جامعة سيدي محمد بن عبد الله بفاس
UNIVERSITE SIDI MOHAMED BEN ABDALLAH DE FES
كلية العلوم ظهر المهراز فاس
FACULTE DES SCIENCES DHAR EL MEHRAZ FES

N° d'Ordre : **07 /2021**

Formation Doctorale : Sciences et Technologies de l'Information et de la Communication -STIC

Spécialité : Informatique

Laboratoire : Laboratoire d'Informatique, Signaux, Automatique et cognitivisme -LISAC

THESE DE DOCTORAT

Présentée par

EL-ALAMI Fatima-zahra

Deep Learning Models for Arabic Text Categorization: Application to Offensive Language Detection

Soutenue le 23/ 01/ 2021 devant le jury composé de :

Pr. SATORI khalid	***Faculté des Sciences Dhar El Mahraz - Fès***	***Président***
Pr. FRIKH Bouchra	***Ecole Supérieure de Technologie- Fès***	***Rapporteur***
Pr. DRISSI EL MALIANI Ahmed	***Faculté des Sciences - Rabat***	***Rapporteur***
Pr. GAHI Youssef	***Ecole Nationale des Sciences Appliquées – Kénitra***	***Rapporteur***
Pr. AOURAGH Si Lhoussain	***Faculté des Sciences Juridiques, Economiques et Sociales – Salé***	***Examinateur***
Pr. ZINEDINE Ahmed	***Faculté des Sciences Dhar El Mahraz - Fès***	**Examinateur**
Pr. OUATIK EL ALAOUI Saïd	***Ecole Nationale des Sciences Appliquées – Kénitra***	**Co-Directeur de thèse**
Pr. EN NAHNAHI Noureddine	***Faculté des Sciences Dhar El Mahraz – Fès***	**Directeur de thèse**

To my father who never saw this adventure

ACKNOWLEDGEMENT

First, I thank Allah the Almighty for giving me the strength and the ability to complete this thesis.

There have been many people that have contributed to this thesis work one way or another. When I think of all these people without whom this work probably could not have been completed, I tell myself that it is not for nothing that this page precedes any thesis presentation. Let these lines serve as a sign of my gratitude to all of them.

In the first place, I would like to express my deepest thanks to my supervisors, **Mr. Said OUATIK EL ALAOUI**, Professor at Ibn Tofail University, National School of Applied Sciences in Kenitra and **Mr. Noureddine EN NAHNAHI** Professor at Sidi Mohamed Ben Abdellah University in Fez, for offering me the opportunity of pursuing this Ph.D., for choosing the subject and the scientific orientation of this thesis work, for their confidence, for providing me the freedom to pursue my research, and for the quality of their supervision. I greatly appreciate the time they made available in their busy schedules. Despite their responsibilities, they were able to be present when needed. I would also like to thank them for their personal and human qualities, which have also contributed a lot to realize this work. Doing research is one of the most enriching activities I have done in my life. However, it is also a challenging and demanding endeavor that, without proper guidance, can lead to frustration and discouragement. I want to thank them for always having their doors open for lively discussion and their valuable comments and suggestions during the realization of this thesis work. Their enthusiasm motivated me a lot.

I am grateful and honored to the jury members for having accepted with great kindness to sit among this jury.

A deep respect and special thanks to **Mr. EL HASSOUNI Mohammed**, the vice dean of research, cooperation and partnerships of the FSDM faculty.

I wish to extend my profound sense of gratitude to **my mother and my sisters** for all the sacrifices they made during my research and provide me with moral support. I would also like to thank my beloved **husband**. Thank you for supporting me for everything, and especially I can't thank you enough for encouraging me throughout this experience.

ABSTRACT

Currently, access to Arabic textual information constitutes a significant and challenging problem for the natural language processing research community. On the one hand, the amount of available information in electronic format is growing enormously and rapidly. On the other hand, the Arabic language's richness and complex morphological structure make text mining an arduous task. Consequently, the development of automatic text categorization systems has become essential in order to explore relevant information.

Our thesis improves Arabic text categorization systems by focusing both on text representation and deep learning methods. Indeed, by exploiting distributed word representation, transfer learning, and deep learning architectures, we overcome classical systems' shortcomings, such as the lack of semantics and high dimensionality of representation space.

In this thesis work, we propose four contributions concerning Arabic text representation and categorization. The first consists of representing Arabic documents using a deep Autoencoder, allowing an appropriate textual representation with a reduced dimension. Our method also considers the semantic aspect of documents by combining both implicit and explicit semantics. In the second contribution, we explore several deep neural models and their combination. Besides, we integrate a retrofitting process to further capture the text semantics. The third contribution adopts transfer learning based on the AraBERT, Arabic Bidirectional Encoder Representations from Transformers. The first variant consists of fine-tuning the parameters of the AraBERT model on the categorization task. The second variant makes use of AraBERT as a feature extraction model. To show our approaches' applicative interest, we propose, as the last contribution, a multilingual offensive language detection application in social networks. Different approaches, such as the joint multilingual approach and that are based on translation, are explored.

The experimental validation of all the contributions is carried out on the OSAC and SOLID datasets.

Keywords: *Categorization*, *Text Representation*, *Arabic Natural Language Processing*, *Machine and Deep Learning*, *Word and Document Embeddings*, *Neural Networks*, *Transfert Learning*, *Social Media*.

RESUMÉ

Actuellement, l'accès à l'information textuelle en langue arabe constitue un problème majeur et représente un défi pour la communauté des chercheurs dans le domaine du traitement automatique du langage naturel. D'une part, la quantité d'information disponible sous format électronique croît de façon extrêmement rapide, et d'autre part, la richesse et la structure morphologique complexe de la langue arabe rendent la fouille de textes une tâche extrêmement ardue. Par conséquent, le développement des systèmes de catégorisation automatique est devenu incontournable afin de retrouver l'information pertinente.

Notre thèse s'inscrit dans le cadre de l'amélioration des systèmes de catégorisation des documents textes en langue arabe en se focalisant à la fois sur le processus de représentation de textes et les méthodes d'apprentissage profond. En effet, en exploitant la représentation distribuée des mots, l'apprentissage par transfert et les architectures d'apprentissage en profondeur, nous remédions aux limitations des systèmes classiques tels que le manque de sémantique et la dimensionnalité élevée de l'espace de représentation.

Dans ce travail de thèse, nous apportons quatre contributions au niveau de la représentation et la catégorisation de textes en langue arabe. La première consiste à représenter les documents arabes via des vecteurs distribués des documents par l'usage d'un auto-encodeur profond permettant une représentation textuelle appropriée à dimension réduite. De plus, notre méthode prend en considération l'aspect sémantique contenu dans les documents en combinant à la fois la sémantique implicite et explicite. Dans la deuxième contribution, nous explorons plusieurs modèles neuronaux profonds ainsi que leur combinaison. De plus, nous intégrons un processus de Retrofitting pour capturer davantage la sémantique du texte. La troisième contribution adopte l'apprentissage par transfert basé sur le modèle AraBERT, *Arabic Bidirectional Encoder Representations from Transformers*. Une première variante consiste à affiner les paramètres du modèle AraBERT sur la tâche de catégorisation. La deuxième variante se base sur AraBERT comme un modèle d'extraction de caractéristiques. Pour montrer l'intérêt applicatif de nos approches, nous proposons, comme dernière contribution, une application de détection du langage offensif dans un contexte multilingue des réseaux sociaux. Différentes approches, telles que l'approche multilingue conjointe et celle basée sur la traduction sont explorées.

La validation expérimentale de l'ensemble des contributions apportées est menée sur les collections OSAC et SOLID.

Mots clés : *Catégorisation*, *Représentation de Textes*, *Traitement Automatique de la Langue Arabe*, *Apprentissage Automatique et Profond*, *Représentations Distribuées des Mots et des documents*, *Réseaux Neuronaux*, *Apprentissage par Transfert*, *Réseaux Sociaux* .

TABLE OF CONTENTS

ACKNOWLEDGEMENT ii
ABSTRACT iii
RESUMÉ iv
LIST OF FIGURES ix
LIST OF TABLES xi
LIST OF ABBREVIATIONS xiii
General Introduction xiii
0.1 Context 1
0.2 Motivations 2
0.3 Contributions 2
0.4 Content Overview 4

1 Arabic Natural Language Processing **5**
1.1 Introduction 5
1.2 Peculiarities of the Arabic Language 5
1.2.1 The Arabic Language 5
1.2.2 Character Sets 6
1.3 Arabic Morphology 7
1.3.1 Arabic as an Agglutinative and a Pro-Drop Language 8
1.4 Derivational Morphology 9
1.5 Inflectional Morphology 11
1.5.1 Verbal Morphology 11
1.5.2 Nominal Morphology 11
1.6 Arabic Syntax 12
1.6.1 Verbal Sentences 12
1.6.2 Nominal Sentences 13
1.6.3 Nominal Phrase Structure 13
1.7 Arabic Natural Language Processing 14
1.7.1 Tokenization 14
1.7.2 Normalization 15
1.7.3 Stop-word Removal 15
1.7.4 Stemming 15
1.7.5 Part-Of-Speech Tagging 16
1.8 Conclusion 18

2 Text Categorization and Deep Learning **20**
2.1 Introduction 20
2.2 Text Categorization Paradigms 21
2.3 A Mathematical Definition of Text Categorization Task 22
2.4 Text Categorization System Architecture 23

2.5 Text Representation . . . 24
2.5.1 Bag-Of-Words . . . 25
2.5.2 Term Frequency and Inverse Document Frequency . . . 25
2.5.3 Latent Semantic Analysis . . . 26
2.5.4 Probabilistic Latent Semantic Analysis . . . 27
2.5.5 Latent Dirichlet Allocation . . . 28
2.5.6 Word2vec . . . 28
2.5.7 Glove . . . 30
2.5.8 Doc2vec . . . 30
2.5.9 Contextualized Word Representations . . . 31
2.6 Feature Selection . . . 34
2.6.1 Pruning . . . 34
2.6.2 Chi-Square . . . 35
2.6.3 Mutual Information . . . 35
2.6.4 Information Gain . . . 35
2.6.5 Document Frequency Thresholding . . . 36
2.6.6 Variance Threshold . . . 36
2.7 Classifiers Overview . . . 36
2.7.1 Logistic Regression . . . 36
2.7.2 Decision Trees . . . 37
2.7.3 Random Forest . . . 37
2.7.4 Naive Bayes . . . 37
2.7.5 Support Vector Machines . . . 38
2.7.6 Boosting Algorithms . . . 38
2.8 Evaluation Metrics . . . 39
2.8.1 Macro-averaging . . . 40
2.8.2 Micro-averaging . . . 41
2.8.3 Weighted-averaging . . . 41
2.9 Deep Learning for Text Categorization . . . 41
2.9.1 Deep Learning Definition . . . 41
2.9.2 Training Artificial Neural Networks . . . 42
2.9.3 History of Deep Neural Networks for Text Categorization . . . 42
2.10 Conclusion . . . 45

3 A Deep Autoencoder-Based Representation for Arabic Text Categorization 47
3.1 Introduction . . . 47
3.2 Motivations . . . 47
3.3 Stacked Autoencoder with Restricted Boltzmann Machines Pretraining . . . 48
3.3.1 Restricted Boltzmann Machines . . . 48
3.3.2 RBMs Autoencoder Architecture . . . 50
3.4 Arabic WordNet . . . 51
3.5 Methodology . . . 52
3.5.1 Preprocessing and Feature Selection Module . . . 52
3.5.2 Representation Module . . . 53
3.5.3 Categorization Module . . . 54
3.6 Experimental Results . . . 57
3.6.1 The Corpus and Performance Measures . . . 57

3.6.2 Discussion . . . 57
3.7 Conclusion . . . 59

4 Deep Neural Models and Retrofitting for Arabic Text Categorization 62
4.1 Introduction . . . 62
4.2 Motivations . . . 62
4.3 Background . . . 64
4.3.1 Convolutional Neural Network . . . 64
4.3.2 Recurrent Neural Network . . . 64
4.4 Methodology . . . 67
4.4.1 Text Preprocessing . . . 67
4.4.2 Word Embedding Model with Subword Information . . . 68
4.4.3 Retrofitted Word Embedding with Semantic Lexicon . . . 69
4.4.4 Deep Neural Network Models . . . 70
4.5 Experimental Results . . . 72
4.5.1 Experiment Setting . . . 72
4.5.2 Comparison with Arabic Text Categorization Methods . . . 73
4.5.3 Results and Discussion . . . 74
4.6 Conclusion . . . 76

5 Contextual Semantic Embeddings based on Fine-tuned AraBERT Model for Arabic Text Multi-class Categorization 78
5.1 Introduction . . . 78
5.2 Motivations . . . 78
5.3 Transformers . . . 79
5.3.1 Scaled Dot-product Attention . . . 80
5.3.2 Multi-head Attention . . . 80
5.3.3 Encoder . . . 81
5.3.4 Decoder . . . 82
5.3.5 Training . . . 82
5.4 BERT . . . 82
5.4.1 Masked Language Model . . . 82
5.4.2 Next Sentence Prediction . . . 83
5.5 AraBERT for Text Categorization . . . 83
5.5.1 Text Preprocessing . . . 83
5.5.2 AraBERT Fine-tuning . . . 84
5.5.3 AraBERT for Feature Extraction . . . 86
5.6 Experiments And Results . . . 88
5.6.1 Dataset . . . 88
5.6.2 Discussion . . . 89
5.7 Conclusion . . . 92

6 A Multilingual Offensive Language Detection Method based on Transfer Learning from Transformer Fine-tuning Model 94
6.1 Introduction . . . 94
6.2 Motivations . . . 94
6.3 Multilingual Offensive Language Detection . . . 95
6.3.1 Offensive Language Detection . . . 95
6.3.2 Multilingual Text Categorization . . . 96

6.4 Methodology . . . 98
6.4.1 Preprocessing . . . 98
6.4.2 Translation . . . 100
6.4.3 BERT Tokenization . . . 100
6.4.4 Tweet Representation . . . 100
6.4.5 Text Classification . . . 100
6.5 Experimental Results . . . 101
6.5.1 The Multilingual Corpora . . . 101
6.5.2 Experimental Setup . . . 101
6.5.3 Comparison Methods . . . 101
6.5.4 Joint-Multilingual Classification Results . . . 102
6.5.5 Joint-Translated Monolingual Results . . . 104
6.6 Conclusion . . . 106
Conclusion and Future Work . . . 108

Appendices

Appendix A Résumé Détaillé de la Thèse en Français **110**
A.1 Introduction . . . 110
A.2 Méthodes Proposées . . . 112
A.2.1 Autoencodeur Profond pour la Catégorisation de Textes en Langue Arabe . . . 113
A.2.2 Modèles Neuronaux Profonds et rétrofit pour la Catégorisation de Texte en Langue Arabe . . . 117
A.2.3 Embeddings Sémantiques et Contextuels basés sur le Modèle AraBERT Affiné pour la Catégorisation Multi-classes de Textes Arabe . . . 119
A.2.4 Une Méthode de Détection du Langage Offensif dans un Contexte Multilingue en se Basant sur l'Apprentissage par Transfert . . . 121
A.3 Conclusion et Perspectives . . . 124
LIST OF PUBLICATIONS . . . 127
REFERENCES . . . 127

LIST OF FIGURES

1.1 Example of POS tagging using Farasa Online Web Demo 19
2.1 Paradigms in text categorization: binary, multi-class and multi-label . . 22
2.2 The text classification system architecture 24
2.3 The CBOW and Skip-gram models architectures [87] 29
2.4 PV-DM and PV-DBOW model architectures [77] 31
2.5 Two possible margins linearly separate positive and negative samples . . 38
3.1 Restricted Boltzmann Machine 50
3.2 Left panel: Pretraining by learning a stack of Restricted Boltzmann Machines. Right panel: Fine-tuning 51
3.3 The proposed Arabic text categorization system architecture 53
3.4 Building feature representation space for texts 54
3.5 Input and output vectors of our representation module 55
3.6 Multi-layer perceptron architecture 56
3.7 Performance analysis of khi2 vs. VT. (a) Using the Bag-of-Concepts representation. (b) Using the Bag-of-Words representation 60
3.8 Performance analysis of Bag-Of-Words representation vs. the Bag-Of-Concepts representation. (a) Performance representation based on Chi-square. (b) Performance representation using VT 61
4.1 Illustration of a CNN architecture for sentence classification [144] . . . 65
4.2 Recurrent Neural Network architecture [96] 66
4.3 Long Short Term Memory Network Memory Block [96] 67
4.4 The flowchart of the proposed Arabic text categorization method using deep neural models 68
4.5 An example of retrofitted Arabic words 69
4.6 The Proposed CNN Model 71
4.7 The Proposed LSTM Model 72
4.8 The Proposed CNN-LSTM Model 73
4.9 Loss functions for CNN-LSTM model 76
4.10 Training and test Loss with retrofitting for the different models 77
5.1 The transformer - model architecture [128] 81
5.2 The overall Arabic text categorization system architecture 84
5.3 The flowchart of fine-tuned AraBERT for ATC 85
5.4 Feature-based approach with AraBERT 87
5.5 Categories distribution in OSAC according to document number 88
5.6 Accuracy Result of three versions of fine-tuned BERT models on OSAC dataset 89
5.7 Performance Result of three versions of fine-tuned BERT models on OSAC dataset 92
6.1 Multiple monolingual classification 96
6.2 Joint-multilingual classification 97

6.3 Joint-translated monolingual classification 98
6.4 The multilingual offensive language detection method flowchart 99
6.5 Emojis preprocessing in Arabic tweets 99
6.6 Performance results of the joint-translated monolingual method on tweet dataset 103
6.7 Accuracy results of the joint-translated method using English language . 105
6.8 Joint-translated monolingual results vs. Join-multilingual classification . 105
A.1 L'architecture du système de catégorisation de textes en utilisant l'Autoencodeur 113
A.2 Panneau de gauche: entraînement par une pile de machines Boltzmann restreintes. Panneau de droite: Ajustement 114
A.3 Performances de khi2 vs. VT. (a) Utilisation de la représentation des sacs de concepts. (b) Utilisation de la représentation des sacs de mots . 116
A.4 Performances de trois versions de modèles BERT affinés sur l'ensemble de données OSAC 120
A.5 Organigramme de la méthode de la détection du langage offensif dans un contexte multilingue 122
A.6 Résultats de performance pour la classification traduite conjointe en anglais 124

LIST OF TABLES

1.1 Different shapes of Arabic letters with their different shapes. Isolated, "I"; as a first letter, "F"; in the middle of the word, "M"; or as a last letter, "L". The IPA column shows their international phonetic representation [94] 8
1.2 Arabic verb subject affixations [54, 53] 11
2.1 Comparaison between state-of-the-art text representation methods . . . 32
2.2 Comparaison between contexrual embbeding representations 34
2.3 Contingency table 39
3.1 Arabic WordNet relation classification [20] 52
3.2 OSAc Corpus 57
3.3 Arabic categorization performance results using different text representations 58
3.4 Comparison between Autoencoder, MLP and SVM classifiers using Bag-Of-Concepts and Chi-square 59
4.1 Comparisons of our models with state-of-the-art ones on OSAc dataset . 74
4.2 Arabic text categorization accuracy results for different models without and with Retrofitting 75
5.1 Example of tokenization process with the two versions of BERT 84
5.2 OSAC dataset statistics 87
5.3 Performance results of the fine-tuned AraBERT and baseline models on OSAc 90
5.4 Performance results of AraBERT vs. mBERT as feature extraction models using different classifiers on OSAc 92
6.1 Performance results of joint-multilingual classification method on the bilingual tweets dataset 103
6.2 Performance results of joint-translated classification using the Arabic language 104
A.1 Résultats de catégorisation de textes en langue arabe en utilisant différentes représentations textuelles 115
A.2 Comparaison entre les classificateurs Autoencodeur, Perceptron multicouches et SVM en utilisant les sacs des concepts et chi-carré 117
A.3 Comparaisons de modèles proposés avec des modèles de l'état de l'art sur OSAc 118
A.4 Résultats de la catégorisation pour différents modèles proposés sans et avec Retrofitting 119
A.5 Performances des modèles AraBERT en comparaison avec l'état de l'art 120
A.6 Performances des modèles AraBERT par rapport à mBERT en tant que modèles d'extraction de caractéristiques en utilisant différents classificateurs sur OSAc 121

A.7 Résultats de performance de la méthode de classification conjointe multilingue 123
A.8 Résultats de performance de la catégorisation en se basant sur la méthode de la traduction conjointe en langue arabe 123

LIST OF ABBREVIATIONS

ANLP	Arabic Natural Language Processing
ANN	Artificial Neural Networks
ATC	Arabic Text Categorization
AWN	Arabic WordNet
Albert	Lite bert for self-supervised learning of language representations
AraBERT	Arabic Bidirectional Encoder Representations from Transformers
BERT	Bidirectional Encoder Representations from Transformers
BOC	Bag-Of-Concepts
BOW	Bag-Of-Words
BiGRU	Bidirectional Gated Recurrent Unit
BiLSTM	Bidirectional Long Short-Term Memory Networks
DL	Deep Learning
ELMo	Embeddings from Language Models
FS	Feature Selection
GRU	Gated Recurrent Unit
Glove	Global Vectors for Word Representation
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
LSTM	Long Short-Term Memory Networks
ML	Machine Learning
MLM	Masked Language Modelling
MLP	Multi-Layer Perceptron
MTC	Multilingual Text Categorization
NLP	Natural Language Processing
NSP	Next Sentence Prediction
PLSA	Probabilistic Latent Semantic Analysis
POS	Part-Of-Speech
PV-DBOW	Distributed Bag of Words version of Paragraph Vector
PV-DM	Distributed Memory version of Paragraph Vector
RBM	Restricted Boltzmann Machine
RNN	Recurrent Neural Network
RoBERTa	Robustly optimized BERT pretraining Approach
SVM	Support Vector Machines
TC	Text Categorization
TF-IDF	Term Frequency-Inverse Document Frequency
VT	Variance Threshold

General Introduction

0.1 Context

Every day, an enormous amount of textual data is generated on the Web, and it is expected to grow. These data contain knowledge that is virtually important to several fields such as business, science, and industry. Therefore, the ability to automatically and efficiently extract interesting knowledge from these data is becoming crucial.
Text mining is a subfield of data mining, which is the process of deriving high-quality information from text. It uses methods that comprise different fields, such as statistics, machine learning, and natural language processing. Text mining [61] process usually involves several phases. The first phase is preparing textual data for the training phase. The aim is to present the textual data in a format compatible with the learning model then apply different methods that improve the text mining efficiency. For instance, stemming and feature selection techniques reduce the number of features and decrease the computational cost. Text preprocessing includes many different techniques such as tokenization, stop-words removal, stemming and lemmatization, and Parts-Of-Speech tagging. Text representation is a vital step in text mining in which text documents are represented according to the learning model requirements. Several techniques were proposed to represent text. For instance, the Bag-Of-Words model and the n-gram model. Besides, term-weighting is also an important step that assigns scores to terms using different techniques such as Term Frequency-Inverse Document Frequency and its variants or more recent methods like word embeddings, FastText and contextual word embeddings. Feature selection is another crucial technique that tends to select a set of the most useful terms. The FS techniques are adopted to simplify model construction and reduce training times and overfitting. All the methods mentioned above are presented in details in chapter 3.
The second phase focuses on the mining models that depend on the application. Mining tasks fall into two groups: supervised learning and unsupervised learning tasks. Each group has a different type of algorithms. Finally, in the last phase, models are evaluated to assess their performance.
Typical text mining tasks include text categorization, clustering, entity extraction, doc-

ument retrieval, sentiment analysis, and document summarization. In this work, we are interested in text categorization that is a supervised learning task aiming to automatically assign a predefined category to a text document. This task is achieved by constructing a model from a set of text documents. There exist three types of classification: binary classification, multi-class classification, and multi-label classification.

0.2 Motivations

The growing share of textual collections in Arabic on the internet makes the development of tools allowing the best organization and management of this information a must. Indeed, text categorization is an important task in several applications, such as information retrieval and question/answering. With the recent breakthroughs in NLP, Arabic Text Categorization has become an active research area since the Arabic language presents various challenges because of its complex morphology and structure. The Inflectional and derivational nature of Arabic makes its analysis more complicated. Text representation constitutes a crucial phase and significantly affects the ATC performance. A wide range of Arabic text representation methods has been proposed. For instance, the traditional text modeling based on the BOW representation [90, 58] achieved state-of-the-art performances in the NLP area. However, it suffers from the dimensionality curse and lack of semantic relationships between different text units. PLSA [23] and LDA [56] are topic modeling methods which are, generally, applied to select more discriminative features but they suffer from inference problem. Pre-trained word embedding models such as Word2vec [87], Doc2vec [77] and Glove [102] have been emerged as an effective representation models for text classification. These pre-trained models on a large corpus aim to build continuous distributed representations for texts and represent words as dense vectors in low-dimensional vector space. However, the former representations suffer from limitations, including the ignorance of the information embedded in lexical databases, the relationships among multiple words, and the meaning of full sentences within the text are not considered.

