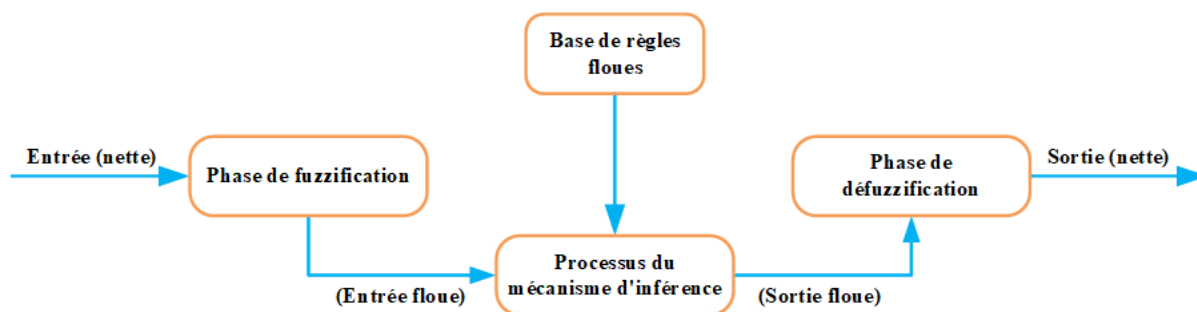


FIGURE 6.7 – Les composantes du système flou de Mamdani.

La première étape avant la procédure de fuzzification est la définition des variables linguistiques d'entrée et de sortie et la définition des termes linguistiques de chaque variable linguistique. Ainsi, dans ce travail, les variables linguistiques d'entrée sont SSN et SSP avec chaque variable prend cinq termes linguistiques qui sont TrèsFaible (entre 0.0 et 0.25), Faible (entre 0.0 et 0.50), Modéré (entre 0.25 et 0.75), Élevé (entre 0.50 et 1) et TrèsÉlevé (entre 0.75 et 1). La variable de sortie est la décision de classification qui comporte trois termes linguistiques : neutre (est compris entre 0.0 et 0.35), négatif (est compris entre 0.35 et 0.65), et positif (est compris entre 0.65 et 1.0). En bref, les entrées de la MFS seront le SSN et SSP, et la sortie sera la décision de classification (DC) telle que décrite dans le Tableau 6.3. Après la définition des variables linguistiques et des termes linguistiques de notre système flou proposé. La phase suivante est le processus de fuzzification qui est une étape importante dans le MFS.

TABLE 6.1 – Paramètres d'entrée et de sortie du système flou utilisé.

Variabes	Variabes linguistiques	Plage	Termes linguistiques	Paramètres
Entrée	SSN	0-1	TrèsFaible	0.0-0.25
			Faible	0.0-0.50
			Modéré	0.25-0.75
			Élevé	0.50-1
			TrèsÉlevé	0.75-1
Entrée	SSP	0-1	TrèsFaible	0.0-0.25
			Faible	0.0-0.50
			Modéré	0.25-0.75
			Élevé	0.50-1
			TrèsÉlevé	0.75-1
Sortie	DC	0-1	Négatif	0.0-0.35
			Neutre	0.35-0.65
			Positif	0.65-1

6.3.5.1 Processus de fuzzification

La méthode de fuzzification est l'opération qui transforme l'ensemble d'entrée clair en un ensemble d'entrée flou en calculant le degré d'appartenance à l'aide de l'une des

fonctions d'appartenance (FAs) les plus populaires. Les variables d'entrée SSP et SSN du MFS sont représentées par les ensembles flous "TrèsFaible", "Faible", "Modéré", "Élevé" et "TrèsÉlevé" en employant des FAs tels que triangulaire, trapézoïdal ou gaussien. Ainsi la variable de sortie DC du MFS est représentée pas les termes linguistiques "Négatif", "Neutre" et "Positif". Les variables et les termes linguistiques sont essentiellement des phrases complètes ou des mots utilisés par le TAL. Lorsque nous définissons les termes et les variables linguistiques, nous assurons qu'aucune donnée numérique n'est utilisée dans les variables et les termes linguistiques. Les deux points importants de cette phase sont la FA utilisés et les ensembles flous définis, car nous les employons pour obtenir les valeurs floues. La transformation d'ensembles d'entrée clairs en ensembles flous s'effectue en utilisant des FAs, et cette fonction de conversion est appelée fuzzification. En d'autres termes, le degré d'appartenance de chaque variable d'entrée à chaque terme linguistique (ensemble flou) est calculé en utilisant un FA particulier. Plusieurs MFs existent dans la littérature sont les fonctions d'appartenance Trapézoïdale, Triangulaire, Gaussienne, 2-D, Gauche-Droite, Sigmoidale et Bell Généralisée. Dans ce travail, nous avons choisi d'appliquer les fonctions d'appartenance Triangulaire, Trapézoïdale et Gaussienne qui sont les plus utilisées. Ces fonctions sont décrites ci-dessous.

Fonction triangulaire : est déterminée par trois paramètres, à savoir ll , v et ul . Où ll est la limite inférieure, ul est la limite supérieure, et la valeur médiane v , où $ll < v < ul$. Elle est décrite par l'équation (6.6) et la figure 6.8 présente sa représentation graphique.

$$\mu_A(x) = \begin{cases} 0 & \text{si } x \leq ll \\ \frac{x-ll}{v-ll} & \text{si } ll \leq x \leq v \\ \frac{ul-x}{ul-v} & \text{si } v \leq x \leq ul \\ 0 & \text{si } c \leq x \end{cases} \quad (6.6)$$

Une autre expression mathématique (6.7) est obtenue en appliquant les fonctions min et max à l'équation précédente.

$$\mu_A(x; ll, v, ul) = \max(\min(\frac{x-ll}{v-ll}, \frac{ul-x}{ul-v}), 0) \quad (6.7)$$

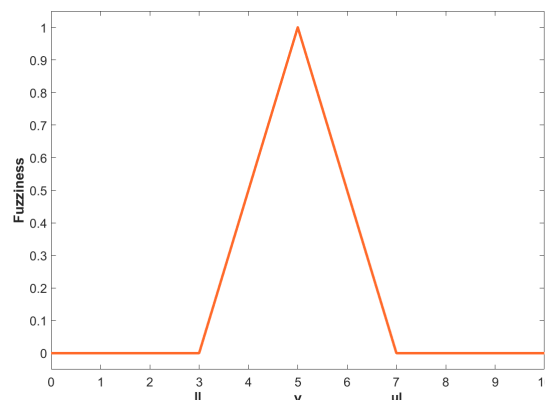


FIGURE 6.8 – Représentation graphique de la fonction triangulaire.

Fonction trapézoïdale : est déterminée par quatre paramètres : ll , lsl , usl et ul . Où ll est la limite inférieure, lsl est la limite inférieure de support, usl est la limite supérieure de support, et ul est la limite supérieure, et $ll < lsl < usl < ul$. Elle est décrite par l'équation (6.8) et la figure 6.9 présente sa représentation graphique.

$$\mu_A(x) = \begin{cases} 0 & \text{si } (x < ll) \text{ or } (x > ul) \\ \frac{x-ll}{lsl-ll} & \text{si } ll \leq x \leq lsl \\ 1 & \text{si } lsl \leq x \leq usl \\ \frac{ul-x}{ul-usl} & \text{si } usl \leq x \leq ul \end{cases} \quad (6.8)$$

Une autre expression mathématique (6.9) est calculée en appliquant les fonctions min et max à l'équation précédente (6.8) :

$$\mu_A(x; ll, lsl, usl, ul) = \max(\min(\frac{x-ll}{lsl-ll}, 1, \frac{ul-x}{ul-usl}), 0) \quad (6.9)$$

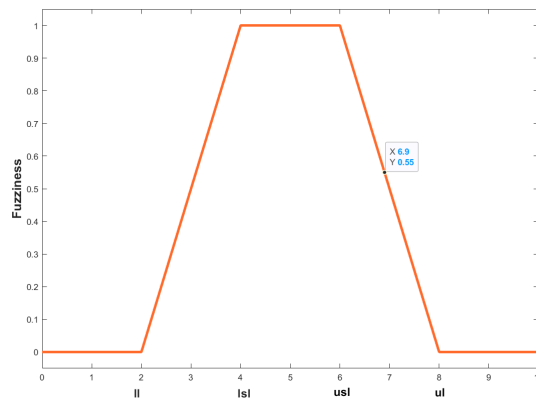


FIGURE 6.9 – Représentation graphique de la fonction trapézoïdale.

Fonction gaussienne : est déterminée par deux paramètres c et s . Où c est la valeur médiane, s est l'écart type, et $s > 0$. Elle est décrite par l'équation (6.10) et la figure 6.10 présente sa représentation graphique.

$$\mu_A(x) = e^{-\frac{(x-c)^2}{2.s^2}} \quad (6.10)$$

6.3.5.2 Définition des règles floues

Après l'étape de fuzzification, l'étape suivante est la définition des règles floues SI-ALORS. Pour les MFS, la définition des règles est considérée comme la phase la plus importante. Ces règles floues SI-ALORS sont généralement formulées de manière appropriée en utilisant des termes linguistiques au lieu de termes numériques. Elles sont principalement reconnues comme des règles floues SI-ALORS qui sont facilement conçues en exploitant des déclarations conditionnelles vagues. Les règles floues SI-ALORS se composent de

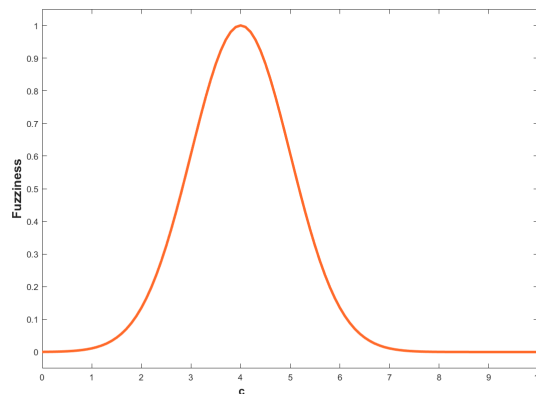


FIGURE 6.10 – Représentation graphique de la fonction gaussienne.

deux segments : un premier bloc qui représente les termes et les variables linguistiques d'entrée, et un dernier bloc ou bloc conséquent qui représente la décision de la classification. Toutes les règles SI-ALORS floues qui possèdent une vérité dans leurs premiers blocs seront libérées et participeront au groupe de conclusion. Chaque règle SI-ALORS floue est libérée à un degré qui est une fonction qui représente le degré auquel son premier bloc correspond à l'entrée. Cette identification vague constitue une base pour l'accomplissement entre les variables linguistiques probables en entrée et vise à réduire le nombre total de règles SI-ALORS floues en demande pour déterminer la relation entre l'entrée et la sortie. Ces règles SI-ALORS floues ont une capacité efficace à résoudre plusieurs problèmes réels. Parce qu'elles sont similaires à la connaissance humaine, et au raisonnement humain qui apparaît souvent sous la forme de règles floues SI-ALORS, à cette phase, nous employons la connaissance experte pour produire un ensemble de règles floues SI-ALORS. Comme expliqué ci-dessous, 25 règles floues sont définies pour le classificateur flou proposé.

- Règle1** : SI SSN est TrèsFaible **ET** SSP est TrèsFaible **ALORS** DC est Neutre
Règle2 : SI SSN est TrèsFaible **ET** SSP est Faible **ALORS** DC est Neutre
Règle3 : SI SSN est TrèsFaible **ET** SSP est Modéré **ALORS** DC est Positif
Règle4 : SI SSN est TrèsFaible **ET** SSP est Élevé **ALORS** DC est Positif
Règle5 : SI SSN est TrèsFaible **ET** SSP est TrèsÉlevé **ALORS** DC est Positif
Règle6 : SI SSN est Faible **ET** SSP est TrèsFaible **ALORS** DC est Neutre
Règle7 : SI SSN est Faible **ET** SSP est Faible **ALORS** DC est Neutre
Règle8 : SI SSN est Faible **ET** SSP est Modéré **ALORS** DC est Positif
Règle9 : SI SSN est Faible **ET** SSP est Élevé **ALORS** DC est Positif
Règle10 : SI SSN est Faible **ET** SSP est TrèsÉlevé **ALORS** DC est Positif
Règle11 : SI SSN est Modéré **ET** SSP est TrèsFaible **ALORS** DC est Négatif
Règle12 : SI SSN est Modéré **ET** SSP est Faible **ALORS** DC est Négatif
Règle13 : SI SSN est Modéré **ET** SSP est Modéré **ALORS** DC est Neutre
Règle14 : SI SSN est Modéré **ET** SSP est Élevé **ALORS** DC est Positif
Règle15 : SI SSN est Modéré **ET** SSP est TrèsÉlevé **ALORS** DC est Positif
Règle16 : SI SSN est Élevé **ET** SSP est TrèsFaible **ALORS** DC est Négatif
Règle17 : SI SSN est Élevé **ET** SSP est Faible **ALORS** DC est Négatif
Règle18 : SI SSN est Élevé **ET** SSP est Modéré **ALORS** DC est Négatif
Règle19 : SI SSN est Élevé **ET** SSP est Élevé **ALORS** DC est Neutre

- Règle20** : SI SSN est Élevé **ET** SSP est TrèsÉlevé **ALORS** DC est Neutre
Règle21 : SI SSN est TrèsÉlevé **ET** SSP est TrèsFaible **ALORS** DC est Négatif
Règle22 : SI SSN est TrèsÉlevé **ET** SSP est Faible **ALORS** DC est Négat
Règle23 : SI SSN est TrèsÉlevé **ET** SSP est Modéré **ALORS** DC est Négatif
Règle24 : SI SSN est TrèsÉlevé **ET** SSP est Élevé **ALORS** DC est Neutre
Règle25 :SI SSN est TrèsÉlevé **ET** SSP est TrèsÉlevé **ALORS** DC est Neutre

6.3.5.3 Processus d'inférence

Le processus d'inférence est exercé pour incorporer les ensembles flous précédemment décrits en prenant en compte les règles SI-ALORS floues prédéfinies et la zone floue attachée individuellement. En général, le processus du moteur d'inférence se compose de trois phases, à savoir les phases d'application, d'implication et d'agrégation. La technique d'inférence Min-Max ou technique d'application est appliquée par la procédure du moteur d'inférence pour calculer les conclusions des règles en utilisant les résultats de la fuzzification et les règles SI-ALORS floues. Le résultat de cette opération est connu comme Le résultat flou. Dans le moteur d'inférence de Mamdani, la valeur réelle de chaque règle SI-ALORS floue est calculée par la conjonction des blocs d'antécédents des règles. Avec la conjonction représentée par $t\text{-norm} = \text{minimum}$ dans le cas du connecteur logique "ET" c'est-à-dire le processus de recherche de la règle avec le bloc antécédent minimum qui est considéré comme la valeur réelle de la règle floue SI-ALORS. Cette opération s'exprime par l'équation suivante (6.11) :

$$\mu_A = \mu_i(SSP) \quad \mathbf{ET} \quad \mu_{ii}(SSN) = \mathbf{min}(\mu_i(SSP), \mu_{ii}(SSN)) \quad (6.11)$$

Où μ_A est le degré d'appartenance obtenu après la phase d'application, $\mu_i(SSP)$ est le degré d'appartenance de la variable SSP, et $\mu_{ii}(SSN)$ est le degré d'appartenance de la variable SSN.

Dans le cas du connecteur logique "OU", le $t\text{-norm} = \text{maximum}$, c'est-à-dire que le mécanisme d'inférence trouve la règle avec le bloc antécédent maximum qui est considéré comme la valeur réelle de la règle floue SI-ALORS. Cette tâche est calculée à l'aide de l'équation suivante (6.12) :

$$\mu_A = \mu_i(SSP) \quad \mathbf{OU} \quad \mu_{ii}(SSN) = \mathbf{max}(\mu_i(SSP), \mu_{ii}(SSN)) \quad (6.12)$$

Où μ_A est le degré d'appartenance obtenu après la phase d'application, $\mu_i(SSP)$ est le degré d'appartenance de la variable SSP, et $\mu_{ii}(SSN)$ est le degré d'appartenance de la variable SSN.

Dans la phase d'application, l'objectif principal est d'extraire la force de déclenchement de chaque règle activée via l'application de la conjonction des deux degrés d'appartenance des deux variables numériques SSP et SSN. À chaque règle SI-ALORS floue activée, une opération d'implication I est appliquée entre le résultat flou obtenu lors de l'étape d'application et la décision de classification de la règle. L'opérateur *minimum* est le plus

utilisé dans le processus d'implication de Mamdani. L'équation suivante (6.13) décrit cette phase d'implication :

$$\mu_I(DC_t) = \mathbf{min}(\mu_A, \mu_i(DC_t) = 1) \quad (6.13)$$

Où $\mu_I(DC_t)$ est le degré d'appartenance obtenu après le processus d'implication, μ_A est le degré d'appartenance obtenu en pratiquant l'opération d'application sur le degré d'appartenance de la variable SSP, et $\mu_i(DC_t) = 1$ est le degré d'appartenance de l'attribut de classification de décision.

Dans la phase d'implication, la force de déclenchement d'une règle SI-ALORS obtenue dans la phase précédente (application) est utilisée pour définir le degré d'appartenance de l'attribut de décision de classification 'DC' vers chaque terme linguistique 'Négatif', 'Neutre' ou 'Positif', en se basant sur le bloc conséquent de la règle floue SI-ALORS.

La phase finale du mécanisme du moteur d'inférence est l'opération d'agrégation des résultats obtenus à partir de l'étape d'implication, c'est-à-dire que toutes les règles ayant la même décision de classification seront agrégées. Il existe de nombreux opérateurs d'agrégation, comme la moyenne géométrique, la moyenne arithmétique, le maximum et le minimum. L'opérateur le plus couramment utilisé est le maximum qui est donné par l'équation suivante (6.14) :

$$\mu_{Ag}(DC_t) = \mathbf{max}(\mu_{I1}(DC_t), \mu_{I2}(DC_t), \dots, \mu_{In}(DC_t)) \quad (6.14)$$

Où $\mu_I(DC_t)$ est le degré d'appartenance obtenu après l'opération d'agrégation, $\mu_{Ii}(DC_t)$ est le degré d'appartenance de l'attribut de décision de classification DC_t .

Dans la phase d'agrégation, la valeur de l'attribut de décision de classification ' V_{DC} ' obtenue à partir de chaque règle SI-ALORS floue nécessite de calculer son degré d'appartenance à son terme linguistique identique (Positif, Neutre ou Négatif) et détermine le degré d'appartenance maximum entre eux.

6.3.5.4 Défuzzification

Après le processus du moteur d'inférence, la phase suivante est le processus de défuzzification qui est utilisé pour convertir l'ensemble flou final obtenu dans l'étape d'agrégation précédente en un nombre réel. La défuzzification est également l'approche qui produit des résultats quantifiables en logique claire qui est réalisée à partir de la définition des ensembles flous et des techniques d'appartenance avec des degrés proportionnels. Il existe plusieurs méthodes de défuzzification couramment utilisées, telles que la *Méthode du Centre de Gravité* (CG), *Approche par Bisecteur de Zone* (BZ), *Procédure du Premier du Maximum* (PM), *Technique du Dernier du Maximum* (DM), *Approche de la Moyenne du Maximum* (MM), *Procédure Moyenne Pondérée* (MP), et la *Méthode du Centre des Sommes* (CS). Dans ce travail, nous avons appliqué quatre méthodes de défuzzification qui sont CG, BZ, MP, et CS. Ces méthodes de défuzzification utilisées sont décrites ci-

dessous :

Méthode du centre de gravité convertit la valeur floue d'entrée en une valeur de sortie nette en calculant le centre de gravité de l'ensemble flou en entrée. La zone totale de la propagation d'AF est utilisée pour surveiller l'action standard, qui est divisée en un certain nombre de sous-zones. La zone et le centroïde (centre de gravité) de chaque sous-zone sont calculés, puis l'intégration de toutes ces sous-zones est calculée pour obtenir la valeur défuzzifiée pour un ensemble flou continu en entrée. Contrairement au cas de l'ensemble flou discret, la somme de toutes ces sous-zones est calculée pour déterminer la valeur défuzzifiée. Dans ce travail, les ensembles flous prennent des valeurs discrètes. Par conséquent, la valeur défuzzifiée d_v est calculée en utilisant la sommation au lieu de faire un calcul d'intégration comme le décrit l'équation suivante (6.15) :

$$d_v = \frac{\sum_{i=1}^n z_i \cdot \mu(z_i)}{\sum_{i=1}^n \mu(z_i)} \quad (6.15)$$

Où z_i indique l'élément d'instance, $\mu(z_i)$ est le degré d'appartenance de l'élément z_i , et n décrit le nombre d'éléments dans l'instance.

Approche par bisecteur de zone calcule l'abscisse de la ligne perpendiculaire qui divise la zone de la fonction d'appartenance obtenue en deux sous-zones de même surface. En d'autres termes, cette méthode sert à calculer la position sous la courbe où les sous-zones ont la même surface, dans laquelle la valeur nette de cette position correspond à la valeur défuzzifiée. C'est l'une des approches les plus largement appliquées. La valeur défuzzifiée d_v est calculée à l'aide de l'équation (6.16) :

$$\int_{\alpha}^{zBOA} \mu_i(z) \cdot d_z = \int_{zBOA}^{\beta} \mu_i(z) \cdot d_z \quad (6.16)$$

Où $\alpha = \min\{z; z \in Z\}$, $\beta = \max\{z; z \in Z\}$ et $z = zBOA$ est la ligne verticale qui divise la zone entre $z=\alpha$, $z=\beta$ $v=0$ et $v=\mu_i(z)$ en deux zones de même région, $\mu_i(z)$ est le degré d'appartenance de l'élément z , et d_z est la dérivée de l'élément z .

Procédure Moyenne Pondérée est adaptée aux ensembles flous d'entrée avec des FAs de sortie identiques et génère des résultats très proches de l'approche du centre de gravité. Cette technique utilise moins de ressources informatiques. Chaque méthode d'appartenance est pondérée par son degré d'appartenance qui a la valeur maximale. La valeur défuzzifiée d_v est déterminée comme le décrit l'équation ci-dessous (6.17) :

$$d_v = \frac{\sum \mu_i(z) \cdot z}{\sum \mu_i(z)} \quad (6.17)$$

Où \sum indique la somme algébrique, z est l'élément qui a le degré d'appartenance maximum, et $\mu_i(z)$ est le degré d'appartenance de l'élément z avec i est le terme linguistique.

Méthode de Centre des Sommes : elle s'agit d'une méthode de défuzzification largement appliquée. Dans cette approche, la zone de chevauchement est calculée deux

fois. Elle est plus rapide que les autres méthodes de défuzzification. Elle applique une somme algébrique à tous les ensembles flous de sortie. Elle est identique à l'approche de la moyenne pondérée ; néanmoins, dans cette approche, les pondérations sont les zones, au lieu des degrés d'appartenance dans l'approche de la moyenne pondérée. La valeur défuzzifiée d_v est calculée à l'aide de l'équation suivante (6.18) :

$$d_v = \frac{\sum_{ii=1}^n z_{ii} \cdot \sum_{j=1}^k \mu_{i_j}(z_{ii})}{\sum_{ii=1}^n \cdot \sum_{j=1}^k \mu_{i_j}(z_{ii})} \quad (6.18)$$

Où n est le nombre total d'ensembles flous utilisés, K est le nombre total de variables linguistiques floues, i_j est le degré d'appartenance pour le j ème ensemble flou.

Sortie de l'approche de défuzzification : après l'application de l'une des méthodes de défuzzification sur la valeur agrégée obtenue dans la phase d'agrégation. La méthode de défuzzification appliquée transforme l'entrée agrégée floue qui est obtenue par l'application de toutes les étapes du processus d'inférence floue en sortie nette. Il s'agit des différents types de variables de sortie (la décision de la classification) qui sont calculés par la valeur défuzzifiée. Par conséquent, nous utilisons des règles de défuzzification pour définir la relation entre la valeur défuzzifiée et la décision de classification.