0.3 Contributions

To tackle the issues above related to Arabic text categorization, including the semantic deficiency, the high dimensionality of feature space, and the complex morphology of Arabic, we propose four main contributions that can be summarized as follows:

1. We propose a deep Autoencoder-based method for Arabic text representation and categorization. This method uses the Bag-Of-Concepts representation to incorpo-

rate explicit semantic relying on Arabic WordNet and exploits Chi-Square measure to select the most informative features. The deep Autoencoder training is performed using successive stacks of Restricted Boltzmann Machines applied to text vectors to produce high-level representations. The learned features are fed to another deep Autoencoder for categorization.
More details on deep Autoencoder-based representation for Arabic text categorization are in chapter 3.

2. We propose to use deep neural networks for Arabic text categorization. Within this context, different models are proposed, including LSTM, CNN, CNN-LSTM, and LSTM-CNN. The design strategy we propose heavily relies on text-domain knowledge. We addressed both the morphological variety of Arabic words and the sparsity of data representation problems. Our method also investigates the benefit of combining implicit and explicit semantics by integrating retrofitting as a post-processing step to improve word representations.
The employed method and obtained results are described in chapter 4.

3. We proposed a contextual semantic embeddings-based method for enhancing Arabic text categorization. The proposed method explores two variants of the transformer AraBERT. The first variant relied on the pre-trained AraBERT fine-tuning, whereas the second one investigated multilingual BERT and AraBERT models as feature extractors combined with several classifiers. The use of contextual information allowed us to handle the Arabic language problems, such as the complexity and ambiguity, by considering the specific model AraBERT.
More details about this study are available in chapter 5.

4. We propose a transfer learning method for identifying the offensive language in a bilingual context where we simultaneously consider Arabic and English tweets. Our method is twofold. First, we tackle the multilingualism problem. Secondly, we cope with offensive language detection. Our method relies on BERT models, including BERT, mBERT, and AraBERT, that are fine-tuned on offensive detection task. Our system consists of several modules: (1) Preprocessing tweets by taking into consideration emojis substitution, (2) Tweet translation for joint translated classification, (3) Tokenizing tweets using BERT tokenizer according to the used multilingualism approach, (4) BERT fine-tuning to produce tweets representation and (5) Tweets prediction applying a sigmoid function on tweets representation. We carried out a set of experiments on a bilingual dataset from the SemEval competition.
To know more about Offensive language detection, transfer learning, or for further details about this study, see chapter 6.

0.4 Content Overview

In the following, we provide an overview of what is presented along with this thesis.
Chapter 1 presents the Arabic language and its peculiarities. This chapter also introduces Arabic natural language processing techniques.
Unveiling the different techniques involved in automatic text categorization by using supervised learning algorithms will be presented in chapter 2. We also introduce deep learning and a brief history of this latter in a text categorization context.
Chapter 3 provides details about our first contribution that concerns the use of deep Autoencoder for text representation and categorization. First, this chapter presents our motivations. Then, it describes Stacked Autoencoder with Restricted Boltzmann Machines pretraining. Then, it presents our methodology. Finally, this chapter ends with a description of the performed evaluations and a discussion of the results.
Chapter 4 presents our second contribution that investigates deep learning networks and retrofitting for Arabic text categorization. We start with our motivations for the use of neural networks. Then, we review related work, including convolutional and recurrent neural networks. Next, we present our method based on deep neural networks and retrofitting. Finally, this chapter exposes the obtained results and discussion.
Chapter 5 present our contribution to using transfer learning for Arabic text categorization. We proceed by presenting our motivations for using transfer learning. Then, we introduce the transformer in detail. Afterward, we present our method containing different phases, such as preprocessing, fine-tuning, and feature extraction. Finally, we detail the performed experiments, evaluations, and obtained results.
Chapter 6 is dedicated to describing our last contribution concerning the application of transfer learning to offensive language detection in a multilingual context.
General conclusion summarizes the contributions made within this work, establishes possible future trends and open issues for Arabic text categorization. The thesis ends with the bibliography followed during our research.

CHAPTER 1

Arabic Natural Language Processing

1.1 Introduction

The Arabic language is among the most ubiquitous languages in the world since it is the fourth used language on the web. The number of speakers reaches more than 420 million speakers over the world.
Natural language processing is a discipline of computer science and artificial intelligence that has been ongoing since the 1950s. Natural language processing merges linguistics and artificial intelligence to enable computers to model, analyze, interpret, and represent natural language. NLP's business value is to make sense of the unstructured data and help organize it into a more structured model. NLP opens the door for sophisticated data analysis since it allows to extract syntactic, morphologic, and semantic information. This information is explored to produce a sophisticated text representation to boost text mining applications.
This chapter is organized as follows: Section 1.2 discusses the Arabic script and the character sets. Section 1.3 describes the general morphology of Arabic. Section 1.4 discusses derivational morphology. Section 1.5 defines inflectional morphology. Section 1.6 introduces Arabic language syntax. Section 1.7 describes the natural Arabic language processing techniques, And finally, the last section concludes.

1.2 Peculiarities of the Arabic Language

1.2.1 The Arabic Language

Arabic is an uncommonly rich language that is related to another linguistic family, particularly Semitic vernaculars, which is different from the Indo-European lingos talked in the West. Arabic is the 6th most spoken language in the world by more than 420

million people as a native language[1]. Besides, 24 countries claim Arabic as an official language, including Morocco, Algeria, Tunisia, Libya, Mauritania, Bahrain, Chad, Comoros, Djibouti, Egypt, Eritrea, Iraq, Jordan, Kuwait, Lebanon, Oman, Palestine, Qatar, Saudi Arabia, Somalia, Sudan, Syria, United Arab Emirates, and Yemen. Besides, there are six sovereign states in which Arabic is considered a national language or recognized as a minority language, including Iran, Turkey, Niger, Senegal, Tanzania, Mali, and Cyprus. Arabic is connected with Islam, and more than 200 million Muslims perform their petitions five times daily utilizing this dialect.
Arabic is known for its complex morphology since it is an exceedingly structured and derivational language [3, 15]. Unlike English, Arabic is written from right to left without any capitalization. Besides, Arabic has tree forms range from Classical Arabic (CA) and Modern Standard Arabic (MSA) to Dialectal Arabic (DA):

- **Classical Arabic (CA)** includes classical historical text such as poetic and Quran based on medieval dialects of Arab Tribes.
- **Modern Standard Arabic (MSA)** is a direct descendant of CA which is used today throughout the Arab world in writing and formal speaking. MSA uses the morphology and syntax structures available in CA.
- **Dialectal Arabic (DA)** is used for daily conversations. Dialects include a lot of slang and culturally relevant vocabulary. colloquial Arabic can be categorized into nine main dialects, including Hassaniya Arabic (Mauritania), Maghrebin Arabic (Morocco, Algeria, Tunisia, and Libya), Egyptian Arabic, Levantine Arabic (Lebanon, Syria, Jordan, and Palestine), Yemeni Arabic (Yemen and southwestern Saudi Arabia), Gulf Arabic (Kuwait, Bahrain, Qatar, the UAE, and Oman), Hejazi Arabic (Western Saudi Arabia), Najdi Arabic (Central Saudi Arabia) and Iraqi Arabic.

1.2.2 Character Sets

Arabic is a Semitic language written from right to left in a cursive, a consonantal script with 29 characters. Besides, alphabet contains three long vowels including ا (alef), و (waw) and ى (yeh). Unlike English, Arabic has no capitalization (lower and upper cases are absent in Arabic). Each letter changes its shape based on the position within words. An Arabic character might have four different shapes: isolated, initial, medial, and final. For instance, the characters ،جـ ،ـجـ ،ـج and ج are four various shapes of the same letter /jiim/. However, the different representations of a character are often mapped to a single base code in the computer encoding systems. Table 1.1

[1] https://www.internetworldstats.com/stats7.htm.

shows the Arabic alphabet characters along with their international phonetic association (IPA) symbols that we use to represent the pronunciation of Arabic words [IPA, 1999]. Arabic knows the presence of diacritics that are used to clarify the pronunciation of characters within a word. Three diacritics are utilized to represent short vowels that can be employed with every consonant character. These diacritics include fatha, damma, and kasra. For example, the consonant ب /b/ with the diacritic fatha بَ, is pronounced /ba/, with the diacritic damma بُ, is pronounced /bu/, and with the diacritic kasra بِ is pronounced /bi/. When we place two identical diacritics above or below the last letter of an Arabic noun; this is called tanween. For instance, the word نُزْهَةٌ picnic is pronounced /nuzhatun/, نُزْهَةً is pronounced /nuzhatan/, and نُزْهَةٍ is pronounced /nuzhatin/. The diacritic shadda as in بّ /bb/ is used to marks consonant doubling. Moreover, the diacritic sukoon is a small circle that is placed above the letter to make a vowel-less consonant as in بْ. The absence of diacritics can lead to syntactical and morphological ambiguity, which makes Arabic natural language processing more complex. For this reason, readers must rely on context to determine implicit diacritics and how the word should be pronounced. To illustrate the aforementioned complexity, we consider the word كتب/ktb/ using the morpho-syntactic analyzer AlKhalil [21], this word has several variants like: كَتَبَ (/kataba/he wrote), كُتُب (/kutub/books), كُتِبَ(/kutiba/is written), or كُتِب (/kutiiba/ wrote). The tatweel (kashida) is a special character that is used in typeset Arabic text. For example, the word كتاب (/kitab/book) can have different shapes as كتـاب and كتـــــــاب.

1.3 Arabic Morphology

The morphology is a domain that studies the internal structure of words by analyzing interactions with both orthography and syntax. It concerns the definition of Morphemes which are the smallest linguistic unit that has a meaning or grammatical function. Arabic is known for its complex and rich morphology [131]. As a result, there is a wealth of terminology; thus, ANLP is challenging [47, 54, 14]. Arabic words are categorized into three categories: verbs, nouns, and particles.

- **A verb** is a word that indicates action at a certain time. Verbs are derived from a set of roots of three or four consonants using patterns. Arabic verbs have a limited number of patterns: ten basic triliteral patterns and two basic quadriliteral patterns [54]. Form I (triliteral pattern 1V2V3) and Form QI (quadriliteral pattern 1V23V4) are considered the most basic patterns.

Table 1.1 Different shapes of Arabic letters with their different shapes. Isolated, "I"; as a first letter, "F"; in the middle of the word, "M"; or as a last letter, "L". The IPA column shows their international phonetic representation [94]

I	F	M	L	IPA	I	F	M	L	IPA	I	F	M	L	IPA
ء	–	–	–	/ʔ/	ر	ر	ـر	ـر	/r/	ف	فـ	ـفـ	ـف	/f/
ا	ا	ـا	ـا	/aa/	ز	ز	ـز	ـز	/z/	ق	قـ	ـقـ	ـق	/q/
ب	بـ	ـبـ	ـب	/b/	س	سـ	ـسـ	ـس	/s/	ك	كـ	ـكـ	ـك	/k/
ت	تـ	ـتـ	ـت	/t̪/	ش	شـ	ـشـ	ـش	/ʃ/	ل	لـ	ـلـ	ـل	/l/
ث	ثـ	ـثـ	ـث	/θ/	ص	صـ	ـصـ	ـص	/sˤ/	م	مـ	ـمـ	ـم	/m/
ج	جـ	ـجـ	ـج	/ʒ/	ض	ضـ	ـضـ	ـض	/d̪ˤ/	ن	نـ	ـنـ	ـن	/n/
ح	حـ	ـحـ	ـح	/ħ/	ط	طـ	ـطـ	ـط	/t̪ˤ/	ه	هـ	ـهـ	ـه	/h/
خ	خـ	ـخـ	ـخ	/x/	ظ	ظـ	ـظـ	ـظ	/ðˤ/	و	و	ـو	ـو	/w/
د	د	ـد	ـد	/d̪/	ع	عـ	ـعـ	ـع	/ʕ/	ي	يـ	ـيـ	ـي	/j/
ذ	ذ	ـذ	ـذ	/ð/	غ	غـ	ـغـ	ـغ	/ɣ/	ى	–	–	ـى	/aa/
ة	–	–	ـة	/t̪/	–	–	–	–		–	–	–	–	–

- **A noun** is a word that has meaning without any association with time. Nouns include nouns that are derived from verbal roots, proper nouns, personal pronouns, demonstrative pronouns, and relative pronouns.
- **Particles** are prepositions and conjunctions to associate phrases, words, or verbs.

1.3.1 Arabic as an Agglutinative and a Pro-Drop Language

word agglutination is one of the linguistic properties which make the Arabic language more challenging. Arabic constructs complex words that can contain affixes and clitics representing various parts of speech or morphemes. Therefore, a verb may embed within itself its subject and object as well as other clitics signifying tense, gender, person, number, and voice [54]. For example, the Arabic sentence ورايتهم /wra'aytuhum/"and I saw them" is written as one word and maybe decomposed into the following four morphemes:

1. و /wa/ Conjunction "and"
2. راى /r'aa/ Past tense Verb "saw"
3. تُ /tu/ Subject Pronoun "I"
4. هم /hum/ Object Pronoun "them"

[54] defined the order and levels in which clitics can attach to the basic form of the word

as follow:

[QST + [CNJ + [PRT + [DET + BASE + PRO]]]]

- QST query conjunction أ
- CNJ coordination and subordination conjunctions:و /wa/ (et) et ف /fa/ (et, so). These conjunctions attach to any word (verbs and nouns).
- PRT represents the class of particles: the mark of future س /sa/ (attaches only to verbs), the prepositions ب /bi/ (with) and ك /ka/ (like) which are attached to nouns and adjectives, and the preposition ل /li/ (for) which attaches to verbs and nouns.
- DET is the definite article ال which can be concatenated only with nouns and adjectives.
- PRO a member of pronominal clitics which represents pronouns that attach to the end of the basic form of the word (noun or a verb).

Arabic is a pro-drop language that means that, in various cases, the subject may be freely dropped and replaced by a personal pronoun manifesting itself as a suffix to the head verb. This characteristic adds further complication and challenges to the computationally complex nature of the Arabic language.

1.4 Derivational Morphology

Derivational morphology concerns new words creating from other words. Arabic words are derived from roots using different patterns. The root الجذر is a radical morpheme composed from a sequence of three, four, or very rarely five consonants [54]. For instance, the Arabic word كاتب /katib/ writer is derived from the verb كتب /kataba/ to write. The notion of the root in Arabic is sometimes confused by researchers with the notions of stem الجذع for several reasons. One reason is that the notion of the root in English and other European languages is closer to that of the stem in Arabic. The second reason is that some researchers of Arab background make the use of diacritics in Arabic orthography's optional, which cause some undiacritized words look like a sequence of root radicals, for instance, the word كتب /katab/ he wrote and its root ك ت ب /k-t-b/. Arabic is a derivational language that is distinguished by the existence of several types of derivational relations between words using pattern or affixation. Some derivational relations cross POS categories, such as the deverbal nouns that are derived from verb forms. But others preserve the POS, e.g., the different semantically related verb forms. Next, we describe some of the common derivations in Arabic.

- verbal noun (المصدر) is an abstract noun that presents the meaning of the verb and indicates an event without specifying its time. However, there are no rules for deriving the verbal nouns from verbs. For instance, نام /nAm/ sleep (verb) نوم /nawm/ sleeping (Noun) and كتب /katab/ write (verb) كتابة /kitaAbah/ writing.

- The active participle (اسم الفاعل) is a derivative noun that indicates who did the action. Verbs having form I use the pattern 1A2i3. For example, the active participle of the verb كتب is كاتب /kAtib/. The corresponding pattern for Form X is musta12i3: the participles of the verb استخدم /Aistaxdam/ use is مستخدِم /mustaxdim/ user.

- The passive participle (اسم المفعول) is a specific form in Arabic which is used commonly to refer to the object of the action. Form I passive participle is ma12uw3, and form X follows the pattern musta12a3. For example, the passive participle of the verbs استخدم and كتب are مكتوب /maktuwb/ written and مستخدَم /mustaxdam/ used.

- Nouns of place and time (أسماء الزمان و المكان) are used to indicate place and time using the patterns ma12a3 and ma12i3, respectively. For example, the noun of place for جلس /jalas/ sit is مجلس /majlis/ council.

- Noun of the instrument (اسم الآلة) is a non to denote instruments used for the verb they are derived from. There exist many nominal patterns to derive the noun of instrument, e.g., مفتاح /miftAH/ key from فتح /fataH/ (open) using the pattern mi12A3, كسّارة /kas˜Arah/ nutcracker from كسّر /kassar/ (smash) following the pattern 1a22A3ah and حاسوب /HAsuwb/ computer from حسب /Hasab/ compute.

- Noun of diminutives (اسم التصغير) is used to derive the diminutive form using the pattern 1u2ay3 among others. For example, شجيرة /šujayrah/ shrub is the diminutive of شجرة /šajarah/ tree. Ya of relatedness (ياء النسبة) is a derivational suffix (يّ /iy˜/) that maps nouns to adjectives related to them. For example, سياسة /siyAsah˜/ politics سياسيّ /siyAsiy˜/.

There are other forms of nouns which are derived from a feminine singular suffix, namely unit noun (اسم الوحدة) such as تمر /tamr/ dates (collective) تمرة /tamrah/ date (singular).

Table 1.2 Arabic verb subject affixations [54, 53]

	Perfective			Imperfective (Indicative, Subjunctive, Jussive)		
	Singular	Dual	Plural	Singular	Dual	Plural
1st	+tu	+nA		+ +(u, a, .)	n+ +(u,a,.)	
2nd	+ta	+tumA	+tum	t+ +(u, a, .)	t+ +(Ani, A, A)	t+ +(uwna, uwA, uwA)
	+ti		+tun~a	t+ +(iyn, iy, iy)		t+ +na
3rd	+a	+A	+uwA	y+ +(u, a, .)	y+ +(Ani, A, A)	y+ +(uwna, uwA, uwA)
	+at	+atA	+na	t+ +(u, a, .)	t+ +(Ani, A, A)	y+ +na

1.5 Inflectional Morphology

The word ”Inflection” mostly designates the phonological changes that a word undergoes as it is being used in context [109]. Inflectional morphology conveys grammatical information such as number, tense, agreement, or case and is expressed using both templatic and concatenative morphology. Arabic has eight major grammatical categories, which are: number, gender, person, voice, definiteness, tense/aspect, case, and mood. Four of these categories apply to adjectives and nouns (number, gender, definiteness, and case), four apply to pronouns (number, gender, person, and case), and six of them apply to verbs (voice, mood, tense/aspect, number, gender, and person). There exist two aspects of Arabic inflectional morphology, including verbal and nominal morphologies.

1.5.1 Verbal Morphology

Arabic verbal morphology contains a number of regular rules. Verbs inflect for aspect or tense (Perfective/past ماضي, Imperfective/present مضارع and imperative/order أمر), mood [134] (Indicative مرفوع, Subjunctive منصوب, Jussive and Imperative مجزوم), the person having three values: speaker متكلم, addressee مخاطب and other غائب, the number having three values: singular, dual or plural, and gender has two values: masculine or feminine. The verbal subject is indicated via affixations, whose form is affected by verbal aspect and mood. Table 1.2 lists of all the verbal subject affixes.

1.5.2 Nominal Morphology

Nominal morphology is more complex than verbal morphology. Arabic nominals inflect for gender, number, state, and case.

- Gender and Number: Arabic has two gender values, including masculine and feminine, and three number values: singular, dual, and plural. These two features are expressed using common morphemes representing some number and some gender combinations. There is a disconnect between the markers of morphemic and functional gender and number. In around 80 % of nominals, functional and

morphemic genders agree and are called sound plurals (الجمع السالم). For example, مدرس /mudar˜is/ teacher [masculine singular], مدرسة /mudar˜is+ah/ teacher [feminine singular], مدرسون /mudar˜is+uwna/ teachers [masculine plural], and مدرسات /mudar˜is+At/ teachers [feminine plural]. Nevertheless, in the other 20 %, functional gender and number do not match morphemic gender and number. The most common patterns of form-function disagreement are Broken Plural (جمع التكسير).

- Definiteness and Construct State: Arabic nouns inflect for the state, which has three values including definite, indefinite, and construct. The definite state is the nominal form that appears most typically with the definite article (ال) like الكتاب /Al+kitAb+u/ the book. The indefinite state is used to express an unspecified instance of a noun like كتاب/kitAbu/ a book. The construct state indicates that the noun is the head of an Idafa construction. The first word is called (مضاف /muDAf /) and possessed by the noun phrase that follows it. For instance, كتاب الطالب /kitAbu AlTAlibi/ the book of the student.
- Case: All Arabic nominals inflect for case, which has three values in Arabic: nominative (مرفوع), accusative (منصوب) or genitive (مجرور).

1.6 Arabic Syntax

The syntax is how linguistic elements (such as words) are arranged together to form phrases or clauses. Arabic involves two types of sentences: nominal and verbal. A nominal sentence (الجملة الإسمية) starts with a noun, whereas a verbal sentence (الجملة الفعلية) starts with a verb. There should be an agreement in number and gender between the verb and the subject in both sentences.

1.6.1 Verbal Sentences

The verbal sentence (V-Sent) consists of Verb, Subject, and Object(s). The prototypical structure of the verbal sentence is expressed in different forms. The most basic form of the V-Sent incorporates a verb with a conjugated (pro-dropped) pronominal subject. In this case, the verb expresses the person, gender, and the number of the subject, e.g., كَتَبْنَا/katab+nA/ we wrote (Verb+Subj). Non-pronominal subjects appear after the verb. The verb agrees with the subject in person and gender but not the number, which stays singular. A verb with a non-pronominal subject in a V-Sent is never plural. The subject has a nominative case. For example, كَتَبَ الوَلَدُ/katab+a Al+walad+u/ the boy wrote.

Pronominal objects appear as part of verbal suffixes whether the subject is pronominal or non-pronominal as in كَتَبَها /katab+a -hA/ he wrote it and كَتَبَها الوَلَدُ /katab+a -hA Al+walad+u/ The boy wrote it.

1.6.2 Nominal Sentences

The Nominal Sentence (N-Sent) consists of Subject-Predicate and Topic-Complement (مبتدأ وخبر). In the simplest N-Sent, the subject is typically a definite noun, proper noun, or pronoun in the nominative case. The predicate is an indefinite nominative noun, proper noun, or adjective that agrees with the subject in number and gender.

- Subject (Nom) + Predicate (Nom): الكِتَابُ جَدِيدٌ (The book is new).
- Subject (Nom) + Predicate (PP-Predicate): the predicate can be a prepositional phrase, e.g., الرَجُلُ فِي البَيْتِ (The man is in the house).
- Subject (Nom) + Predicate (N-Sent): The predicate can also be another N-Sent. For example, البَيْتُ بَابُهُ جَدِيدٌ (The house, its door is new).
- Subject (Nom) + Predicate (V-Sent [Verb + Subject + Objective]): the predicate can be a verbal phrase, الأَوْلاَدُ كَتَبُوا القِصَص (The boys wrote the stories).
- Predicate (PP-Predicate) + Subject (Nom): فِي البَيْتِ رَجُلٌ (Man is in the house) or (There is a man in the house).

1.6.3 Nominal Phrase Structure

The most basic nominal phrase comprises a noun or an adjective with or without the definite article. Several types of nominal modifiers exist.

- Adjectives that always agree in the definiteness and case with nouns. In addition, adjectives of rational nouns agree in gender and number. However, broken plural adjectives are form-wise singular with ad hoc form-based gender. For instance, the word المَهَرَة /Al+maharah/ is feminine and singular by form but masculine and plural functionally. Concerning adjectives of irrational nouns, they agree with the nouns in gender and number when they are singular or dual. For example, مَكَاتِبُ جَدِيدَةٌ /makAtibu jadiydahũ/ new offices, the adjective جَدِيدَةٌ is feminine singular does not agree with مَكَاتِبُ which is plural masculine.
- Idafa construction (الإضافة) is a possessive/genitive construction allowing to relate two nouns: the possessor (مضاف /mudaf/) and the possessed (مضاف إليه /muDAf Ailayhi/). The possessor is in the construct state while the possessed has

a genitive case. For example, مَفَاتِيحُ السَيأْرَةِ /mafAtiyHu Al + sayArati/ the keys of the car. This can be extended recursively to create an Idafa chain.

- Tamyiz construction (التمييز) connects two nouns. The first noun called the specified (المُمَيَزْ /Almumay˜az/) that heads and governs the second noun, the specifier (المُمَيِنْ /Almumay˜iz/) qualifying the first noun. The specifier is always singular in number and accusative in case. For example, أَكْثَرُ بَيَاضاً /Aktaru bayADAã/ whiter.
  There exist other nominal modifiers such as apposition construction (البدل) and relative clauses (جملة الوصل). Besides, Verbal nouns such as deverbal nouns and active participles behave like verbs so they can take an accusative object argument and other verbal modifiers. For example, مَعْرِفَةُ الرجلِ الحقِيقَةَ.