Règles de défuzzification : Voici toutes les règles de défuzzification possibles où d_v signifie la valeur défuzzifiée tandis que d_c signifie la décision de la classification. Ces règles sont utilisées pour déterminer la sortie nette finale, qui est soit négative, neutre ou positive. **Règle1 :** SI ($0.0 \leq d_v \leq 0.35$) **ALORS** $d_c =$ Négative
Règle2 : SI ($0.35 < d_v \leq 0.65$) **ALORS** $d_c =$ Neutre
Règle3 : SI ($0.65 < d_v \leq 1.0$) **ALORS** $d_c =$ Positive.

L'algorithme suivant décrit toutes les étapes de notre classificateur flou basé sur le MFS.

6.4 Parallélisation de la méthode proposée

L'une des plus grandes lacunes de nos réseaux de neurones en profondeur (CNN+FFNN) est leur long temps d'exécution. Ce problème de temps d'exécution empêche les modèles d'apprentissage en profondeur entraînés d'obtenir rapidement des informations plus précises et d'effectuer les tâches requises. Pour remédier à ce problème de temps d'exécution, nous avons appliqué le cadre Hadoop [266] à notre approche proposée qui est un cadre utile servant à améliorer l'efficacité de la prévision et l'évolutivité de notre modèle d'apprentissage en profondeur flou proposé. La plateforme Hadoop permet de paralléliser notre CAPF entre plusieurs nœuds de calcul. Ce cadre utilise son système de fichiers distribués Hadoop pour stocker les deux jeux de données de médias sociaux utilisés dans ce travail (sentiment140, et COVID-19 Sentiments) à classifier et la décision de la classification, et son modèle de programmation MapReduce qui traite nos tâches d'apprentissage en profondeur floues de manière parallèle à l'aide de plusieurs mappeurs et réducteurs, comme l'illustre la figure 6.11.

Algorithm 8 Notre classificateur flou basé sur le système flou de Mamdani.

Input : Les deux scores sentimentaux SSN et SSP obtenus dans la phase d'apprentissage en profondeur. Chaque variable prend cinq termes linguistiques qui sont : TrèsFaible (entre 0.0 et 0.25), Faible (entre 0.0 et 0.50), Modéré (entre 0.25 et 0.75), Élevé (entre 0.50 et 1) et TrèsÉlevé (entre 0.75 et 1).

Output : Décision de la classification qui est déterminée par trois étiquettes : neutre, négative ou positive.

Phase 1 : Définition des variables linguistiques d'entrée et de sortie et définition des termes linguistiques de chaque variable linguistique.

Phase 2 : Processus de fuzzification

2.1 Utilisation d'une fonction d'appartenance Gaussienne, Triangulaire ou Trapézoïdale.

2.2 Calcule le degré d'appartenance de chaque terme linguistique en utilisant la fonction d'appartenance sélectionnée.

2.3 Transformation de tout ensemble net en ensemble flou.

Phase 3 : Génère des règles floues SI-ALORS basées sur nos connaissances d'expertise.

Phase 4 : Processus du moteur d'inférence

4.1 Phase d'application

4.2 Phase d'implication

4.3 Phase d'agrégation

Phase 5 : Processus de défuzzification

5.1 Utilisation des méthodes de défuzzification CG, BZ, MP, ou CS.

5.2 Transformation de la valeur floue agrégée obtenue lors de la phase d'agrégation en valeur nette ou réelle par l'application d'une méthode de défuzzification.

5.3 A partir de la valeur nette obtenue, découvrez la décision de classification.

Règle1 : SI ($0.0 \leq d_v \leq 0.35$) **ALORS** $d_c =$ Négative

Règle2 : SI ($0.35 < d_v \leq 0.65$) **ALORS** $d_c =$ Neutre

Règle3 : SI ($0.65 < d_v \leq 1.0$) **ALORS** $d_c =$ Positive

Où d_v est la valeur défuzzifiée, et d_c est la décision de la classification.

return d_c

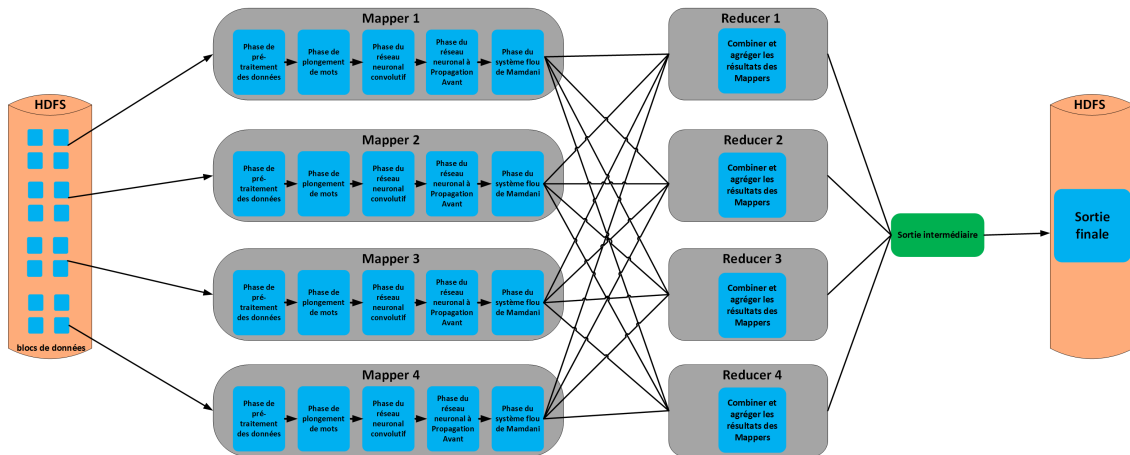


FIGURE 6.11 – Architecture parallèle de notre contribution utilisant MapReduce.

La mise en œuvre de notre CAPF sur le modèle de programmation MapReduce consiste principalement en trois étapes : la phase de Map, la phase de Combinage et la phase de Reduce, présentées brièvement comme suit.

- **Phase Map** : La phase map se compose de quatre mappers ; chaque mapper lit un ou plusieurs chunks de données depuis HDFS sous la forme d'une paire clé-valeur. Le mapper applique le pré-traitement des données à chaque chunk, puis le transforme en données numériques basées sur la phase du plongement de mots, et le fait passer par notre modèle d'apprentissage en profondeur (CNN+FFNN), et enfin applique le classificateur flou au chunk traité. Après avoir traité tous les chunks de données, les résultats obtenus à l'aide de notre CAPF sont transformés en un ensemble de paires clé-valeur intermédiaires et sont écrits sur le disque local.
- **Regroupement par clés** : Le modèle de programmation MapReduce effectue cette opération. Son objectif principal est d'agréger toutes les valeurs intermédiaires obtenues dans le Mapper avec la même clé intermédiaire dans un tableau de valeurs et de le transmettre au Reducer.
- **Phase Reduce** : Dans notre travail, la phase de Reduce se compose de quatre réducteurs ; chaque réducteur reçoit la liste des valeurs des intermédiaires de tous les mappers. Le réducteur travaille sur une clé simultanément et agrège la liste des valeurs associées à cette clé dans un ensemble plus réduit [266]. Enfin, les sorties de tous les réducteurs sont combinées et fusionnées en une seule sortie intermédiaire et la sortie résultante est écrite en tant que paire clé-valeur de sortie sur HDFS, comme le montre la figure 6.11.

Notre méthode proposée a été mise en œuvre sur les deux jeux de données massifs Sentiment140 et COVID-19 Sentiments. Dans un premier temps, nous avons utilisé le HDFS pour stocker et partager le jeu de données massif en parallèle entre tous les serveurs de calcul du cluster Hadoop. Après avoir stocké le jeu de données dans HDFS, l'étape suivante consiste à entraîner notre CAPF. Dans la deuxième étape, nous avons utilisé le modèle de programmation MapReduce pour paralléliser notre approche CAPF en tâches entre tous les nœuds de calcul du cluster Hadoop. L'entrée de chaque tour de l'algorithme MapReduce est un tweet à classifier, et le résultat est un tweet classifié avec la décision Négative, Neutre ou Positive. Le résultat de la classification de chaque tweet

sera également stocké dans HDFS. Toutes ces étapes sont décrites dans la figure 6.11, et l'algorithme 9 présente l'algorithme MapReduce appliqué dans ce travail pour classifier les tweets en utilisant notre CAPF.

Algorithm 9 Notre algorithme de programmation MapReduce

Input : Blocs de données

Output : Décision de classification

SI (un mot dans le tweet n'est pas en Anglais) ALORS

Traduire (le mot en Anglais)

$T = \text{pré-traitementDesDonnées}$ (blocs de données)

$W = \text{plongementDeMots}(T)$

$C = \text{CNN}(W)$; en utilisant l'algorithme 6

$SSN = \text{FFNN}(C)$: en appliquant l'algorithme 7

$SSP = \text{FFNN}(C)$: en appliquant l'algorithme 7

$\mu(SSN) = \text{Fuzzification}(SSN)$; en utilisant l'une des FAs décrites dans la section 6.3.5.1.

$\mu(SSP) = \text{Fuzzification}(SSP)$; en utilisant l'une des FAs décrites dans la section 6.3.5.1.

$A = \text{Application}(\mu(SSN), \mu(SSP), \text{Règles floues SI-ALORS})$; comme décrit dans 6.3.5.3

$I = \text{Implication}(A, \mu(DC_t)=1)$; comme décrit dans 6.3.5.3

$Ag = \text{Agrégation}(I)$; comme décrit dans 6.3.5.3

$d_v = \text{Défuzzification}(Ag)$; en utilisant l'une des méthodes de défuzzification décrites dans 6.3.5.4

SI ($0.0 \leq d_v \leq 0.35$) ALORS $d_c = \text{Négative}$

SI ($0.35 < d_v \leq 0.65$) ALORS $d_c = \text{Neutre}$

SI ($0.65 < d_v \leq 1.0$) ALORS $d_c = \text{Positive}$

Où d_v est la valeur défuzzifiée, et d_c est la décision de la classification.

return d_c

6.5 Exemple de l'application

Cette section illustre notre CAPF à l'aide d'un exemple. En effet, nous allons expliquer les étapes suivies pour classifier chaque tweet S , selon trois étiquettes de classe (Négatif, Neutre, ou Positif). La première étape de notre algorithme consiste à vérifier que tous les mots de chaque tweet S sont écrits en Anglais. Si ce n'est pas le cas, nous utilisons la fonction de traduction pour traduire ces mots depuis d'autres langues vers la langue Anglaise. Par exemple, nous avons $S = \text{"The deep neural networks are very efficiencccccyy to process data ; but they are inefficiency with the ingrained ambiguity in NL que demande d'autre solutions."}$; la partie "que demande d'autres solutions" du tweet S est traduite en "that needs more solutions". Ensuite, après l'application de techniques de prétraitement, nous avons obtenu le tweet S comme un ensemble de jetons (T) comme décrit dans le tableau 6.4.

TABLE 6.2 – Ensemble de jetons qui représentent le tweet à classifier.

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12
deep	neural	networks	very	efficiency	process	data	inefficiency	ambiguity	natural language	needs	solutions

Après avoir obtenu l'ensemble des jetons, nous avons appliqué l'approche FastText de plongement de mots pour transformer les données textuelles en données numériques, comme le montre la figure 6.12.

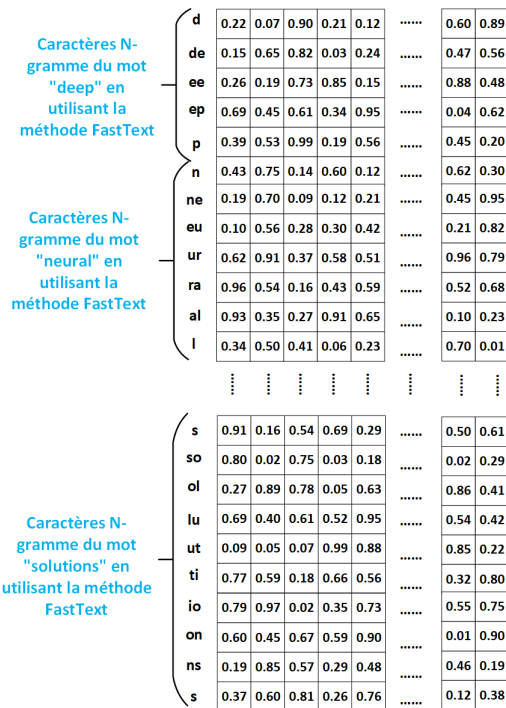


FIGURE 6.12 – Matrice de plongement de mots du tweet S.

Comme l'illustre la figure 6.12, la méthode de plongement de mots FastText applique les caractères avec le n-gramme=2 à chaque mot du tweet S . Par exemple, les caractères avec le n-gramme=2 pour le mot "deep" sera (d, de, ee, ep, p) . Ensuite, FastText utilise soit CBOW soit Skip-Gram pour calculer le plongement de mots pour chaque mot n-gramme et génère la matrice de plongement de mots du tweet S . Une fois la matrice de plongement obtenue, CNN extrait automatiquement les caractéristiques essentielles à partir de cette matrice. Donc, la figure 6.13 décrit les étapes de CNN.

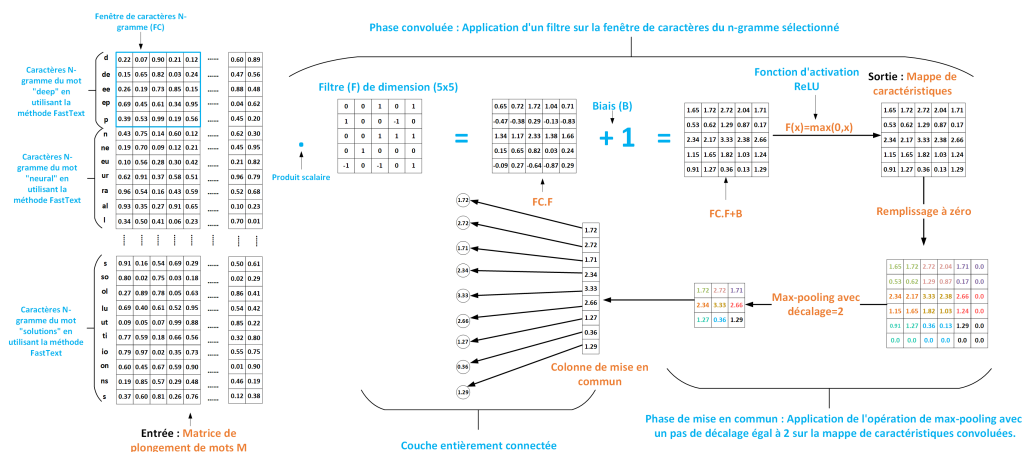


FIGURE 6.13 – Application des étapes du CNN.

Comme le montre la figure 6.13, la première étape du CNN est l'opération de convolution qui sert à appliquer plusieurs filtres à chaque fenêtre de caractères n-gramme CW . Pour clarifier l'opération de convolution de manière précise, la figure 6.13 illustre uniquement l'application d'un filtre F à une fenêtre de caractères n-gramme CW , et nous obtenons la matrice $F.CW$. Ensuite, nous ajoutons le biais b égal à 1, et nous obtenons la matrice $F.CW+1$. Enfin, nous appliquons la fonction d'activation ReLU à la matrice $F.CW+1$, et nous obtenons la première mappe de caractéristiques. Après l'opération de convolution, l'étape suivante est l'opération de pooling, dans laquelle nous réduisons la dimension de la mappe de caractéristiques. Dans cet exemple, nous utilisons le max-pooling avec un pas de décalage égal à 2. Pour cela, il faut padder à zéro la mappe des caractéristiques pour obtenir ses dimensions divisibles par 2. Ensuite, nous appliquons le max-pooling à la matrice paddée, et nous obtenons la colonne de pooling. Enfin, nous passons la colonne de pooling obtenue à la couche entièrement connectée. Après avoir extrait les caractéristiques essentielles en utilisant CNN, la phase suivante est l'application de FFNN pour calculer les valeurs SSN et SSP. Par conséquent, la figure 6.14 présente les étapes du FFNN.

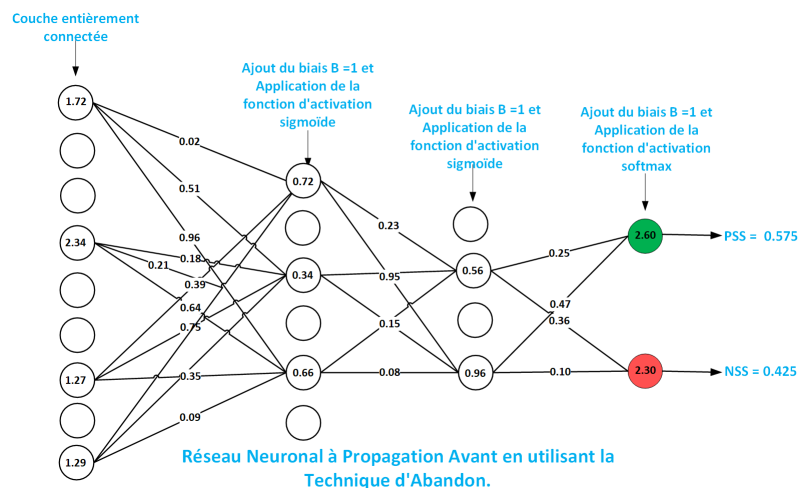


FIGURE 6.14 – Application des étapes du FFNN.

Comme présenté dans la figure 6.14, la première couche de notre version simple du FFNN est la couche entièrement connectée, suivie de deux couches cachées et de la couche de sortie. Nous avons utilisé au niveau des deux couches cachées la fonction d'activation sigmoïde, et au niveau de chaque nœud de neurone de la couche de sortie, nous avons appliqué la méthode d'activation softmax. L'entrée de ce réseau est une couche entièrement connectée obtenue dans la phase du CNN, et les sorties sont à la fois SSN et SSP. La valeur de chaque neurone caché H_{nv} est calculée en multipliant les poids entièrement connectés à ce neurone caché par sa valeur et en ajoutant le biais b puis en calculant la fonction d'activation sigmoïde. L'exemple suivant illustre le calcul de la valeur du premier neurone caché de la première couche cachée :

$$\begin{aligned}
 H_{nv} &= f((w_{i_1} * v_1 + w_{i_2} * v_2 + w_{i_3} * v_3 + w_{i_4} * v_4) + B) = f((1.72 * 0.02 + 2.34 * 0.15 + \\
 &1.27 * 0.39 + 1.29 * 0.75) + b) = f(0.0344 + 1 + 0.351 + 1 + 0.4953 + 1 + 0.9675 + 1) = \\
 &f(5.85) = \frac{1}{1+e^{5.85}} = \frac{1}{1+347.23} = \frac{1}{348.23} = \mathbf{0.00287}
 \end{aligned}$$

La même manière est suivie pour calculer la valeur de chaque nœud de neurone dans la couche de sortie. La seule différence est qu'au lieu d'utiliser la fonction d'activation sigmoïde comme nous l'avons fait pour calculer la valeur de chaque neurone caché, nous employons la fonction d'activation softmax pour calculer la valeur de chaque nœud de neurone dans la couche de sortie. L'exemple suivant explique comment nous calculons la valeur des deux neurones de la couche de sortie :

$$ON_{v_1} = (wi_1*v_1 + wi_2*v_2) + B = (0.56*0.25 + 0.96*0.47) + B = 0.14 + 1 + 0.4512 + 1 = 2.60$$

$$ON_{v_2} = (wi_3*v_1 + wi_4*v_2) + B = (0.56*0.36 + 0.96*0.10) + B = 0.2016 + 1 + 0.096 + 1 = 2.30$$

Par conséquent, les valeurs SSP et SSN sont calculées en appliquant la fonction d'activation softmax sur les deux valeurs ON_{v_1} , et ON_{v_2} comme suit :

$$SSP = \frac{e^{2.60}}{e^{2.60} + e^{2.30}} = \frac{13.46}{13.46 + 9.97} = \frac{13.46}{23.43} = \mathbf{0.575}$$

$$SSN = \frac{e^{2.30}}{e^{2.60} + e^{2.30}} = \frac{9.97}{13.46 + 9.97} = \frac{9.97}{23.43} = \mathbf{0.425}$$

Après avoir calculé les deux valeurs **SSP=0.575** et **SSN=0.425** en utilisant notre CAPF proposé, la phase suivante est l'application de la MFS. Par conséquent, la première étape de la MFS est l'application du processus de fuzzification aux valeurs nettes du SSN et du SSP, c'est-à-dire que nous utilisons la fonction d'appartenance triangulaire (ou trapézoïdale, ou gaussienne) pour calculer le degré d'appartenance du SSN et du SSP aux ensembles flous TrèsFaible, Faible, Modéré, Élevé et TrèsÉlevé. Dans cet exemple, nous avons utilisé l'AF triangulaire représenté par l'équation (6.6) et illustré par la figure 6.8. Le processus de calcul est basé de la figure 6.15 comme suit :

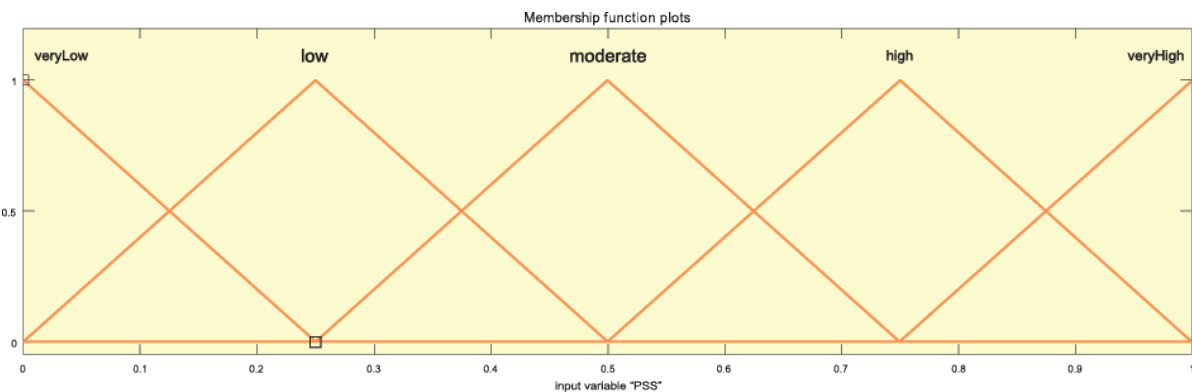


FIGURE 6.15 – Degré d'appartenance de chaque terme linguistique.

Pour le terme linguistique TrèsFaible, et les paramètres scalaires optimaux sont $l=0$; $v=0,125$; et $ul=0,25$; ensuite, nous avons utilisé ces paramètres pour calculer les degrés d'appartenance des deux variables linguistiques SSN et SSP à l'ensemble flou TrèsFaible. Les résultats sont les suivants :

— Nous avons $SSP=0.575 \geq ul=0.25$ Alors ; $\mu_{\text{TrèsFaible}}(SSP) = 0$

— Nous avons $SSN=0.425 \geq ul=0.25$ Alors ; $\mu_{\text{TrèsFaible}}(SSN) = 0$

Par conséquent, les valeurs des paramètres de chaque fonction d'appartenance utilisée ont été déterminées expérimentalement, et nous prenons les valeurs optimales de ces paramètres qui fournissent de meilleures performances de classification.

Pour le terme linguistique Faible, les paramètres scalaires optimaux sont $ll=0$; $v=0,25$; et $ul=0,5$; ensuite, nous avons utilisé ces paramètres pour calculer les degrés d'appartenance des deux variables linguistiques SSN et SSP à l'ensemble flou Faible. Les résultats sont les suivants :

- Nous avons $SSP=0.575 \geq ul=0.5$ Alors ; $\mu_{\text{Faible}}(\text{SSP}) = \mathbf{0}$
- Nous avons $v=0.25 \leq SSN=0.425 \leq ul=0.5$ Alors ; $\mu_{\text{Faible}}(\text{SSN}) = \frac{ul-SSN}{ul-v} = \frac{0.5-0.425}{0.5-0.25} = \mathbf{0.3}$

Pour le terme linguistique Modéré, les paramètres scalaires optimaux sont $ll=0.25$; $v=0.5$; et $ul=0.75$; ensuite, nous avons utilisé ces paramètres pour calculer les degrés d'appartenance des deux variables linguistiques SSN et SSP à l'ensemble flou Modéré. Les résultats sont les suivants :

- Nous avons $v=0.5 \leq SSP=0.575 \leq ul=0.75$ Alors ;
 $\mu_{\text{Modéré}}(\text{SSP}) = \frac{ul-SSP}{ul-v} = \frac{0.75-0.575}{0.75-0.5} = \mathbf{0.7}$
- Nous avons $ll=0.25 \leq SSN=0.425 \leq v=0.5$ Alors ;
 $\mu_{\text{Modéré}}(\text{SSN}) = \frac{SSN-ll}{v-ll} = \frac{0.425-0.25}{0.5-0.25} = \mathbf{0.7}$

Pour le terme linguistique Élevé, les paramètres scalaires optimaux sont $ll=0,5$; $v=0,75$; et $ul=1$; ensuite, nous avons utilisé ces paramètres pour calculer les degrés d'appartenance des deux variables linguistiques SSN et SSP à l'ensemble flou Élevé. Les résultats sont les suivants :

- Nous avons $ll=0.5 \leq SSP=0.575 \leq v=0.75$ Alors ;
 $\mu_{\text{Élevé}}(\text{SSP}) = \frac{SSP-ll}{v-ll} = \frac{0.575-0.50}{0.75-0.50} = \mathbf{0.30}$
- Nous avons $SSN=0.425 \leq ll=0.5$ Alors ;
 $\mu_{\text{Élevé}}(\text{SSN}) \mathbf{0}$

Pour le terme linguistique TrèsÉlevé, les paramètres scalaires optimaux sont $ll=0,75$; $v=0,875$; et $ul=1$; ensuite, nous avons utilisé ces paramètres pour calculer les degrés d'appartenance des deux variables linguistiques SSN et SSP à l'ensemble flou TrèsÉlevé. Les résultats sont les suivants :

- Nous avons $SSP=0.575 \leq ll=0.75$ Alors ; $\mu_{\text{TrèsÉlevé}}(\text{SSP}) = \mathbf{0}$
- Nous avons $SSN=0.425 \leq ll=0.75$ Alors ; $\mu_{\text{TrèsÉlevé}}(\text{SSN}) = \mathbf{0}$

L'étape suivante après le processus de fuzzification est le processus d'application des 25 règles floues SI-ALORS créées. L'objectif principal de cette opération est de découvrir

la force de déclenchement de chaque règle SI-ALORS floue activée par l'application de la conjonction des deux variables numériques calculées SSP et SSN dans la phase de fuzzification. Les étapes de calcul de ce processus sont présentées comme suit :

- Règle1 : **SI** (SSN est TrèsFaible) = 0 **ET** (SSP est TrèsFaible) = 0 **ALORS** (DC is neutre) = $\text{Min}(0,0) = 0$
- Règle2 : **SI** (SSN is TrèsFaible) = 0 **ET** (SSP est Faible) = 0 **ALORS** (DC est neutre) = $\text{min}(0,0) = 0$
- Règle3 : **SI** (SSN is TrèsFaible) = 0 **ET** (SSP est Modéré) = 0.7 **ALORS** (DC is positif) = $\text{min}(0,0.7) = 0$
- Règle4 : **SI** (SSN est TrèsFaible) = 0 **ET** (SSP est Élevé) = 0.30 **ALORS** (DC est positif) = $\text{min}(0,0.30) = 0$
- Règle5 : **SI** (SSN est TrèsFaible) = 0 **ET** (SSP est TrèsÉlevé) = 0 **ALORS** (DC est positif) = $\text{min}(0,0) = 0$
- Règle6 : **SI** (SSN est Faible) = 0.3 **ET** (SSP est TrèsFaible) = 0 **ALORS** (DC est neutre) = $\text{min}(0.3,0) = 0$
- Règle7 : **SI** (SSN est Faible) = 0.3 **ET** (SSP est Faible) = 0 **ALORS** (DC est neutre) = $\text{min}(0.3,0) = 0$
- Règle8 : **SI** (SSN est Faible) = 0.3 **ET** (SSP est Modéré) = 0.7 **ALORS** (DC est positif) = $\text{min}(0.3,0.7) = 0.3$
- Règle9 : **SI** (SSN est Faible) = 0.3 **ET** (SSP est Élevé) = 0.3 **ALORS** (DC est positif) = $\text{min}(0.3,0.3) = 0.3$
- Règle10 : **SI** (SSN est Faible) = 0.3 **ET** (SSP est TrèsÉlevé) = 0 **ALORS** (DC est positif) = $\text{min}(0.3,0) = 0$
- Règle11 : **SI** (SSN est Modéré) = 0.7 **ET** (SSP est TrèsFaible) = 0 **ALORS** (DC est Négatif) = $\text{min}(0.7,0) = 0$
- Règle12 : **SI** (SSN est Modéré) = 0.7 **ET** (SSP est Faible) = 0 **ALORS** (DC est Négatif) = $\text{min}(0.7,0) = 0$
- Règle13 : **SI** (SSN est Modéré) = 0.7 **ET** (SSP est Modéré) = 0.7 **ALORS** (DC est neutre) = $\text{min}(0.7,0.7) = 0.7$
- Règle14 : **SI** (SSN est Modéré) = 0.7 **ET** (SSP est Élevé) = 0.375 **ALORS** (DC est positif) = $\text{min}(0.7,0.3) = 0.3$
- Règle15 : **SI** (SSN est Modéré) = 0.7 **ET** (SSP est TrèsÉlevé) = 0 **ALORS** (DC est

$$\text{positif})=\min(0.7,0)=0$$

- Règle16 : **SI** (SSN est Élevé)= 0 **ET** (SSP est TrèsFaible)= 0 **ALORS** (DC est Négatif)= $\min(0,0)=0$
- Règle17 : **SI** (SSN est Élevé)= 0 **ET** (SSP est Faible) = 0 **ALORS** (DC est Négatif)= $\min(0,0)=0$
- Règle18 : **SI** (SSN est Élevé)= 0 **ET** (SSP est Modéré) = 0.7 **ALORS** (DC est Négatif)= $\min(0,0.7)=0$
- Règle19 : **SI** (SSN est Élevé)= 0 **ET** (SSP est Élevé)=0.3 **ALORS** (DC est neutre)= $\min(0,0.3)=0$
- Règle20 : **SI** (SSN est Élevé)= 0 **ET** (SSP est TrèsÉlevé)=0 **ALORS** (DC est neutre)= $\min(0,0)=0$
- Règle21 : **SI** (SSN est TrèsÉlevé)=0 **ET** (SSP est TrèsFaible)=0 **ALORS** (DC est Négatif)= $\min(0,0)=0$
- Règle22 : **SI** (SSN est TrèsÉlevé)=0 **ET** (SSP est Faible)=0 **ALORS** (DC est Négatif)= $\min(0,0)=0$
- Règle23 : **SI** (SSN est TrèsÉlevé)=0 **ET** (SSP est Modéré)=0.7 **ALORS** (DC est Négatif)= $\min(0,0.7)=0$
- Règle24 : **SI** (SSN est TrèsÉlevé)=0 **ET** (SSP est Élevé)=0.3 **ALORS** (DC est neutre)= $\min(0,0.30)=0$
- Règle25 : **SI** (SSN est TrèsÉlevé)=0 **ET** (SSP est TrèsÉlevé)=0 **ALORS** (DC est neutre)= $\min(0,0)=0$

L'étape suivante est le processus d'implication. L'objectif principal de la phase d'implication, tel que nous l'avons présenté précédemment, est le calcul du degré d'appartenance de l'attribut de décision de classification "DC" à chaque terme linguistique "Négatif", "Neutre" ou "Positif", en fonction de la force de déclenchement d'une règle SI-ALORS obtenue dans la phase d'application précédente, et du bloc conséquent de la règle floue SI-ALORS. les étapes de calcul sont présentées ci-dessous :

$$\text{— Règle1 : } \mu_1(\text{Neutre})=\min(0,1)=0$$

$$\text{— Règle2 : } \mu_2(\text{Neutre})=\min(0,1)=0$$

$$\text{— Règle3 : } \mu_3(\text{Positif})=\min(0,1)=0$$

$$\text{— Règle4 : } \mu_4(\text{Positif})=\min(0,1)=0$$

- Règle5 : $\mu_5(\text{Positif}) = \min(0,1) = 0$
- Règle6 : $\mu_6(\text{Neutre}) = \min(0,1) = 0$
- Règle7 : $\mu_7(\text{Neutre}) = \min(0,1) = 0$
- Règle8 : $\mu_8(\text{Positif}) = \min(0.3,1) = 0.3$
- Règle9 : $\mu_9(\text{Positif}) = \min(0.3,1) = 0.3$
- Règle10 : $\mu_{10}(\text{Positif}) = \min(0,1) = 0$
- Règle11 : $\mu_{11}(\text{Négatif}) = \min(0,1) = 0$
- Règle12 : $\mu_{12}(\text{Négatif}) = \min(0,1) = 0$
- Règle13 : $\mu_{13}(\text{Neutre}) = \min(0.625,1) = 0.7$
- Règle14 : $\mu_{14}(\text{Positif}) = \min(0.375,1) = 0.3$
- Règle15 : $\mu_{15}(\text{Positif}) = \min(0.625,1) = 0.625$
- Règle16 : $\mu_{16}(\text{Négatif}) = \min(0,1) = 0$
- Règle17 : $\mu_{17}(\text{Négatif}) = \min(0,1) = 0$
- Règle18 : $\mu_{18}(\text{Négatif}) = \min(0,1) = 0$
- Règle19 : $\mu_{19}(\text{Neutre}) = \min(0,1) = 0$
- Règle20 : $\mu_{20}(\text{Neutre}) = \min(0,1) = 0$
- Règle21 : $\mu_{21}(\text{Négatif}) = \min(0,1) = 0$
- Règle22 : $\mu_{22}(\text{Négatif}) = \min(0,1) = 0$
- Règle23 : $\mu_{23}(\text{Négatif}) = \min(0,1) = 0$
- Règle24 : $\mu_{24}(\text{Neutre}) = \min(0,1) = 0$
- Règle25 : $\mu_{25}(\text{Neutre}) = \min(0,1) = 0$

Après le processus d'implication, le processus suivant est le processus d'agrégation, qui vise à agréger chaque étiquette de classe (positive, neutre ou négative) entre elles et à trouver le degré d'appartenance maximal parmi les valeurs agrégées. Ainsi, les étapes de calcul sont effectuées comme suit :

- $\mu(\mathbf{Positif}) = \mu_3(\text{Positif}) \vee \mu_4(\text{Positif}) \vee \mu_5(\text{positif}) \vee \mu_8(\text{Positif}) \vee \mu_9(\text{Positif}) \vee \mu_{10}(\text{Positif}) \vee \mu_{14}(\text{Positif}) \vee \mu_{15}(\text{Positif}) = \mathbf{\max(0,0,0,0,3,0,3,0,3,0)} = \mathbf{0.3}$
- $\mu(\mathbf{Négatif}) = \mu_{11}(\text{Négatif}) \vee \mu_{12}(\text{Négatif}) \vee \mu_{16}(\text{Négatif}) \vee \mu_{17}(\text{Négatif}) \vee \mu_{18}(\text{Négatif}) \vee \mu_{21}(\text{Négatif}) \vee \mu_{22}(\text{Négatif}) \vee \mu_{23}(\text{Négatif}) = \mathbf{\max(0,0,0,0,0,0,0,0)} = \mathbf{0}$
- $\mu(\mathbf{Neutre}) = \mu_1(\text{Neutre}) \vee \mu_2(\text{Neutre}) \vee \mu_6(\text{Neutre}) \vee \mu_7(\text{Neutre}) \vee \mu_{13}(\text{Neutre}) \vee \mu_{19}(\text{Neutre}) \vee \mu_{20}(\text{Neutre}) \vee \mu_{24}(\text{Neutre}) \vee \mu_{25}(\text{Neutre}) = \mathbf{\max(0,0,0,0,0,7,0,0,0,0)} = \mathbf{0.7}$

Enfin, et après le processus d'agrégation, nous utilisons la méthode de défuzzification du centroïde de la zone pour défuzzifier les sorties agrégées floues afin d'obtenir un résultat net qui indique l'étiquette de classe. Ce processus est réalisé en suivant les étapes décrites ci-dessous :

Étape 1 : D'après la figure 6.16, nous calculons le centroïde de l'étiquette négative qui sera égal à $(0,0+0,35)/2 = \mathbf{0,175}$, le centroïde de l'étiquette neutre qui sera égal à $(0,35+0,65)/2 = \mathbf{0,5}$, et le centroïde de l'étiquette positive qui sera égal à $(0,65+1)/2 = \mathbf{0,825}$

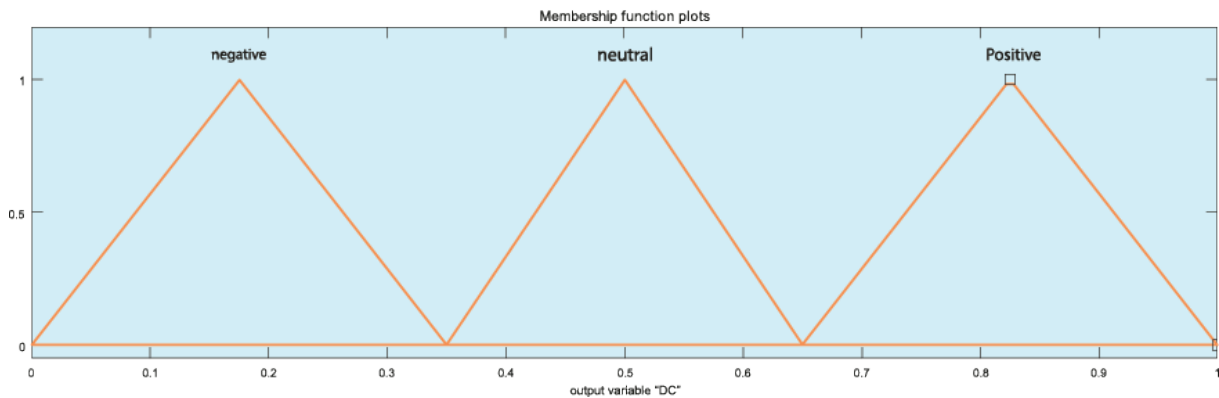


FIGURE 6.16 – Degré d'appartenance de chaque étiquette de classe.

Étape 2 : Selon la méthode de défuzzification du centroïde de la zone introduite dans l'équation (6.15). La valeur défuzzifiée d_v est calculée en utilisant le centroïde calculé dans la première étape et le degré d'appartenance de chaque étiquette comme suit :

$$d_v = \frac{0 \cdot 0,175 + 0,7 \cdot 0,5 + 0,825 \cdot 0,3}{0 + 0,7 + 0,3} = \mathbf{0.60}.$$

Étape 3 : D'après l'étape 2, la valeur défuzzifiée $d_v = 0,60$ est comprise entre 0,35 et 0,65. Par conséquent, la décision de classification de la phrase **S** est **Neutre**.

D'après la figure 6.17 nous remarquons que la valeur défuzzifiée est égale à 0.60 pour SSP=0.575 et SSN=0.425. Cette valeur défuzzifiée est comprise entre 0.35 et 0.65. Par conséquent, la décision de classification de la phrase *S* est *neutre*.

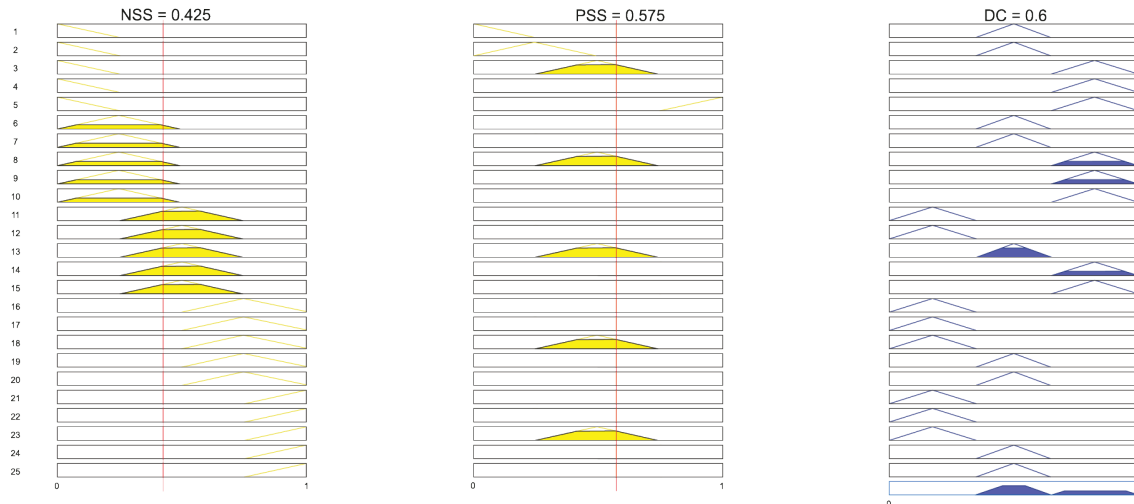


FIGURE 6.17 – Valeur défuzzifiée calculée à l’aide de l’examen des règles dans Matlab.

6.6 Expériences et résultats

Cette section décrit les résultats expérimentaux de notre classificateur d’apprentissage en profondeur parallèle et flou (CNN+FFNN+MFS+Hadoop). Ces résultats expérimentaux sont fournis par l’application de notre classificateur d’apprentissage en profondeur parallèle et flou avec d’autres méthodes de la littérature sur les deux jeux de données utilisés, comme présenté dans le chapitre 3. En général, dans cette étude, nous divisons le jeu de données utilisé en un jeu de données d’apprentissage qui représente 90 % du jeu de données global, et un jeu de données de test représentant seulement 10 % du jeu de données global. Ensuite, nous sauvegardons les deux jeux de données obtenus (apprentissage et test) dans HDFS. Une fois le stockage terminé, nous appliquons plusieurs techniques de pré-traitement des données sur les jeux de données d’apprentissage et de test pour réduire et supprimer les données bruitées. Ensuite, nous mettons en œuvre l’approche du plongement de mots la plus efficace pour transformer les données textuelles en données numériques. En outre, nous appliquons notre CAPF proposé sur l’ensemble de données de test, et nous stockons les données classifiées dans le HDFS. De plus, nous accomplissons la classification des sentiments basée sur notre classificateur proposé de manière parallèle en mettant en œuvre le cadre Hadoop avec son HDFS et son paradigme de programmation MapReduce. Dans ce travail, le cluster Hadoop simple est composé de cinq nœuds de calcul : un nœud de calcul maître et quatre nœuds de calcul slaves. Pour évaluer notre classifieur par rapport aux autres méthodes de la littérature, nous avons calculé dix métriques d’évaluation présentées dans la section 3.9 du chapitre 3. Ces dix mesures comprennent TC , TR , TE , TPV , TNF , TNV , TPF , PR , SK , et SF . Nous évaluons également notre approche en termes de complexité, de convergence et de stabilité.

Dans ce travail, nous avons réalisé les expériences en cinq étapes. Tout d’abord, nous avons conservé les mêmes paramètres de notre CAPF. Ensuite, nous avons changé les approches adoptées pour le plongement de mots (Word2Vec, GloVe, FastText) afin de déterminer l’approche la plus efficace pour transformer les données textuelles en données numériques. Après avoir obtenu la procédure du plongement de mots la plus performante en termes de taux de classification, nous l’avons utilisée dans la suite de ce travail. Deuxième-

mement, nous avons appliqué la technique du plongement de mots efficace obtenue à l'expérience précédente, et nous avons conservé les mêmes paramètres de notre modèle d'apprentissage en profondeur proposé (CNN+FFNN). Ensuite, nous avons changé l'approche de fuzzification adoptée et les méthodes de défuzzification utilisées. L'objectif principal de cette expérience est de déterminer l'approche de fuzzification la plus efficace et la méthode de défuzzification la plus performante parmi toutes les méthodes de fuzzification et de défuzzification utilisées dans ce travail en termes de taux de classification. Troisièmement, après avoir déterminé la technique du plongement de mots la plus efficace, la meilleure méthode de fuzzification et l'approche de défuzzification la plus performante. Nous avons formé plusieurs classificateurs d'apprentissage en profondeur flous (CAPFs) en utilisant différents paramètres pour chaque couche. L'objectif principal de cette expérience est de définir l'ensemble des paramètres qui renforceront la performance et la précision de notre CAPF. Quatrièmement, nous avons parallélisé notre CAPF proposé en utilisant le cadre Hadoop avec son HDFS et son paradigme de programmation MapReduce. Enfin, nous avons comparé le modèle le plus efficace de notre proposition CAPF avec des approches similaires de la littérature et démontré la performance de notre proposition CAPF.

6.6.1 Première expérience

Dans cette expérience, nous avons évalué l'efficacité de Word2vec, GloVe, Fast-text en termes de TR et de TC. Nous avons combiné ces méthodes du plongement de mots avec notre proposition CAPF pour prouver et vérifier leurs performances. Alors, nous avons gardé les mêmes paramètres de notre CAPF proposé comme décrit dans le tableau 6.3, et nous avons changé la méthode du plongement de mots adoptée (Word2Vec, GloVe, et FastText), puis nous avons appliqué chaque combinaison sur les deux jeux de données massifs utilisés comme présenté dans le tableau 6.4.

Le tableau 6.3 présente les paramètres des deux modèles d'apprentissage en profondeur CNN et FFNN utilisés. En outre, pour le classificateur MFS, nous avons utilisé la fonction d'appartenance triangulaire pour effectuer la fuzzification et le centroïde de la zone pour effectuer la défuzzification.

Par ailleurs, le cadre Hadoop est mis en œuvre dans notre travail permet de paralléliser les tâches d'apprentissage du plongement de mots entre cinq machines : un nœud maître et quatre nœuds esclaves. Le cadre Hadoop utilise son HDFS pour stocker le jeu de données à plonger et l'ensemble des vecteurs de représentation (le résultat obtenu en appliquant la méthode du plongement de mots), et le paradigme de programmation MapReduce pour analyser et traiter les tâches de notre CAPF. L'objectif principal du Hadoop est de paralléliser le processus de plongement sur plusieurs machines afin de réduire les TR et TE.

Le tableau 6.4 montre les TR et TE de chaque méthode du plongement de mots sans application du cadre Hadoop.

D'après les résultats de nos expériences, comme le révèle le tableau 6.4, nous avons remarqué que la méthode GolVe prend respectivement moins de temps d'exécution que les autres techniques qui est égal à 7.32s, et 4.52s dans le cas des jeux de données sentiment140

TABLE 6.3 – Paramètres de notre CAPF que nous avons utilisés pour évaluer les approches du plongement de mots.

Paramètres de configuration du CNN		Paramètres de configuration du FFNN	
Paramètres	valeurs	Paramètres	Valeurs
Dimension du vocabulaire	40,000	N. de couches entièrement connectées	1
Matrice du plongement d'entrée	500x500	N. de couches cachées	2
N. de couche de convolution	1	Couche de sortie	1
N. de couche de pooling	1	N. de noeuds de neurones dans la couche d'entrée	250
Fonction de la couche de pooling	Max-pooling	N. de noeuds de neurones dans la couche de sortie	2
Nombre de filtres	15	N. de noeuds de neurones dans la première couche cachée	125
Dimension du filtre	4,7	N. de noeuds de neurones dans la deuxième couche cachée	63
Padding	0	Fonction d'activation	sigmoid, softmax
Fonction d'activation	ReLU	Abandon	0.5
Régulariseur	L2	Régulariseur	L2

TABLE 6.4 – TR et TE de chaque méthode du plongement de mots sans l'application du cadre Hadoop.

Evaluation criteria	TR(%)			TE(s)		
	Word2vec	GolVe	FastText	Word2vec	GolVe	FastText
Plongement de mots sentiment140	20.59	35.68	11.02	13.43	7.32	20.15
COVID-19 Sentiments	15.54	21.39	8.66	08.30	4.52	12.46

et COVID-19 Sentiments. Mais elle a respectivement un taux d'erreur plus élevé, qui est égal à 35.68 % et 21.39 % dans le cas des jeux de données sentiment140 et COVID-19 Sentiments. FastText a respectivement un taux d'erreur inférieur à celui des techniques Word2vec et GloVe, qui est égal à 11.02 % et 8.66 % dans le cas des ensembles de données sentiment140 et COVID-19 Sentiments. Mais elle a un temps d'exécution plus élevé qui est égal à 20,15 s, et 12,46 s dans le cas des jeux de données sentiment140 et COVID-19 Sentiments, respectivement. En résumé, la méthode du plongement FastText est la plus efficace en termes de taux d'apprentissage.