1.7 Arabic Natural Language Processing

Over the last few years, Arabic natural language processing has gained increasing importance, and several state-of-the-art systems have been developed for a wide range of applications, including text classification, machine translation and information retrieval. These applications had to deal with several complex problems pertinent to the nature and structure of the Arabic language. Next, we describe the ANLP techniques and present some ANLP tools.

1.7.1 Tokenization

Tokenization refers to the division of sequences of characters in a text into morphemes called tokens. For agglutinative languages like English, most text tokenization methods rely on several word separators, such as punctuation marks, white spaces, and numbers. However, tokenization in Arabic presents a real problem [13]. For example, a single Arabic word may comprise up to four different tokens like ورايتهم /wa + r'aa + tu + hum/ and I saw them. Hence, knowledge of the constraints on concatenating affixes and clitics within Arabic words is required. A distinction is needed between clitics that are syntactic units and affixes that mark grammatical inflections such as tense, number, and person agreement.

1.7.2 Normalization

One of the most critical dilemmas facing researchers and developers of Arabic computational linguistics is normalization. Some Arabic letters share the same orthographical forms and are only differentiated by adding certain marks such as a dot, a hamza (Hamza Al-Qata قطع همزة and Hamza Al-Wasl وصل همزة) or a madda placed above or below the letter. For example, the alif in Arabic (ا) maybe three different letters depending on whether it has a hamza above as in (أ) or a hamza below as in (إ) or a madda above as in (آ). Recognizing these marks above or below a letter is essential to be able to distinguish between apparently similar letters. However, texts that are written in MSA often do not incorporate voweling, nor do they adhere to the inclusion of marks above or below some Arabic letters. To tackle this problem, the common solution in Arabic NLP systems is to normalize the input text. [76] replace أ، إ، آ with simply an alif (ا). Further, they normalize the taa marbuuTa (ة) with (ه) and the ئ ، ي with (ى).

1.7.3 Stop-word Removal

Stop-words appear very frequently in a text collection and are considered to add little document-specific information. These words are often omitted to reduce the vocabulary size, avoid the noise, and consider only relevant or representative terms of the content. Stop word removing relies mainly on stop-word dictionaries. Several researchers proposed different list of stop-words [67, 42, 22].

1.7.4 Stemming

Stemming is the process to reduce different variants of the same word with different endings to a common stem [98]. Stemmers have improved performance in information retrieval tasks, especially with a highly inflectional language like the Arabic language. Stemming techniques allow to unify vocabulary, reduce term variants and feature space representation, and increase the likelihood of matching documents [111]. For the Arabic language, there are three different approaches for stemming, including the root-based stemmer, the light stemmer, and the statistical stemmer.
Root-Based stemmer uses morphological analysis to extract the root of a given Arabic word. Many algorithms have been proposed in this area. [5] proposed an algorithm that finds the word's root by matching the word with all possible patterns with all possible affixes attached to it without removing any prefixes or suffixes. [7] developed a morphology system using different algorithms to find roots and patterns. This algorithm omits the longest possible prefix from the word and then extracts the root by checking

the first five letters. The basic hypothesis is that the root must be included in the first five letters of the word. [67] developed an algorithm removing prefixes and suffixes, checking that it's not dropping part of the root and then matching the remaining word against the patterns of the same length to extract the root.
The light stemming approach aims at removing all suffixes and prefixes from the word form. Light stemming techniques rely mainly on the use of a list of suffixes and prefixes to reduce words to their stems. Several works proposed light stemming algorithms. [74] have proposed various light stemming algorithms: light1, light2, light3, and light8 sharing the same normalization technique. These algorithms vary depending on the number and depth of prefixes and suffixes to be removed. These algorithms remove the affixes in a specific order from left to right. The latest version, namely light10 [75], is the most efficient. It consists of combining the other versions. The evaluation was carried out using TREC 2001 and TREC 2002 collections. [26] have proposed two light stemming techniques for Arabic texts. The first one consists of grouping words according to their English translation. Words having the same translation are represented by the same word with the shortest form. The second technique relies on removing prefixes and suffixes. The list of 26 prefixes and 22 suffixes are constructed according to their grammatical role and their frequency of occurrence in the distinct words of the TREC 2001 collection. [64] introduced a light stemming method using TREC 2001 to build the list of possible stems for each term in the collection. The algorithm proceeds by removing prefixes and suffixes according to their occurrences in the words of the corpus. Farasa stemmer is a technique based on SVM-rank using linear kernels that employs various features and lexicons to rank possible segmentations of a word. The features include likelihoods of stems, prefixes, suffixes, their combinations; presence in lexicons containing valid stems or named entities; and underlying stem templates [1]. Farasa is state-of-the-art Arabic segmenters and outperforms Stanford and MADAMIRA [99].

1.7.5 Part-Of-Speech Tagging

Part-Of-Speech tagging is the procedure of assigning one POS tag to every word-form or every word's segments. The tags are selected from a predefined tag set. Arabic morphological analysis is essential to ANLP assignments. POS tagging is usually done in the first steps of NLP tasks such as machine translation, information extraction, word sense disambiguation, and text categorization [116, 138, 104].
Due to the inflectional and complex nature of the Arabic language and the deficiency of short vowels, morphological analysis of Arabic is not an easy task. The analysis involves handling of a high degree of ambiguity [11]. Indeed, several morphological analyzers and POS taggers were proposed for the Arabic language. Next, we describe

some analyzers and POS taggers systems.

1.7.5.1 MADA+TOKAN

MADA [54] is a popular suite that has multiple tools for ANLP[2]. MADA processes raw Arabic text to provide a list of applications: POS tagging, diacritization, lemmatization, stemming, and glossing. MADA is written in Perl and uses the Support Vector Machines (SVM) model trained on Penn Arabic Treebank (PATB) to select a proper analysis from the list provided by Buckwalter Arabic Morphological Analyser (BAMA). MADA uses 19 features, 14 of which are morphological features, to rank the list of possible analysis. The reported accuracy of predicting the correct POS tag is 96.1% [99].
TAGSET: MADA targets the finest possible POS tagset. It supports the mapping to four different possible tagsets: ALMORGEANA, CATiV, PATB, or Buckwalter.

1.7.5.2 AMIRA Toolkit

AMIRA is a toolkit of three main tools: tokenizer, POS tagger, and base phrase chunker. The POS tagger uses the YamChi toolkit, an SVM-based sequence classification toolkit. The toolkit does not depend on deep morphology information; instead, it learns from the surface data. AMIRA was trained on PATB. The reported accuracy of predicting the correct POS tag using the default tagset is 96%.
TAGSET: AMIRA can output the tags in one of three tagsets: RTS, Extended RTS, Extended RTS with personal information. Extended RTS with personal information has about 72 tags, and those tags encode gender, number, and definiteness.

1.7.5.3 Stanford POS tagger and segmenter (ST)

Stanford NLP group released a list of ANLP tools[3], including a POS tagger [125] and Arabic word segmenter [34]. The POS tagger is shipped with a model for Arabic trained on the Penn Arabic Treebank (PATB). It uses the Maximum Entropy approach to assign a POS tag to a segmented text (using Stanford Arabic Word Segmenter). Stanford Arabic Word Segmenter uses Conditional Random Fields (CRF) classifier to normalize the text and split off clitics from base words in a similar segmentation schema to one used in the PATB.

[2] `https://www.sketchengine.eu/arabic-mada-system-tagset//stats7.htm`.
[3] `https://nlp.stanford.edu/projects/arabic.shtml`.

TAGSET Bies tags of 25 basic tags. Authors augmented the tagset by adding DT (determiner) to the beginning of nominal tags.

1.7.5.4 MADAMIRA suite (MA)

MADAMIRA [99] is a suite that combines two previously mentioned systems: MADA and AMIRA. MADAMIRA ported the two systems into JAVA programming language, allowing it to be portable, extensible, and even faster[4]. MADAMIRA supports MSA and Egyptian Arabic. One added feature to MADAMIRA is the server mode feature, which allows the user to run MADAMIRA in the background and then send HTTP requests for tokenization, tagging, etc. While the accuracy has not improved, the tagging speed has improved over MADA substantially (16-21x faster). The reported accuracy of predicting the correct POS tag is 95.9% [99].
TAGSET: The tagset used by MADAMIRA extends the MADA tagset by having some tags for Egyptian Arabic processing.

1.7.5.5 Farasa

Farasa [143] is a toolkit for segmentation/tokenization module, POS tagger, Arabic text Diacritizer, and Dependency Parser[5]. Farasa is different from other POS taggers as it can jointly segment, pos-tag, and parse the text, which avoids error propagation in the pipelined structure and should exploit syntactic information for POS tagging. This is particularly useful for tagging classical Arabic as it is different in vocabulary from MSA, but it shares similar syntax. The reported accuracy of predicting the correct POS tag of MSA is 97.43% and of CA is 84.44%. Fig. 1.1 shows an example of a sentence tagged with Farasa.
TAGSET: FARASA has a tagset of 16 basic tags.

1.8 Conclusion

In this chapter, we have introduced the Arabic language and explained its morphology. We have presented the characters used in Arabic and their different representations in Arabic text. We have also explained the different morphologies of Arabic, including derivational morphology and inflectional morphology. Moreover, we have described Arabic syntax and detailed verbal and nominal sentences. On the other hand, we have presented the Arabic natural language processing techniques, including tokenization,

[4] `http://nlp.ldeo.columbia.edu/madamira/`.
[5] `https://farasa.qcri.org/POS/`.

Normalization, Stemming, and Part-of-speech tagging. In the next chapter, we continue with a review of text categorization systems and deep learning.

تحتوي اللغة العربية 28 حرفاً مكتوباً

NUMBER	GENDER	POS	WORD	ID
na	na	V	تحتوي	1
S	F	NOUN	اللغة	2
S	F	ADJ	العربية	3
P	M	NUM	28	4
S	M	NOUN	حرفا	5
S	M	ADJ	مكتوبا	6

Fig. 1.1 Example of POS tagging using Farasa Online Web Demo

CHAPTER 2

Text Categorization and Deep Learning

2.1 Introduction

Automatic text categorization received a surge of attention because of the large amount of data available on the internet. The recent breakthroughs in natural language processing paved the road for proposing successful automatic text categorization methods. Text categorization is an important task for several text mining applications that generally involves a sequence of steps, including:

Text preprocessing such as tokenizing, removing stopword, stemming...

Feature extraction to convert the unstructured textual data into structured data using some techniques such as TF-IDF [70], Word2Vec [87], Doc2vec [77], GloVe [102], ...

Feature selection to select the most pertinent features.

Classification during which we start by training the chosen classifier on the training dataset then test it on unlabeled data from the evaluation set.

Evaluation which is the final phase in the text categorization system. There exist many metrics to evaluate the categorization performances, including accuracy, F-measure, recall, and precision.

Deep learning invades the text categorization area by proposing more effective algorithms to overcome the classical algorithms' limitations. We aim in this chapter to provide a historical introduction to the field of text categorization and deep learning. First, we introduce the basic building blocks of a categorization system, including text representation, feature selection, classification, and evaluation. Along with that discussion, we will also introduce deep learning for text categorization to help the reader understand the possibilities of current artificial neural networks for text.

2.2 Text Categorization Paradigms

The automatic assignment of labels to documents using full-text data has been included inside the area of text categorization. In this area, machine learning and information retrieval meet, affording solutions to problems with real word collections. Text categorization is a supervised learning task responsible for the automatic classification of textual documents under predefined categories. As shown in Fig. 2.1, three paradigms in text categorization exist, including the binary case, the multi-class case, and the multi-label case [91].

- In the binary categorization, a sample can belong to exactly one or two given categories.

- In the multi-class categorization, a sample belongs to one class of a set of multiple classes.

- Finally, in the multi-label categorization, a sample can belong to various categories at the same time.

In binary classification, a classifier is trained basing on supervised algorithms in order to attribute a sample textual document to one of two possible sets. These two sets are usually referred to as belonging samples (positive) and not belonging samples (samples) respectively or two disjoint classes. To do so, several algorithms can be performed, such as Linear Regression, Naïve Bayes, and Support Vector Machines [62]. The binary case is a base case from which we can build the two other cases. Multi-class and multi-label tasks consist of training a binary classifier for every class. Whenever the binary base case returns a measure of confidence on the categorization attributing either the top-ranked one (multi-class task) or a given number of the top-ranked ones (multi-label task). Another approach for multi-label categorization relies on return all the classes whose binary classifiers provide a positive answer for the sample. In our work, we are interested in studying the multi-class categorization for Arabic. In the next section, we present a definition of the elements involved in the text categorization assignment.

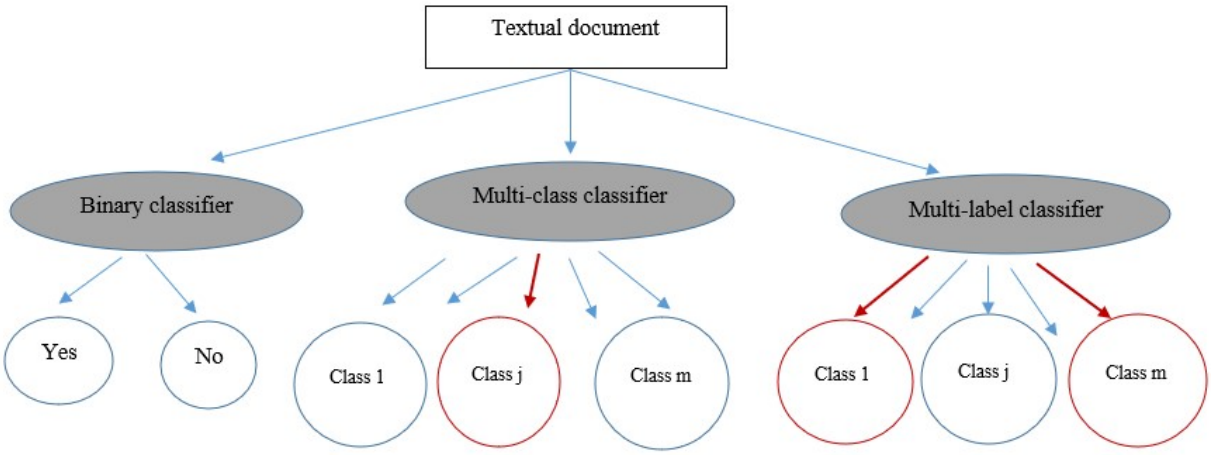


Fig. 2.1 Paradigms in text categorization: binary, multi-class and multi-label

2.3 A Mathematical Definition of Text Categorization Task

We define some notations to clarify the text categorization assignment and setting a base terminology to model mathematically the text categorization.
Given

- A set of textual documents as $X = \{x_1, x_2, \ldots, x_N\}$ and
- A set of classes (or labels) as $C = \{c, c_2, \ldots, c_M\}$
- A binary classifier is defined as a function $\Phi_b : X \rightarrow \{true, false\}$ mapping documents to assignment assessment.
- A m-class classifier is a function $\Phi_{mc} : X \rightarrow C$ mapping documents to labels.
- A m-label classifier is a function $\Phi_{ml} : X \rightarrow 2^C$ where $2^C \subseteq C$ is the power set of C.

Therefore, a multi-label categorization system as the algebraic structure:

$$< X, C, \Phi_{ml} > \tag{2.1}$$

And a multi-class categorization system as the algebraic structure:

$$< X, C, \Phi_{mc} > \tag{2.2}$$

The used machine learning algorithms will be based on documents representation on the feature space. Hence, every text will be transformed to a vector of features as follows:

$$F = \{f_1, f_2, \ldots, x_T\} \tag{2.3}$$

Where F is the set of all the corpus features. To represent the values of features in a document x_i, we use:

$$F = \{w_{i,1}, f_{i,2}, \ldots, x_{i,l}\} \tag{2.4}$$

Where $w_{i,k}$ is the weight of the feature f_k in text x_i.

2.4 Text Categorization System Architecture

A text categorization system is composed from several components (see Fig. 2.2).

- **Feature extraction**: Text in the unstructured format must be processed and indexed to become structured. Two steps are given:

  1. Feature identification in which we identify all the features which will be preserved: words, bi-grams, stem, or other items. This phase encompasses the natural processing techniques discussed above.

  2. Feature weighting containing the weight calculation of the selected features. There exist several approaches as BOW and TF-IDF that we will discuss in the section (text representation).

- **Dimensionality reduction**: Text categorization suffer from the curse of dimensionality. Therefore, various methods have been proposed to reduce the dimensionality of feature space using stemming, lemmatization, or feature selection techniques. Another approach is feature transformation, in which we replace terms with other entities. In this case, we try to enclose the existing words by grouping them into semantic sets such as the latent semantic analysis [32]. During the feature extraction and dimensionality reduction phase, the text will be transformed into samples that can be handled by learning algorithms.

- **Classifier training**: This step is considered as the core of the categorization system. Here machine learning methods are utilized to build an autonomous classifier based on supervised learning algorithms. There exist a wide range of algorithms. Fortunately, several researches have been done comparing different classifiers for text categorization such as Naive Bayes, Decision Trees, Support Vector Machines, Boosting, Multi-layer perceptron that will be described in the section (classifiers overview). As textual documents have been converted into a vector of features in former stages, these algorithms can be applied.

To sum up, we have to convert the textual document into a set of features that will be handled by the categorization algorithm. The learning process depends on the used algorithm. In general, a training collection is split into training and evaluation sets. The

involved algorithms may differ from one to another, but in general, this architecture is common for most classification problems.

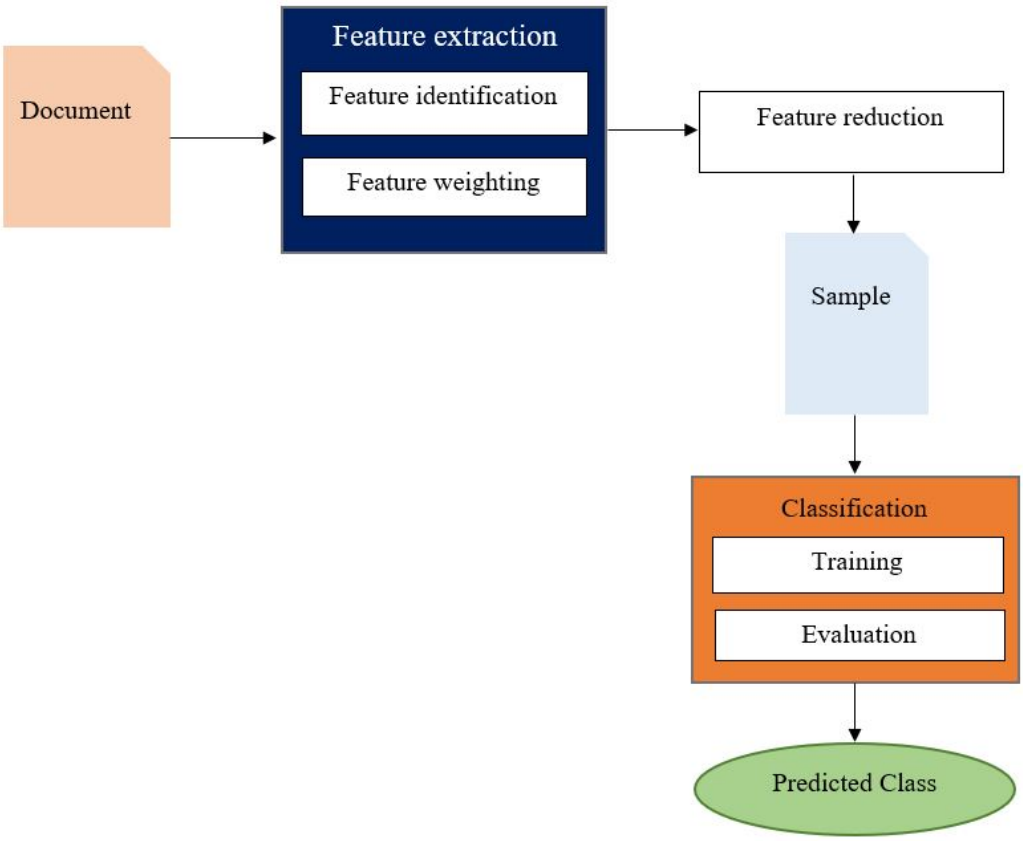


Fig. 2.2 The text classification system architecture

2.5 Text Representation

Text documents are very complex to be processed directly since they comprise unstructured data that need to be converted to numeric format before being fed to any machine learning or deep learning algorithm. This step is considered a critical step in text mining applications. To do so, there are different ways to convert text data into a vector format. First, text can be represented as a string of characters that is the simplest way to represent text as it can be used to represent any textual data in any natural language. However, this representation is often inefficient when it comes to semantic analysis, which is needed in a text categorization task. A word-level representation is the first level of semantic richness as words are the basic unit of any human communication natural language. For this purpose, other techniques were proposed, such as BOW or term frequency.
Next, we describe in details the most popular text representation methods.

2.5.1 Bag-Of-Words

Bag-Of-words is a technique where the sentence is represented by the occurrence or absence of a word in the given sentence. This technique first constructs a corpus containing all the used words in sentences. Then, the occurrence of the word is marked as 1, while its absence is marked as 0. This technique considers each word as a potential independent feature. BOW uses to be merged with stemming or lemmatization algorithm and dimensionality reduction techniques since it suffers from the curse of dimensionality problem due to the large set of resulting features and the meaning losing. This technique fails to capture the sequence and meaning present in the document. Moreover, the information about the frequency of the most repeated words in the document is not retained.

2.5.2 Term Frequency and Inverse Document Frequency

To overcome the limitations of the BOW technique, Term Frequency and Inverse Document Frequency was proposed. It is a numerical statistic that reflects the importance of a word to a document or corpus by calculating the number of times the given word occurred in a particular document and how rarely it occurs in all other documents. TF-IDF for a word in a document is calculated by multiplying two different metrics: The term frequency and the inverse document frequency. The term frequency of a word in a document. It calculates the number of times a term occurs in a document. Then, there are ways to adjust the frequency by the length of a document or by the raw frequency of the most frequent word in a document. Term frequency is given by the following equation:

$$TF(t,d) = 0.5 + 0.5\frac{f_{t',d}}{max\{f_{t',d} : t' \in d\}} \tag{2.5}$$

where $f_{t,d}$ is raw count.
The inverse document frequency of the word across a set of documents. This means how common or rare a word is in the entire document set. This metric can be calculated by taking the total number of documents and dividing it by the number of documents that contain this word. It is given by:

$$IDF(t,D) = log\frac{N}{|\{d \in D : t \in d\}|} \tag{2.6}$$

Where N is the total number of documents and the $|\{d \in D : t \in d\}|$ the number of documents where the term t appears. If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$.

Then TF–IDF is computed as:

$$TF - IDF = TF(t, d).IDF(t, D) \tag{2.7}$$

Unlike BOW model, TF-IDF converts the text into vectors based on term frequency and inverse document frequency formula rather than 1 or 0. TF-IDF outperforms in most of the cases the BOW model since the scores are calculated from the TF-IDF model differentiates a particular document from its counter one. For instance, the food text contains the words related to food, and in the same way, the business text contains the words related to business such as دولار، منتجات، مال, etc. While finding the vectors using the TF-IDF vectorizer, the words such as كرة ، ملعب were given high scores in the sports article at the same time the words such as دولار، أسهم were given a high score in the business texts. TF-IDF also reduces the importance of stop-words; even though the term frequency for stop-words is very high, the inverse document score is very low, the overall product of term frequency and inverse document frequency will be low for stop-words. However, TF-IDF vector-based representation suffers from the drawback of losing syntactic and semantic relationships among words. Furthermore, the sparse representation of a document or sentence leads to the curse of dimensionality. To deal with these limitations, some researchers argue that the derived features of LSA, which are linear combinations of the original TF-IDF features, can capture some aspects of basic linguistic notions such as synonymy and polysemy.

2.5.3 Latent Semantic Analysis

Latent Semantic Analysis is a classical tool for automatically extracting similarities between documents through dimensionality reduction [72]. To do so, a term-document matrix is constructed using weights corresponding to the importance of the term in the specific document (TF-IDF) and then is reduced via singular value decomposition to a lower-dimensional space called the concept space. More formally, the term-document matrix X of dimension $n.m$ (n terms and m documents) can be decomposed into U and V orthogonal matrices and Σ a diagonal matrix through singular value decomposition as follow:

$$X = U\Sigma V^T \tag{2.8}$$

This can be represented through a rank proximation of a smaller dimensionally space. This approximation is written as:

$$X_k = U_k\Sigma V_k^T \tag{2.9}$$

This representation is often used to compare documents in this new space, but as the problem is symmetrical, it can be used to compare words. The U_k matrix of dimensions $n.k$ represents the model of words in the new k-dimensional concept space. We can thus compare the relative similarity of each word by taking the cosine distance between their representations.
However, this method has some limitations since it is a linear model, so not the best solution to handle non-linear dependencies. Moreover, the latent topic dimension cannot be chosen for arbitrary numbers. It depends on the rank of the matrix, so it cannot go beyond that.