Le tableau 6.5 présente les résultats obtenus après l'incorporation du cadre Hadoop avec les méthodes du plongement de mots.

D'après les tableaux 6.4 et 6.5, nous déduisons que le cadre Hadoop peut diminuer le temps d'exécution et augmenter le taux d'apprentissage. Par exemple, il réduit respecti-

TABLE 6.5 – TR et TE de chaque méthode du plongement de mots avec l'application du Hadoop.

Evaluation criteria	TR(%)			TE(s)		
	Word2vec	GolVe	FastText	Word2vec	GolVe	FastText
Plongement de mots sentiment140	15.16	27.44	8.28	1.68	0.46	3.03
COVID-19 Sentiments	10.68	14.08	5.51	0.66	0.096	1.49

vement le temps d'exécution de la méthode FastText de (20.15 s, 12.46 s) à (3.03 s, 1.49 s) pour les deux jeux de données sentiment140 et COVID-19 Sentiments. Presque, nous pouvons dire que FastText est la méthode la plus efficace. Par conséquent, nous utiliserons uniquement la méthode du plongement de mots FastText dans la suite de ce travail.

6.6.2 Deuxième expérience

Le but majeur de cette deuxième expérience est de trouver l'approche de fuzzification la plus efficace et la meilleure méthode de défuzzification. Dans cette expérience, la méthode de plongement de mots adoptée est le FastText, et la configuration des paramètres des deux modèles d'apprentissage en profondeur CNN et FFNN utilisés est la même que celle présentée dans le tableau 6.3. La partie rénovée de notre proposition concerne le classificateur flou appliqué. Dans notre classificateur flou, nous utilisons les trois fonctions d'appartenance pour le processus de fuzzification, à savoir les fonctions d'appartenance triangulaire, trapézoïdale et gaussienne, et les quatre approches de défuzzification, à savoir la méthode du centre de gravité, l'approche par bisecteur de zone, la procédure de la moyenne pondérée et la méthode du centre des sommes. Ainsi, nous combinons chaque fonction de fuzzification avec différentes approches de défuzzification, comme montré dans le tableau 6.6. Cette agrégation génère 12 combinaisons, à savoir FA Triangulaire/Centre de gravité, FA Triangulaire/Bisecteur de zone, FA Triangulaire/Moyenne pondérée, FA Triangulaire/Centre des sommes, FA Trapézoïdale/Centre de gravité, FA Trapézoïdale/Bisecteur de zone, FA Trapézoïdale/Moyenne pondérée, FA Trapézoïdale/Centre des sommes, FA Gaussienne/Centre de gravité, FA Gaussienne/Bisecteur de zone, FA Gaussienne/Moyenne pondérée, and FA Gaussienne/Centre des sommes. Le tableau 6.6 présente les TR, TC et TE obtenus pour chaque méthode de fuzzification/défuzzification après l'application de notre proposition sur le jeu de données Sentiment140 sans utilisation du cadre Hadoop.

Dans cette expérience, nous avons gardé les mêmes paramètres pour le modèle d'apprentissage en profondeur, comme détaillé dans le tableau 6.3, et nous avons appliqué le plongement de mots FastText. Par conséquent, à chaque fois nous avons changé les méthodes de fuzzification/défuzzification afin de découvrir les approches de fuzzification/défuzzification les plus efficaces. D'après le tableau 6.6, nous constatons que la meilleure méthode de fuzzification est la FA gaussienne et que la méthode de défuzzification la plus efficace est l'approche du centre des sommes par rapport aux autres approches. En outre, nous avons utilisé la FA gaussienne pour transformer les valeurs de SSN et SSP en ensembles flous "TrèsFaible", "Faible", "Modéré", "Élevé" et "TrèsÉlevé". Après avoir obtenu la sortie du processus d'inférence, nous avons défuzzifié cette sortie

TABLE 6.6 – TR, TC et TE de l'agrégation des méthodes de fuzzification/défuzzification sans Hadoop.

Méthode de fuzzification	Méthode de défuzzification	TR(%)	TC(%)	TE(s)
FA Trapézoïdale	Centre de gravité	16.33	83.67	29.06
	Bisecteur de zone	21.95	78.05	31.56
	Moyenne pondérée	24.08	75.92	25.37
	Centre des sommes	12.64	87.36	35.09
FA Triangulaire	Centre de gravité	18.02	81.98	21.64
	Bisecteur de zone	25.41	74.59	28.03
	Moyenne pondérée	24.98	75.02	22.18
	Centre des sommes	14.36	85.64	27.92
FA Gaussienne	Centre de gravité	13.56	86.44	23.46
	Bisecteur de zone	20.03	79.97	25.84
	Moyenne pondérée	23.45	76.55	24.39
	Centre des sommes	10.25	89.75	22.01

floue en sortie nette à l'aide de la méthode du centre des sommes. Cette agrégation FA gaussienne/Centre des Sommes augmente le TC à 89.75 % et diminue le TR à 10.25 % ; de plus, cette combinaison est meilleure en termes de temps d'exécution qui est égal à 22.01s.

Pour démontrer l'efficacité de l'application du Hadoop sur les approches de fuzzification et des méthodes de défuzzification , nous suivons le même procédé que précédemment. Le tableau 6.3 décrit le résultat obtenu.

TABLE 6.7 – TR, TC et TE de l'agrégation des méthodes de fuzzification/défuzzification avec Hadoop.

Méthode de fuzzification	Méthode de défuzzification	TR(%)	TC(%)	TE(s)
FA Trapézoïdale	Centre de gravité	12.26	87.74	5.812
	Bisecteur de zone	17.02	82.98	6.312
	Moyenne pondérée	18.91	81.09	5.074
	Centre des sommes	8.79	91.21	7.018
FA Triangulaire	Centre de gravité	14.56	85.44	4.328
	Bisecteur de zone	19.62	80.38	5.606
	Moyenne pondérée	17.34	82.66	4.436
	Centre des sommes	9.86	90.14	5.584
FA Gaussienne	Centre de gravité	7.49	92.51	4.692
	Bisecteur de zone	16.07	83.93	5.168
	Moyenne pondérée	18.05	81.95	4.878
	Centre des sommes	5.13	94.87	4.402

D'après le tableau 6.3 nous remarquons que le cadre Hadoop réduit le temps d'exécution des douze agrégations de fuzzification/défuzzification. En outre, elle augmente le TC et diminue le TR et TE. Par exemple, dans le cas de la combinaison FA gaussien/Centre des sommes le TE est réduit de 22.01 s comme indiqué dans le tableau 6.6 à 4.402s comme présenté dans le tableau 6.7. Le TC est augmenté de 89.75 % comme illustré dans le tableau 6.6 à 94.87 % comme présenté dans le tableau 6.7. Le TR est réduit

de 10.25 % comme indiqué dans le tableau 6.6 à 5.13 % comme illustré dans le tableau 6.7. Par conséquent, cette expérience nous a permis d'apprendre deux remarques. Premièrement, le cadre Hadoop possède une capacité significative à améliorer les performances de notre proposition CAPF. Deuxièmement, la combinaison FA Gaussienne/Centre des Sommes a prouvé son efficacité par rapport aux onze autres combinaisons. Pour cette raison, nous avons décidé d'utiliser cette combinaison FA Gaussienne/Centre des Sommes comme méthodes de fuzzification/défuzzification dans notre CAPF proposé.

6.6.3 Troisième expérience

Dans cette expérience, plusieurs CAPFs ont été construits en jouant sur différents paramètres pour chaque couche. Les paramètres de configuration utilisés pour le modèle d'apprentissage en profondeur (CNN+FFNN) que nous proposons, tels que le nombre de couches de convolution, le nombre de couches de pooling, le nombre de couches entièrement connectées, le nombre de couches cachées, les fonctions d'activation appliquées, la dimension du filtre, le régularisateur, le nombre de filtres, les options d'abandon, la dimension de la fenêtre sélectionnée à partir de la matrice de plongement de mots, la dimension du vocabulaire et le nombre d'époques, sont décrits dans le tableau 6.8.

TABLE 6.8 – Configuration des paramètres de notre modèle d'apprentissage en profondeur (CNN+FFNN).

Paramètres	Valeurs
N. de C de convolution	1, 2, 3, 4, 5, 6, 7
N. de C de pooling	1, 2, 3, 4, 5, 6, 7
Matrice de plongement en entrée	500x500
N. de C entièrement connectées	1
Fonctions d'activation	sigmoid,softmax,ReLU
Dimension du filtre	3, 4, 5, 7, 9, 10, 11, 12
Régularisateur	L2
N. de filtres	15, 45, 90, 180, 360, 270, 225, 200, 190, 183, 185, 187, 184
N. des couches cachées	3, 4, 5, 6, 7, 8
Options d'abandon	0.5
Dimension de la fenêtre	256
Dimension du vocabulaire	40,000
Nombre d'époques	15

Pour l'évaluation de cette proposition, nous prenons soit 1,2,3,4,5,6, ou 7 couches de convolution, également nous prenons soit 1,2,3,4,5,6, ou 7 couches de pooling, puis nous varions les dimensions des filtres tels que 3x3, 4x4, 5x5, 7x7, 9x9, 10x10, 11x11, 12x12 et le nombre de ces filtres employés varie de 15 à 135. De plus, nous prenons soit 3, 4, 5, 6, 7 ou 8 couches cachées. Pour les autres paramètres tels que la matrice de plongement en entrée, l'option d'abandon, les fonctions d'activation appliquées, le régularisateur, la dimension de la fenêtre, la dimension du vocabulaire, et le nombre d'époques ont été maintenus constants avec les changements effectués parce que ces paramètres n'ont pas

démontré une amélioration de la performance dans notre contribution. Dans tous les CAPFs construits, le nombre de couches de convolution, le nombre de couches de pooling a été modifié conjointement avec d'autres paramètres utilisés tels que la dimension des filtres, le nombre de couches cachées, le nombre de filtres, la méthode du plongement de mots est définie en FastText, et les approches de fuzzification/défuzzification sont définies en méthodes FA Gaussienne/Centre des Sommes. La configuration des paramètres des 32 CAPFs parallèles est illustrée dans le tableau 6.9.

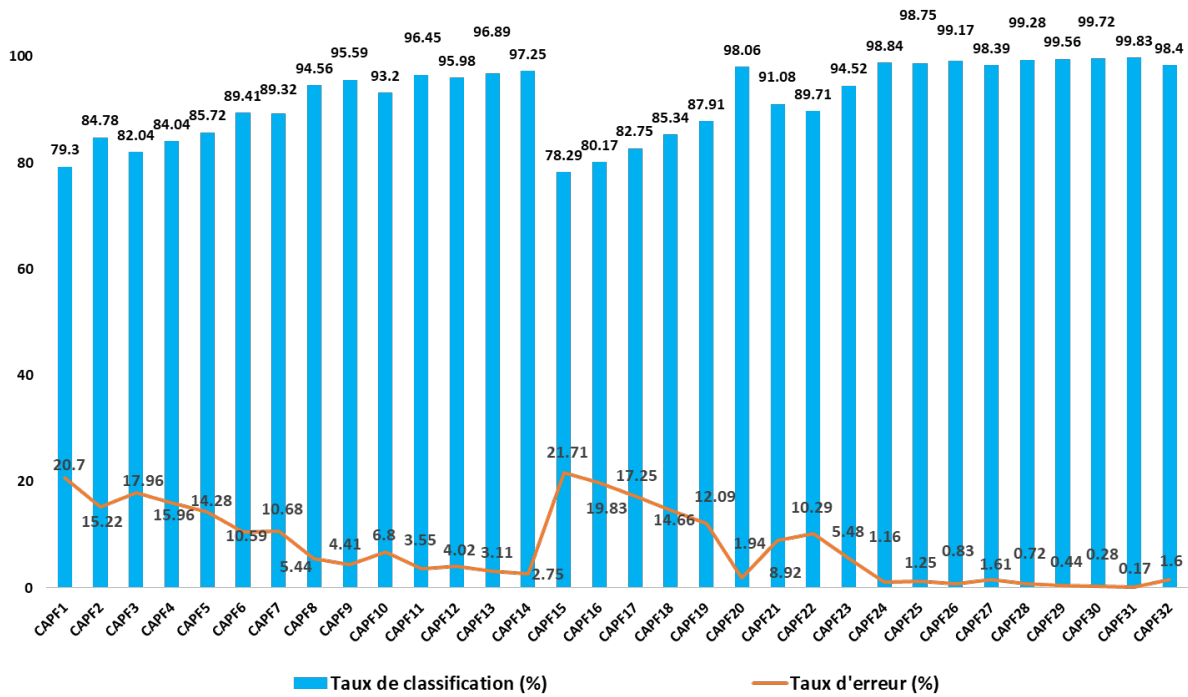


FIGURE 6.18 – TC et TR de tous les CAPFs évalués de notre contribution.

Comme présenté dans le tableau 6.9, la différence entre CAPF1 et CAPF2 est le nombre de couches cachées, tel que le CAPF1 a trois couches cachées et CAPF2 a quatre couches cachées, et concernant la dimension des filtres tel que le CAPF1 a utilisé deux filtres de dimension 4x4 et 5x5, et CAPF2 a appliqué deux filtres de dimension 5x5 et 7x7. D'après le tableau 6.10, nous remarquons que le taux de classification de CAPF1 est égal à 91.30% et celui de CAPF2 est égal à 94.04%. Donc, il y a une amélioration significative pour passer de CAPF1 à CAPF2. En se basant sur CAPF1 et CAPF2, nous avons construit deux nouveaux modèles, appelés CAPF3 et CAPF4, dans lesquels nous avons fixé le nombre de couches cachées, et nous avons fait varier la dimension des filtres utilisés pour trouver les causes qui rendent CAPF2 meilleur que CAPF1. A partir de CAPF2 et CAPF4, nous constatons que le filtre 7x7 fait augmenter le taux de classification de 0.74%. De même, à partir de CAPF1 et CAPF4, nous remarquons que l'utilisation de quatre couches cachées dans CAPF4 au lieu de trois couches cachées dans CAPF1 augmente la classification de 4.74

À ce point, le modèle le plus efficace parmi les quatre modèles construits est le CAPF2. Donc, à partir de ce modèle CAPF2, nous allons construire deux autres modèles, qui sont appelés CAPF5 et CAPF6. Dans le CAPF5, nous ajoutons un nouveau filtre de dimension 9x9, et dans le CAPF6, nous augmentons le nombre de filtres utilisés à 45, et nous ajoutons

TABLE 6.9 – Configuration des paramètres pour nos CAPFs parallèles.

Nom	N. de C de convolution	N. de C de pooling	N. de C cachées	N. de filtres	D. des filtres
CAPF1	1	1	3	15	4,5
CAPF2	1	1	4	15	5, 7
CAPF3	1	1	3	15	5, 7
CAPF4	1	1	4	15	4, 5
CAPF5	1	1	4	15	5, 7, 9
CAPF6	1	1	4	45	5, 7, 9
CAPF7	1	1	4	45	5, 7, 9, 11
CAPF8	1	1	4	90	5, 7, 9, 11
CAPF9	1	1	4	90	5, 7, 9, 10
CAPF10	1	1	4	90	5, 7, 9, 12
CAPF11	1	1	5	90	5, 7, 9, 10
CAPF12	1	1	4	180	5, 7, 9, 10
CAPF13	1	1	5	180	5, 7, 9, 10
CAPF14	1	1	6	180	5, 7, 9, 10
CAPF15	1	1	5	360	5, 7, 9, 10
CAPF16	1	1	5	270	5, 7, 9, 10
CAPF17	1	1	5	225	5, 7, 9, 10
CAPF18	1	1	5	200	5, 7, 9, 10
CAPF19	1	1	5	190	5, 7, 9, 10
CAPF20	1	1	5	183	5, 7, 9, 10
CAPF21	1	1	5	185	5, 7, 9, 10
CAPF22	1	1	5	187	5, 7, 9, 10
CAPF23	1	1	5	184	5, 7, 9, 10
CAPF24	1	1	7	183	5, 7, 9, 10
CAPF25	2	2	5	183	5, 7, 9, 10
CAPF26	2	2	7	183	5, 7, 9, 10
CAPF27	2	2	8	183	5, 7, 9, 10
CAPF28	3	3	7	183	5, 7, 9, 10
CAPF29	4	4	7	183	5, 7, 9, 10
CAPF30	5	5	7	183	5, 7, 9, 10
CAPF31	6	6	7	183	5, 7, 9, 10
CAPF32	7	7	7	183	5, 7, 9, 10

TABLE 6.10 – TC, TR et TE des différents CAPF parallèles proposés.

Nom	TC(%)	TR(%)	TE(s)
CAPF1	79.30	20.7	4.446
CAPF2	84.78	15.22	4.960
CAPF3	82.04	17.96	6.009
CAPF4	84.04	15.96	5.750
CAPF5	85.72	14.28	7.120
CAPF6	89.41	10.59	7.978
CAPF7	89.32	10.68	10.862
CAPF8	94.56	5.44	11.970
CAPF9	95.59	4.41	12.621
CAPF10	93.20	23.29	12.968
CAPF11	96.45	3.55	13.200
CAPF12	95.98	4.02	15.157
CAPF13	96.89	3.11	16.402
CAPF14	97.25	2.75	19.586
CAPF15	78.29	21.71	41.170
CAPF16	80.17	19.83	32.865
CAPF17	82.75	17.25	29.145
CAPF18	85.34	14.66	25.007
CAPF19	87.91	12.09	21.18
CAPF20	98.06	1.94	18.145
CAPF21	91.08	8.92	19.03
CAPF22	89.71	10.29	19.35
CAPF23	94.52	5.48	19.11
CAPF24	98.84	1.16	19.445
CAPF25	98.75	1.25	20.92
CAPF26	99.17	0.83	22.03
CAPF27	98.39	1.61	24.59
CAPF28	99.28	0.72	25.12
CAPF29	99.56	0.44	25.94
CAPF30	99.72	0.28	26.64
CAPF31	99.83	0.17	25.97
CAPF32	98.40	3.98	27.58

également le filtre de dimension 9x9 aux autres filtres employés. Par conséquent, le filtre avec la dimension 9x9 fait augmenter le taux de classification de 0.94 %, et l'augmentation du nombre de filtres utilisés à 45 conduit à une amélioration du taux de classification de 3.69 %.

À ce moment actuel, le CAPF6 est meilleur modèle que tous les modèles suggérés précédemment ; de même, nous construisons deux autres modèles basés sur le CAPF6 qui sont appelés CAPF7 et CAPF8. Dans CAPF7, nous attachons un filtre avec la dimension 11x11 aux différents filtres adoptés dans CAPF6, et dans CAPF8, nous ajoutons également un filtre avec la dimension 11x11, et nous boostons le nombre de filtres adoptés à 90.

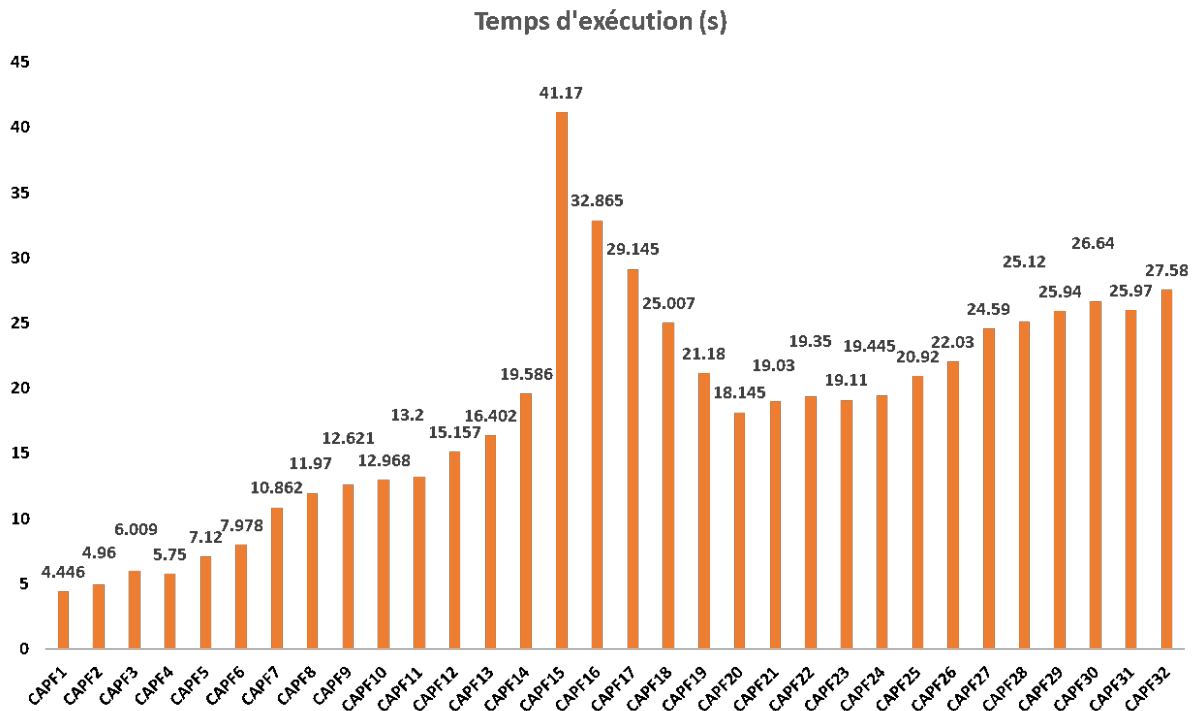


FIGURE 6.19 – TE de tous les CAPFs évalués de notre contribution.

Grâce à une étude comparative entre CAPF7 et CAPF8, nous constatons que le taux de classification est diminué de 0.09 % lorsque nous ajoutons le filtre avec la dimension 11x11. Par conséquent, dans la prochaine expérience, nous utiliserons un filtre avec la dimension 10x10 et un filtre avec la dimension 12x12 pour découvrir le filtre optimal parmi 9x9, 10x10, 11x11 et 12x12 filtres. Dans le modèle CAPF8, le taux de classification est augmenté de 5.24 %. On peut donc conclure que l'augmentation du nombre de filtres employés conduit à une augmentation du taux de classification.

Actuellement, le modèle le plus performant est le modèle CAPF8. En conséquence, nous avons construit les modèles CAPF9 et CAPF10 à partir de ce modèle CAPF8. Ces nouveaux modèles conservent la même configuration que CAPF8. La seule différence concerne la dimension du filtre ajouté, notamment dans le modèle CAPF9, nous supprimons le filtre avec la dimension 11x11 et nous adoptons un nouveau filtre avec la dimension 10x10, ainsi dans le modèle CAPF10, nous éliminons le filtre avec la dimension 11x11 et nous ajoutons un nouveau filtre avec la dimension 12x12. Le filtre avec la dimension 11x11 est supprimé parce qu'il fait baisser la performance de la classification comme indiqué précédemment à propos de la performance du modèle CAPF7. Conformément au résultat obtenu, et comme expliqué dans le tableau 6.10, le filtre avec la dimension 10x10 employé dans le CAPF9 fait augmenter la performance de classification de 1.03 %, et le filtre avec la dimension 12x12 utilisé dans le CAPF10 fait diminuer la performance de 0.97 % par rapport au CAPF8. Dans cette expérience, nous observons que la dimension du filtre qui va de 3x3 à 10x10 conduit à une augmentation du taux de classification. Cependant, une fois que nous arrivons à une dimension de filtre supérieure ou égale à la dimension 11x11, la performance de classification commencera à diminuer. Par conséquent, nous avons considéré la dimension 10x10 comme étant une dimension optimale

pour notre CAPF proposé.