2.5.4 Probabilistic Latent Semantic Analysis

Probabilistic latent semantic analysis is a statistical technique proposed by [60]. It derives a low-dimensional representation of the observed variables regarding their affinity to certain hidden variables, just as in latent semantic analysis. Compared to standard latent semantic analysis, which stems from linear algebra and downsizes the occurrence tables (usually via a singular value decomposition), probabilistic latent semantic analysis is based on a mixture decomposition derived from a latent class model. More formally, considering observations in the form of co-occurrences (w, d) of words and documents, PLSA models the probability of each co-occurrence as a mixture of conditionally independent multinomial distributions:

$$P(w, d) = P(d) \sum_{c} P(c|d)P(w|c) \tag{2.10}$$

Where c being the words' topic and d is a given document. Note that the number of topics is a hyperparameter that must be chosen in advance and not estimated. For each document d, a latent class is chosen conditionally to the document according to $P(c|d)$, and a word is then generated from that class according to $P(w|c)$.
While Hofmann's work is a useful step toward probabilistic modeling of text, it is incomplete since it provides no probabilistic model at the level of documents. Because each document is represented as a list of numbers, however, there is no generative probabilistic model for these numbers. This results several issues, including the increase of the number of parameters in the model linearly with the size of the corpus that leads to the overfitting, and it is not clear how to assign a probability to a document outside of the training set. To overcome all these limitations, LDA was proposed.

2.5.5 Latent Dirichlet Allocation

Latent Dirichlet Allocation is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words [18].
The input of LDA is a collection of documents and a few parameters. The output is a probabilistic model describing (a) how much words belong to topics and (b) how associated topics are with documents. A list of topics, often containing some topics repeatedly, is generated at random on the basis of (b). That list is the same length as the number of words in the document. Then, that list of topics is transformed into a list of words by turning each topic into a word on the basis of (a). LDA is a generative model. This means that it works with the probability of observing a particular dataset given some assumptions about how that dataset was created. At its core is the assumption that a document is generated by a small number of "topics". An LDA "topic" is a probability distribution, assigning to each word in the collection vocabulary a probability. Topics are considered hypothetical and unobservable, saying that they do not actually exist in documents. This means that, firstly, we know that documents are not actually generated from a set of topics. Instead, we are using the concept of a topic as a simplified model for what must be a more complicated process, the process of writing a document. Secondly, documents do not contain information about what topics are present, what words those topics contain, and how much of each topic is in each document. Therefore, we must infer the topic characteristics from a collection of collections of words. Each document is usually assumed to be generated by a few of the total number of possible topics. So, every word in every document is assumed to be attributable to one of the document's topics. However, LDA is computationally very expensive on large data sets. For this reason, word and document embeddings models were proposed.

2.5.6 Word2vec

The challenge for the natural language processing researchers is to understand the meaning of the text by capturing the semantic relationship between words within the sentence. Word Embedding is one solution where words or phrases from the vocabulary are mapped to vectors of real numbers in a matrix by preserving the semantic and syntactic relationship. The weights are assigned in such a way that the weight of a word is related to the words surrounding it. Word2Vec is one of the most popular technique to learn word embeddings using neural networks that was developed by [87]. Word2Vec uses two model architectures to produce a distributed representation of words: Skip Gram and Common Bag Of Words (CBOW).

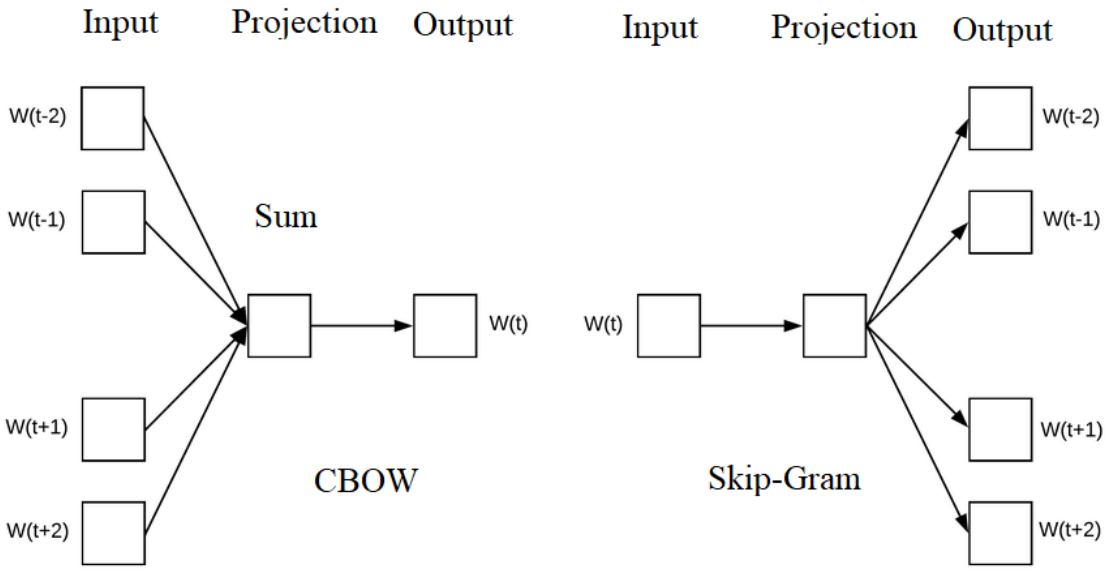


Fig. 2.3 The CBOW and Skip-gram models architectures [87]

2.5.6.1 CBOW

This method takes the context of each word as the input to predict the word corresponding to the context. In the process of predicting the target word, the vector representation of the target word is learned. The training complexity of this algorithm is calculated as:

$$Q = N \times D + D \times log_2(V) \tag{2.11}$$

Where V is the size of the vocabulary, N are words that are encoded previously, and D are the word representations. It is called bag-of-words model because the order of words in the history does not influence the projection. Unlike standard bag-of-words model, CBOW uses continuous distributed representation of the context. The model architecture is shown at Fig. 2.3.

2.5.6.2 Skip-gram

This architecture is similar to CBOW, but instead of predicting the current word based on the context, it attempts to maximize classification of a word using another word in the same sentence. Each current word is utilized as an input to a log-linear classifier with continuous projection layer in order to foresee words within a certain range before and after the current word. Increasing the range improves quality of the resulting word vectors, however it also leads to the computational complexity raise. The training complexity of this architecture is:

$$Q = C \times (D + D \times long_2(V)) \tag{2.12}$$

Where C is the maximum distance of the words.

2.5.7 Glove

Glove is a model for distributed word representation that is proposed by [102]. The model is an unsupervised learning algorithm for obtaining words vector representations. This is achieved by mapping words into a meaningful space where the distance between words is related to semantic similarity. More precisely, it is based on matrix factorization techniques on the word-context matrix. A large matrix of co-occurrence information is constructed. Each "word" (the rows) and how frequently it has been seen in some "context" (the columns) are counted. Usually, we scan our corpus in the following manner: for each term, we look for context terms within some area defined by a window-size before the term and a window-size after the term. Also, we give less weight for more distant words. To reduce the representation space dimensionality the matrix of context is factorized to yield a lower-dimensional matrix, where each row now yields a vector representation for each word. In general, this is done by minimizing a "reconstruction loss". This loss tries to find the lower-dimensional representations.

2.5.8 Doc2vec

Doc2vec [77] is unsupervised model to learn distributed and continuous representation for document regardless of its length based on the word2vec model. However, documents do not come in logical structures such as words, so the basic idea was to use the word2vec model, and added another Paragraph ID vector as illustrated in Fig. 2.4. The model in the figure is a small extension to the CBOW model. But instead of using just words to predict the next word, another feature vector which is document-unique is added. Therefore, the document vector D is trained simultaneously with the word vector W, and in the end of training, it holds a numeric representation of the document. This model is called PV-DM. It acts as a memory that remembers what is missing from the current context or as the topic of the paragraph. While the word vectors represent the concept of a word, the document vector intends to represent the concept of a document. As illustrated in 2.4, another algorithm that is similar to skip-gram word2vec model may be used. It is called PV-DBOW, this algorithm is faster and consumes less memory, since it does not need to save the word vectors.

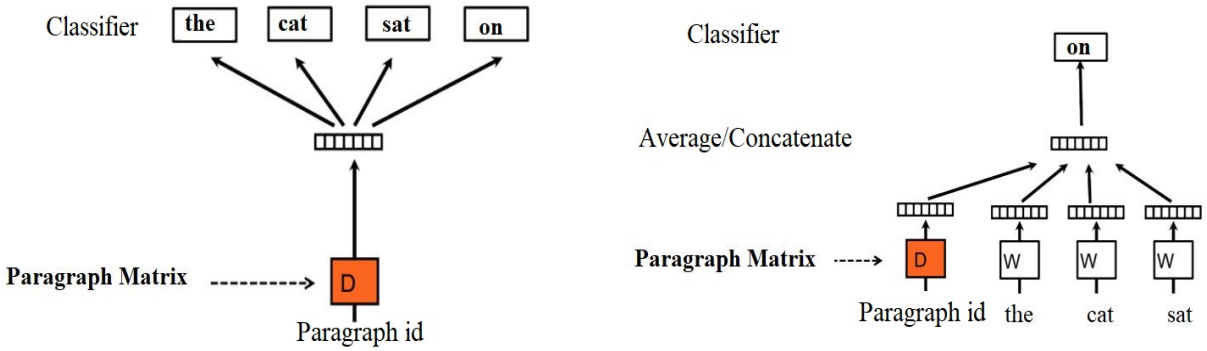


Fig. 2.4 PV-DM and PV-DBOW model architectures [77]

Before diving into the current and emerging text representations based on transfer learning techniques, we present in Table 2.1 the comparison between the above text representations by giving the advantages and limitations of each method.

2.5.9 Contextualized Word Representations

2.5.9.1 ELMO

Contextualized word representations are another word embedding techniques that are intended to produce word representation based on word context. The context2vec [86] is an unsupervised model for efficiently learning generic context embedding of wide sentential contexts, using bidirectional LSTM. [103] built upon this technique to create the deep contextualized word representations technique. The main idea behind these word embedding techniques is that the resulting word vectors are learned from a bidirectional language model (biLM), which consist of both forward and backward language models. These forward L-layer LSTM and backward L-layer LSTM are applied to encode the left and right contexts, respectively. The contextualized representations are the concatenation of the left-to-right and right to-left representations. To utilize ELMo in downstream tasks, the (L + 1)- layer representations (including the global word embedding) for each token k are aggregated as:

$$ELMO_k^{task} = \gamma^{task} \sum_{j=0}^{L} S_j^{task} h_{k,j} \tag{2.13}$$

Where S^{task} task are layer-wise weights normalized by the softmax used to linearly combine the (L+1)-layer representations of the token k, $h_{k,j}$ are the hidden representations and γ^{task} is a task-specific constant.
Given a pre-trained model ELMo, it can be incorporated into a task-specific architecture for improving the performance.

Table 2.1 Comparaison between state-of-the-art text representation methods

Approaches	Methods	Advantages	Limitations
Classical approach	-Bag-of-words -TFIDF - N-gram	-Widely used -Still accomplish highly competitive results	-Lack of semantic -High dimensionality of features space
Statistical approach	-LSA -PLSA -LDA	-Dimensionality reduction -Extracts semantic	-Insufficient revealed semantic in large dataset. -Exact inference is intractable -Fix the number of components -It is difficult to fit the models to data
Semantic approach	-Bag-of-concepts	-Integrate semantic -Enrich vocabulary	-Insufficient resources -Problems of mapping
Implicit semantic	-Word2vec -Doc2vec	-Provide highly discriminative features -Low dimensional vectors -Learn pertinent implicit text semantics	-Cannot preserve contextual information -Fitting Out-Of-Vocabulary words is difficult

2.5.9.2 BERT

BERT [33] proposes a MLM objective, where some of the tokens of the input sequence are randomly masked, and then these masked positions are predicted taking the corrupted sequence as input. BERT applies a Transformer encoder to attend to bi-directional contexts during pre-training. In addition, BERT uses an NSP objective. Given two input sentences, NSP predicts whether the second sentence is the actual next sentence of the first sentence. Moreover, BERT uses special tokens to obtain a single contiguous sequence for each input sequence. Specifically, the first token is always a special classification token [CLS], and sentence pairs are separated using a special token [SEP]. BERT adopts a pre-training followed by fine-tuning scheme. The final hidden state of [CLS] is used for sentence-level tasks and the final hidden state of each token is used for token-level tasks. BERT obtains new state-of-the-art results on several natural language processing tasks such as question answering and sentiment analysis.

2.5.9.3 RoBERTa

RoBERTa is new variant of BERT. [83] proposed to makes some changes range from training the model longer with larger batches and more data, removing the NSP objective, training on longer sequences to dynamically changing the masked positions during pretraining.

2.5.9.4 DistilBERT

DistilBERT [113] is a method to pre-train a smaller general purpose language representation model, which can then be finetuned with good performances on a wide range of tasks like its larger counterparts. This model leverage knowledge distillation during the pre-training phase and show that it is possible to reduce the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster.

2.5.9.5 ALBERT

ALBERT is another variant of BERT in which [71] proposed two parameter-reduction techniques including factorized embedding parameterization and cross-layer parameter sharing to lower memory consumption and speed up training. Furthermore, ALBERT uses a sentence-order prediction objective instead of the NSP objective since they argue that NSP lacks difficulty.

2.5.9.6 XLNet

The XLNet model is a generalized autoregressive pretraining method. [135] identified two weaknesses of BERT: (1) BERT neglects dependency between the masked positions and suffers from a pretrain-finetune discrepancy and (2) The symbols such as [MASK] are introduced by BERT during pre-training, yet they never occur in real data, resulting in a discrepancy between pre-training and fine-tuning. To overcome these limitations, XLNet proposes a new auto-regressive method based on permutation language modelling [126] without introducing any new symbols. Furthermore, XLNet integrates ideas from Transformer-XL [30] into pretraining to learn long-range dependencies.

2.5.9.7 BART

The BART model [81] introduces additional noising functions beyond MLM for pre-training sequence-to-sequence models. First, the input sequence is corrupted using an arbitrary noising function. Then, the corrupted input is reconstructed by a Transformer network trained using teacher forcing [132]. BART evaluates a wide variety of noising functions, including token masking, token deletion, text infilling, document rotation, and sentence shuffling (randomly shuffling the word order of a sentence). The best performance is achieved by using both sentence shuffling and text infilling.
We present a comparaison between the contextual embedding methods in Table 2.3.

Table 2.2 Comparaison between contexrual embbeding representations

	BERT	RoBERTa	DistilBERT	ALBERT	XLNet
Size (millions)	-Base: 110 -Large: 340	-Base: 110 -Large: 340	-Base: 66	-Base: 12 -Large: 18	-Base~110 -Large~340
Training Time	-Base: 8×V100×12d -Large: 280×V100×1d	1024 ×V100×1d (4-5 times more than BERT)	-Base: 8×V100× 3.5d (4 times less than BERT)	-Large: 1.7 times faster than BERT	-Large: 512 TPU chips×2d (5 times more than BERT)
Method	Bidirectional Transformer with MLM and NSP	BERT without NSP, using dynamic masking	BERT distillation	BERT with reduced parameters, uses sentence-order prediction objective instead of the NSP	Bidirectional Transformer with permutation-based modeling
Data	BooksCorpus+English Wikipedia=16GB	BERT+ 144 GB additional=160GB	16GB BERT	16GB BERT	-Base: 16GB BERT -Large: 16GB BERT+97GB additional

2.6 Feature Selection

Feature selection methods enable to identify correctly the relevant features in a text. This has a vital importance to the task of text categorization. FS methods used in data pre-processing to achieve efficient data reduction and to improve text categorization performances. Next, we provide a brief overview of several popular feature selection methods commonly used for text categorization.

2.6.1 Pruning

Pruning methods are typically applied prior on the feature set to eliminate very rare words (word appearing two or fewer times) and very frequent words (for example stop-words). Since, these features are not usually relevant and useful for classifying a text document. The choice of elimination threshold can have an effect on categorization accuracy, so it is crucial to choose it properly. Eradicating extremely frequent words can be done using a threshold or a list of stopwords that are language and domain specific.

2.6.2 Chi-Square

Chi-Square is a statistical feature selection method that is widely applied within the field of text classification. The Chi-Square (χ^2) statistics measures divergence from the expected distribution, assuming that feature occurrence is independent of class value. The basic assumption is that a term whose frequency strongly depends on the category in which it occurs will be more useful for discriminating it [49]. The Chi-Square measures the lack of independence between a term t and a category c as follow:

$$\chi^2(t,c) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)} \tag{2.14}$$

Where A is the number of times t and c co-occur, B is the number of times the t occurs without c, C is the number of times c occurs without t , D is the number of times neither c nor t occurs, and N is the total number of documents. The χ^2 statistic has a natural value of zero if t and c are independent.

2.6.3 Mutual Information

Information theory was initially developed to find fundamental limits on compressing and reliably communicating data [120]. Entropy is a key measure of information in this theory. Since it is able of quantifying the uncertainty of random variables and scaling the amount of information shared by them effectively and it is calculated as follow:

$$I(t,c) = log\frac{(A \times N)}{(A+C) \times (A+B)} \tag{2.15}$$

I statistic has a natural value of zero if t and c are independent.

2.6.4 Information Gain

Information gain is a method to measures the term goodness. It measures the number of bits of information obtained for category prediction by knowing the presence or absence of term in a textual document. It can generalize to any number of classes [49]. Given a term t_j and a category c_k, the information gain $IG(t_j, c_k)$ is defined by:

$$IG(t_j,c_k) = p(t_j,c_k)log(\frac{p(t_j,c_k)}{p(t_j)p(c_k)}) + p(\bar{t}_j,c_k)log(\frac{p(\bar{t}_j,c_k)}{p(\bar{t}_j)p(c_k)}) \tag{2.16}$$

Only the words for which the value of the criterion is the most important are considered as characteristics for c_k.

2.6.5 Document Frequency Thresholding

This method relies on the hypothesis according to which a term belonging to a few numbers of documents is not a good feature for the categorization task [73]. So, only the terms, that appear in a number of documents higher to a defined threshold, are selected. This threshold can be determined using a training set. Given a term t_j, this criterion can be computed globally on the collection $DFG(t_j)$ or on each category c_k $DFL(t_j, c_k)$:

$$DFG(t_j) = p(t_j) \tag{2.17}$$

$$DFL(t_j, c_k) = p(t_j|c_k) \tag{2.18}$$

2.6.6 Variance Threshold

The main idea of VT method is that a feature which does not vary much within itself, it generally has very little predictive power. Variance is a simple baseline approach for feature selection. It removes all features whose variance does not meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples. Variance threshold doesn't consider the relationship of features with the target variable.
In this work we use Chi-square and VT since they have shown to be effective for Arabic text categorization.

2.7 Classifiers Overview

Machine learning algorithms are modeled to perform classification assignments. Several classical algorithms were proposed to deal with text categorization such as Logistic Regression for binary classification, Decision Trees, Random Forest, Support Vector Machines, Naive Bayes, and Boosting algorithms. In this section, we present an overview of these classification algorithms.

2.7.1 Logistic Regression

Logistic regression is a technique used for the binary classification to predict output in the range of 0 and 1. The logistic regression classifier utilizes the weighted combination of the input features and passes them through a sigmoid function. Sigmoid function transforms any real number input to a number between 0 and 1.

2.7.2 Decision Trees

Decision trees algorithms match a textual document x_i against the successive nodes of a tree to decide whether a document belong to the class or not. A decision tree is a tree whose internal nodes are tests and whose leaf nodes are categories. The decision taken in a node is based on the value of some feature or by applying a decision rule. For instance, we use the information gain to build a CART tree [117] by selecting the features which reduce the entropy among possible classes and then the tree will split nodes until reaching a final node that is associated to a class. Another model is C4.5 [112] that produces trees by varying numbers of branches per node. To reduce the over-fitting, both trees have to be pruned once generated.

2.7.3 Random Forest

Random forest classifier comprises an ensemble of decision trees. At each leaf of Random Forest classifier, it uses a voting mechanism to decide on the outcome. Random forest works most of the times better than decision tree classifier as the output is based on an ensemble of trees. Random forest is also a simple approach of bootstrapping as at each stage the algorithm output is decided on different decision tree outputs with varied sampling rate.

2.7.4 Naive Bayes

Bayesian classifiers [51] assign the most likely class to a given example described by its feature vector. This classifier uses a probabilistic way to estimate the probability of output class label. Naive Bayes classifier assumes conditional independence between the input attributes given the class labels to avoid any level of overfitting. Naive Bayes classifier has shown to be successful in text classification challenges despite its unrealistic assumption. Naive Bayes classifier works on Bayes theorem defined as follow:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.19}$$

The output probability of a given class c is calculated as follow:

$$P(X|c) = \prod_{i=1}^{n} P(X_i|c) \tag{2.20}$$

Where $X = (X_1, X_2, \ldots, X_n)$ is a feature vector and n is the number of possible class labels.

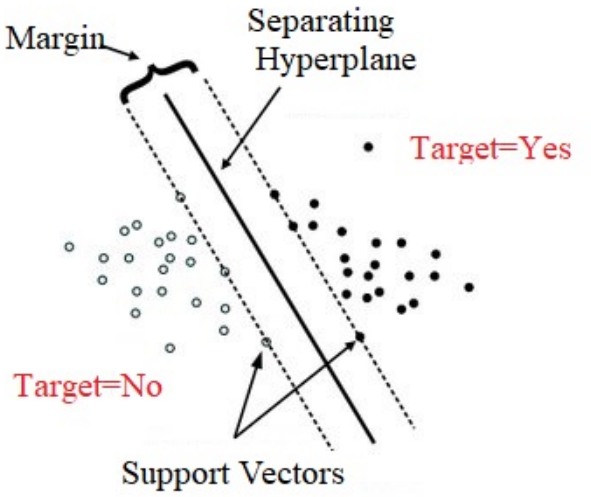


Fig. 2.5 Two possible margins linearly separate positive and negative samples

2.7.5 Support Vector Machines

Support vector machines applied the theory developed by [127] and later [62] proposed them for text categorization to take advantages of its robustness in high dimensional spaces. The basic idea is to find those samples (support vectors) delimiting the widest frontier between positive and negative samples in the feature space (see Fig. 2.5). The width of such border is known as the margin hyperplane, and SVM tries to find the maximal margin by applying constraint quadratic optimization. To get the hyperplane $w + b$, the decision function is:

$$\hat{\Phi}(x) = \sum_{i=1}^{S} \alpha_i y_i K(x, w_i) + b \tag{2.21}$$

Where S is the number of support vectors, α_i is a value that makes the algorithm allows a certain error, $w_i + b$ is the hyperplane and K is the kernel function that can map the space to another one in the non-linear cases.
SVM have been achieved good performances in many text classification assignments due to several advantages including their robustness in high dimensional feature spaces and in sparsity cases, considering every feature as important.

2.7.6 Boosting Algorithms

Boosting is a technique in machine learning for merging classifier in an interative manner to improve the performance. The final classifier is composed by several weak classi-

fiers that are generated via the learning process. Adaboost [114] have proven promising results in text classification task. This algorithm works by maintaining a distribution of weights which is updated to force weak learners concentrate on those samples whose classification is "harder". This algorithm focuses on the binary case. Two variants were proposed to deal with multi-class classification including *Adaboost.MR* , which is based in ranking loss minimization and *Adaboost.MH* that relies on Hamming loss minimization. There are more recent boosting algorithms such as SmoothBoost [115] and [50] .

2.8 Evaluation Metrics

The evaluation of a text categorization system relies on test samples which have been already labeled by human experts. [79] presented an interesting review on text categorization evaluation. To perform the evaluation, the start point is the comparison between human-assigned key words and computer-assigned ones. As shown in the contingency table (Table 2.3) four situations are possible:

- TP_i: *True Positive*. Assessments on which system and human expert agree for a label assignment.
- FP_i : *False Positive*. Labels assigned by the system which does not agree with expert assignment.
- FN_i : *False Negative*. Those labels the system failed to predict correctly as they were by expert.
- TN_i : *True Negative*. Non assigned labels that were discarded by the human expert.

Table 2.3 Contingency table

	Assigned by expert?	
Assigned by classifier?	Yes	No
Yes	TP_i	FP_i
No	FN_i	TN_i

We compute those numbers for every category i.
By combining these values, the evaluation metrics such as accuracy, precision (p), recall (r) and f1-score can be computed as follows:

$$p = \frac{TP}{TP + FP} \tag{2.22}$$

$$r = \frac{TP}{TP + FN} \tag{2.23}$$

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.24}$$

Even though most of the systems seem to produce high accuracy, is all due to imbalanced data. The models are reliable on other metrics such as precision, recall, and f1-score. The precision (Eqn. 2.22) indicates the fraction of correct assigned labels while recall (Eqn. 2.23) measures the fraction of expert's labels found by the system. Since, we are looking for a system having both high precision and recall, we consider f1-score that is defined as an overall measure to evaluate the overall performance obtained by the categorization system.

$$f1 - score = \frac{2pr}{p + r} \tag{2.25}$$

Formulae given before provide only measures for a single class. However, we need to do an average over classes. To obtain global values, three averaging approaches exist: micro-averaging, macro-averaging and weighted mean averaging introduced first by [80].

2.8.1 Macro-averaging

Macro-averaging works by computing the evaluation measure first at class. Then, all the measures are averaged into a global one. Hence, the precision and recall will be calculated as follows:

$$precision_j = \frac{\sum_i TP_{ij}}{\sum_i TP_{ij} + \sum_i FP_{ij}} \tag{2.26}$$

$$recall_j = \frac{\sum_i TP_{ij}}{\sum_i TP_{ij} + \sum_i FN_{ij}} \tag{2.27}$$

$$precision^M = \frac{\sum_j precision_j}{n} \tag{2.28}$$

$$recall^M = \frac{\sum_j recall_j}{n} \tag{2.29}$$

Where n is the total number of document in the collection and j is a class.

2.8.2 Micro-averaging

This method computes defined measures by adding all the individual TP_i, TN_i, FP_i and FN_i of every class to global ones. The micro precision and recall are given by:

$$precision^m = \frac{\sum_i \sum_j TP_{ij}}{\sum_i \sum_j TP_{ij} + \sum_i \sum_j FP_{ij}} \tag{2.30}$$

$$recall^m = \frac{\sum_i \sum_j TP_{ij}}{\sum_i \sum_j TP_{ij} + \sum_i \sum_j FN_{ij}} \tag{2.31}$$

Where TP_{ij}, FP_{ij} and FN_{ij} are the number of true positives, false positives and false negatives found for the class i in the evaluation document j.

2.8.3 Weighted-averaging

Weighted precision and recall give the weighted mean of precision and recall with weights equal to class probability.

2.9 Deep Learning for Text Categorization

2.9.1 Deep Learning Definition

Deep learning [78] is a subdomain of ML that incorporates computational models and algorithms imitating the architecture of the biological neural networks in brain, namely Artificial Neural Networks. The brain tries to compare the new received information with already known information to make sense of it. The brain decodes the information through labelling and assigning the items to various categories, and DL follows the same concept. Deep learning refers to the number of layers in an Artificial Neural Network. A deep leaning model contains three types of layer: the input layer which receives the input data, the output layer that produces the result of data processing and the hidden layer which extracts the patterns within the data. Deep neural networks are able to perform more complex tasks since the data move from one hidden layer to another to recompose complex features. DL systems enable machines to process data with a nonlinear approach and work exceptionally well on unstructured data and have higher accuracy than ML algorithms.

2.9.2 Training Artificial Neural Networks

Neural networks learn the desired function using big amounts of data and an iterative algorithm called backpropagation [108]. First, we feed the network with data then we compare the output with a desired one based on a loss function. Afterwards, the weights are readjusted based on the difference. The adjustment of weights is performed using a non-linear optimization technique called stochastic gradient descent that is the quintessential algorithm for training neural networks. However, this algorithm does not guarantee to reach a global minimum when optimizing non-convex error function especially in deep neural networks training, stochastic gradient descent updates the model parameters θ as follows [105]:

$$\theta_{(i+1)} = \theta_i \mu_i \nabla(\theta_i) \tag{2.32}$$

The i sub-index denotes the sequential nature of the training algorithm, where the parameters are updated iteratively following the above equations. To start, θ_i parameters are randomly initialized and the backpropagation algorithm finds the update direction $\nabla(\theta_i)$ by means of computing the gradient of the error function with respect to the parameters to optimize. Finally, μ_i is the learning rate that controls how much the parameters θ_i are updated.

Over the years researchers came up with amazing improvements in the field of deep learning and proposed different architectures. Each architecture was targeted on a specific problem. We devote the next section to present an overview of using artificial neural network for text categorization.

2.9.3 History of Deep Neural Networks for Text Categorization

Deep learning models have achieved promising performances and outperformed classical machine learning based approaches in various text classification tasks such as sentiment analysis, topic categorization and question answering.

Various works investigated the use of the word and document embeddings [87, 88, 77] for text categorization. For instance, [43] explored both word vectors averaging and document embeddings to enhance Arabic text categorization using a large Arabic text corpus. [36] also investigated the usage of document embeddings, word senses, and word sense disambiguation to train text representations. However, these models do not consider contextual information. Moreover, as documents can vary significantly in length and word composition, it is difficult to control for over/under-fitting without running further diagnostics that add additional complexity.

Several works explore the MLP for text classification. [100] introduced a process for

text classification to classify Mobile SMS. The system comprises several steps as preprocessing and assignment weight to each term and finally classification using the MLP. [101] presented a work on the for the task of text classification. They proceeded by text preprocessing and then applied term weighting methods such as TF and TF.RF. Finally, they built an MLP with 20 hidden layers. For the Arabic language, [89] proposed three techniques to classify Arabic text including: support vector machines, naïve bayesian, and MLP. The authors constructed an MLP containing three layers with hyperbolic tangent activation function in the hidden layer, followed by a linear output layer. The number of input layers is 400 and 600 and the number of output layers is 8 (number of categories in corpus).

Recurrent Neural Network based models consider text as a sequence of words and capture word dependencies and text structures for text classification. There exist two popular RNN models namely, Long Short-Term Memory and Gated Recurrent Unit. Those models were widely utilized for text classification in several works. [122] proposed three RNN-based architectures to leverage supervised data from many related tasks and model text sequence with multi-task learning. [93] proposed a method a classify text using LSTM network and their modifications including Bidirectional LSTM network and Gated Recurrent Unit. The authors compared their method against BOW. [119] have built a Tree-LSTM model which is a generalization of LSTM to tree-structured network typologies in order to learn rich semantic representations. The authors argue that Tree-LSTM model outperforms chain-structured LSTM for NLP tasks because natural language exhibits syntactic properties that would naturally combine words to phrases. The authors validate the effectiveness of the proposed model on two tasks including sentiment classification and predicting the semantic relatedness of two sentences. Besides, [147] proposed to extend the chain-structured LSTM to tree structures using a memory cell to store the history of multiple child cells or multiple descendant cells in a recursive process. They proved that their model provides a principled way of considering long-distance interaction over hierarchies. [145] introduced a Bidirectional LSTM model with two-dimensional max pooling to reflect text features. [28] proposed a machine reading simulator that processes text incrementally from left to right then performs shallow reasoning with memory and attention. Their method extends the LSTM architecture with a memory network in place of a single memory cell. [63] explore a text region embedding method using LSTM for supervised and semi-supervised text categorization. [149] explored unified structure to investigate the effects of word embedding and GRU for text classification. [123] proposed a full attention-based bidirectional GRU neural network, namely FABG. FABG utilizes a Bi-GRU to learn the semantic information of text and full attention mechanism to learn the weights of previous and current outputs of the Bi-GRU at each step. Finally, the text representation is produced

through a pooling layer. The authors argue that FABG can learn more information that enhances the effect of text classification.

RNNs are trained to recognize patterns across time, whereas Convolutional Neural Networks learn to recognize patterns across space. These latter are proven to be successful in image classification, face recognition, object detection and other applications. Even though Convolutional networks are influential in the field of computer vision, they are used in text processing applications as well. CNNs achieve state-of-the-art results on problems such as text classification. Various researches studied the CNN in this area. [65] proposed one of the first CNN-based models for text classification. This model is called the Dynamic CNN (DCNN) and uses dynamic k-max pooling. DCNN consists of several layers the first one constructs a sentence matrix using the embedding for each word in the sentence. Then convolutional layers with dynamic pooling layers given by dynamic k-max pooling is used to generate a feature map over the sentence. [68] introduced a much simple CNN-based model for text classification. This CNN model utilizes only one layer of convolution on top of word vectors obtained from Word2vec. Kim also compared four different approaches to learning word embeddings: (1) CNN-rand, where all word embeddings are randomly initialized and then modified during training; (2) CNN-static employs Word2vec embeddings that stay fixed during model training; (3) CNN-non-static, where the Word2vec embeddings are fine-tuned during training for each task; and (4) CNN multi-channel, where two sets of word embedding vectors are utilized, both are initialized using Word2vec. [82] presented a new CNN-based model that makes two modifications to the architecture of [68]. The authors adopted a dynamic max-pooling scheme to captures more fine-grained features from different regions of the text. In addition, they inserted a hidden bottleneck layer between pooling and output layer to reduce model size and improve model performance. [142] explored character-level CNNs for text classification. Firstly, the authors feed the characters in a fixed-sized to the model, encoded as one-hot vectors, then passes them through a deep CNN model. [29] presented a Very Deep CNN model for text processing. It operates directly at the character level and uses only small convolutions and pooling operations. This study shows that the performance of CNN increases with the depth. [144] examined the impact of different word embedding methods and pooling mechanisms on CNN architecture. They proved that Word2vec and GloVe outperforms one-hot vectors and that max-pooling surpasses other pooling techniques. [53] investigated the impact of word embeddings and proposed to use them via a multi-channel CNN model. [130] proposed a CNN-based model that combines explicit and implicit representations of short text for classification. For the Arabic language, [97] presented a deep convolutional network for Arabic Sentiment Analysis. Also, [46] proposed deep learning models including CNN for single-label as well as multi-label text classifica-

tion. Many Hybrid models have been developed to combine LSTM and CNN architectures to capture local and global features of sentences and documents. [147] proposed a Convolutional LSTM (C-LSTM) network that uses a CNN to extract a sequence of higher-level phrase (n-gram) representations, which are fed to a LSTM network to obtain the sentence representation. Similarly, [144] proposed a Dependency Sensitive CNN for document modeling. This model is a hierarchical that uses LSTM to learn the sentence vectors which are fed to the convolution and max-pooling layers to generate the document representation. [27] developed a CNN-RNN model for multi-label text categorization. [122] merged a CNN to learn sentence representations, and a gated RNN to learn a document representation. For Arabic, [9] proposed a combined CNN and LSTM model for arabic sentiment analysis. [10] studied several deep neural networks in conjunction with stemming methods for Arabic text categorization.
One of the computational bottlenecks suffered by RNNs is the sequential processing of text. CNNs are less sequential than RNNs, Although the computational cost to capture relationships among words in a sentence increases with length of the sentence raise. Transformers [128] overcome this drawback by applying self-attention to allow Transformers to be more parallelization than CNNs and RNNs, which makes it possible to efficiently train very big models on large amounts of data on GPU. [136] proposed a hierarchical attention network for text classification. This model has an hierarchical structure that mirrors the hierarchical structure of documents. Moreover, two levels of attention mechanisms applied at the word and sentence-level to attend more and less important content when constructing the document representation. [146] extended the hierarchical attention model to cross-lingual sentiment classification. [129] introduced an attention framework that measures the compatibility of embeddings between text sequences and labels. One of the most widely used autoencoding transformers is BERT [33]. It achieved state-of-the-art on sentiment analysis and other NLP tasks. [12] developed an Arabic version of BERT namely Arabert and tested it on several NLP assignments for Arabic.

2.10 Conclusion

In this chapter, we first presented the fundamental text categorization concepts and described the general architecture of a text categorization system that consists of several modules. We introduced text representation methods that fall into three categories: classical methods such as BOW and TF-IDF, latent methods including LSA, PLSA, and LDA, and embedding methods. We presented the most known feature selection methods. Moreover, we cited the most important classical algorithms for text classification and shed light on the evaluation metrics. Besides, we present deep learning before div-

ing into the history of neural networks for text categorization. We noticed that research works on the Arabic language are limited. Almost the methods mentioned above do not consider the information embedded in lexical datasets as Arabic WordNet. Indeed, exploring this information can enhance ATC performances. Furthermore, almost all works focus on sentiment analysis tasks. To overcome these shortcomings, in our first contribution [35], we propose a deep Autoencoder text representation and classical classifiers for Arabic text categorization. Our first work's promising performances were a big motivation to enhance the obtained results by integrating more sophisticated feature selection methods and adopting BOC representation. Also, using deep neural networks in the categorization phase helped to boost the performance. In the next chapter, we present in detail this contribution.

CHAPTER 3

A Deep Autoencoder-Based Representation for Arabic Text Categorization

3.1 Introduction

In the former chapters, we introduced the most important notions for text categorization and Arabic NLP. Moreover, we introduced the most important modules in a text categorization system. Then, we afforded an overview of the deep learning-based methods for text categorization.
Text representation is a fundamental assignment in text categorization. Therefore, to enhance Arabic text categorization, it is necessary to build an efficient text representation reducing the feature space dimensionality and reflecting text semantics. Several text categorization techniques were studied for other languages (English, French...). Nevertheless, research on Arabic text categorization is still limited.
Along with this chapter, we introduce a deep Autoencoder-based representation for Arabic text categorization [41]. The remainder of this chapter is sorted out as pursues. In section 3.2, we introduce the problem and the motivations of our work. Section 3.3 introduces Stacked Autoencoder based on Restricted Boltzmann Machines. Section 3.4 describes the proposed Arabic text categorization method. Section 3.5 exposes the experimental results. Finally, the last section concludes.

3.2 Motivations

Some intriguing properties of the Arabic language are variety, richness, and complex morphology. For this reason, several researches were devoted to proposing methods for Arabic text categorization that is considered a challenging task. These methods fall into these categories, including classical methods, knowledge-based methods, latent representations methods, and deep learning methods. The classical Bag-Of-Words and character-level n-gram approaches were widely used and still accomplish highly

competitive results [6, 2, 137, 95]. However, these representations fail to extract similarities between words and phrases, leading to feature space dimensionality sparsity and explosion. Moreover, by handling words as independent tokens, semantic dependencies cannot be captured. To capture document semantics, several knowledge-based methods were successfully applied to Arabic text categorization [55]. Besides, numerous models for capturing low dimensional and latent document representations as LDA and LSA were proposed [148, 4]. The previous latent representations' major drawbacks range from the inference problem since they cannot handle nonlinear dependencies and the high memory cost to the processing time. To overcome these limitations, distributed word and document representations were proposed [87, 88, 77, 102]. These techniques were explored in the context of Arabic text categorization [43, 36]. However, by averaging together word representations, temporal information is lost, and word order is ignored. Moreover, in Doc2vec as documents can vary significantly in length and word composition, it is difficult to control for over/under-fitting without running further diagnostics that add additional complexity.
To conquer the above shortcomings, we propose an Arabic text categorization method adopting Restricted Boltzmann Machines deep Autoencoder and Bag-Of-Concepts to produce high-level representation. Besides, we investigate deep learning neural networks for Arabic text categorization.

3.3 Stacked Autoencoder with Restricted Boltzmann Machines Pretraining

3.3.1 Restricted Boltzmann Machines

A Restricted Boltzmann Machine is a network that characterizes a layer of unconnected "visible" input units having undirected and symmetrical associations with another single layer of "hidden" units. It refers to a graph with no intra-layer connections [24].
Data in RBM can be initialized randomly, and then alternating Gibbs sampling is performed to update them. Given the current states of the units in one layer, all the units of the other layer are then updated simultaneously. This process is repeated until the entire system is sampling from its equilibrium distribution. RBMs are generative models, meaning that the training period results in a probability distribution of the training data being learned. When the model is later used for testing, it may encounter new, unfamiliar data, but the probability distribution can account for these previously unseen occurrences, yielding a likely, probabilistic output [52].
As shown in Fig. 3.1, RBM network comprises two layers of hidden h_j and visible v_i units. A weight is assigned to each connection between the units of the two layers and

thus a weight matrix $w_{i,j}$ is composed. After that, alternative Gibbs sampling is applied to this matrix to update weights. Given these, a joint configuration (v, h) of the visible and hidden units has energy given by:

$$E(v,h) = \sum_{i \in visible} a_i v_i \sum_{j \in hidden} b_j h_j \sum_{i,j} v_i h_j w_{i,j} \tag{3.1}$$

The network assigns a probability to every possible pair of a visible and a hidden vector via the following equation:

$$p(v,h) = \frac{1}{Z} e^{-E(v,h)} \tag{3.2}$$

where the "partition function", Z, is given by summing over all possible pairs of visible and hidden vectors:

$$Z = \sum_{v,h} e^{-E(v,h)} \tag{3.3}$$

In RBM, the hidden unit activations are mutually independent given the visible unit activations and conversely, the visible unit activations are mutually independent, given the hidden unit activations. So we can calculate the conditional probability of a configuration of the visible units v, given a configuration of the hidden units h: $p(v|h)$, and conversely the conditional probability of h given v: $p(h|v)$. More formally, given m visible units and n hidden units, the activation probabilities for these units are defined in Equation 3.4 and Equation 3.5:

$$p(h_j = 1|v) = \sigma(b_j + \sum_{i=1}^{m} w_{i,j} v_i) \tag{3.4}$$

$$p(v_i = 1|h) = \sigma(a_i + \sum_{j=1}^{n} w_{i,j} h_j) \tag{3.5}$$

Where, σ denotes the logistic sigmoid. v_i and h_j are the binary states of the visible unit i and hidden unit j . a_i and b_j are their corresponding biases.
The probability that the network assigns to a visible vector, v, is given by summing over all possible hidden vectors:

$$p(v) = \frac{1}{Z} \sum_{h} e^{-E(v,h)} \tag{3.6}$$

The weights can be adjusted by:

$$\frac{\delta log p(v)}{\delta W_{i,j}} = < v_i h_j >_{data} - < v_i h_j >_{model} \tag{3.7}$$

$$\Delta W_{i,j} = \epsilon(< v_i h_j >_{data} - < v_i h_j >_{model}) \tag{3.8}$$

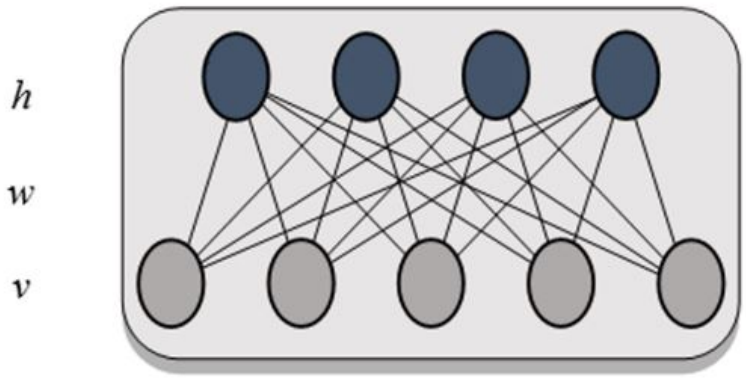


Fig. 3.1 Restricted Boltzmann Machine

Where the angle brackets are used to denote expectations under the distribution specified by the subscript that follows, and ϵ is the learning rate. The same learning rule is applied to the biases a_i and b_j.

3.3.2 RBMs Autoencoder Architecture

An Autoencoder is a feedforward neural network containing one or multiple hidden layers. The main purpose is reconstructing the input data in an unsupervised manner [121]. The Autoencoder consists of an encoder which reverses the input information into low-dimensional representations and a decoder that reconstructs the original information from the encoder output. It is essential to alleviate the distance between the Autoencoder input and output utilizing backpropagation. However, this latter is not appropriate for deep neural systems seeing that the training will be moderate. Hence, RBMs pretraining for each hidden layer by [57] to accelerate the training procedure through parameters initialization was employed. It begins with a traditional one-hidden layer then creates a pile of RBMs, where outputs from the concealed layer are the contributions for preparing the following RBM layer. This greedy layer-bylayer training can be repeated a few times to pretrain a deep model.
After the pretraining, the individual RBMs at each level is unrolled to have a symmetric deep Autoencoder topology. Then, as shown in Fig. 3.2, fine-tuning is used to diminish the input vectors reconstruction error through the whole system utilizing backpropagation. This backpropagation can be performed via square error function or cross entropy function [124]. In this work, we utilized the square error which is characterized as follow in Equation 3:

$$SquareError = \sum_{j} (\hat{x_j} - x_j)^2 \tag{3.9}$$

Where, j is a variable.
$\hat{x_j}$ is the vector of observed values being predicted.

x_j vector of real values.

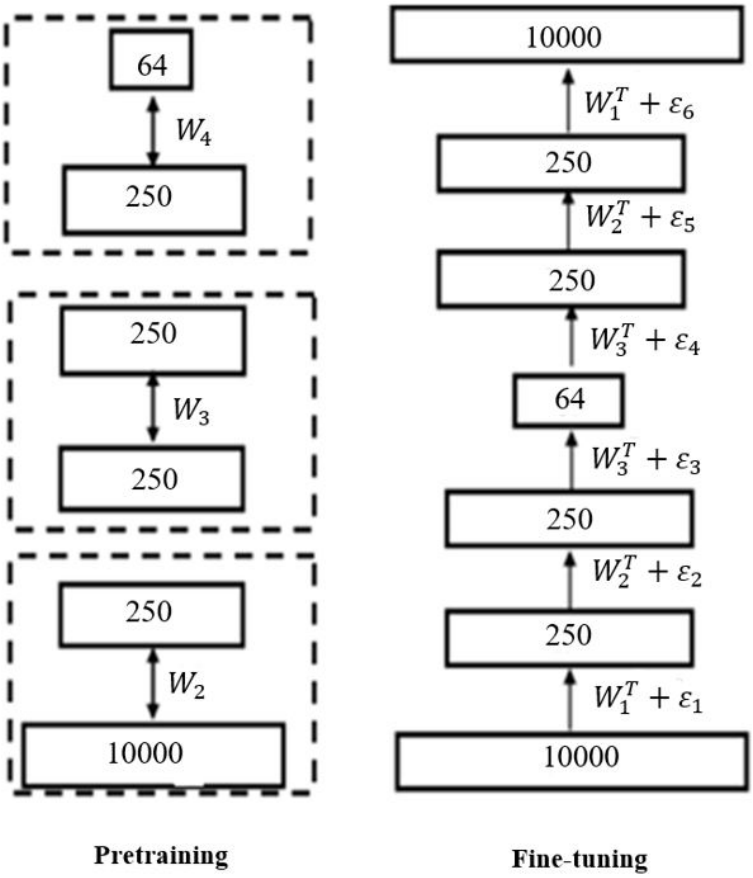


Fig. 3.2 Left panel: Pretraining by learning a stack of Restricted Boltzmann Machines. Right panel: Fine-tuning

3.4 Arabic WordNet

WordNet is an English lexical database where nouns, verbs, adjectives and adverbs are grouped into sets of synonyms called synsets [84]. These synsets are related to each other through different semantic and lexical relations. Therefore, the WordNet can be viewed as a directed graph. The Arabic WordNet is the Arabic version of the English WordNet. The Arabic WordNet database structure is composed of four principal entity types including item, word, form and link. Items are conceptual entities, including synsets, ontology classes and instances. A word entity is a word sense. A form is a special form that is considered as dictionary information. Links are relations between synsets. They are classified according to the part-of-speech of the related synsets (verb, noun, adjective, and adverb) or according to their type (lexical, semantic and lexico-semantic relations). Table 3.1 shows different relations in the Arabic WordNet according to their classification type [20].

Table 3.1 Arabic WordNet relation classification [20]

Type	*Relation*	*Example*	*Frequency*
	Has holo part	أوروبا has holo part فرنسا, إسبانيا	697
	Has subvent	بَلَعَ has subvent أكَلَ	128
	Has instance	مرسيليا has instance مدينة	1067
	See also	عَرَفَ See also إكْتَشَفَ	192
	Causes	ذكَّرَ Causes تَذكَّرَ	75
	Has holo member	has holo الأوروبي الاتحاد فرنساmember,	334
	Verb group	أجَازَ -سَمَحَ	152
	Region term	اليونان Region term حروب البلقان	35
	Category term	catégorie لواء, عقيد جيشtermes	548
	Has holo made of	has holo made of ماء هيدروجين, أكسجين	60
	Be in state	إمْكَان Be in state قادر	83
	Usage term	أسبرين Usage term اسم تجاري	3
Lexical relations	Near synonym	انسان near synonym فرد	122
	Near antonym	أم Near antonym أب	722
Lexico-semantic relations	Related to	إحْتَفَظ related to إحْتَفَاظ	4774
	Has derived	أبَويّ has derived والِدَان	178

3.5 Methodology

As illustrated in Fig. 3.3, our method consists of various modules. First, we incorporate the knowledge contained in Arabic WordNet using Bag-Of-Concepts representation to enrich the vocabulary and feature selection techniques to enhance text vector quality. Then, we use a deep Restricted Boltzmann Machines Autoencoder to learn higher-level feature representation. Afterward, the prepared representations are fed to a deep Autoencoder classifier.