Conformément à toutes les expériences réalisées précédemment, CAPF9 est le modèle le plus efficace. En conséquence, nous formons deux modèles différents CAPF11 et CAPF12 à partir de la configuration de ce modèle CAPF9. Dans le modèle CAPF11, nous varions le nombre de couches cachées, et pour le modèle CAPF12, nous augmentons le nombre de filtres. D'après les résultats expérimentaux, nous constatons que l'augmentation du nombre de couches cachées entraîne une augmentation de 0.86 % du taux de classification pour le modèle CAPF11. En outre, nous observons que le modèle CAPF12 augmente la performance de 0.39 % car le nombre de filtres utilisés augmente. À partir de ces résultats, nous construisons un nouveau modèle CAPF13 qui augmente le nombre de couches cachées et le nombre de filtres employés en même temps. Le résultat expérimental prouve que le modèle CAPF13 a atteint le meilleur taux de classification à ce moment qui est égal à 96.89 %.

Une fois de plus, nous créons deux nouveaux modèles, qui sont les CAPF14 et CAPF15. Dans le CAPF14, nous augmentons le nombre de couches cachées à 6, et dans le CAPF15 nous augmentons le nombre de filtres employés à 360. D'après les résultats empiriques, nous montrons que le CAPF14 augmente le taux de classification de 0.36 % par rapport au CAPF13. Mais l'augmentation du nombre de filtres utilisés dans le CAPF15 provoque une baisse significative du taux de classification par rapport au CAPF13. Par comparaison, il a réduit le taux de classification de 96.89 % dans CAPF13 à 78.29 % dans CAPF15. Ces diminutions considérables démontrent que la valeur optimale du nombre de filtres est proche de 180. Selon l'expérience précédente, nous créons quatre autres nouveaux modèles, à savoir CAPF16, CAPF17, CAPF18 et CAPF19. L'objectif de ces modèles est de trouver l'intervalle possible dans lequel la valeur optimale du nombre de filtres utilisés est existante. Dans le CAPF16, nous avons utilisé 270 filtres adoptés, pour le modèle CAPF17, nous avons adopté 225, également dans le CAPF18, nous avons appliqué 200. Enfin, dans le CAPF19, nous avons employé 190 filtres. D'après les résultats expérimentaux, le taux de classification augmente lorsque le nombre de filtres utilisés est proche de 180. Donc ; la valeur optimale se trouve dans l'intervalle [180, 190].

Pour déterminer la valeur optimale du nombre de filtres qui doit adopter, nous établissons trois modèles, à savoir CAPF20, CAPF21 et le modèle CAPF22. Dans le modèle CAPF20, nous avons utilisé 185 filtres, et le nombre de filtres utilisés dans le modèle CAPF21 est égal à 183. De même, pour le modèle CAPF22, nous avons appliqué 187 filtres. Le résultat préliminaire a prouvé que la valeur optimale du nombre de filtres utilisés se situe dans l'intervalle [183,185]. Par conséquent, dans la précédente expérience, nous avons calculé le taux de classification pour 183 et ceux de 185. Dans cette expérience, nous allons créer un nouveau modèle appelé CAPF23, dans lequel le nombre de filtres utilisés est égal à 184. Les résultats expérimentaux acquis du modèle CAPF20 qui est appliqué un nombre de filtres égal à 183 montrent que son taux de classification est égal à 97.25%, qui est actuellement le taux de classification le plus élevé. Donc, nous déduisons que la valeur optimale des filtres qui devrait appliquer est égal à 183.

Après avoir utilisé le modèle CAPF20, nous modifions le nombre de couches cachées afin de déterminer le nombre le plus efficace. Pour ce faire, nous développons quatre modèles, à savoir CAPF24, CAPF25, CAPF26 et CAPF27. Dans le modèle CAPF24, nous

avons utilisé 7 couches cachées. Aussi, nous changeons le nombre de couches de convolution à 2, et nous modifions le nombre de couches de pooling à 2 dans le modèle CAPF25. Puis dans le modèle CAPF26, nous modifions en même temps le nombre de couches de convolution, de couches de pooling et de couches cachées. De plus, nous changeons le nombre de couches cachées à 8 dans le modèle CAPF27. Conséquemment, CAPF24 améliore le taux de classification de 0.78 %. Ainsi, le CAPF25 fait augmenter le taux de classification de 0.67 % par rapport au CAPF20. En conséquence, le taux de classification dans le modèle CAPF26 est monté à 99.17 %. Mais le CAPF27 a un taux de classification inférieur au CAPF26 qui est égal à 98.39 %. D'après les modèles CAPF26 et CAPF27, nous concluons que le nombre optimal de couches cachées est de 7 couches.

A partir des résultats empiriques précédents, nous établissons cinq modèles, dans lesquels nous faisons varier en même temps le nombre de couches de convolution et de pooling. Ces modèles sont CAPF28, CAPF29, CAPF30, CAPF31, et CAPF32, comme décrit dans le Tableau 6.9. Dans le rapport des résultats expérimentaux pour les cinq derniers modèles, nous remarquons que 6 est le nombre optimal de couches de convolution et de pooling. Conformément aux résultats exposés dans le tableau 6.10 et dans la figure 6.18, nous remarquons que le plus efficace CAPF dans ce travail est le CAPF31 avec un taux de classification égal à 99.83 %. La figure 6.20 illustre l'architecture finale de notre classificateur d'apprentissage en profondeur parallèle et flou.

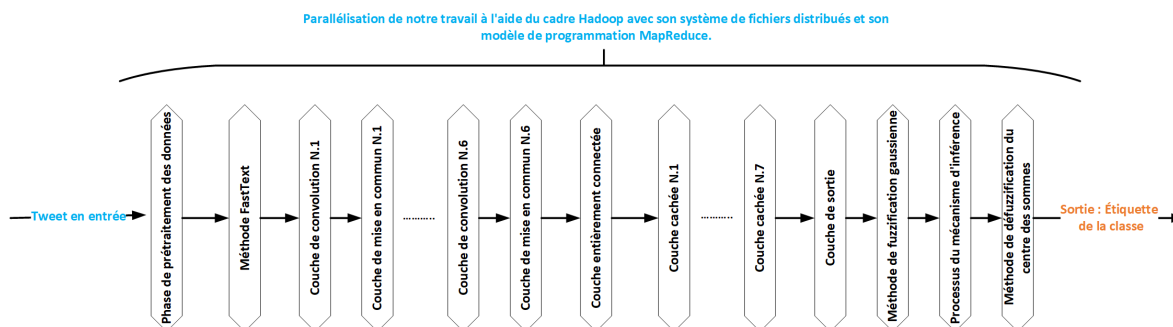


FIGURE 6.20 – Architecture de notre proposition après des multiples expériences.

D'après le tableau 6.10 et la figure 6.19, nous remarquons que notre proposition a un taux de classification plus élevé mais consomme un temps d'exécution plus élevé. Donc, pour éviter ce problème nous décidons d'utiliser plus de nœuds de calcul dans le cluster Hadoop. La figure 6.21 représente le résultat expérimental après l'exécution de notre proposition sur plusieurs nœuds de calcul dans le cluster Hadoop. A partir de cette figure 6.21, nous observons que le temps d'exécution est passé de 25.97s avec cinq machines de calcul à 0.0089s avec douze machines de calcul.

6.6.4 Quatrième expérience

L'objectif de cette expérience est de comparer notre proposition avec plusieurs autres méthodes de la littérature. Les travaux sélectionnés sont : Une approche titrée "Multi-task learning model based on Multi-scale CNN and LSTM for sentiment classification" proposé par Jin *et al.* [252]. Ce travail fusionne CNN et RNRLC pour améliorer les performances de l'analyse des sentiments et il a atteint un taux de classification égal à 86.25 %. Une

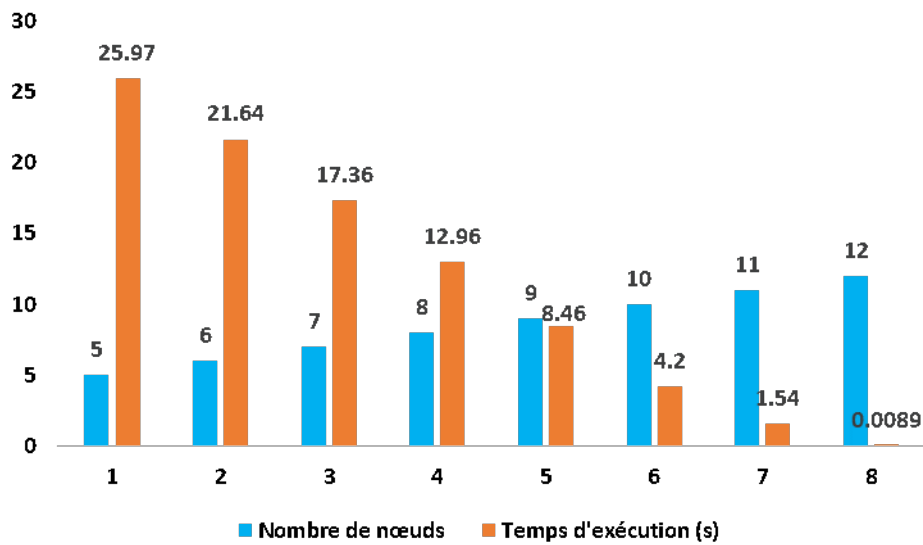


FIGURE 6.21 – TE de notre approche illustrée dans la figure 6.20.

méthode intitulée "Stacked Residual Recurrent Neural Networks With Cross-Layer Attention for Text Classification" proposée par Lan *et al.* [254]. Cette approche intègre le RNR résiduel empilé et la technique d'attention en couche croisée. Elle vise à capturer et à détecter davantage de caractéristiques linguistiques et à les utiliser pour l'analyse des sentiments. Elle a atteint un taux de classification de 89 %. Une autre approche appelée "Comparison Enhanced Bi-LSTM with MultiHead Attention (CE-B-MHA)" développée par Lin *et al.* [255]. Elle combine l'attention multi-têtes pour extraire les caractéristiques globales et la force du RNRBLMCT (Bi-LSTM) pour découvrir les caractéristiques des séquences locales. Une autre méthode intitulée "hybrid method for bilingual text sentiment classification based on deep learning" mise en œuvre par Liu *et al.* [256]. Cette méthode intègre l'algorithme d'apprentissage automatique BN, MVS avec le modèle d'apprentissage en profondeur RNR et RNRLMCT. Enfin, la méthode intitulée "Intelligent asset allocation via market sentiment views" conçue par Xing *et al.* [259] qui propose de combiner également le regroupement évolutif avec le modèle d'apprentissage en profondeur LSTM. Notre proposition et toutes ces approches choisies dans la littérature sont appliquées aux deux jeux de données utilisés dans ce travail, à savoir les jeux de données Sentiment140 et COVID-19_Sentiments.

La figure 6.22 illustre les résultats obtenus en termes de TC et TE en appliquant notre proposition et d'autres méthodes sélectionnées sur le jeu de données Sentiment140.

La figure 6.23 illustre les résultats obtenus en termes de TC et TE après l'application de notre CAPF et d'autres techniques sélectionnées dans la littérature sur le jeu de données COVID-19_Sentiments.

D'après les figures 6.22 et 6.23, nous observons que notre proposition basée sur le modèle d'apprentissage en profondeur (CNN+FFNN), le cadre Hadoop et le système flou de Mamdani surperforme les autres approches appliquées (Jin *et al.* [252], Lan *et al.* [254], Lin *et al.* [255], Liu *et al.* [256], et Xing *et al.* [259]) avec un taux de classification égale à 99.83%, 99.99%, et un temps d'exécution égal à 0.0089s, et 0.00534s respectivement sur le

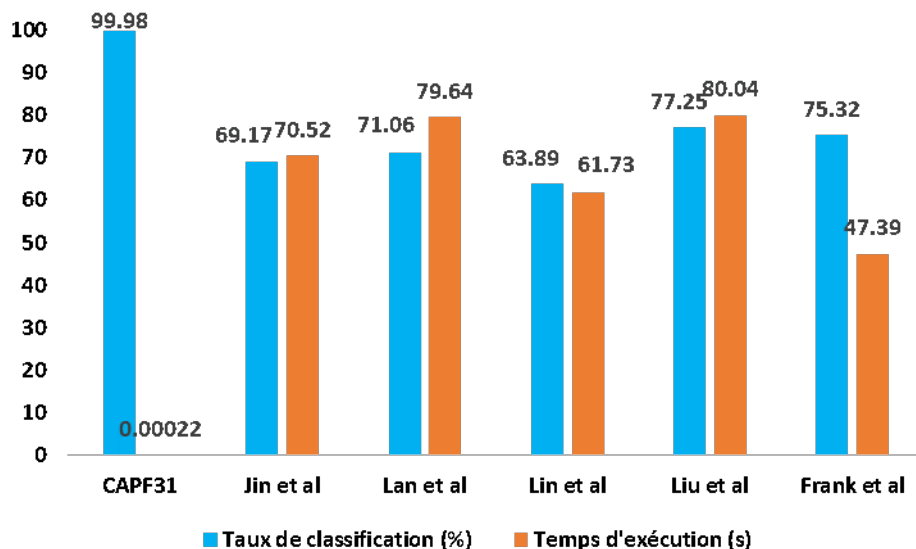


FIGURE 6.22 – TC et TE en appliquant notre approche et d'autres méthodes sur Sentiment140.

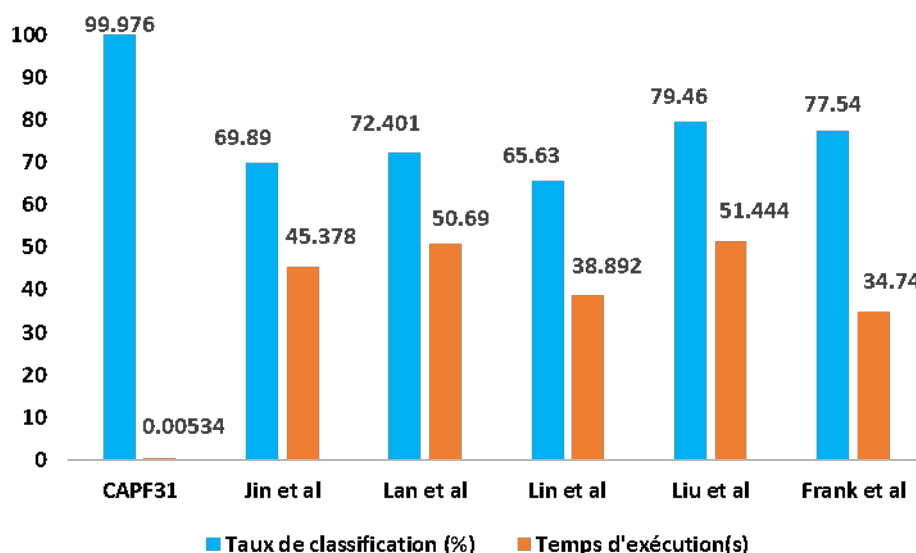


FIGURE 6.23 – TC et TE en appliquant notre approche et d'autres approches sur COVID-19_Sentiments.

jeu de données Sentiment140 et le jeu de données COVID-19_Sentiments. L'efficacité et les performances élevées de notre proposition sont dues à l'application de la logique floue, à l'intégration des modèles d'apprentissage en profondeur CNN+FFNN, et à l'utilisation de douze nœuds de calcul dans le cluster Hadoop.

Pour une évaluation plus approfondie du classificateur d'apprentissage en profondeur parallèle et flou que nous proposons, nous avons réalisé une autre expérience qui compare notre proposition avec les mêmes approches sélectionnées précédemment. Mais dans ce cas, les critères d'évaluation utilisés seront *TPV*, *TNF*, *TNV*, *TPF*, *PR*, *SK*, et *SF* comme présentées dans la section 3.9 du chapitre 3. Nous gardons toujours les mêmes jeux de

données (Sentiment140 et COVID-19_Sentiments) pour effectuer cette étude comparative. Les résultats expérimentaux de cette expérience sont illustrés dans le Tableau 6.11.

TABLE 6.11 – *TPV*, *TNF*, *TNV*, *TPF*, *PR*, *SK*, et *SF* pour notre CAPF et les autres approches.

		TPV	TNF	TNV	TPF	PR	SK	SF
COVID-19_Sentiments	Notre CAPF31	99.98	0.02	98.61	1.39	98.55	97.96	98.35
	Jin <i>et al.</i> [252]	70.89	29.11	69.08	30.92	70.85	67.49	71.14
	Lan <i>et al.</i> [254]	72.64	27.36	71.51	28.49	69.54	70.66	69.20
	Lin <i>et al.</i> [255]	65.76	34.24	61.49	38.51	58.27	57.37	62.81
	Liu <i>et al.</i> [256]	78.64	21.36	76.98	23.02	74.23	72.15	69.45
	Xing <i>et al.</i> [259]	76.45	23.55	73.26	26.74	42.72	50.52	62.42
Sentiment140	Notre CAPF31	99.81	0.19	98.91	1.09	97.75	98.96	96.54
	Jin <i>et al.</i> [252]	67.74	32.26	68.41	31.59	69.18	71.87	66.94
	Lan <i>et al.</i> [254]	69.32	30.68	69.91	30.80	70.29	72.16	68.42
	Lin <i>et al.</i> [255]	61.304	30.696	62.04	37.96	69.12	65.25	67.59
	Liu <i>et al.</i> [256]	75.87	24.13	74.05	25.95	70.92	72.62	69.33
	Xing <i>et al.</i> [259]	73.19	27.81	75.31	24.96	72.29	75.14	72.48

Selon les résultats expérimentaux décrits dans le tableau 13, nous observons que notre proposition (CNN+FFNN+MFS+Hadoop) surpasse les autres approches sélectionnées dans la littérature pour les deux jeux de données (Sentiment140 et COVID-19_Sentiments) et au niveau du TPV(99.98 %, 99.81 %), TNF(0.02 %, 0.19 %), TNV(98.61 %, 98.91 %), TPF(1.39 %, 1.09 %) PR(98.55 %, 97.75 %), SK(97.96 %, 98.96 %) et SF(98.35 %,96.54 %).

6.6.5 Cinquième expérience

Dans cette dernière expérience, nous avons évalué l'efficacité de notre CAPF31 proposé en termes de complexité, de convergence et de stabilité. Cette expérience sert à comparer notre CAPF proposé, Jin *et al.* [252], Lan *et al.* [254], Lin *et al.* [255], Liu *et al.* [256], et Xing *et al.* [259] approches et de découvrir la méthode la plus efficace parmi toutes les approches évaluées en termes de complexité, de convergence et de stabilité.

6.6.5.1 Complexité

La complexité d'un modèle est la mesure du temps d'exécution et de l'espace mémoire utilisé par le modèle. Dans cette sous-section, nous avons évalué la complexité temporelle et spatiale de notre proposition CAPF31 et des autres approches. Par ailleurs, le tableau 6.12 présente les résultats expérimentaux obtenus après avoir mesuré la complexité spatiale de notre modèle CAPF31 et des autres modèles choisis en termes de nombre d'opérations exécutées et de nombre de paramètres.

D'après le tableau 6.12, nous constatons que notre CAPF31 proposé a effectué des opérations nombreuses avec une taille égale à (42M,102M) respectivement pour les jeux de données COVID-19_Sentiments et Sentiment140. En outre, la taille des paramètres

TABLE 6.12 – Complexité spatiale de notre proposition CAPF31 et des autres approches.

Jeux de données	Algorithmes	No. d'opérations	No. de paramètres
COVID-19_Sentiments	Notre CAPF31	42M	21.76M
	Jin <i>et al.</i> [252]	79.5M	51.10M
	Lan <i>et al.</i> [254]	98M	46.5M
	Lin <i>et al.</i> [255]	57.37M	27.80M
	Liu <i>et al.</i> [256]	102M	70.11M
	Xing <i>et al.</i> [259]	45M	29.64M
Sentiment140	Notre CAPF31	102M	47.82M
	Jin <i>et al.</i> [252]	203.75M	107.21M
	Lan <i>et al.</i> [254]	294M	79.5M
	Lin <i>et al.</i> [255]	130.13M	63.4M
	Liu <i>et al.</i> [256]	306M	150.33M
	Xing <i>et al.</i> [259]	105.78M	56.43M

de notre CAPF31 est égale à (21.76M,47.82M) respectivement pour les jeux de données COVID-19_Sentiments et Sentiment140. Comme le montrent les résultats expérimentaux, notre proposition de CAPF31 nécessite une complexité spatiale de calcul beaucoup plus faible que celle de autres approches.

Le tableau 6.13 montre les résultats empiriques obtenus après avoir mesuré la complexité temporelle de notre modèle CAPF31 et d'autres modèles choisis en termes de temps consommé pour l'apprentissage et de temps consommé pour le test.

TABLE 6.13 – Complexité temporelle de notre modèle CAPF31 et d'autres modèles.

Jeux de données	Algorithmes	Temps d'apprentissage	Temps de test
COVID-19_Sentiments	Notre CAPF31	0.00534s	0.00178s
	Jin <i>et al.</i> [252]	45.378s	16.50s
	Lan <i>et al.</i> [254]	50.69s	18.10s
	Lin <i>et al.</i> [255]	38.892s	13.84s
	Liu <i>et al.</i> [256]	51.444s	17.148s
	Xing <i>et al.</i> [259]	34.74s	12.86s
Sentiment140	Notre CAPF31	0.0089s	0.00287s
	Jin <i>et al.</i> [252]	75.63s	26.17s
	Lan <i>et al.</i> [254]	81.15s	27.05s
	Lin <i>et al.</i> [255]	64.82s	23.15s
	Liu <i>et al.</i> [256]	85.74s	31.75s
	Xing <i>et al.</i> [259]	57.9s	18.67s

D'après le tableau 6.13, nous observons que notre CAPF31 proposé a consommé un temps d'apprentissage égal à (0.00534s, et 0.0089s) respectivement pour les jeux de données COVID-19_Sentiments et Sentiment140. Aussi, le temps de test de notre modèle CAPF31 est égal à (0.00178s, et 0.00287s) respectivement pour les jeux de données COVID-19_Sentiments et Sentiment140. Comme le décrit le résultat empirique, notre proposition CAPF31 requiert une complexité temporelle de calcul beaucoup plus faible que celle de autres approches.

6.6.5.2 Convergence

Le classificateur proposé sera démontré s'il est convergent ou non convergent en trouvant un nombre particulier de tours d'entraînement dans lequel le classificateur proposé satisfait la condition décrite dans l'équation 6.19. Cette équation définit la condition de la convergence :

$$E_{rp} - E_{rc} \geq T_{re} \quad (6.19)$$

Où E_{rp} est l'erreur moyenne du classificateur du tour d'apprentissage précédent, E_{rc} est l'erreur moyenne du classificateur du tour d'apprentissage actuel, et T_{re} est la valeur seuil qui définit la valeur du taux de convergence, et après avoir effectué plusieurs expériences, nous avons fixé ce taux seuil à 0.0001. Par conséquent, l'erreur moyenne du classificateur suggéré est mesurée à l'aide de l'équation suivante :

$$E = \frac{1}{2} \times \frac{\sum_{j=1}^S \sum_{i=1}^C (y - y_{label})^2}{S} \quad (6.20)$$

Où S est le nombre total d'instances dans l'ensemble de données utilisé, C est le nombre total d'étiquettes de classe (dans notre cas, il y a trois étiquettes de classe qui sont négatives, neutres et positives), y est la décision de classification souhaitée en sortie, et y_{label} la décision de classification obtenue en sortie. Si l'équation (6.19) décrite précédemment est vérifiée, notre classificateur proposé peut être considéré comme convergent, et l'algorithme est exécuté jusqu'à ce que l'erreur moyenne du classificateur satisfasse la condition. Dans le cas contraire, notre classificateur proposé ne converge pas. La figure 6.24 montre le taux de convergence de notre CAPF31 proposé pour les deux jeux de données utilisés. D'après la figure 6.24, nous constatons que notre CAPF31 proposé a convergé vers la valeur seuil de 0,0001 après que notre CAPF31 ait atteint les itérations 200, et 400 dans le cas des jeux de données COVID-19_Sentiments et Sentiment140, respectivement.