3.5.1 Preprocessing and Feature Selection Module

The first module in our categorization system is the preprocessing and feature selection module. The Arabic text preprocessing consisted of three steps:

- Tokenization;

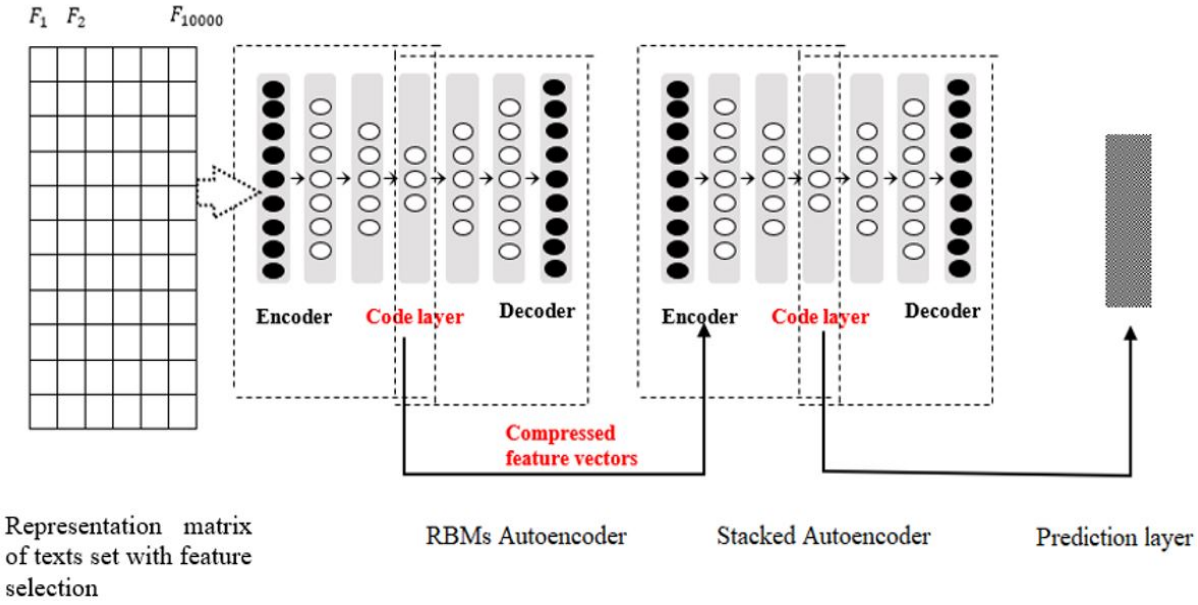


Fig. 3.3 The proposed Arabic text categorization system architecture

- Removing stop-words, punctuation marks, numbers and words written in different languages;
- Normalizing words by replacing letters (أ إ آ) with (ا), letters (ؤ ء) with (ا) and (ا) with (ى).

After text preprocessing and as illustrated in Fig. 3.4, we mapped bag-of-words to bag-of-concepts using Arabic WordNet 2.0.1 to capture the semantics and enrich the Arabic text representations. The mapping strategy added concepts to terms using a threshold to select only words having important frequencies. After that, we performed Chi-Square and VT feature selection techniques to rank the best features by eliminating non-discriminant terms. In this work, the number of chi-square features is fixed empirically at 10000 since it gives the best result. Concerning the VT, we fixed the threshold at 0.001.

3.5.2 Representation Module

As shown in Fig. 3.5, after selecting the best features using Chi-Square or Variance Threshold, the outputted vectors are fed to the Autoencoder module. To train the deep Autoencoder, we use Restricted Boltzmann Machines pre-training, as we mentioned in the previous section. During this step, a pile of Restricted Boltzmann Machines is generated. Then, each Restricted Boltzmann Machine is unfolded to have a symmetric architecture to produce the input vectors resultant from the previous module. Afterward, we fine-tune weights using the square error function to reduce the reconstruction error

between the input and output data. We construct a deep Autoencoder with seven layers. We have evaluated different architectures and we adopt 10000-250-250-64-250-250-10000 architecture (see Fig. 3.2) basing on its good results. After freezing the decoder layer, we obtain low-dimensional 64-length vectors for each textual document. To accelerate the pre-training process, we subdivide the training set into small mini-batches. For the fine-tuning, we use the conjugate gradient method.

3.5.3 Categorization Module

After learning text representations based on deep RBMs Autoencoder, these representations are fed to the categorization module. We used two deep neural networks, including the MPL and the Autoencoder.

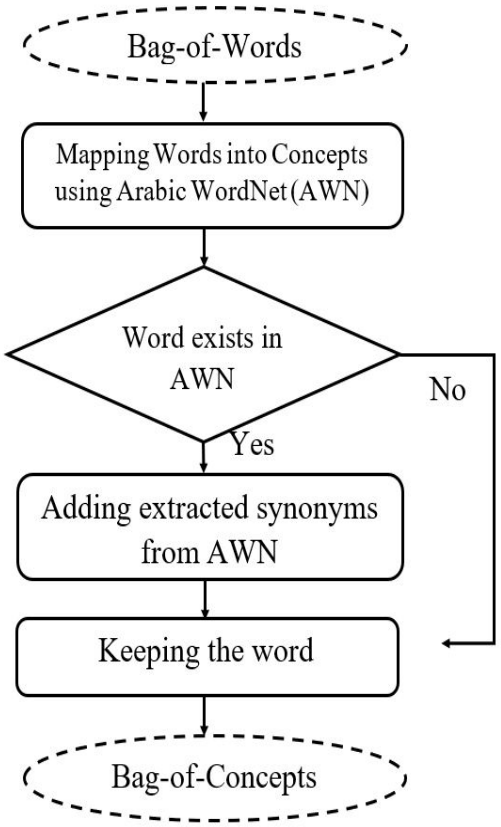


Fig. 3.4 Building feature representation space for texts

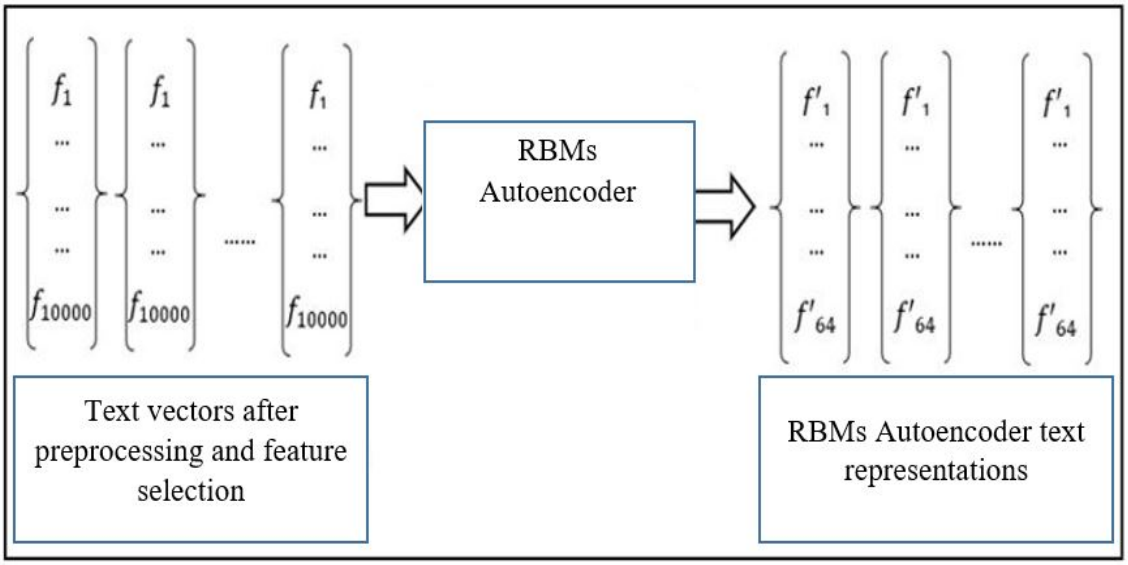


Fig. 3.5 Input and output vectors of our representation module

3.5.3.1 Autoencoder

We have adopted a deep Autoencoder classifier owing to its ability to reconstruct the inputs without needing any labels and its effectiveness in classification applications. We train our Autoencoder using a stack of layers to reconstruct the inputs. After its training, we drop the decoder layer and connect a softmax layer to the encoder features layer to map vector representations to the suitable class among ten categories. After several experiments, we use a stack of layers (three layers for the encoder and three layers for the decoder: 64-700-64-64-64-700-64). We utilize *relu* activation function and *adam* optimizer in the training.

3.5.3.2 Multi-layer Perceptron Classifier

Modern artificial neural networks are premised on the perceptron, that is a prediction out of the following formula [107]:

$$\hat{y} = \sigma_{(0)} W_{(0)} + b_{(0)} \tag{3.10}$$

Although the perceptron is a single-layered linear model for this raison it can only learn linearly separable patterns. Multi-layer Perceptron is a fully connected feedforward neural network where every neuron in a layer is connected with all the other neurons in the next layers (see Fig. 3.6). The described structure is originated back in 1958. MLP is able to learn non-linear relationships between the data. MLP having more than one

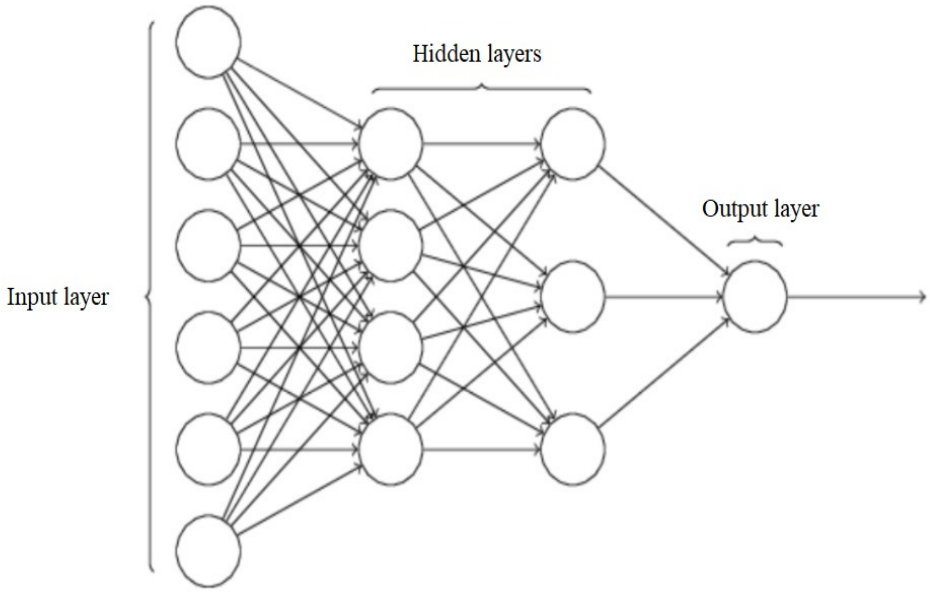


Fig. 3.6 Multi-layer perceptron architecture

hidden layer ($L \geq 1$). For example, for $L = 2$ the MLP is defined as follows:

$$\hat{y} = \sigma_{(2)}(W_{(2)}h_{(2)} + b_{(2)}) \tag{3.11}$$

$$h_{(2)} = \sigma_{(1)}(W_{(1)}h_{(1)} + b_{(1)}) \tag{3.12}$$

$$h_{(1)} = \sigma_{(0)}(W_{(0)}x + b_{(0)}) \tag{3.13}$$

Where the predicted output is $\hat{y}$, σ is an activation function, W is the weight matrix, b is the bias vector, the output of the hidden layers are in $h_{(l)}$, and l is to indicate the index of the current layer (ranging from $0 \geq l \leq L$). Moreover, the dimension can be different in each layer, input, and output. Each layer is characterized by a non-linear activation function $\sigma_{(l)}$ that can be differ from a layer to another. The Popular activation functions are the sigmoid, the tanh or the rectified linear unit (ReLU). A Multi-Layer Perceptron operates on numerical vectors, so the inputs are needed to be coded as ordered n-tuples of floating point numbers. The numbers are called "features". Multiple features can be catenated into a feature vector. When a feature is essentially the name of a category, it is usually coded as an integer number, but in the case of a Multi-Layer Perceptron classifier, we should convert these to binary vectors. The same should be done with the outputs of the classification task that are the names of the categories. Categories can also be encoded as binary vectors. Multi-Layer Perceptron maps the data from feature space to categorization output space to predict the encoding vector that is closed to the outputs selected.

In this work, we applied an MLP with four hidden layers with 1000 nodes for each layer. We explore the *relu* activations and *adam* optimizer.

Table 3.2 OSAc Corpus

Categories	Documents number
Economy	3102
History	3233
Education and Family	3608
Religious and Fatwas	3171
Sport	2419
Health	2296
Astronomy	557
Low	944
Stories	726
Cooking and Recipes	2372

3.6 Experimental Results

In this section, we expose the corpus description and the used text categorization performance indicators. Moreover, we describe the experiments, detail and analyze the results.

3.6.1 The Corpus and Performance Measures

To evaluate multi-class categorization on Arabic text, we use OSAc dataset that was introduced by [110]. OSAc is the largest freely public Arabic text corpus. The corpus includes 22.428 text documents, each document belongs to 1 of 10 categories which are described in Table 3.2. The corpus contains 18 million words and more than 449.600 distinct words after stop-words removal. To measure our Arabic text categorization system's effectiveness, we utilized three indicators, including precision, recall, and F-measure.

3.6.2 Discussion

We evaluated our method against the leading text representation models since they are well-known and used in Arabic. Moreover, they enabled the production of fixed-length vectors for text categorization tasks. These representations included Doc2vec, LSA, and TF-IDF, and they were integrated with the SVM classifier. The illustrated results in Table 3.3 demonstrate that the proposed method improves Arabic text categorization

performance. Our model adopted the Bag-of-Concepts representation, and the Chi-Square outperformed all baselines and achieved up to 93% and 94% in terms of F-measure and precision, respectively. This was not surprising since the classical method based on Bag-of-Words representation afforded only 88% of F-measure since it ignores the semantics and word contexts. Moreover, the proposed method captured exact inference and handled complex semantic tasks since it considered nonlinear dependencies in comparison with the LSA representation, which could not. The LSA representation achieved only 89% in terms of F-measure. Apart from this, a significant reduction of feature space dimensionality was derived from our technique (64 fixed-length vectors) compared with other methods, 300 dimensions for Doc2vec and LSA. Overall, the obtained results showed that the proposed technique enhanced Arabic text categorization by reducing the feature representation space's dimensions and represented each textual document by a vector containing only 64 features. Also, our proposed method reflected the hidden document structures due to Autoencoder architecture using.

To study the feature selection impact on the effectiveness of our system, we utilized two different feature selection methods: the Chi-Square and VT. The results are shown in Fig 3.7. From this figure, it can be seen that the Chi-square was more effective than the VT for both Bag-of-Words and Bag-of-Concepts representations since it selected the best features based on the probability of interdependence between the term and category. Chi-Square using Bag-of-Concepts performed 94%, 91%, and 93% for precision, recall, and F-measure, respectively.

Table 3.3 Arabic categorization performance results using different text representations

Methods	F-measure	Precision	Recall
The proposed method	0.93	0.94	0.91
Classical Bag-Of-Words	0.88	0.91	0.88
Doc2vec	0.89	0.91	0.88
Latent Semantic Analysis	0.89	0.91	0.89

In contrast, the VT using Bag-of-Concepts achieved only 92%, 89%, and 90% for precision, recall, and F-measure, respectively. In respect of the Bag-of-Words representation, the Chi-Square reached less than 91% for all measures in comparison with the VT, which achieved 86% and 83% for precision and F-measure, respectively.
To show the Bag-Of-Concepts representation effects on the system performance, we conducted different experiments using Bag-Of-Concepts and Bag-Of-Words. The results are shown in Fig 3.8. It is noticed that our model using the Bag-Of-Concepts representation surpasses the model using Bag-Of-Words representation regardless of the feature selection method. This is due to the vocabulary enrichment that the mapping to Bag-Of-Concepts does.
To evaluate the impact of adopting the deep Autoencoder classifier, we compared its performance with Multi-Layer Perceptron and SVM, applying the khi2 measure and using Bag-Of-Concepts representation. The results are illustrated in Table 3.4; we observe that the deep Autoencoder outperforms the other classifiers slightly and achieves the best performance, specifically the best F-measure by a score of 93%.

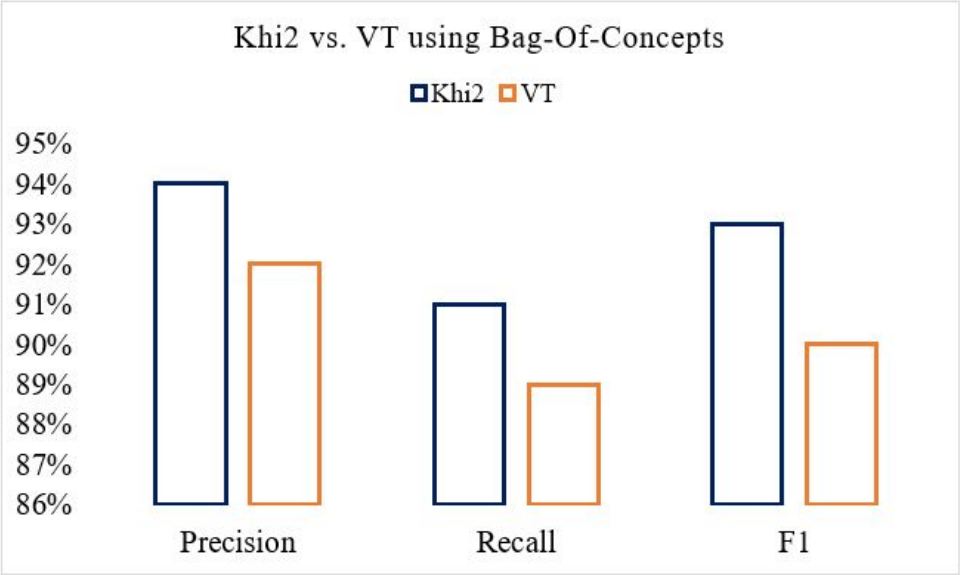
Table 3.4 Comparison between Autoencoder, MLP and SVM classifiers using Bag-Of-Concepts and Chi-square

Methods	**Deep Autoencoder**	**MLP**	**SVM**
Precision	0.94	0.93	0.93
Recall	0.91	0.90	0.91
F-measure	0.93	0.91	0.92

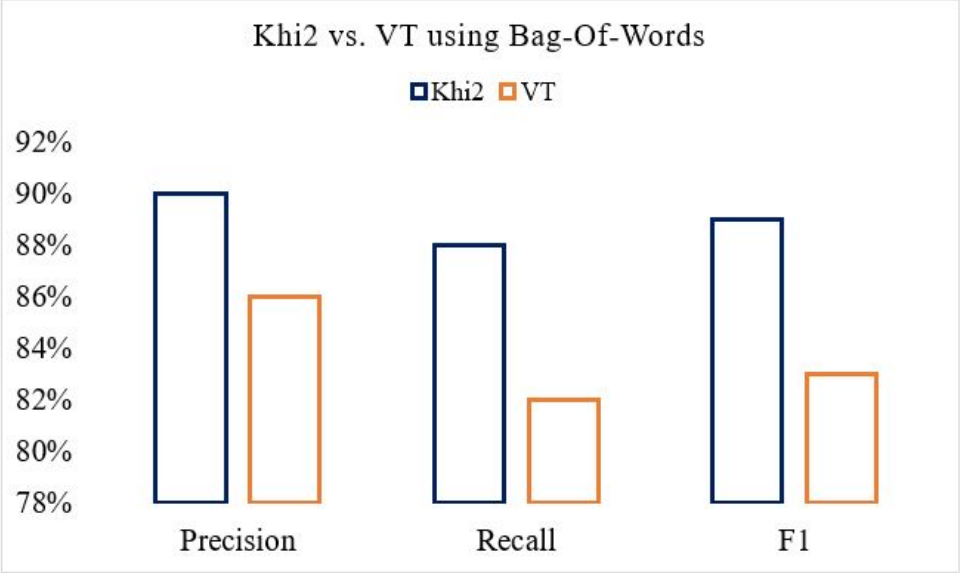
3.7 Conclusion

In this chapter, we proposed our Arabic text categorization method based on Bag-Of-Concepts and deep Autoencoder representations. It involves explicit semantic relying on Arabic WordNet and uses Chi-Square measure to select the most pertinent features. Successive stacks of Restricted Boltzmann Machines are applied to text vectors to produce high-level representations. The learned features are fed to another deep Autoencoder for categorization. An exhaustive set of experiments has been carried out and shown that using Autoencoder as text representation model and classifier outperformed state-of-the-art techniques and achieved the best results with Chi-Square by 94% and 93% respectively for precision and F-measure. The principal advantages are: (1) Integrating the explicit semantic in order to improve text vector quality; (2) Modeling the semantic structure within texts; (3) Reducing representation dimensionality; (4) Exploring deep learning networks for Arabic text categorization.

However, the autoencoder suffers from some limitations ranging from the requirement of a large quantity of data, the processing time to capturing as much information as possible rather than as much relevant information as possible. In the next chapter, we present our second contribution concerning exploring deep neural networks for Arabic text categorization and integrating the Retrofitting technique in the text representation.

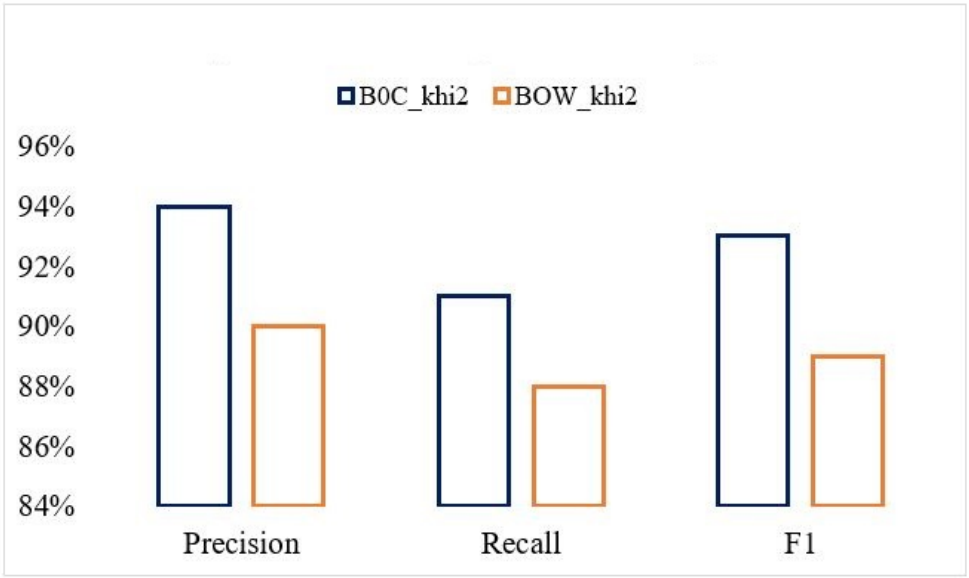


(a)

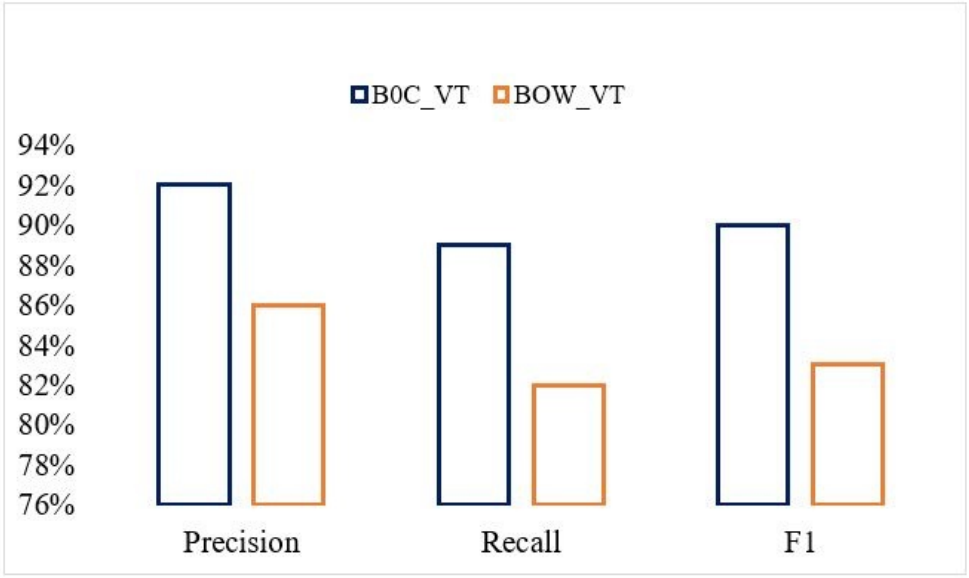


(b)

Fig. 3.7 Performance analysis of khi2 vs. VT. (a) Using the Bag-of-Concepts representation. (b) Using the Bag-of-Words representation



(a)



(b)

Fig. 3.8 Performance analysis of Bag-Of-Words representation vs. the Bag-Of-Concepts representation. (a) Performance representation based on Chi-square. (b) Performance representation using VT

CHAPTER 4

Deep Neural Models and Retrofitting for Arabic Text Categorization

4.1 Introduction

In the previous chapter, we studied the contribution of the Deep Autoencoder exploration in the process of text representation and categorization. We also presented the integration of explicit semantics using Arabic WordNet to enhance the quality of text vectors. This chapter introduces other types of deep neural network models that have shown promising performance and indicated great data modeling capacities in managing large and substantial datasets. We present our method investigating Convolution Neural Networks, Long Short-Term Memory, and their combination for Arabic text categorization [40, 37, 38]. This work additionally handles the morphological variety of Arabic words by exploring the word embeddings model using position weights and subword information. To guarantee the nearest vector representations for related words, our method adopts a strategy for refining Arabic vector space representations using semantic information embedded in lexical resources. Several experiments utilizing different architectures have been conducted on the OSAc dataset.
The rest of this chapter is organized as follows. Section 4.2 presents our motivations. Section 4.3 describes the background. Section 4.4 introduces the proposed method for Arabic text categorization using different models. Section 4.6 illustrates the experimental results. Finally, section 4.7 concludes the study.

4.2 Motivations

Text categorization system remains a challenging task due to the following two reasons: (1) The high dimensionality of feature space, which decreases the categorization system's performance; (2) The existence of redundant and noisy features that misguide the TC results. To address these issues, various feature representation methods have been

proposed. The most known representations are BOW, pLSA, LDA, word embeddings, and doc2vec. However, BOW cannot capture semantics within texts and fails to reflect similarities among words. PLSA and LDA are topic modeling methods which are, generally, applied to select more discriminative features, but they suffer from inference problem. More efficient representations as word embeddings or document embeddings are defined as a set of language modeling techniques able to present words of the vocabulary or text as low dimensional vectors of real numbers through neural language models. They ignore the information embedded in the lexical database. These representations have shown good performance in Arabic text categorization. However, the information embedded in the lexical database is ignored.

While several TC systems have been proposed for other languages (English, French, etc.), Arabic TC still faces numerous difficulties in addition to the challenges discussed above. This could be explained by the complexity of the Arabic language owing to the fact that it is inflectional and derivational.

To solve the shortcomings above, we explore deep neural models and Retrofitting for Arabic text categorization. The Retrofitting is defined as a graph-based learning technique utilizing lexical relational resources to train higher quality semantic vectors. It is employed for further enhancement. On the other hand, deep neural networks can achieve the best results in many NLP tasks. Convolutional Neural Networks can achieve good performances in highlighting the best features and empowering deeper models. Long Short-Term Memory networks are a kind of Recurrent Neural Networks that enable associations between cells to construct a directed graph along a sequence. They have demonstrated great capabilities in highlighting the dynamic behavior of sequential data. The main contributions of this work can be summarized as follows:

- We propose a more sophisticated method for Arabic word representation by combining the implicit and explicit semantic using the Retrofitting technique. This allows us to improve the word embeddings by exploring the knowledge in AWN.
- Inspired from [48] and based on the model of [19], we introduce a Retrofitting process that takes advantage of using subword information training and treating the morphological diversity of Arabic words.
- Finally, this work investigates CNN, LSTM, and their combination in Arabic text categorization by building these models on top of the word embeddings layer. This gives a significant advantage in terms of the salient pattern's detection in the sequence and the inference capturing.

4.3 Background

4.3.1 Convolutional Neural Network

Convolutional neural networks [66] are neural networks consisting of input, an output layer, and multiple hidden layers. The hidden layers typically comprise convolutional layers, pooling layers, fully connected layers, and normalization layers. CNNs use convolutional operators instead of connecting each neuron with all the next ones. In a way, they try to regularize feedforward networks to avoid overfitting, which makes them very good at identifying spatial relationships between the data.
CNN replaces any MLP layer with a convolutional layer as follow:

$$h_l^{(k)} = \sigma_{(l)}(W_{(l)}^{(k)} \times h_{(l-1)} + b_{(l)}) \tag{4.1}$$

Where $W_{(l)}^{(k)}$ is the filter weights for the filter k which capture a local context and the layer l.
Convolutional networks may include local or global pooling layers to streamline the underlying computation. Pooling layers reduce the data dimensions by combining neuron clusters' outputs at one layer into a single neuron in the next layer. Local pooling combines small clusters, typically 2×2. Global pooling acts on all the neurons of the convolutional layer. Besides, pooling may compute a max or average. Max pooling uses the maximum value from each of a cluster of neurons in the last layer. Average pooling uses the average value from each of a cluster of neurons in the last layer. Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional MLP. The flattened matrix goes through a fully connected layer to classify the images. The below block diagram, Fig. 4.1, shows the complete CNN architecture for sentence classification.

4.3.2 Recurrent Neural Network

Recurrent neural networks are used to model temporal dependencies [45]. Fig. 4.2 shows the RNN architecture. The Network is unfolded to see the internal architecture of RNNs. The latent representation $h_{(1)}^{(t)}$ depends on the current time-step $t : h_{(1)}^{(t)} = f(x^{(t)})$ and the previous time-step: $h_{(1)}^{(t)} = f(h_{(1)}^{(t-1)}, x^{(t)})$. The traditional formulation of RNNs for a single-layered RNN ($L = 1$) is defined as follow:

$$\hat{y}^{(t)} = \sigma_{(1)}(W_{(1)}h_{(1)}^{(t)} + b_{(1)}) \tag{4.2}$$

$$h_{(1)}^{(t)} = \sigma_{(0)}(W_{(0)}x^{(t)} + W_{rec}h_{(1)}^{(t-1)}) + b_{(0)} \tag{4.3}$$

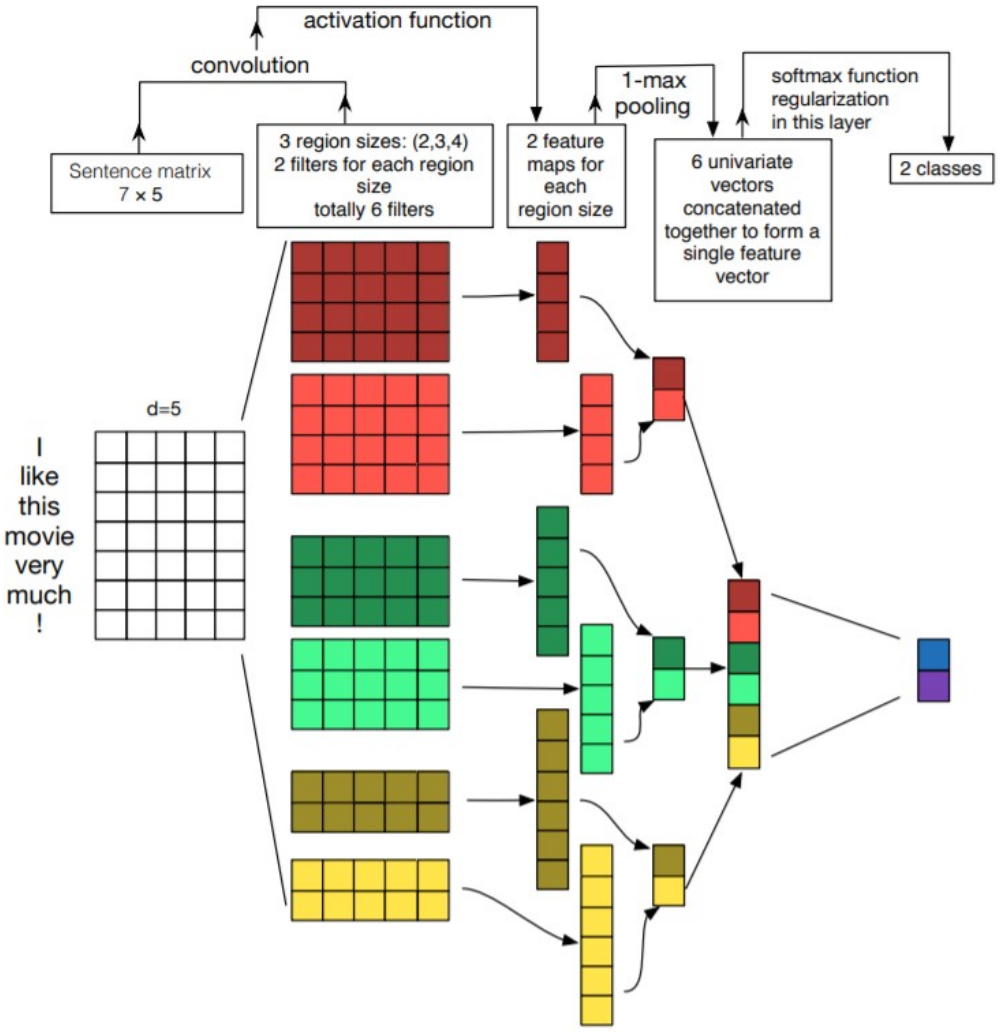


Fig. 4.1 Illustration of a CNN architecture for sentence classification [144]

Where $\hat{y}^{(t)}$ denotes the predicted output and W_{rec} is the recurrent weights.
RNNs are used to capture the sequence and structural information present in text. Word2vec transformation techniques fail to capture perception about the next word in the sequence. Unfortunately, RNNs suffers from vanishing gradient problem and long-term dependency [17]. Such as words that are related are in longer distance between the other words in the sentence. Long Short-Term Memory networks were proposed to overcome the drawback of RNNs. In the next section, we discuss LSTM in detail.

4.3.2.1 Long Short Term Memory Networks

Long Short Term Memory networks, a RNN variant, are first introduced by [59]. LSTM contains memory cells and has three types of gates: the input gate, output gate and forget

gate. LSTMs are defined as follows :

$$Input : i^{(t)} = \sigma(Wx^{(t)} + W_{rec}h^{(t-1)} + b) \tag{4.4}$$

$$state : s^{(t)} = g_i^{(t)} i^{(t)} + g_f^{(t)} s^{(t-1)} \tag{4.5}$$

$$Output : h^{(t)} = \sigma(s^{(t)}) g_0^{(t)} \tag{4.6}$$

Where σ can be any non-linearity, but the $tanh$ is oftentimes used. The input representation formulation $i^{(t)}$ is the same as the traditional RNN; the state representation decides what's important to keep: if "present" $i^{(t)}$ or "past" information $s^{(t-1)}$; and the output representation is to control what's visible from the output. This behavior is controlled by a set of sigmoidal gates: $g_i^{(t)}$ (that reads the input), $g_f^{(t)}$ (that forgets from the past), and $g_0^{(t)}$ (that writes the output), that are defined by:

$$Gates : g_?^{(t)} = \sigma(b_? + W_? x^{(t)} + W_{(rec?)} h^{(t-1)}) \tag{4.7}$$

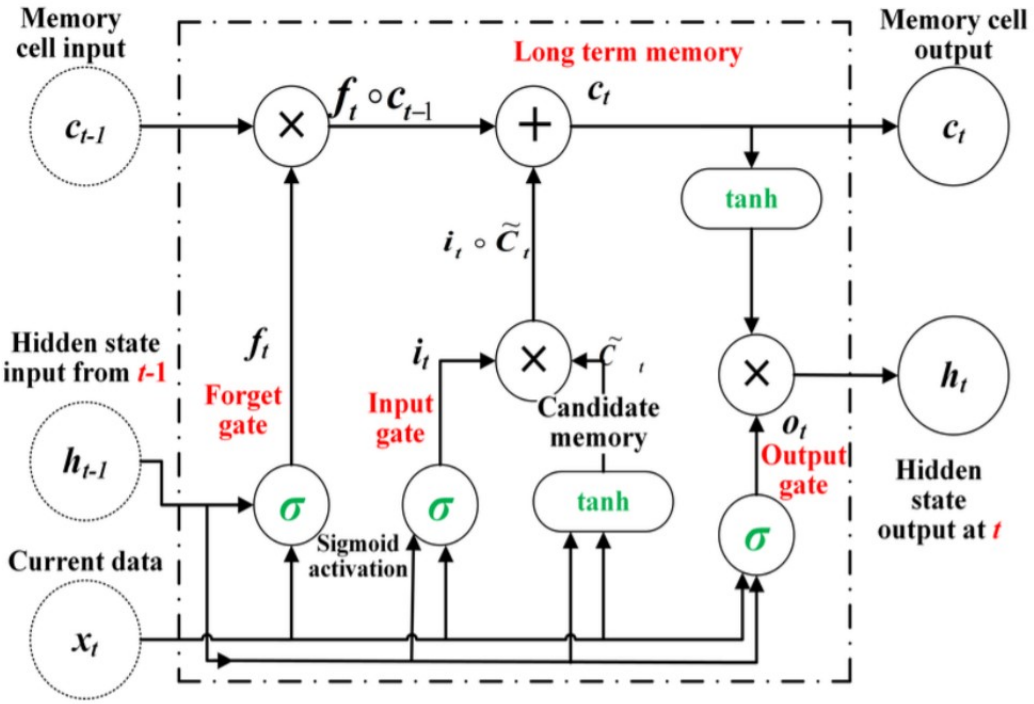


Fig. 4.2 Recurrent Neural Network architecture [96]

Where ? the sub-index standing for input, forget or output. In text, LSTMs are designed to capture the long term dependencies that exist within it with the help of the memory unit. Fig. 4.3 illustrates the internal structure of LSTM network with Memory unit.

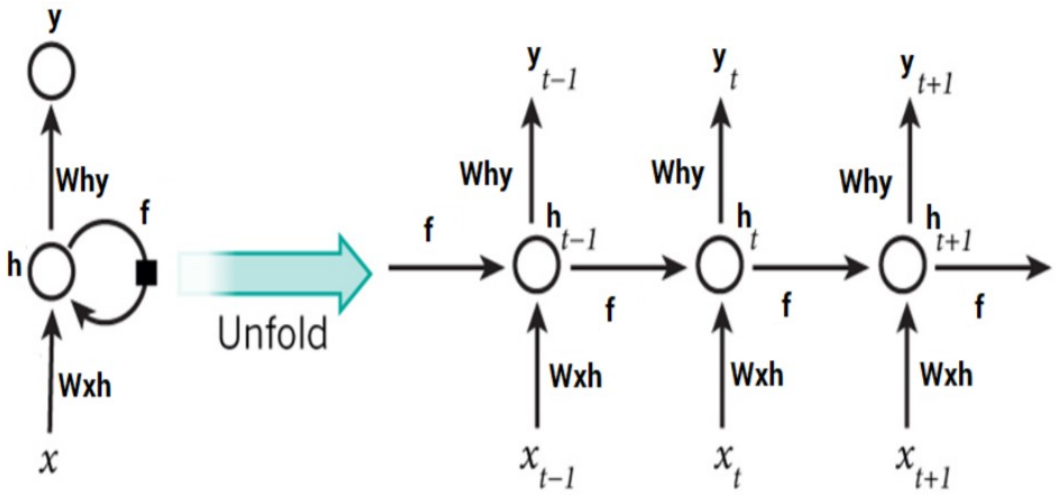


Fig. 4.3 Long Short Term Memory Network Memory Block [96]

4.4 Methodology

Our Arabic text categorization system is made up of several stages: (1) Text preprocessing; (2) Word embeddings model based on subword information; (3) Retrofitting the model using Arabic WordNet lexical resource; (4) Combining CNN, LSTM models up on the retrofitted word embeddings. Fig. 4.4 shows the flowchart of the proposed Arabic text categorization method using deep neural models.

4.4.1 Text Preprocessing

The first step consists of the preprocessing process during which we remove punctuation marks, numbers, words written in different languages, and stop-words. Each text is then tokenized to a set of words. For instance, taking the following text:
«الشركة نجحت في الخروج من مصيدة الافلاس من خلال التحول الى شركة جديدة»
The obtained tokens after the preprocessing are (company, الشركة), (succeeded, نجحت), (breaking, الخروج), (bankruptcy, الافلاس), (trap, مصيدة), (switching, التحول), (company, شركة) and (new, جديدة). After this stage and as Fig. 4.4 shows, we train a matrix of size $x \times r$ where r is the length of the word vectors, and x is the maximum length of texts. Let maxlen be the maximum length of texts in the training and test sets. Each text, including a number of terms less than maxlen, will be padded to have the same length

as the maximum text length. However, for texts having a length longer than maxlen, we cut extra words at the end of these texts to attain maxlen. After that, text representations are fed to one of the neural network models (LSTM, CNN, LSTM-CNN, or CNN-LSTM).

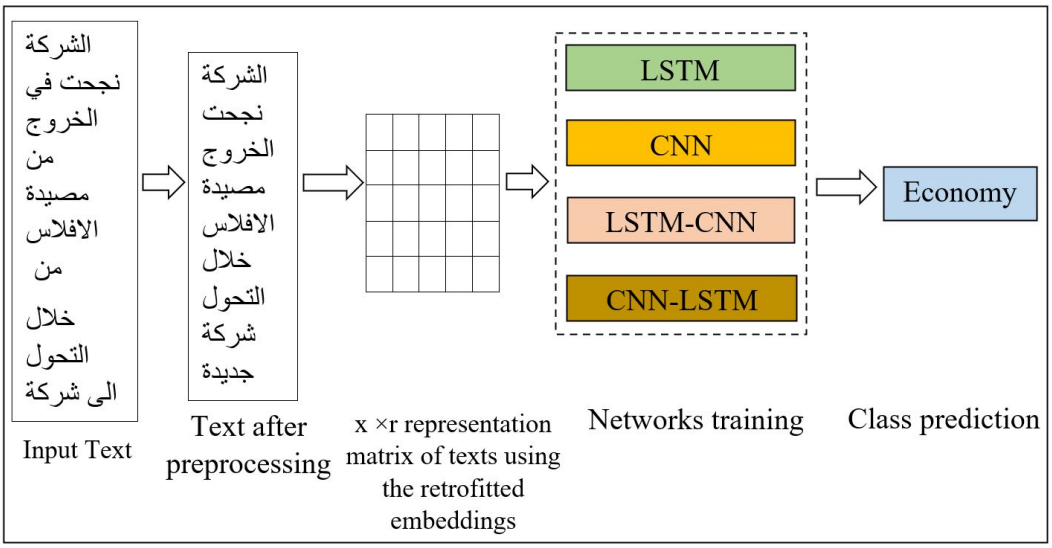


Fig. 4.4 The flowchart of the proposed Arabic text categorization method using deep neural models

4.4.2 Word Embedding Model with Subword Information

Word embedding representations as Word2vec and Glove ignore the morphology of words by assigning a distinct vector to each word [19]. These representations cannot be appropriate for the Arabic language since it has a rich vocabulary and rare words. To solve this problem, Facebook AI Research lab released a new word embedding method called FastText. The main idea is that each word is represented as a bag of character n-gram. The newest version of this model trained on several languages, including the Arabic language, was presented by [19]. For example, the word مساعدة with $n = 3$ is represented by the set of character n-grams:مساعدة ،مس ،مسا ،ساع ،اعد ،عدة ،دة. Suppose we have a dictionary of n-grams of size G and given the word w, which is associated as a vector representation z_g to each n-gram g. Given a context vector representation v_c of the context word c, the word embeddings are learned by maximizing the scoring function as follows:

$$s(w, c) = \sum_{g \in g_w} z_g^T v_c \tag{4.8}$$

This straightforward model allows sharing the representations across words, thus permitting us to learn solid representations for uncommon words.

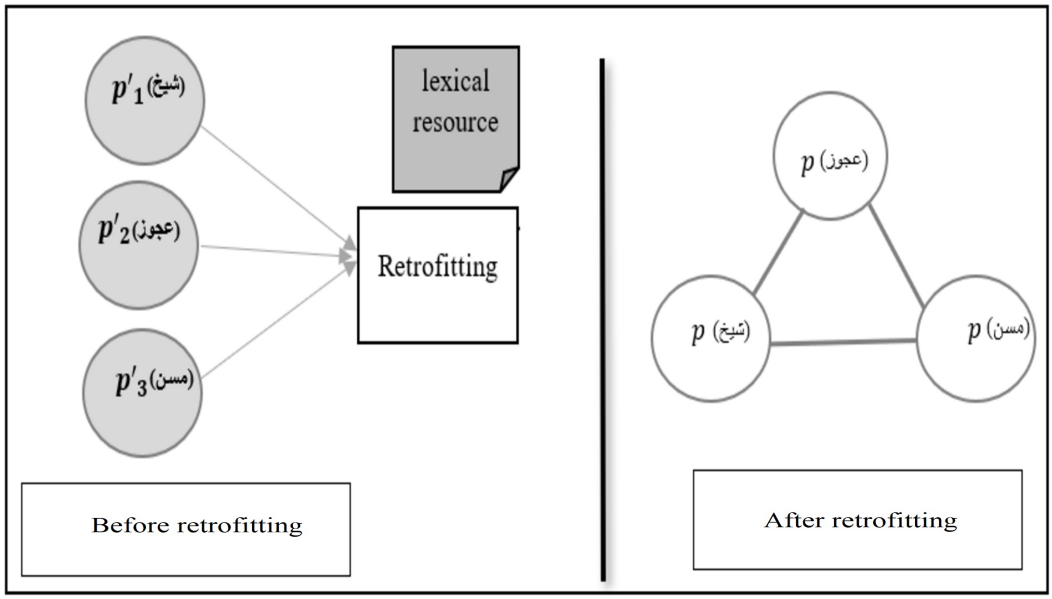


Fig. 4.5 An example of retrofitted Arabic words

4.4.3 Retrofitted Word Embedding with Semantic Lexicon

Word embedding models can be improved by exploring the data contained in semantic dictionaries as Arabic WordNet. In line with the work of [48], we investigate here refining Arabic word vector representations by using semantic information embedded in the semantic lexicon Arabic WordNet. More formally, let $V = w_1, ..., w_j$ the vocabulary and θ the Arabic WordNet ontology that includes the semantic relations among words in the vocabulary, it is represented as an undirected graph $(V, E), E \subset V \times V$ that contains one vertex for each word type and edges (w_j, w_j). The matrix P' represents the collection of vector representations $(p_i)'$ for each w_i using the word embeddings model trained on Wikipedia and Web crawl resources by [19]. The main objective is to learn and store new knowledge in a matrix $P = (p_1, ..., p_n)$ in such a manner that vocabulary words are semantically close to their partners in P' and neighboring terms in θ. The resultant word vectors are close to the observed value $(p_i)'$ and neighbor vectors $p_j \forall j, (i, j) \in E$ hence the target to be minimized moves toward becoming:

$$\Omega(P) = \sum_{i=1}^{n} [\alpha_i ||p_i - p'_i||^2 + \sum_{(i,j)\in E} \beta_{ij} ||p_i - p_j||^2] \tag{4.9}$$

where α and β values control the relative strengths of associations. By considering the first derivative of Ω equals to zero, P_i is calculated as follows:

$$p_i = \frac{\sum_{j:(i,j)\in E} \beta_{ij} p_j + \alpha_i p'_j}{\sum_{j:(i,j)\in E} \beta_{ij} p_j + \alpha_i} \tag{4.10}$$

Fig. 4.5 illustrates a Retrofitting example for the Arabic words (مسن، شيخ، عجوز) which are synonyms; The grey nodes are the original embeddings which are labeled with the corresponding vectors in P', these representations are retrofitted using the Arabic WordNet in order to find the associations between them; The white nodes are labeled with the P vectors and they are the retrofitted representations. The Retrofitting boosts semantic relations between these three words by reproducing the nearest representations (white nodes) for these semantically similar words.
Note that all retrofitted word embeddings have a fixed size of 300. Based on the model proposed by [19] and Arabic WordNet, we fine-tune these word embeddings along with other model parameters during the training phase.

4.4.4 Deep Neural Network Models

4.4.4.1 CNN Model

After the processing, the filters are passed across the text representations matrix, different sizes of filters (32, 64, 128) are used. Each filter detects specific features in texts using the *relu* activation function to represent them in the features map. After the convolutional layer, the max-pooling layer is a way to downsample the incoming features map. We employ the max function to select the highest values, which capture the most important features in texts and reduce the computation in the advanced layers. The max-pooling layer outputs are merged into one matrix and then passed to a fully connected layer. For regularization, we apply dropout to prevent co-adaptation, and we fix it at 0.5. Since this work studies multi-class classification problem, the *softmax* function is used in the output layer, and texts are categorized into ten categories. The Fig. 4.6 shows the proposed architecture of the CNN model.

4.4.4.2 LSTM Model

The retrofitted text vectors are passed to an LSTM classifier consisting of one hidden layer with 100 hidden units. Adding dropout is a way to avoid overfitting. It is fixed at 0.5, and then this layer is connected to a fully connected network. For text categorization, we consider the output of the hidden state at the last time step of LSTM as the document representation and add a softmax layer on the top. We train the entire model by minimizing the *cross-entropy* error and applying *Adam* optimizer. The Fig. 4.7 shows the proposed architecture of the LSTM model.