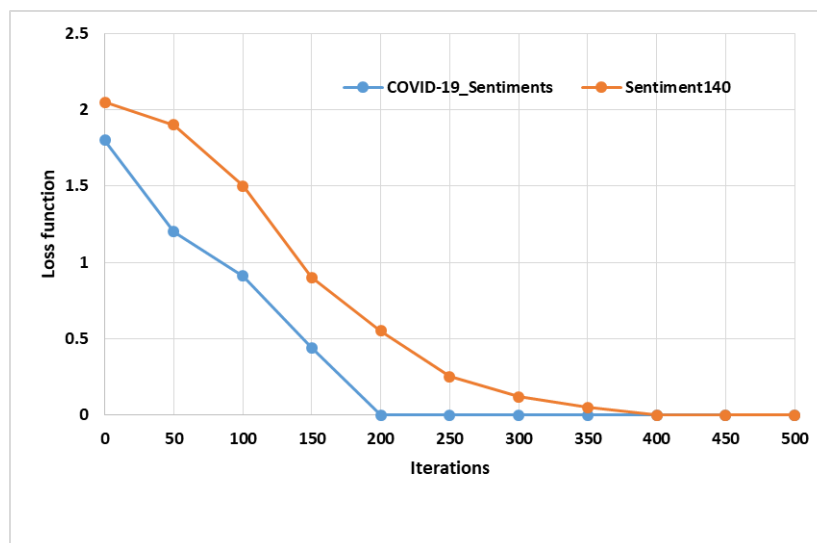


FIGURE 6.24 – Taux de convergence de notre CAPF31 pour les deux jeux de données.

Le tableau 6.15 représente le taux de convergence de notre CAPF31 et les autres approches. D'après le tableau 6.15, nous déduisons que notre proposition CAPF31 converge

très rapidement.

TABLE 6.14 – Taux de convergence de notre CAPF31 et les autres approches.

Jeux de données	Algorithmes	Itérations
COVID-19_Sentiments	Notre CAPF31	200
	Jin <i>et al.</i> [252]	1020
	Lan <i>et al.</i> [254]	1425
	Lin <i>et al.</i> [255]	800
	Liu <i>et al.</i> [256]	1289
	Xing <i>et al.</i> [259]	500
Sentiment140	Notre CAPF31	400
	Jin <i>et al.</i> [252]	1523
	Lan <i>et al.</i> [254]	2008
	Lin <i>et al.</i> [255]	1264
	Liu <i>et al.</i> [256]	1749
	Xing <i>et al.</i> [259]	712

6.6.5.3 Stabilité

Nous déterminons si notre CAPF31 est stable ou non en calculant l'écart type moyen correspondant au taux de classification de chaque modèle en appliquant cinq validations croisées différentes du jeu de données utilisé. Notre CAPF31 est entraîné avec les mêmes hyperparamètres et configurations mais avec cinq validations croisées différentes du jeu de données. Le tableau 6.15, montre le taux de classification moyen (TCM) et l'écart-type moyen (ETM) obtenus du CAPF31, Jin *et al.* [252], Lan *et al.* [254], Lin *et al.* [255], Liu *et al.* [256], et Xing *et al.* [259].

TABLE 6.15 – Stabilité de notre classificateur CAPF31 et les autres approches.

Jeux de données	Algorithmes	TCM (%)	TCM (%)
COVID-19_Sentiments	Notre CAPF31	99.97	0.18
	Jin <i>et al.</i> [252]	69.54	2.45
	Lan <i>et al.</i> [254]	73.12	2.27
	Lin <i>et al.</i> [255]	64.80	3.52
	Liu <i>et al.</i> [256]	79.04	1.49
	Xing <i>et al.</i> [259]	77.62	1.79
Sentiment140	Notre CAPF31	99.81	0.23
	Jin <i>et al.</i> [252]	66.78	2.84
	Lan <i>et al.</i> [254]	70.43	2.68
	Lin <i>et al.</i> [255]	62.25	3.76
	Liu <i>et al.</i> [256]	76.52	2.09
	Xing <i>et al.</i> [259]	74.38	2.35

D'après le tableau 6.15, nous constatons que notre CAPF31 est pratiquement toujours capable d'atteindre un taux de classification moyenne plus élevée avec un écart-type moyen très faible. Cela signifie que notre CAPF31 est plus stable que les autres approches.

6.7 Conclusions

La diversité des plateformes de médias sociaux entraîne une utilisation massive par les personnes, qui considèrent ces plateformes comme un outil de communication efficace. Par conséquent, les commentaires des utilisateurs sur ces plateformes ont généré de grandes données d'analyse des sentiments à étudier. À présent, la TALN, le cadre Hadoop et les modèles d'apprentissage en profondeur fournissent un ensemble d'outils visant à capturer et à détecter les sentiments exprimés par les utilisateurs dans les jeux de données massifs collectés à partir des plateformes de médias sociaux. Dans ce travail, un nouveau classificateur parallèle d'apprentissage en profondeur floue est développé. Ce classificateur intègre des méthodes de pré-traitement des données, des approches de prolongement de mots, un modèle d'apprentissage en profondeur CNN+FFNN et un système floue de Mamdani. L'objectif principal de cette proposition est de déterminer une relation significative entre les approches de prolongement des mots et les deux modèles d'apprentissage en profondeur utilisés (CNN+FFNN). Il a également comme objectif de traiter les ambiguïtés inhérentes aux données en appliquant le système floue de Mamdani. Le classificateur proposé est parallélisé à l'aide d'Hadoop pour éviter le problème de long temps d'exécution et améliorer le taux de classification.

Conclusion Générale et Perspectives

Dans cette thèse, nous avons étudié et analysé des nouveaux problèmes de l'analyse de sentiment en utilisant les technologies Big Data et les techniques de la fouille de données spécifiquement les techniques d'apprentissage automatique et de théorie de la logique floue. En premier lieu, nous avons souligné l'importance des techniques de pré-traitement de données en tant que solution prometteuse pour éliminer les données bruyantes et les informations indésirables afin d'améliorer la qualité de données avant de les utiliser par le processus de classification et nous avons comparé et mis en œuvre plusieurs extracteurs et sélecteurs de caractéristiques pour déterminer le plus efficace extracteur/sélecteur qui permettent de détecter et capturer efficacement les données pertinentes à partir des tweets analysés et de réduire la dimensionnalité des caractéristiques élevées. Deuxièmement, nous avons proposé trois approches hybride pour effectuer la classification des sentiments avec une haute performance en termes de taux positif vrai, spécificité, taux positif faux, Taux négatif faux, Taux d'erreur, précision, taux de classification, statistique kappa, score F1, temps d'exécution, complexité, convergence et stabilité.

Nous avons commencé ce rapport par une introduction aux problématiques fréquemment étudiées dans la fouille d'opinion. Ensuite, nous avons présenté un état de l'art des travaux réalisés sur l'application des technologies du Big data et de la fouille de données dans le domaine d'analyse de sentiments. Nous avons également décrit le cadre général de notre thèse, nos motivations et nos contributions. Ensuite, nous avons présenté les bases et les principes fondamentaux du Big Data et de la fouille de données utilisés dans cette thèse pour étudier, analyser et classifier les sentiments exprimés dans les tweets collectés à partir de Twitter.

Après, nous avons proposé un algorithme ID3 basé sur le calcul de gain d'information pondéré au lieu du gain d'information calculé dans l'ID3 traditionnel. L'amélioration de l'ID3 a été intégrée dans le processus de la fouille d'opinion pour la classification des sentiments. Notre processus de la fouille d'opinion s'est effectué en cinq étapes :

- Étudier et analyser les performances de chaque technique de traitement automatique du Langage naturel appliqué sur nos données.
- Étudier et comparer les performances de plusieurs extracteurs de caractéristiques pour détecter et capturer efficacement les données pertinentes à partir des tweets.
- Analyser et comparer les performances de plusieurs sélecteurs de caractéristiques pour réduire la dimensionnalité des caractéristiques élevées.
- Application de notre ID3 amélioré pour classifier les tweets.
- Cette contribution est mise en œuvre dans un environnement distribué Hadoop.

L'objectif principal de cette proposition est d'améliorer la performance de l'analyse des sentiments en se basant sur de multiples expériences pour choisir à chaque étape la technique appropriée.

Dans la suite, nous avons élaboré un nouveau modèle d'analyse des sentiments qui vise à améliorer le processus d'extraction et de sélection des caractéristiques à l'aide du réseau de neurones convolutifs, à traiter l'ambiguïté enracinée dans le traitement du langage naturel de données en utilisant le système à base de règles floues et à traiter et stocker la quantité volumineuse de données sentimentales en employant le framework Hadoop. Ce modèle proposé se compose de six parties :

- Mettre en oeuvre plusieurs tâches de prétraitement pour supprimer les données bruyantes et éliminer les informations indésirables afin d'améliorer la qualité de nos données.
- Appliquer la méthode de prolongement des mots FastText qui vectorise les tweets analysés suite aux résultats expérimentaux obtenus dans la première contribution.
- Utiliser le réseau neuronal convolutif pour extraire et sélectionner automatiquement les caractéristiques pertinentes à partir des tweets analysés afin de résoudre le problème des méthodes d'extraction manuelle qui extraient des caractéristiques incomplètes et prend beaucoup de temps pour produire le résultat final.
- Implémenter le système à base de règles floues par fuzzifié la sortie du réseau neuronal convolutif en appliquant la méthode de fuzzification gaussienne afin de traiter les données incertaines et ambiguës.
- Appliquer l'algorithme C4.5 flou pour créer l'arbre flou et générer les règles floues au lieu de la création manuelle des règles floues.
- Appliquer l'approche générale de raisonnement flou sur les règles floues pour classifier les nouveaux tweets et nous avons également mis en oeuvre notre modèle sous framework Hadoop.

Due aux études dans la littérature qui démontrent que la précision des algorithmes d'apprentissage automatique en profondeur augmente avec la croissance de la taille des données, nous avons combiné les deux modèles d'apprentissages automatiques en profondeur qui sont le réseau neuronal convolutif et le réseau de neurones à propagation avant avec le système flou de Mamdani. Notre nouveau classificateur comprend principalement cinq aspects :

- Appliquer les étapes de prétraitement des données afin de réduire les données bruyantes et d'améliorer la qualité des données.
- Utiliser les méthodes de prolongement lexical de mots pour convertir les données textuelles en données numériques.
- Appliquer notre modèle d'apprentissage en profondeur hybride qui combine le réseau neuronal convolutif et le réseau de neurones à propagation avant afin de construire un processus automatique efficace pour extraire les caractéristiques à partir des données non structurées et pour calculer les deux scores sentimentaux positifs et négatifs.
- Mettre en oeuvre le système flou de Mamdani comme un classificateur flou pour classifier les résultats des deux modèles d'apprentissage en profondeur en trois classes : Neutre, Négatif et Positif
- Paralléliser notre classificateur sous Hadoop afin de surmonter le problème du long temps d'exécution.

Notre travail est des nouvelles contributions apportées aux nombreuses études sur

la classification et l'apprentissage automatique. Pour une évaluation plus complète, il serait judicieux dans le futur d'élargir le volume et le type des données. Comme, il serait pertinent de tester d'autres types de classifieurs, surtout ceux qui supportent les deux modèles supervisé et non supervisé. Ainsi, nos travaux futurs consistent à combiner notre approche avec les réseaux de capteurs sans fil. L'objectif principal de ces travaux futurs est de classifier les données collectées par les nœuds de capteurs, en prenant en compte de multiples paramètres associés à la détection, à l'extraction de caractéristiques et à l'agrégation de données.

Bibliographie

- [1] M. H. ur Rehman, I. Yaqoob, K. Salah, M. Imran, P. P. Jayaraman, and C. Perera, “The role of big data analytics in industrial internet of things,” vol. 99, pp. 247–259. [14](#), [33](#)
- [2] B. Kosko, *Fuzzy Thinking : The New Science of Fuzzy Logic*. New York : Hyperion, reprint édition ed., 1994. [24](#), [176](#)
- [3] L. A. Zadeh, “Fuzzy logic = computing with words,” *IEEE Transactions on Fuzzy Systems*, vol. 4, pp. 103–111, May 1996. Conference Name : IEEE Transactions on Fuzzy Systems. [24](#), [176](#)
- [4] P. Bühlmann and S. van de Geer, “Statistics for big data : A perspective,” vol. 136, pp. 37–41. [27](#)
- [5] K. M. Potter, F. H. Koch, C. M. Oswald, and B. V. Iannone, “Data, data everywhere : detecting spatial patterns in fine-scale ecological information collected across a continent,” vol. 31, no. 1, pp. 67–84. [27](#)
- [6] M. M. Mariani and S. Fosso Wamba, “Exploring how consumer goods companies innovate in the digital age : The role of big data analytics companies,” vol. 121, pp. 338–352. [27](#)
- [7] Y. Chen, *Big Data and Its Applications in Coal Research and Exploration*. [27](#)
- [8] A. McAfee and E. Brynjolfsson, “Big Data : The Management Revolution,” p. 9. [27](#)
- [9] R. Iqbal, F. Doctor, B. More, S. Mahmud, and U. Yousuf, “Big data analytics : Computational intelligence techniques and application areas,” *Technological Forecasting and Social Change*, vol. 153, p. 119253, Apr. 2020. [27](#)
- [10] C. L. Philip Chen and C.-Y. Zhang, “Data-intensive applications, challenges, techniques and technologies : A survey on Big Data,” *Information Sciences*, vol. 275, pp. 314–347, Aug. 2014. [27](#)
- [11] B. Logica and R. Magdalena, “Using Big Data in the Academic Environment,” *Procedia Economics and Finance*, vol. 33, pp. 277–286, Jan. 2015. [27](#)
- [12] M. Chen, S. Mao, and Y. Liu, “Big Data : A Survey,” *Mobile Networks and Applications*, vol. 19, pp. 171–209, Apr. 2014. [28](#)
- [13] A. A. Hussein, “How Many Old and New Big Data V’s Characteristics, Processing Technology, And Applications (BD1),” *International Journal of Application or Innovation in Engineering and Managemen*, vol. 9, no. 9, 2020. [28](#)
- [14] Ishwarappa and J. Anuradha, “A brief introduction on big data 5vs characteristics and hadoop technology,” vol. 48 of *International Conference on Computer, Communication and Convergence (ICCC 2015)*, pp. 319–324. [28](#)
- [15] B. Zhou, “Keyword Search on Large-Scale Structured, Semi-Structured, and Unstructured Data,” in *Handbook of Data Intensive Computing* (B. Furht and A. Escalante, eds.), pp. 733–751, New York, NY : Springer, 2011. [28](#)

- [16] F. Cauteruccio, P. L. Giudice, L. Musarella, G. Terracina, D. Ursino, and L. Virgili, "A Lightweight Approach to Extract Interschema Properties from Structured, Semi-Structured and Unstructured Sources in a Big Data Scenario," *International Journal of Information Technology & Decision Making*, June 2020. Publisher : World Scientific Publishing Company. 28
- [17] N. Deshai, B. V. D. S. Sekhar, P. V. G. D. P. Reddy, and V. V. S. S. S. Chakravarthy, "Processing Real World Datasets using Big Data Hadoop Tools," *JSIR Vol.79(07) [July 2020]*, July 2020. Accepted : 2020-08-24T06 :21 :56Z Publisher : NISCAIR-CSIR, India. 28
- [18] J. Bhogal and I. Choksi, "Handling Big Data Using NoSQL," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, pp. 393–398, Mar. 2015. 28
- [19] G. Press, "A Very Short History Of Big Data," *Forbes*. Section : Innovation. 28
- [20] Y. Noh, "Imagining library 4.0 : Creating a model for future libraries." 28
- [21] O. Romero, V. Herrero, A. Abelló, and J. Ferrarons, "Tuning small analytics on big data : Data partitioning and secondary indexes in the hadoop ecosystem." 28
- [22] S. Sagiroglu and D. Sinanc, "Big data : A review." 28
- [23] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics : A technology tutorial." Conference Name : IEEE Access. 29
- [24] F. X. Diebold, "A Personal Perspective on the Origin(s) and Development of 'Big Data' : The Phenomenon, the Term, and the Discipline, Second Version," SSRN Scholarly Paper ID 2202843, Social Science Research Network, Rochester, NY, Nov. 2012. 29
- [25] S. Lohr, "The Origins of 'Big Data' : An Etymological Detective Story," Feb. 2013. Section : Technology. 29
- [26] K. Brodlie, R. Allendes Osorio, and A. Lopes, "A review of uncertainty in data visualization." 29
- [27] D. Laney, "3-D Data Management : Controlling Data Volume, Velocity and Variety," *Application Delivery Strategies by META Group Inc.*, vol. 949, Feb. 2001. 29, 30
- [28] V. Rajaraman, "Big data analytics." 29, 31
- [29] Y. Demchenko, C. de Laat, and P. Membrey, "Defining architecture components of the Big Data Ecosystem," in *2014 International Conference on Collaboration Technologies and Systems (CTS)*, (Minneapolis, MN, USA), pp. 104–112, IEEE, May 2014. 29, 31
- [30] J. Wu, S. Guo, J. Li, and D. Zeng, "Big data meet green challenges : Greening big data," vol. 10, no. 3, pp. 873–887. Conference Name : IEEE Systems Journal. 29
- [31] S. G. Alonso, I. de la Torre Díez, J. J. P. C. Rodrigues, S. Hamrioui, and M. López-Coronado, "A systematic review of techniques and sources of big data in the healthcare sector," vol. 41, no. 11, p. 183. 33
- [32] J.-G. Lee and M. Kang, "Geospatial big data : Challenges and opportunities," vol. 2, no. 2, pp. 74–81. 33
- [33] S. Stephens-Davidowitz, "The cost of racial animus on a black candidate : Evidence using google search data," vol. 118, pp. 26–40. 33

- [34] M. Ge, H. Bangui, and B. Buhnova, “Big Data for Internet of Things : A Survey,” *Future Generation Computer Systems*, vol. 87, pp. 601–614, Oct. 2018. [33](#)
- [35] D. Sanchez, O. Solarte, V. Bucheli, and H. Ordonez, “Evaluating The Scalability of Big Data Frameworks,” *Scalable Computing : Practice and Experience*, vol. 19, pp. 301–307, Sept. 2018. [34](#)
- [36] A. H. Ali, “A Survey on Vertical and Horizontal Scaling Platforms for Big Data Analytics,” *International Journal of Integrated Engineering*, vol. 11, pp. 138–150, Sept. 2019. Number : 6. [34](#), [35](#)
- [37] A. Abualkishik, “HADOOP AND BIG DATA CHALLENGES,” *Journal of Theoretical and Applied Information Technology*, vol. 97, p. 3488, June 2019. [35](#)
- [38] S. K. and J. Editor, “HADOOP WORKFLOWS,” vol. 1, Aug. 2020. [35](#)
- [39] D. Zburivsky, *Hadoop Cluster Deployment*. Packt Publishing Ltd, Nov. 2013. [35](#)
- [40] K. G. Srinivasa and A. K. Muppalla, “Guide to High Performance Distributed Computing : Case Studies with Hadoop, Scalding and Spark,” *Computer Communications and Networks*, pp. 185–217, Cham : Springer International Publishing, 2015. [35](#)
- [41] Y. Luo, S. Luo, J. Guan, and S. Zhou, “A RAMCloud storage system based on HDFS : Architecture, implementation and evaluation,” vol. 86, no. 3, pp. 744–750. [36](#)
- [42] H. H. Le, S. Hikida, and H. Yokota, “NameNode and DataNode coupling for a power-proportional hadoop distributed file system,” in *Database Systems for Advanced Applications* (W. Meng, L. Feng, S. Bressan, W. Winiwarter, and W. Song, eds.), pp. 99–107, Springer. [36](#)
- [43] C.-T. Yang, W.-C. Shih, L.-T. Chen, C.-T. Kuo, F.-C. Jiang, and F.-Y. Leu, “Accessing medical image file with co-allocation HDFS in cloud,” vol. 43-44, pp. 61–73. [37](#)
- [44] A. Alam and J. Ahmed, “Hadoop architecture and its issues,” in *2014 International Conference on Computational Science and Computational Intelligence*, vol. 2, pp. 288–291. [37](#)
- [45] D. Dev and R. Patgiri, “A Deep Dive into the Hadoop World to Explore Its Various Performances,” in *Techniques and Environments for Big Data Analysis : Parallel, Cloud, and Grid Computing* (B. S. P. Mishra, S. Dehuri, E. Kim, and G.-N. Wang, eds.), *Studies in Big Data*, pp. 31–51, Cham : Springer International Publishing, 2016. [37](#)
- [46] B. J. Mathiya and V. L. Desai, “Apache hadoop yarn parameter configuration challenges and optimization,” in *2015 International Conference on Soft-Computing and Networks Security (ICSNS)*, pp. 1–6. [38](#)
- [47] Y. Kim, T. Araragi, J. Nakamura, and T. Masuzawa, “A distributed NameNode cluster for a highly-available hadoop distributed file system,” in *2014 IEEE 33rd International Symposium on Reliable Distributed Systems*, pp. 333–334. ISSN : 1060-9857. [38](#)
- [48] J. Dean and S. Ghemawat, “MapReduce : a flexible data processing tool,” vol. 53, no. 1, pp. 72–77. [38](#)

- [49] J. Dean and S. Ghemawat, “MapReduce : simplified data processing on large clusters,” vol. 51, no. 1, pp. 107–113. [38](#)
- [50] J. Dean and S. Ghemawat, “Mapreduce : Simplified data processing on large clusters,” *Commun. ACM*, vol. 51, p. 107–113, Jan. 2008. [39](#)
- [51] J. Lin and C. Dyer, “Data-Intensive Text Processing with MapReduce,” *Synthesis Lectures on Human Language Technologies*, vol. 3, pp. 1–177, Jan. 2010. Publisher : Morgan & Claypool Publishers. [39](#)
- [52] R. Buyya, C. Vecchiola, and S. T. Selvi, *Mastering Cloud Computing : Foundations and Applications Programming*. Newnes, Apr. 2013. Google-Books-ID : wqKkqH-JhPJQC. [40](#)
- [53] C. Swa and Z. Ansari, “Apache Pig - A Data Flow Framework Based on Hadoop Map Reduce,” *International Journal of Engineering Trends and Technology*, vol. 50, pp. 271–275, Aug. 2017. [40](#)
- [54] A. Fuad, A. Erwin, and H. P. Ipung, “Processing performance on apache pig, apache hive and MySQL cluster,” in *Proceedings of International Conference on Information, Communication Technology and System (ICTS) 2014*, pp. 297–302. [40](#)
- [55] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy, “Hive : A warehousing solution over a map-reduce framework,” vol. 2, p. 1626–1629, Aug. 2009. [40](#), [41](#)
- [56] B. N. Kumar and L. B. Babu, “Integrating Hadoop with Relational Databases using Sqoop,” *International Journal of Engineering Research*, vol. 4, no. 34, p. 3, 2016. [41](#)
- [57] S. Rathee, *Big data and Hadoop with components like Flume, Pig, Hive and Jaql*, vol. 15. [42](#)
- [58] P. J. Dendek, A. Czczko, M. Fedoryszak, A. Kawa, P. Wendykier, and u. Boliowski, *Chrum : The Tool for Convenient Generation of Apache Oozie Workflows*. Springer International Publishing. [42](#)
- [59] C. Artho, Q. Gros, G. Rousset, K. Banzai, L. Ma, T. Kitamura, M. Hagiya, Y. Tanabe, and M. Yamamoto, “Model-based API testing of apache ZooKeeper,” in *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pp. 288–298. [42](#)
- [60] S. G. Walunj and K. Sadafale, “An online recommendation system for e-commerce based on apache mahout framework,” in *Proceedings of the 2013 annual conference on Computers and people research, SIGMIS-CPR '13*, pp. 153–158, Association for Computing Machinery. [43](#)
- [61] G. Wang, J. Koshy, S. Subramanian, K. Paramasivam, M. Zadeh, N. Narkhede, J. Rao, J. Kreps, and J. Stein, “Building a replicated logging system with apache kafka,” vol. 8, no. 12, pp. 1654–1655. [43](#)
- [62] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The Hadoop Distributed File System,” in *2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, pp. 1–10, May 2010. ISSN : 2160-1968. [44](#)
- [63] A. S. Tanenbaum and M. Van Steen, *Distributed systems : principles and paradigms*. Prentice-Hall, 2007. [44](#)

- [64] M. Alian, D. Kim, and N. S. Kim, "pd-gem5 : Simulation Infrastructure for Parallel/Distributed Computer Systems," *IEEE Computer Architecture Letters*, vol. 15, pp. 41–44, Jan. 2016. Conference Name : IEEE Computer Architecture Letters. [44](#), [45](#)
- [65] L. Ho, J. Wu, and P. Liu, "Optimal Algorithms for Cross-Rack Communication Optimization in MapReduce Framework," in *2011 IEEE 4th International Conference on Cloud Computing*, pp. 420–427, July 2011. ISSN : 2159-6190. [45](#), [52](#)
- [66] "Fault Tolerance in Cluster Computing System," pp. 408–412, Oct. 2011. [45](#), [46](#), [52](#), [53](#)
- [67] R. Buyya, "High performance cluster computing : Architectures and systems (volume 1)," *Prentice Hall, Upper SaddleRiver, NJ, USA*, vol. 1, no. 999, p. 29, 1999. [45](#), [53](#)
- [68] L. A. Barroso, J. Dean, and U. Holzle, "Web search for a planet : The Google cluster architecture," *IEEE Micro*, vol. 23, pp. 22–28, Mar. 2003. Conference Name : IEEE Micro. [45](#)
- [69] X. Wu, X. Zhu, G. Wu, and W. Ding, "Data mining with big data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, pp. 97–107, Jan. 2014. Conference Name : IEEE Transactions on Knowledge and Data Engineering. [46](#), [51](#)
- [70] J. Long, Z. Gao, H. Ren, and A. Lian, "Urban traffic congestion propagation and bottleneck identification." [46](#), [52](#)
- [71] L. Zhengyou and C. Tao, "A Distributed Parallel Algorithm for Web Page Inverted Indexes Construction on the Cluster Computing Systems," in *2009 International Forum on Information Technology and Applications*, vol. 2, pp. 33–36, May 2009. [46](#), [53](#)
- [72] V. Dhar, M. Jarke, and J. Laartz, "Big data." [46](#)
- [73] J. Dean and S. Ghemawat, "MapReduce : simplified data processing on large clusters," *Communications of the ACM*, vol. 51, pp. 107–113, Jan. 2008. [46](#), [47](#), [50](#), [53](#)
- [74] W. Fan and A. Bifet, "Mining big data : Current status, and forecast to the future," *SIGKDD Explor. Newsl.*, vol. 14, p. 1–5, Apr. 2013. [46](#), [51](#), [52](#), [53](#)
- [75] F. Es-Sabery and A. Hair, "Big data solutions proposed for cluster computing systems challenges : A survey," in *Proceedings of the 3rd International Conference on Networking, Information Systems and Security, NISS2020*, (New York, NY, USA), Association for Computing Machinery, 2020. [47](#), [49](#)
- [76] J. Bent, D. Thain, A. Arpaci-Dusseau, R. Arpaci-dusseau, and M. Livny, "Explicit Control in a Batch-Aware Distributed File System," Mar. 2004. [48](#)
- [77] M. Vaidya and S. Deshpande, "Comparative analysis of various distributed file systems performance evaluation using map reduce implementation," in *2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, pp. 1–6, Dec. 2016. ISSN : 5090-2806. [48](#), [50](#)
- [78] S. Wu, G. Chen, K. Chen, F. Li, and L. Shou, "HM : A Column-Oriented MapReduce System on Hybrid Storage," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, pp. 3304–3317, Dec. 2015. Conference Name : IEEE Transactions on Knowledge and Data Engineering. [48](#)

- [79] S. Ghemawat, H. Gombioff, and S.-T. Leung, “The google file system.” 49
- [80] S. Ghemawat, H. Gombioff, and S.-T. Leung, “The Google File System,” in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, (Bolton Landing, NY), pp. 20–43, 2003. 49
- [81] J. Nandimath, E. Banerjee, A. Patil, P. Kakade, S. Vaidya, and D. Chaturvedi, “Big data analysis using apache hadoop.” 50
- [82] M. R, Tejus, C. R.K, and B. S, “A Big Data MapReduce Hadoop distribution architecture for processing input splits to solve the small data problem,” 2016. 50, 51, 52
- [83] J. Dean, “Experiences with mapreduce, an abstraction for large-scale computation,” in *Proceedings of the 15th International Conference on Parallel Architectures and Compilation Techniques*, PACT '06, (New York, NY, USA), p. 1, Association for Computing Machinery, 2006. 51
- [84] Z. Xiao and Y. Xiao, “Accountable MapReduce in cloud computing,” in *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1082–1087, Apr. 2011. 51
- [85] S. Sumathi and S. N. Sivanandam, *Introduction to Data Mining and its Applications*. Studies in Computational Intelligence, Berlin Heidelberg : Springer-Verlag, 2006. 55
- [86] P. Mohebbi and P. Jafari, “Survey Data Mining and its Application in Industrial Engineering,” 55
- [87] V. Plotnikova, M. Dumas, and F. Milani, “Adaptations of data mining methodologies : a systematic literature review,” *PeerJ Computer Science*, vol. 6, p. e267, May 2020. 55
- [88] A. Gandomi and M. Haider, “Beyond the hype : Big data concepts, methods, and analytics,” *International Journal of Information Management*, vol. 35, pp. 137–144, Apr. 2015. 55
- [89] G. Mariscal, s. Marbán, and C. Fernández, “A survey of data mining and knowledge discovery process models and methodologies,” *The Knowledge Engineering Review*, vol. 25, no. 2, p. 137–166, 2010. 56
- [90] W. Lee, S. J. Stolfo, and K. W. Mok, “Mining in a data-flow environment : Experience in network intrusion detection,” in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, (New York, NY, USA), p. 114–124, Association for Computing Machinery, 1999. 56
- [91] U. Fayyad, “Data mining and knowledge discovery in databases : implications for scientific databases,” in *Proceedings. Ninth International Conference on Scientific and Statistical Database Management (Cat. No.97TB100150)*, pp. 2–11. 56, 57
- [92] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “The KDD process for extracting useful knowledge from volumes of data,” *Communications of the ACM*, vol. 39, pp. 27–34, Nov. 1996. 56
- [93] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “Knowledge Discovery and Data Mining : Towards a Unifying Framework,” p. 7. 56
- [94] T. M. Mitchell, “Machine learning and data mining,” *Commun. ACM*, vol. 42, p. 30–36, Nov. 1999. 57

- [95] J. F. Elder, “A Statistical Perspective on KDD,” p. 7, 1995. 57
- [96] W. Zhang, “Machine Learning Approaches to Predicting Company Bankruptcy,” *Journal of Financial Risk Management*, vol. 06, no. 04, pp. 364–374, 2017. 58
- [97] T. Hofmann, B. Schölkopf, and A. J. Smola, “Kernel methods in machine learning,” *Annals of Statistics*, vol. 36, pp. 1171–1220, June 2008. Publisher : Institute of Mathematical Statistics. 60
- [98] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer Science & Business Media, June 2013. Google-Books-ID : EqgACAAAQBAJ. 60
- [99] R. Polikar, “Ensemble based systems in decision making,” *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006. 60
- [100] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression : a statistical view of boosting (With discussion and a rejoinder by the authors),” *Annals of Statistics*, vol. 28, pp. 337–407, Apr. 2000. Publisher : Institute of Mathematical Statistics. 60
- [101] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001. 60
- [102] C. M. Bishop and P. o. N. C. C. M. Bishop, *Neural Networks for Pattern Recognition*. Clarendon Press, Nov. 1995. Google-Books-ID : T0S0BgAAQBAJ. 60
- [103] N. Noceti and F. Odone, “Semi-supervised learning of sparse representations to recognize people spatial orientation,” in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 3382–3386, Oct. 2014. ISSN : 2381-8549. 61
- [104] J. H. Min and C. Jeong, “A binary classification method for bankruptcy prediction,” vol. 36, no. 3, pp. 5256–5263. 61
- [105] J. Pennington, R. Socher, and C. Manning, “GloVe : Global Vectors for Word Representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, Oct. 2014. 66
- [106] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” *arXiv :1310.4546 [cs, stat]*, Oct. 2013. arXiv : 1310.4546. 67
- [107] S. Wu and U. Manber, “Fast text searching : allowing errors,” vol. 35, no. 10, pp. 83–91. 68
- [108] A. Jain and D. Zongker, “Feature selection : evaluation, application, and small sample performance,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 153–158, 1997. 69
- [109] M. Rogati and Y. Yang, “High-performing feature selection for text classification,” CIKM '02, (New York, NY, USA), p. 659–661, Association for Computing Machinery, 2002. 69
- [110] G. Miner, J. Elder IV, A. Fast, T. Hill, R. Nisbet, and D. Delen, *Practical text mining and statistical analysis for non-structured text data applications*. Academic Press. 69
- [111] C. E. Shannon, “A mathematical theory of communication,” *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, p. 3–55, Jan. 2001. 69

- [112] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, Mar. 1986. [70](#)
- [113] R. I. Lerman and S. Yitzhaki, "A note on the calculation and interpretation of the gini index," vol. 15, no. 3, pp. 363–368. [70](#)
- [114] A. Meyer-Baese and V. Schmid, "Chapter 6 - Statistical and Syntactic Pattern Recognition," in *Pattern Recognition and Signal Analysis in Medical Imaging (Second Edition)* (A. Meyer-Baese and V. Schmid, eds.), pp. 151–196, Oxford : Academic Press, Jan. 2014. [72](#)
- [115] M. Bramer, *Principles of Data Mining*. Undergraduate Topics in Computer Science, London : Springer-Verlag, 3 ed., 2016. [72](#)
- [116] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, pp. 660–674, May 1991. Conference Name : IEEE Transactions on Systems, Man, and Cybernetics. [73](#)
- [117] D. Thakur, N. Markandaiah, and D. S. Raj, "Re optimization of ID3 and C4.5 decision tree," in *2010 International Conference on Computer and Communication Technology (ICCT)*, pp. 448–450, Sept. 2010. [75](#)
- [118] B. Lausen, W. Sauerbrei, and M. Schumacher, "Classification and Regression Trees (CART) Used for the Exploration of Prognostic Factors Measured on Different Scales," in *Computational Statistics* (W. A. Müller, P. Schuster, P. Dirschedl, and R. Ostermann, eds.), pp. 483–496, Heidelberg : Physica-Verlag HD, 1994. Series Title : Contributions to Statistics. [75](#)
- [119] K. Polat and S. Güneş, "A novel hybrid intelligent method based on C4.5 decision tree classifier and one-against-all approach for multi-class classification problems," *Expert Systems with Applications*, vol. 36, pp. 1587–1592, Mar. 2009. [75](#), [76](#)
- [120] V. F. Rodriguez-Galiano, B. Ghimire, J. Rogan, M. Chica-Olmo, and J. P. Rigol-Sanchez, "An assessment of the effectiveness of a random forest classifier for land-cover classification," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 67, pp. 93–104, Jan. 2012. [75](#), [76](#)
- [121] U. M. Fayyad and K. B. Irani, "The attribute selection problem in decision tree generation," in *Proceedings of the tenth national conference on Artificial intelligence, AAAI'92*, (San Jose, California), pp. 104–110, AAAI Press, July 1992. [75](#)
- [122] S. Zhang, X. Li, M. Zong, X. Zhu, and D. Cheng, "Learning k for kNN classification," vol. 8, no. 3, pp. 43 :1–43 :19. [76](#)
- [123] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, pp. 273–297, Sept. 1995. [77](#)
- [124] M. A. Chandra and S. S. Bedi, "Survey on SVM and their application in imageclassification," vol. 13, no. 5, pp. 1–11. [77](#)
- [125] V. K. Chauhan, K. Dahiya, and A. Sharma, "Problem formulations and solvers in linear SVM : a review," vol. 52, no. 2, pp. 803–855. [77](#), [78](#)
- [126] T. Joachims, "Text categorization with Support Vector Machines : learning with many relevant features," in *Proceedings of the 10th European Conference on Machine Learning, ECML'98*, (Berlin, Heidelberg), pp. 137–142, Springer-Verlag, Apr. 1998. [78](#)

- [127] A. Lundborg, "Text classification of short messages," *LU-CS-EX 2017-14*, 2017. 78
- [128] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, "Machine learning : a review of classification and combining techniques," *Artificial Intelligence Review*, vol. 26, pp. 159–190, Nov. 2006. 79
- [129] L. S. Camargo and T. Yoneyama, "Specification of training sets and the number of hidden neurons for multilayer perceptrons," *Neural Computation*, vol. 13, pp. 2673–2680, Dec. 2001. 79
- [130] M. A. Kon and L. Plaskota, "Information complexity of neural networks," *Neural Networks : The Official Journal of the International Neural Network Society*, vol. 13, pp. 365–375, Apr. 2000. 79
- [131] C. Neocleous and C. Schizas, "Artificial Neural Network Learning : A Comparative Review," in *Methods and Applications of Artificial Intelligence* (I. P. Vlahavas and C. D. Spyropoulos, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 300–313, Springer, 2002. 79
- [132] K. Fukushima, "Neocognitron : A hierarchical neural network capable of visual pattern recognition," *Neural Networks*, vol. 1, pp. 119–130, Jan. 1988. 80
- [133] Y. Chen, "Convolutional Neural Network for Sentence Classification," Aug. 2015. Accepted : 2015-08-26T15 :45 :03Z Publisher : University of Waterloo. 81
- [134] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8609–8613. ISSN : 2379-190X. 82
- [135] H. Yousuf and S. Salloum, "Survey Analysis : Enhancing the Security of Vectorization by Using word2vec and CryptDB," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 4, pp. 374–380, 2020. 82
- [136] F. Es-SABERY and A. Hair, "An Improved ID3 Classification Algorithm Based On Correlation Function and Weighted Attribute*," in *2019 International Conference on Intelligent Systems and Advanced Computing Sciences (ISACS)*, pp. 1–8, Dec. 2019. 85
- [137] D. Tang, B. Qin, and T. Liu, "Deep learning for sentiment analysis : successful approaches and future challenges," *WIREs Data Mining and Knowledge Discovery*, vol. 5, no. 6, pp. 292–303, 2015. 91
- [138] A. Severyn and A. Moschitti, "Twitter sentiment analysis with deep convolutional neural networks," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '15*, (New York, NY, USA), p. 959–962, Association for Computing Machinery, 2015. 91, 92
- [139] R. Feldman, "Techniques and applications for sentiment analysis," *Commun. ACM*, vol. 56, p. 82–89, Apr. 2013. 91
- [140] R. Sharma, S. Nigam, and R. Jain, "Opinion Mining of Movie Reviews At Document Level," *International Journal on Information Theory*, vol. 3, pp. 13–21, July 2014. 91
- [141] A. Tripathy, A. Agrawal, and S. K. Rath, "Classification of sentiment reviews using n-gram machine learning approach," *Expert Systems with Applications*, vol. 57, pp. 117–126, 2016. 91, 93

- [142] K. K. Pawar, P. P. Shrishrimal, and R. R. Deshmukh, "Twitter Sentiment Analysis : A Review," vol. 6, no. 4, p. 9, 2015. 91
- [143] H. Lu and J. C.-C. Lin, "Predicting customer behavior in the market-space : a study of rayport and sviokla's framework," *Information & Management*, vol. 40, no. 1, pp. 1–10, 2002. 91
- [144] J. R. Saura, P. R. Palos-Sanchez, M. B. Correia, J. R. Saura, P. R. Palos-Sanchez, and M. B. Correia, "Digital Marketing Strategies Based on the E-Business Model : Literature Review and Future Directions," Jan. 1. Archive Location : digital-marketing-strategies-based-on-the-e-business-model ISBN : 9781522570745 Publisher : IGI Global. 91
- [145] Z. Kechaou, M. B. Ammar, and A. M. Alimi, "Improving e-learning with sentiment analysis of users' opinions," in *2011 IEEE Global Engineering Education Conference (EDUCON)*, pp. 1032–1038, Apr. 2011. ISSN : 2165-9567. 91
- [146] T. H. Nguyen and K. Shirai, "Topic modeling based sentiment analysis on social media for stock market prediction," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1 : Long Papers)*, (Beijing, China), pp. 1354–1364, Association for Computational Linguistics, July 2015. 91
- [147] P. Bhatt and C. M. Pickering, "Public perceptions about nepalese national parks : A global twitter discourse analysis," vol. 34, no. 6, pp. 685–702. Publisher : Routledge _eprint : <https://doi.org/10.1080/08941920.2021.1876193>. 92
- [148] D. Jiang, X. Luo, J. Xuan, and Z. Xu, "Sentiment Computing for the News Event Based on the Social Media Big Data," *IEEE Access*, vol. 5, pp. 2373–2382, 2017. Conference Name : IEEE Access. 92
- [149] M. U. Salur and I. Aydin, "A Novel Hybrid Deep Learning Model for Sentiment Classification," *IEEE Access*, vol. 8, pp. 58080–58093, 2020. Conference Name : IEEE Access. 92
- [150] F. Zhu and X. M. Zhang, "Impact of Online Consumer Reviews on Sales : The Moderating Role of Product and Consumer Characteristics," *Journal of Marketing*, vol. 74, pp. 133–148, Mar. 2010. Publisher : SAGE Publications Inc. 92
- [151] N. O. F. Daeli and A. Adiwijaya, "Sentiment Analysis on Movie Reviews using Information Gain and K-Nearest Neighbor," *Journal of Data Science and Its Applications*, vol. 3, pp. 1–7, May 2020. Number : 1. 93
- [152] N. Zainuddin and A. Selamat, "Sentiment analysis using Support Vector Machine," in *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, pp. 333–337, Sept. 2014. 93
- [153] M. Rathi, A. Malik, D. Varshney, R. Sharma, and S. Mendiratta, "Sentiment Analysis of Tweets Using Machine Learning Approach," in *2018 Eleventh International Conference on Contemporary Computing (IC3)*, pp. 1–3, Aug. 2018. ISSN : 2572-6129. 93
- [154] W. P. Ramadhan, S. T. M. T. A. Novianty, and S. T. M. T. C. Setianingsih, "Sentiment analysis using multinomial logistic regression," in *2017 International Conference on Control, Electronics, Renewable Energy and Communications (ICCREC)*, pp. 46–49, Sept. 2017. 93

- [155] C. Troussas, M. Virvou, K. J. Espinosa, K. Llaguno, and J. Caro, “Sentiment analysis of Facebook statuses using Naive Bayes classifier for language learning,” in *IISA 2013*, pp. 1–6, July 2013. [93](#)
- [156] Y. Al Amrani, M. Lazaar, and K. E. El Kadiri, “Random Forest and Support Vector Machine based Hybrid Approach to Sentiment Analysis,” *Procedia Computer Science*, vol. 127, pp. 511–520, Jan. 2018. [93](#)
- [157] P. S. Patil and N. V. Dharwadkar, “Analysis of banking data using machine learning,” in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp. 876–881, Feb. 2017. [93](#)
- [158] P. Burnap and M. L. Williams, “Us and them : identifying cyber hate on Twitter across multiple protected characteristics,” *EPJ Data Science*, vol. 5, pp. 1–15, Dec. 2016. Number : 1 Publisher : SpringerOpen. [93](#)
- [159] P. Larrañaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armañanzas, G. Santafé, A. Pérez, and V. Robles, “Machine learning in bioinformatics,” *Briefings in Bioinformatics*, vol. 7, pp. 86–112, Mar. 2006. [93](#)
- [160] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, “Abusive Language Detection in Online User Content,” in *Proceedings of the 25th International Conference on World Wide Web, WWW '16*, (Republic and Canton of Geneva, CHE), pp. 145–153, International World Wide Web Conferences Steering Committee, Apr. 2016. [93](#)
- [161] I. Habernal, T. Ptáček, and J. Steinberger, “Sentiment Analysis in Czech Social Media Using Supervised Machine Learning,” in *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, (Atlanta, Georgia), pp. 65–74, Association for Computational Linguistics, June 2013. [93](#)
- [162] K. Reynolds, A. Kontostathis, and L. Edwards, “Using Machine Learning to Detect Cyberbullying,” in *2011 10th International Conference on Machine Learning and Applications and Workshops*, vol. 2, pp. 241–244, Dec. 2011. [93](#)
- [163] M. S. Neethu and R. Rajasree, “Sentiment analysis in twitter using machine learning techniques,” in *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pp. 1–5, July 2013. [93](#)
- [164] A. Sharma and S. Dey, “A comparative study of feature selection and machine learning techniques for sentiment analysis,” in *Proceedings of the 2012 ACM Research in Applied Computation Symposium, RACS '12*, (New York, NY, USA), pp. 1–7, Association for Computing Machinery, Oct. 2012. [93](#), [95](#)
- [165] F. Es-SABERY and A. Hair, “An Improved ID3 Classification Algorithm Based On Correlation Function and Weighted Attribute*,” in *2019 International Conference on Intelligent Systems and Advanced Computing Sciences (ISACS)*, pp. 1–8, Dec. 2019. [94](#), [98](#), [116](#)
- [166] G. Patil, V. Galande, M. V. Kekani, and K. Dange, “Sentiment Analysis Using Support Vector Machine,” vol. 2, no. 1, p. 6, 2007. [95](#)
- [167] K. P. P. Shein and T. T. S. Nyunt, “Sentiment Classification Based on Ontology and SVM Classifier,” in *2010 Second International Conference on Communication Software and Networks*, pp. 169–172, Feb. 2010. [95](#)

- [168] P. Gamallo, M. Garcia, and S. Fernandez-Lanza, “TASS : A Naive-Bayes strategy for sentiment analysis on Spanish tweets,” p. 7. [95](#)
- [169] M. Anjaria and R. M. R. Guddeti, “A novel sentiment analysis of social networks using supervised learning,” *Social Network Analysis and Mining*, vol. 4, p. 181, Mar. 2014. [95](#)
- [170] R. M. Duwairi, “Sentiment analysis for dialectical Arabic,” in *2015 6th International Conference on Information and Communication Systems (ICICS)*, pp. 166–170, Apr. 2015. [95](#)
- [171] V. K. Soni and S. Pawar, “Emotion based social media text classification using optimized improved ID3 classifier,” in *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pp. 1500–1505, Aug. 2017. [95](#), [117](#), [119](#), [120](#), [121](#), [122](#), [123](#), [124](#)
- [172] P. V. Ngoc, C. V. T. Ngoc, T. V. T. Ngoc, and D. N. Duy, “A C4.5 algorithm for english emotional classification,” *Evolving Systems*, vol. 10, pp. 425–451, Sept. 2019. [95](#), [117](#), [119](#), [120](#), [121](#), [122](#), [123](#), [124](#)
- [173] Liu Yuxun and Xie Niuniu, “Improved ID3 algorithm,” in *2010 3rd International Conference on Computer Science and Information Technology*, vol. 8, pp. 465–468, July 2010. [95](#)
- [174] R.-m. Chai and M. Wang, “A more efficient classification scheme for ID3,” in *2010 2nd International Conference on Computer Engineering and Technology*, vol. 1, pp. V1–329–V1–332, Apr. 2010. [95](#)
- [175] S. Elyassami and A. Idri, “Applying Fuzzy ID3 Decision Tree for Software Effort Estimation,” *arXiv :1111.0158 [cs]*, Nov. 2011. arXiv : 1111.0158. [95](#)
- [176] K. Zou, W. Sun, H. Yu, and F. Liu, “ID3 Decision Tree in Fraud Detection Application,” in *2012 International Conference on Computer Science and Electronics Engineering*, vol. 3, pp. 399–402, Mar. 2012. [95](#)
- [177] V. Srinivasan, G. Rajenderan, J. Vandar Kuzhali, and M. Aruna, “Fuzzy fast classification algorithm with hybrid of ID3 and SVM,” *Journal of Intelligent & Fuzzy Systems*, vol. 24, pp. 555–561, Jan. 2013. Publisher : IOS Press. [95](#)
- [178] X. J. Chen, Z. G. Zhang, and Y. Tong, “An Improved ID3 Decision Tree Algorithm,” *Advanced Materials Research*, vol. 962–965, pp. 2842–2847, 2014. Conference Name : Resources and Sustainable Development III ISBN : 9783038351375 Publisher : Trans Tech Publications Ltd. [95](#)
- [179] B. Ding, Y. Zheng, and S. Zang, “A New Decision Tree Algorithm Based on Rough Set Theory,” in *2009 Asia-Pacific Conference on Information Processing*, vol. 2, pp. 326–329, July 2009. [95](#)
- [180] L. Zhu and Y. Yang, “Improvement of Decision Tree ID3 Algorithm,” in *Collaborate Computing : Networking, Applications and Worksharing* (S. Wang and A. Zhou, eds.), Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, (Cham), pp. 595–600, Springer International Publishing, 2017. [95](#)
- [181] N. Kaewrod and K. Jearanaitanakij, “Improving ID3 Algorithm by Ignoring Minor Instances,” in *2018 22nd International Computer Science and Engineering Conference (ICSEC)*, pp. 1–5, Nov. 2018. [95](#)

- [182] A. Rajeshkanna and K. Arunesh, "ID3 Decision Tree Classification : An Algorithmic Perspective based on Error rate," in *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pp. 787–790, July 2020. [95](#)
- [183] B. Lakshmi Devi, V. Varaswathi Bai, S. Ramasubbareddy, and K. Govinda, "Sentiment Analysis on Movie Reviews," in *Emerging Research in Data Engineering Systems and Computer Communications* (P. Venkata Krishna and M. S. Obaidat, eds.), Advances in Intelligent Systems and Computing, (Singapore), pp. 321–328, Springer, 2020. [96](#), [97](#), [117](#), [119](#), [120](#), [121](#), [122](#), [123](#), [124](#)
- [184] J. Guerreiro and P. Rita, "How to predict explicit recommendations in online reviews using text mining and sentiment analysis," *Journal of Hospitality and Tourism Management*, vol. 43, pp. 269–272, June 2020. [96](#), [97](#)
- [185] R. P. Mehta, M. A. Sanghvi, D. K. Shah, and A. Singh, "Sentiment Analysis of Tweets Using Supervised Learning Algorithms," in *First International Conference on Sustainable Technologies for Computational Intelligence* (A. K. Luhach, J. A. Kosa, R. C. Poonia, X.-Z. Gao, and D. Singh, eds.), Advances in Intelligent Systems and Computing, (Singapore), pp. 323–338, Springer, 2020. [96](#), [97](#)
- [186] J. Zhang, "Sentiment analysis of movie reviews in Chinese," p. 42. [96](#), [97](#)
- [187] A. López-Chau, D. Valle-Cruz, and R. Sandoval-Almazán, "Sentiment Analysis of Twitter Data Through Machine Learning Techniques," in *Software Engineering in the Era of Cloud Computing* (M. Ramachandran and Z. Mahmood, eds.), Computer Communications and Networks, pp. 185–209, Cham : Springer International Publishing, 2020. [96](#), [97](#)
- [188] H. A. Addi, R. Ezzahir, and A. Mahmoudi, "Three-level binary tree structure for sentiment classification in Arabic text," in *Proceedings of the 3rd International Conference on Networking, Information Systems & Security, NISS2020*, (New York, NY, USA), pp. 1–8, Association for Computing Machinery, Mar. 2020. [96](#), [97](#), [117](#), [118](#), [119](#), [120](#), [121](#), [122](#), [123](#), [124](#)
- [189] R. Patel and K. Passi, "Sentiment Analysis on Twitter Data of World Cup Soccer Tournament Using Machine Learning," *IoT*, vol. 1, pp. 218–239, Dec. 2020. Number : 2 Publisher : Multidisciplinary Digital Publishing Institute. [96](#), [97](#), [117](#), [118](#), [119](#), [120](#), [121](#), [122](#), [123](#), [124](#)
- [190] Y. Wang, Q. Chen, J. Shen, B. Hou, M. Ahmed, and Z. Li, "Aspect-level sentiment analysis based on gradual machine learning," *Knowledge-Based Systems*, vol. 212, p. 106509, Jan. 2021. [96](#), [97](#), [117](#), [119](#), [120](#), [121](#), [122](#), [123](#), [124](#)
- [191] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification," *Information Processing & Management*, vol. 50, pp. 104–112, Jan. 2014. [100](#)
- [192] M. Anandarajan, C. Hill, and T. Nolan, "Text Preprocessing," in *Practical Text Analytics : Maximizing the Value of Text Data* (M. Anandarajan, C. Hill, and T. Nolan, eds.), Advances in Analytics and Data Science, pp. 45–59, Cham : Springer International Publishing, 2019. [100](#)
- [193] O. Levy and Y. Goldberg, "Dependency-Based Word Embeddings," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, (Baltimore, Maryland), pp. 302–308, Association for Computational Linguistics, June 2014. [101](#), [183](#)

- [194] M. Ghiassi, J. Skinner, and D. Zimbra, "Twitter brand sentiment analysis : A hybrid system using n-gram analysis and dynamic artificial neural network," *Expert Systems with Applications*, vol. 40, pp. 6266–6282, Nov. 2013. [101](#), [183](#)
- [195] I. Santos, N. Nedjah, and L. d. M. Mourelle, "Sentiment analysis using convolutional neural network with fastText embeddings," in *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pp. 1–5, Nov. 2017. [101](#), [183](#)
- [196] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model : a statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, pp. 43–52, Dec. 2010. [101](#), [183](#)
- [197] L. V. D. Maaten, E. Postma, and J. V. D. Herik, "Dimensionality reduction : A comparative review," 2013. [101](#)
- [198] F. Es-Sabery and A. Hair, "Big Data Solutions Proposed for Cluster Computing Systems Challenges : A survey," in *Proceedings of the 3rd International Conference on Networking, Information Systems & Security, NISS2020*, (New York, NY, USA), pp. 1–7, Association for Computing Machinery, Mar. 2020. [104](#)
- [199] H. Kaur, "A LITERATURE REVIEW FROM 2011 TO 2014 ON STUDENT'S ACADEMIC PERFORMANCE PREDICTION AND ANALYSIS USING DECISION TREE ALGORITHM," 2018. [125](#)
- [200] F. Es-SABERY and A. Hair, "An Improved ID3 Classification Algorithm Based On Correlation Function and Weighted Attribute*," in *2019 International Conference on Intelligent Systems and Advanced Computing Sciences (ISACS)*, pp. 1–8, Dec. 2019. [125](#)
- [201] I. Abdallah, V. Dertimanis, H. Mylonas, K. Tatsis, E. Chatzi, N. Dervilis, K. Worden, and E. Maguire, "Fault diagnosis of wind turbine structures using decision tree learning algorithms with big data," in *Proceedings of the European Safety and Reliability Conference*, pp. 3053–3061, 2018. [125](#), [126](#)
- [202] S. Goswami, S. Chakraborty, S. Ghosh, A. Chakrabarti, and B. Chakraborty, "A review on application of data mining techniques to combat natural disasters," *Ain Shams Engineering Journal*, vol. 9, pp. 365–378, Sept. 2018. [125](#)
- [203] X. Wu, X. Zhu, G. Wu, and W. Ding, "Data mining with big data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, pp. 97–107, Jan. 2014. Conference Name : IEEE Transactions on Knowledge and Data Engineering. [126](#)
- [204] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, June 1965. [126](#), [127](#), [130](#), [192](#)
- [205] W. Pedrycz and F. Gomide, "An introduction to fuzzy sets : analysis and design," 1998. [126](#)
- [206] P. Ducange, F. Marcelloni, and A. Segatori, "A MapReduce-based fuzzy associative classifier for big data," in *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–8, Aug. 2015. [127](#)
- [207] M. Afzaal, M. Usman, A. C. M. Fong, S. Fong, and Y. Zhuang, "Fuzzy Aspect Based Opinion Classification System for Mining Tourist Reviews," Oct. 2016. ISSN : 1687-7101 Pages : e6965725 Publisher : Hindawi Volume : 2016. [127](#)
- [208] M. Abdul-Jaleel, Y. H. Ali, and N. J. Ibrahim, "Fuzzy logic and Genetic Algorithm based Text Classification Twitter," in *2019 2nd Scientific Conference of Computer Sciences (SCCS)*, pp. 93–98, Mar. 2019. [128](#)

- [209] M. C. Pegalajar, L. G. B. Ruiz, M. Sánchez-Marañón, and L. Mansilla, “A Munsell colour-based approach for soil classification using Fuzzy Logic and Artificial Neural Networks,” *Fuzzy Sets and Systems*, vol. 401, pp. 38–54, Dec. 2020. [128](#)
- [210] P. Melin, F. Olivas, O. Castillo, F. Valdez, J. Soria, and M. Valdez, “Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic,” *Expert Systems with Applications*, vol. 40, pp. 3196–3206, June 2013. [128](#)
- [211] E. Rubio, O. Castillo, F. Valdez, P. Melin, C. I. Gonzalez, and G. Martinez, “An Extension of the Fuzzy Possibilistic Clustering Algorithm Using Type-2 Fuzzy Logic Techniques,” Jan. 2017. ISSN : 1687-7101 Pages : e7094046 Publisher : Hindawi Volume : 2017. [128](#)
- [212] D. Sánchez, P. Melin, and O. Castillo, “Optimization of modular granular neural networks using a firefly algorithm for human recognition,” *Engineering Applications of Artificial Intelligence*, vol. 64, pp. 172–186, Sept. 2017. [128](#)
- [213] M. Nilashi, O. Ibrahim, M. Dalvi, H. Ahmadi, and L. Shahmoradi, “Accuracy Improvement for Diabetes Disease Classification : A Case on a Public Medical Dataset,” *Fuzzy Information and Engineering*, vol. 9, pp. 345–357, Sept. 2017. [128](#)
- [214] D. Bhamare and P. Suryawanshi, “Review on Reliable Pattern Recognition with Machine Learning Techniques,” *Fuzzy Information and Engineering*, vol. 10, pp. 362–377, July 2018. Publisher : Taylor & Francis _eprint : <https://doi.org/10.1080/16168658.2019.1611030>. [128](#)
- [215] A. Chatterjee, S. Mukherjee, and S. Kar, “A Rough Approximation of Fuzzy Soft Set-Based Decision-Making Approach in Supplier Selection Problem,” *Fuzzy Information and Engineering*, vol. 10, pp. 178–195, Apr. 2018. Publisher : Taylor & Francis _eprint : <https://doi.org/10.1080/16168658.2018.1517973>. [128](#)
- [216] R. C. Prati, F. Charte, and F. Herrera, “A first approach towards a fuzzy decision tree for multilabel classification,” in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pp. 1–6, July 2017. ISSN : 1558-4739. [129](#)
- [217] V. Levashenko and P. Martinová, “FUZZY DECISION TREE FOR PARALLEL PROCESSING SUPPORT,” 2005. [129](#), [131](#)
- [218] R. D. Suryawanshi and D. M. Thakore, “Decision Tree Classification Implementation with Fuzzy Logic,” 2012. [129](#), [131](#)
- [219] X. Wang, L. Dong, and J. Yan, “Maximum Ambiguity-Based Sample Selection in Fuzzy Decision Tree Induction,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, pp. 1491–1505, Aug. 2012. Conference Name : IEEE Transactions on Knowledge and Data Engineering. [129](#)
- [220] S. Sebastian and T. V. Ramakrishnan, “Multi-fuzzy Sets : An Extension of Fuzzy Sets,” *Fuzzy Information and Engineering*, vol. 3, pp. 35–43, Mar. 2011. Publisher : Taylor & Francis _eprint : <https://doi.org/10.1007/s12543-011-0064-y>. [129](#), [130](#)
- [221] Y. Bai and D. Wang, “Fundamentals of Fuzzy Logic Control — Fuzzy Sets, Fuzzy Rules and Defuzzifications,” in *Advanced Fuzzy Logic Technologies in Industrial Applications* (Y. Bai, H. Zhuang, and D. Wang, eds.), Advances in Industrial Control, pp. 17–36, London : Springer, 2006. [130](#), [155](#)
- [222] H. Liu and M. Cocea, “Fuzzy rule based systems for interpretable sentiment analysis,” pp. 129–136, Feb. 2017. [130](#), [131](#)

- [223] Wang Xizhao and Jiarong Hong, “On the handling of fuzziness for continuous-valued attributes in decision tree generation,” *Fuzzy Sets and Systems*, vol. 99, pp. 283–290, Nov. 1998. [130](#), [131](#)
- [224] F. Berzal, J.-C. Cubero, N. Marín, and D. Sánchez, “Numerical Attributes in Decision Trees : A Hierarchical Approach,” in *Advances in Intelligent Data Analysis V* (M. R. Berthold, H.-J. Lenz, E. Bradley, R. Kruse, and C. Borgelt, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 198–207, Springer, 2003. [130](#), [131](#)
- [225] *Classical Sets and Fuzzy Sets*, ch. 2, pp. 25–47. John Wiley and Sons, Ltd, 2010. [131](#), [192](#)
- [226] F. Afsari, M. Eftekhari, E. Eslami, and P.-Y. Woo, “Interpretability-based fuzzy decision tree classifier a hybrid of the subtractive clustering and the multi-objective evolutionary algorithm,” *Soft Computing*, vol. 17, pp. 1673–1686, Sept. 2013. [131](#)
- [227] H. Ishibuchi, “A fuzzy reasoning method for handling fuzzy rules with different specificity levels,” in *18th International Conference of the North American Fuzzy Information Processing Society - NAFIPS (Cat. No.99TH8397)*, pp. 110–114, June 1999. [140](#), [141](#)
- [228] A. Krouska, C. Troussas, and M. Virvou, “The effect of preprocessing techniques on Twitter sentiment analysis,” in *2016 7th International Conference on Information, Intelligence, Systems Applications (IISA)*, pp. 1–5, July 2016. [145](#)
- [229] P. Chandrasekar and K. Qian, “The Impact of Data Preprocessing on the Performance of a Naive Bayes Classifier,” in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, pp. 618–619, June 2016. ISSN : 0730-3157. [145](#)
- [230] K. W. Church, “Word2Vec,” *Natural Language Engineering*, vol. 23, pp. 155–162, Jan. 2017. Publisher : Cambridge University Press. [149](#)
- [231] Y. Seddiq, Y. A. Alotaibi, S.-A. Selouani, and A. H. Meftah, “Distinctive Phonetic Features Modeling and Extraction Using Deep Neural Networks,” *IEEE Access*, vol. 7, pp. 81382–81396, 2019. Conference Name : IEEE Access. [150](#)
- [232] S. Dara and P. Tumma, “Feature Extraction By Using Deep Learning : A Survey,” in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 1795–1801, Mar. 2018. [150](#)
- [233] M. Jogin, Mohana, M. S. Madhulika, G. D. Divya, R. K. Meghana, and S. Apoorva, “Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning,” in *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pp. 2319–2323, May 2018. [150](#)
- [234] M. Farooq and E. Sazonov, “Feature Extraction Using Deep Learning for Food Type Recognition,” in *Bioinformatics and Biomedical Engineering* (I. Rojas and F. Ortuño, eds.), Lecture Notes in Computer Science, (Cham), pp. 464–472, Springer International Publishing, 2017. [150](#)
- [235] I. S. Damanik, A. P. Windarto, A. Wanto, Poningsih, S. R. Andani, and W. Saputra, “Decision Tree Optimization in C4.5 Algorithm Using Genetic Algorithm,” *Journal of Physics : Conference Series*, vol. 1255, p. 012012, Aug. 2019. Publisher : IOP Publishing. [155](#), [161](#), [162](#)

- [236] A. Cherfi, K. Noura, and A. Ferchichi, “Very Fast C4.5 Decision Tree Algorithm,” *Applied Artificial Intelligence*, vol. 32, pp. 119–137, Apr. 2018. Publisher : Taylor & Francis _eprint : <https://doi.org/10.1080/08839514.2018.1447479>. 155, 162
- [237] J. Lee, “AUC4.5 : AUC-Based C4.5 Decision Tree Algorithm for Imbalanced Data Classification,” *IEEE Access*, vol. 7, pp. 106034–106042, 2019. Conference Name : IEEE Access. 155, 162
- [238] F. Xu, Z. Pan, and R. Xia, “E-commerce product review sentiment classification based on a naïve Bayes continuous learning framework,” *Information Processing & Management*, vol. 57, p. 102221, Sept. 2020. 162, 163, 165, 166, 167, 169, 170, 172
- [239] S. Liao, J. Wang, R. Yu, K. Sato, and Z. Cheng, “CNN for situations understanding based on sentiment analysis of twitter data,” *Procedia Computer Science*, vol. 111, pp. 376–381, Jan. 2017. 162, 163, 165, 166, 167, 168, 169, 170, 172
- [240] S. U. Maheswari and S. S. Dhenakaran, “Aspect based Fuzzy Logic Sentiment Analysis on Social Media Big Data,” in *2020 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0971–0975, July 2020. 162, 163, 165, 166, 167, 168, 169, 170, 172
- [241] F. Es-Sabery, A. Hair, J. Qadir, B. Sainz-De-Abajo, B. García-Zapirain, and I. D. L. Torre-Díez, “Sentence-Level Classification Using Parallel Fuzzy Deep Learning Classifier,” *IEEE Access*, vol. 9, pp. 17943–17985, 2021. Conference Name : IEEE Access. 164
- [242] M. Anagha, R. R. Kumar, K. Sreetha, and P. C. R. Raj, “Fuzzy logic based hybrid approach for sentiment analysis of Malayalam movie reviews,” in *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, pp. 1–4, Feb. 2015. 174
- [243] J. B. Sathe and M. P. Mali, “A hybrid Sentiment Classification method using Neural Network and Fuzzy Logic,” in *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, pp. 93–96, Jan. 2017. 174
- [244] M. Biltawi, W. Etaiwi, S. Tedmori, and A. Shaout, “Fuzzy Based Sentiment Classification in the Arabic Language,” in *Intelligent Systems and Applications* (K. Arai, S. Kapoor, and R. Bhatia, eds.), Advances in Intelligent Systems and Computing, (Cham), pp. 579–591, Springer International Publishing, 2019. 174
- [245] S. Modha, T. Mandl, P. Majumder, and D. Patel, “Tracking Hate in Social Media : Evaluation, Challenges and Approaches,” *SN Computer Science*, vol. 1, p. 105, Mar. 2020. 175
- [246] B. Lakshmi Devi, V. Varaswathi Bai, S. Ramasubbareddy, and K. Govinda, “Sentiment Analysis on Movie Reviews,” in *Emerging Research in Data Engineering Systems and Computer Communications* (P. Venkata Krishna and M. S. Obaidat, eds.), Advances in Intelligent Systems and Computing, (Singapore), pp. 321–328, Springer, 2020. 175
- [247] R. Bose, R. K. Dey, S. Roy, and D. Sarddar, “Sentiment Analysis on Online Product Reviews,” in *Information and Communication Technology for Sustainable Development* (M. Tuba, S. Akashe, and A. Joshi, eds.), Advances in Intelligent Systems and Computing, (Singapore), pp. 559–569, Springer, 2020. 175
- [248] A.-S. Uban and L. P. Dinu, “On Transfer Learning for Detecting Abusive Language Online,” in *Advances in Computational Intelligence* (I. Rojas, G. Joya, and

- A. Catala, eds.), *Lecture Notes in Computer Science*, (Cham), pp. 688–700, Springer International Publishing, 2019. 175
- [249] H. Rosa, N. Pereira, R. Ribeiro, P. C. Ferreira, J. P. Carvalho, S. Oliveira, L. Coheur, P. Paulino, A. M. Veiga Simão, and I. Trancoso, “Automatic cyberbullying detection : A systematic review,” *Computers in Human Behavior*, vol. 93, pp. 333–345, Apr. 2019. 175
- [250] D. Sharma, M. Sabharwal, V. Goyal, and M. Vij, “Sentiment Analysis Techniques for Social Media Data : A Review,” in *First International Conference on Sustainable Technologies for Computational Intelligence* (A. K. Luhach, J. A. Kosa, R. C. Poonia, X.-Z. Gao, and D. Singh, eds.), *Advances in Intelligent Systems and Computing*, (Singapore), pp. 75–90, Springer, 2020. 175
- [251] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari, “A State-of-the-Art Survey on Deep Learning Theory and Architectures,” *Electronics*, vol. 8, p. 292, Mar. 2019. Number : 3 Publisher : Multidisciplinary Digital Publishing Institute. 175
- [252] N. Jin, J. Wu, X. Ma, K. Yan, and Y. Mo, “Multi-Task Learning Model Based on Multi-Scale CNN and LSTM for Sentiment Classification,” *IEEE Access*, vol. 8, pp. 77060–77072, 2020. Conference Name : IEEE Access. 177, 223, 224, 226, 227, 229
- [253] F. Almeida and G. Xexéo, “Word Embeddings : A Survey,” *arXiv :1901.09069 [cs, stat]*, Jan. 2019. arXiv : 1901.09069. 177
- [254] Y. Lan, Y. Hao, K. Xia, B. Qian, and C. Li, “Stacked Residual Recurrent Neural Networks With Cross-Layer Attention for Text Classification,” *IEEE Access*, vol. 8, pp. 70401–70410, 2020. Conference Name : IEEE Access. 178, 224, 226, 227, 229
- [255] Y. Lin, J. Li, L. Yang, K. Xu, and H. Lin, “Sentiment Analysis With Comparison Enhanced Deep Neural Network,” *IEEE Access*, vol. 8, pp. 78378–78384, 2020. Conference Name : IEEE Access. 178, 224, 226, 227, 229
- [256] G. Liu, X. Xu, B. Deng, S. Chen, and L. Li, “A hybrid method for bilingual text sentiment classification based on deep learning,” in *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pp. 93–98, May 2016. 178, 224, 226, 227, 229
- [257] A. Jain and V. Jain, “Sentiment classification of twitter data belonging to renewable energy using machine learning,” *Journal of Information and Optimization Sciences*, vol. 40, pp. 521–533, Feb. 2019. Publisher : Taylor & Francis _eprint : <https://doi.org/10.1080/02522667.2019.1582873>. 178
- [258] A. Jain and V. Jain, “Sentiment classification of twitter data belonging to renewable energy using machine learning,” *Journal of Information and Optimization Sciences*, vol. 40, pp. 521–533, Feb. 2019. Publisher : Taylor & Francis _eprint : <https://doi.org/10.1080/02522667.2019.1582873>. 178
- [259] F. Z. Xing, E. Cambria, and R. E. Welsch, “Intelligent Asset Allocation via Market Sentiment Views,” *IEEE Computational Intelligence Magazine*, vol. 13, pp. 25–34, Nov. 2018. Conference Name : IEEE Computational Intelligence Magazine. 179, 224, 226, 227, 229

- [260] J.-n. González, F. Pla, and L.-F. Hurtado, “ELiRF-UPV at SemEval-2017 Task 4 : Sentiment Analysis using Deep Learning,” in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, (Vancouver, Canada), pp. 723–727, Association for Computational Linguistics, Aug. 2017. [179](#)
- [261] K. Wu, M. Zhou, X. S. Lu, and L. Huang, “A fuzzy logic-based text classification method for social media data,” in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1942–1947, Oct. 2017. [179](#)
- [262] S. Vashishtha and S. Susan, “Fuzzy rule based unsupervised sentiment analysis from social media posts,” *Expert Systems with Applications*, vol. 138, p. 112834, Dec. 2019. [179](#)
- [263] M. Abdul-Jaleel, Y. H. Ali, and N. J. Ibrahim, “Fuzzy logic and Genetic Algorithm based Text Classification Twitter,” in *2019 2nd Scientific Conference of Computer Sciences (SCCS)*, pp. 93–98, Mar. 2019. [180](#)
- [264] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” *arXiv :1310.4546 [cs, stat]*, Oct. 2013. arXiv : 1310.4546. [183](#)
- [265] Y. Li and B. Shen, “Research on sentiment analysis of microblogging based on LSA and TF-IDF,” in *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, pp. 2584–2588, Dec. 2017. [183](#)
- [266] F. Es-Sabery and A. Hair, “Big Data Solutions Proposed for Cluster Computing Systems Challenges : A survey,” in *Proceedings of the 3rd International Conference on Networking, Information Systems & Security, NISS2020*, (New York, NY, USA), pp. 1–7, Association for Computing Machinery, Mar. 2020. [200](#), [202](#)