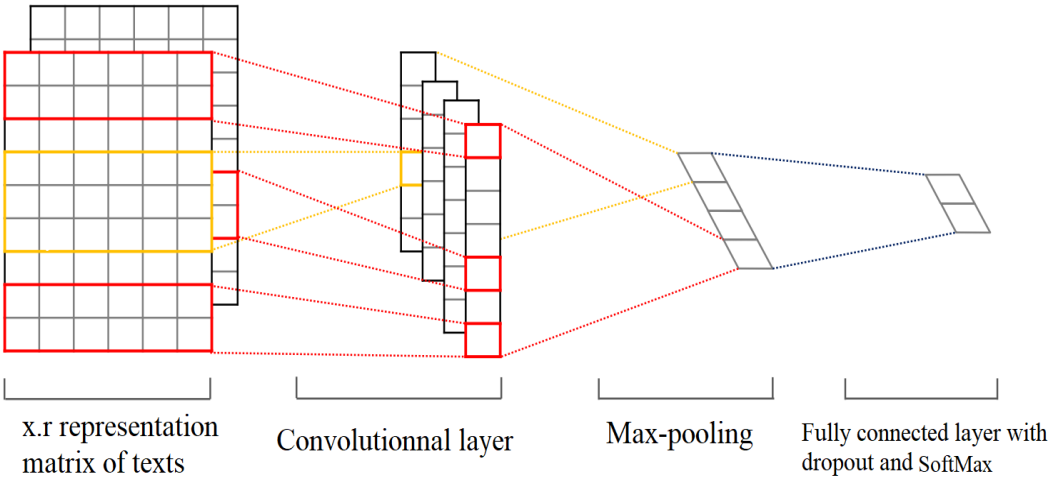


Fig. 4.6 The Proposed CNN Model

4.4.4.3 LSTM-CNN

The LSTM-CNN model comprises an underlying LSTM layer that receives text representations. The LSTM layer's output is then fed into a convolution layer to extract local features. Finally, the convolution layer's resultant vectors are pooled to a smaller dimension. Afterward, the pooling layer's output is connected to a fully connected layer then outputted as one of ten corpus's labels.

4.4.4.4 CNN-LSTM

Fig. 4.8 illustrates the CNN-LSTM model. The input layer comprises a matrix of retrofitted embeddings then the dropout is applied on it. Afterward, the matrix is scanned using a filter of size 64; the filter detects multiple features in texts. After the convolutional layer, the max-pooling is applied. LSTM layer takes the outputted vectors from the max-pooling layer as inputs, this layer contains a set number of cells, and the input of each cell is the output from the max-pooling layer. The generated vectors from LSTM are passed to a fully connected layer. The last step of the model is adding a *softmax* layer on the top. We train the entire model by minimizing the *cross-entropy* error. The advantage of this model is that the convolution layer picks up the pertinent features and the LSTM layer allows to use the sequence property of these features to learn the input's text ordering.

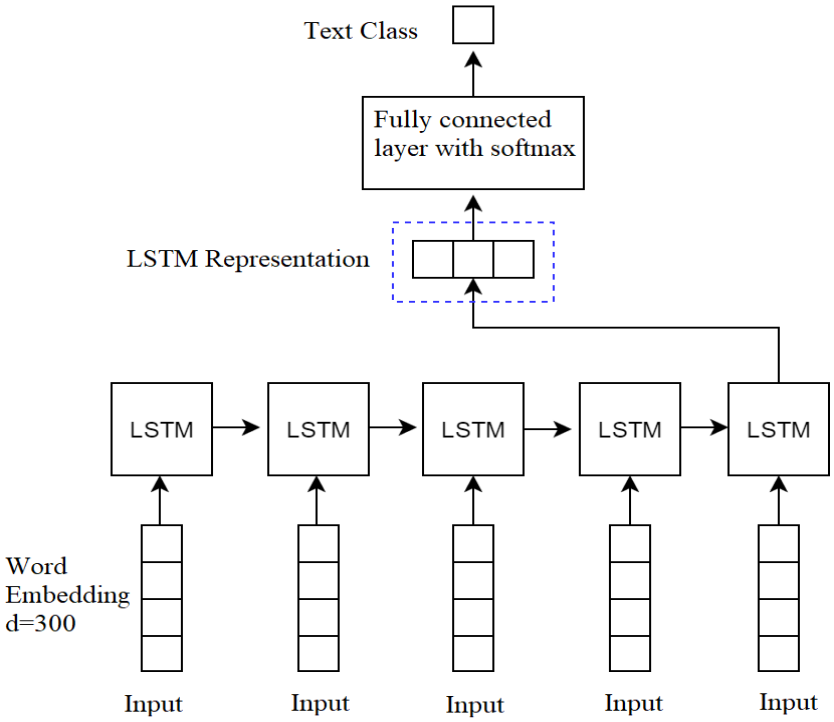


Fig. 4.7 The Proposed LSTM Model

4.5 Experimental Results

4.5.1 Experiment Setting

To validate the impact of using neural network models and the Retrofitting on Arabic text categorization, several experiments were conducted on the publicly available dataset OSAc. The proposed models were implemented using the Keras library based on TensorFlow Backend, Numpy, Pandas, Sklearn, and others. All experiments were conducted on a computer with a 4-core Intel i7 @2.40GHz CPU, Intel GPU, and 16GB of RAM. We have employed the pre-trained vectors taken from the Arabic word embeddings [19]. The Retrofitting on the models is performed using the latest version of Arabic WordNet 2.0.1, which contains 11.269 synsets corresponding to 23.841 words and 22 link types, totaling 161.705 links. AWN synsets belong to one of 5 parts-of-speech: noun (6.438), verb (2.536), adjective (456), adjective satellite (158), and adverb (110). The training set contains the remaining 75% of the whole dataset while the test set contains the staying 25%. To avoid overfitting, dropout is utilized. The dropout layer selected randomly dropped neurons with a dropping rate of 0.5. To make a fair comparison, the same classification setup is adopted for all models. The performance of models was computed via the accuracy and F-measure measurements.

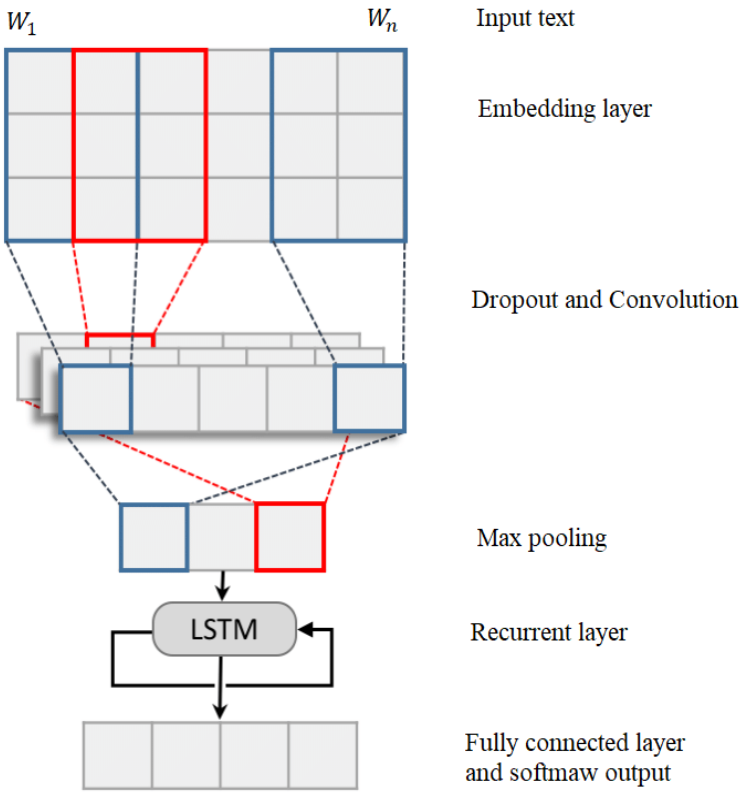


Fig. 4.8 The Proposed CNN-LSTM Model

4.5.2 Comparison with Arabic Text Categorization Methods

We compare our models with the state-of-the-art Arabic text categorization methods.

- TFIDF_SVM: We employ BOW based on the light stemmer and TF-IDF measure to present documents in feature space, and as a classifier, we chose SVM.
- TFIDF+khi2_SVM: We use BOW representations after preprocessing the texts by excluding stop-words and stemming all the terms of the dataset. Then, we utilize chi-square to select the most pertinent features. All the document representations are trained using 5000 best features. Afterward, these representations are fed to the SVM classifier to attribute a category to each text in the corpus.
- LDA_SVM: To generate the representation based on topics modeling LDA method, stop-words were removed, and texts were stemmed. We build an LDA model of 300 topics (the same dimension as the pre-trained embeddings). We exploit the SVM in the categorization phase.
- Doc2vec_SVM, Skip-gram_SVM, and Glove_SVM: These models are trained by [43]. They studied documents embeddings models (Doc2vec and Glove) and Word2vec. Text representations are fixed at 300 dimensions.

Table 4.1 Comparisons of our models with state-of-the-art ones on OSAc dataset

Models	F-measure
TFIDF_SVM	0.89
TFIDF+Khi2_SVM	0.96
LDA_SVM	0.96
Skip-gram_SVM	0.96
Glove_SVM	0.95
Doc2vec_SVM	0.94
CNN-32	0.98
CNN-64	0.98
CNN-128	0.99
LSTM	0.93
CNN-LSTM	0.99
LSTM-CNN	0.92

4.5.3 Results and Discussion

Table 4.1 illustrates the F-measure results of CNNs, LSTM, CNN-LSTM, and LSTM-CNN models using Retrofitting compared with state-of-the-art models. To understand the effects of the CNN architecture parameters, especially the number of filters, on the performance of this model, experiments are conducted for a various number of filters (32, 64, and 28). From Table 4.1, it is pointed out that the CNN model gives the best performance in the case of a number of filters equals 128. Besides, it is noticed that CNN-128 and CNN-LSTM achieve the best results for Arabic text categorization and outperform CNN-32, CNN-64, LSTM, and LSTM-CNN. They also perform better than state-of-the-art models including TFIDF_SVM, TFIDF+Khi2_SVM, LDA_SVM, Skip-gram_SVM, Glove and Doc2Vec by, 10%, 3%, 3%, 3%, 4% and 5% respectively. However, LSTM and LSTM-CNN models outperform only TF-IDF.

To show the effect of Retrofitting on Arabic text categorization, we implement our models without and with Retrofitting. The results in terms of the training and the test accuracy are shown in Table 4.2. For the CNN-LSTM model, the training accuracy attains 99% without and with Retrofitting, while test accuracy knows an improvement of 1% utilizing Retrofitting. About CNN models, the accuracy gets an improvement for both the training and the test phases when Retrofitting is employed. Concerning the LSTM model, the training accuracy approaches 93%, and the test accuracy reaches 92% using Retrofitting. Note that the LSTM-CNN model is more sensitive to Retrofitting since the accuracy has been improved by almost 12% for the training and the test phases. From

Table 4.2, it can also be underlined that the CNN-LSTM model achieves a test accuracy of 1% respectively and 8% higher than the CNN model and the LSTM model. These results indicate that the Retrofitting generally enhances text categorization performance because it fosters the analogical vectors to be closer by making semantically similar words near in the representation space. Moreover, the combination of CNN and LSTM tackles both CNN's capacity to recognize local patterns and the LSTM's capacity to saddle the content's ordering. However, the ordering of the layers plays a crucial role in how well models perform since the LSTM-CNN model achieves respectively an accuracy of 7% and 1% less than the CNN-LSTM model and the LSTM model.

Fig. 4.9 illustrates the loss function for the CNN-LSTM model using Retrofitting. It can be realized that the training loss decreases until 0.0084 and the validation loss until 0.03. This result demonstrates the good performance of this model in comparison with others.

Fig. 4.10 shows the training and validation loss for CNNs and LSTM models with Retrofitting technique. The training loss decreases gradually until the value of 0.05 at the end of the training for CNNs (32, 64, and 128), whereas the validation loss cannot decrease less than 0.4. For LSTM, the training loss decreases and increases with every epoch until the last epoch when it approaches zero value. The validation loss keeps changing until ending at a value of 0.55 in the last epoch.

Overall, neural network models achieve generally good performance since they can capture contextual information and do not suffer from the data sparsity problem. Adding the Retrofitting technique allows capturing more semantics because the quality of word vector embeddings can be improved using semantic knowledge from the AWN lexicon.

Table 4.2 Arabic text categorization accuracy results for different models without and with Retrofitting

	Without Retrofitting		**With Retrofitting**	
Models	**Training**	**Test**	**Training**	**Test**
CNN-128	0.989	0.9743	0.9995	0.9868
CNN-64	0.9994	0.9816	0.9996	0.987
CNN-32	0.9975	0.9832	0.9992	0.9822
LSTM	0.9229	0.9196	0.9304	0.9263
CNN-LSTM	0.9977	0.9898	0.9988	0.9923
LSTM-CNN	0.8055	0.789	0.9256	0.919

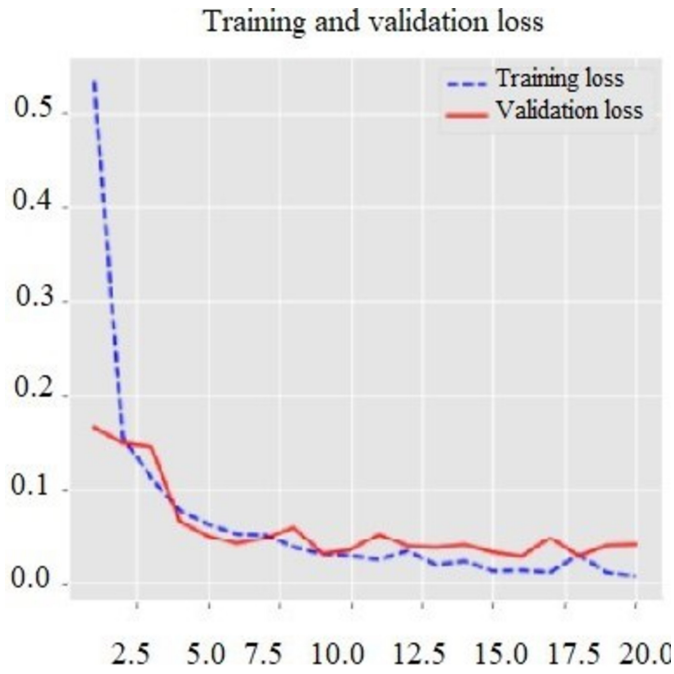


Fig. 4.9 Loss functions for CNN-LSTM model

4.6 Conclusion

In this chapter, we presented our method for Arabic text categorization. This latter has explored deep neural-based models such as CNN, LSTM, and their combination for Arabic text categorization. This allows to capture the contextual information within texts and fill up the gap related to the lack of semantics problem. This work has addressed both the morphological variety of Arabic words and the sparsity of data representation. It has also investigated the benefit of combining implicit and explicit semantics by integrating Retrofitting as a post-processing step to improve word representations. Experiments on the OSAc dataset show that the CNN model using 128 filters with the Retrofitting achieves good performance and reaches F-measure up to 99%. Moreover, the combination of both models (CNN-LSTM) performs the best results without and with Retrofitting.

As of late, Transfer learning constitutes state-of-the-art in the field of DL. In the next chapter, We will show the usefulness of transfer learning in Arabic text categorization.

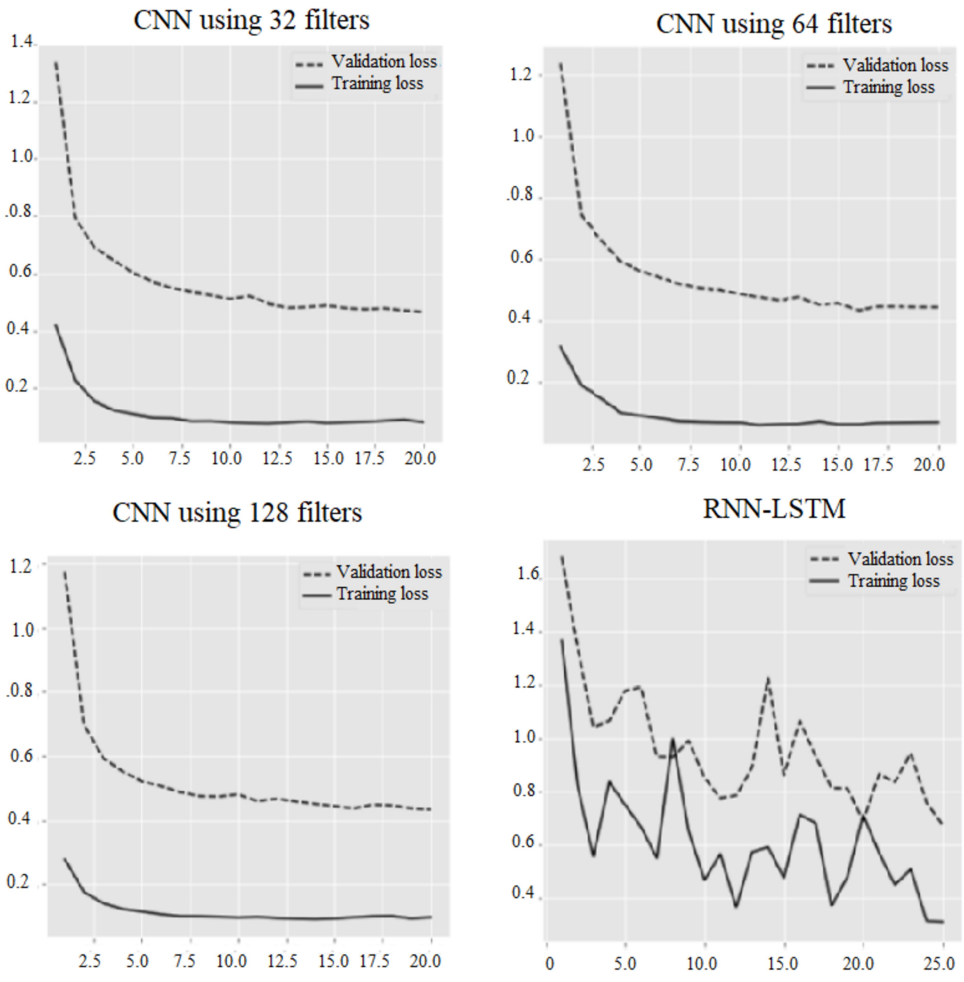


Fig. 4.10 Training and test Loss with retrofitting for the different models

CHAPTER 5

Contextual Semantic Embeddings based on Fine-tuned AraBERT Model for Arabic Text Multi-class Categorization

5.1 Introduction

Language model pretraining has achieved promising performance in learning text representation. Pre-trained word embeddings models have advanced a wide range of natural language processing applications such as Text Categorization, Question Answering, and Information Retrieval. However, these models ignore contextual information and the meaning of full sentences within the text. To overcome these limitations, contextual embeddings have proven to be useful for learning universal sentence representations. This chapter investigates the potential of the pre-trained Arabic BERT model to showcase its usefulness for Arabic text Multi-class categorization [39]. On the one hand, we exploit the Arabic BERT model after fine-tuning its parameters on the OSAC dataset to boost the categorization efficiency. On the other hand, we inquire into AraBERT performance, as a feature extractor model, by combining it with several classifiers, including CNN, LSTM, Bi-LSTM, MLP, and SVM.
The remainder of this chapter is sorted out as pursues: Section 5.2 introduces our motivations. Section 5.3 and 5.4 present the transformer architecture and BERT. Section 5.5 describes the proposed Arabic text categorization method. Section 5.6 exposes the experimental results. Finally, the last section concludes this study.

5.2 Motivations

With the recent breakthroughs in NLP, Arabic text categorization has become an active research area. A wide range of Arabic text representation methods has been proposed. However, none of the proposed methods consider relationships among multiple words and the meaning of full sentences within the text. For example, the two sentences هذا الرجل عينه and هذا الرجل عينه مصابة will have identical representation based on

word embedding and BOW representations while their meanings are completely different.
To deal with the above problem, contextual embeddings models have been shown to be useful for learning universal sentence representations. These pre-trained models allow taking into account the word context and meaning of sentences. For instance, ELMo [103] is a dynamic language modeling technique to learn the embeddings depending on the context even when the word is polysemous. It is a combination of two language models in both directions that form a multi-layer RNN. Unlike the concatenation of a left-to-right and right-to-left models that ELMo adopts, BERT [33] applies a bidirectional training of the transformer to generate more accurate word representations. There are three existing versions of BERT that treat the Arabic language, including the multilingual BERT (mBERT) and two versions of AraBERT.
We aim to handle the most encountered Arabic challenges such as ambiguity and complexity in the context of text categorization. For this purpose, we adopt the key technical innovation language modeling AraBERT. We also investigate the AraBERT model as a feature extractor with various classifiers, including CNN, LSTM, BiLSTM, MLP, and SVM. The obtained representation is able to consider contextual information within the text and capture semantic relationships between words in a sentence. To the best of our knowledge, no existing work has employed AraBERT for Arabic text multi-class categorization. The main contributions of this work are as follows:

1. We propose an effective ATC method using the contextual semantic embeddings to deal with the Arabic language's ambiguity and complexity problems.
2. We investigate the pre-trained AraBERT, then we fine-tune its parameters on ATC task, and we showcase that the fine-tuned AraBERT model achieves high performance.
3. We combine AraBERT features with various classifiers. We investigate contextual embeddings to build effective and suitable sentence representation.
4. We evaluate the AraBERT performance against its multilingual counter-part for Arabic text categorization. We conduct several comparative experiments to show the proposed approach vigor.

5.3 Transformers

The transformer is a deep learning model that is mostly used in natural language processing [128] as they are starting to make recurrent networks obsolete. Like LSTM, Transformer is an architecture to handle sequential data. Unlike RNNs, Transformers

do not require that the sequential data be processed in the order. They are based on a concept called attention, that is used to force the network to focus on a particular data point.
Transformers, in fact, consist of a number of stacked encoders (form the encoder layer), a number of stacked decoders (the decoder layer), and a bunch of attention layers (self-attentions and encoder-decoder attention). The function of each encoder layer is to process its input to generate encodings containing information about which parts of the inputs are relevant to each other. It passes its set of encodings to the next encoder layer as inputs. Each decoder layer does the opposite, taking all the encodings and processes them, using their incorporated contextual information to generate an output sequence. To achieve this, each encoder and decoder layer makes use of an attention mechanism, which for each input, weighs the relevance of every other input and draws information from them accordingly to produce the output. Each layer decoder also has an additional attention mechanism, which draws information from previous decoders' outputs before the decoder layer draws information from the encodings. The encoder and decoder layers have a feed-forward neural network for additional processing of the outputs and contain residual connections and layer normalization steps (see Fig. 5.1).

5.3.1 Scaled Dot-product Attention

The basic building blocks of the transformer are scaled dot-product attention units. When a sentence is passed into a Transformer model, attention weights are calculated between every token simultaneously. The attention unit produces embeddings for every token in the context that contains information not only about the token itself but also a weighted combination of other relevant tokens weighted by the attention weights. The attention can be expressed by the following equation:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{5.1}$$

Q is a matrix that contains the query (vector representation of one word in the sequence), K are all the keys (vector representations of all the words in the sequence), V are the values, which are again the vector representations of all the words in the sequence and $\sqrt{d_k}$ is the square root of the dimension of the key vectors.

5.3.2 Multi-head Attention

One set of (W_Q, W_K, W_V) matrices is called an attention head where W_Q is the query weights matrix, W_K is the key weights matrix, and W_V is the query weights; and each

layer in a Transformer model has multiple attention heads. While one attention head attends to the tokens that are relevant to each token, with multiple attention heads, the model can learn to do this for different definitions of ”relevance”. Since Transformer models have multiple attention heads, they can capture many levels and types of relevance relations, from surface-level to semantic. The multiple outputs for the multi-head attention layer are concatenated to pass into the feed-forward neural network layers.

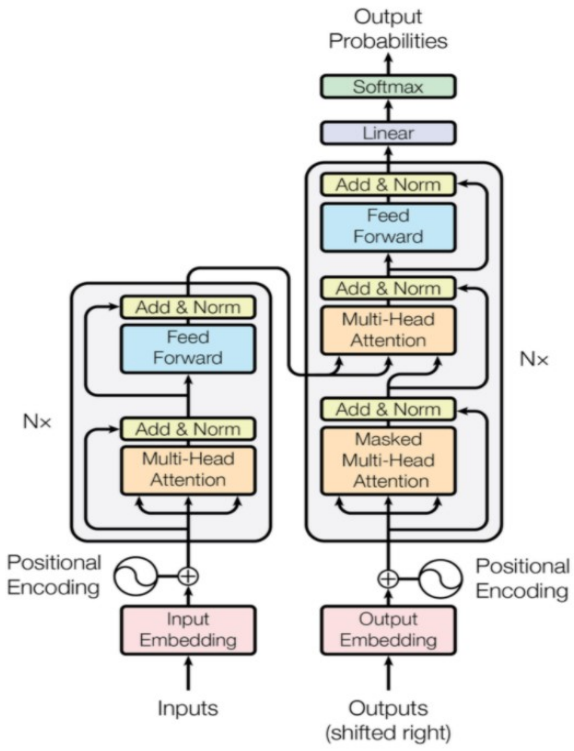


Fig. 5.1 The transformer - model architecture [128]

5.3.3 Encoder

Each encoder consists of two major components: a self-attention mechanism and a feed-forward neural network. The self-attention mechanism takes in a set of input encodings from the previous encoder and weighs their relevance to each other to generate a set of output encodings. The feed-forward neural network then further processes each output encoding individually. These output encodings are finally passed to the next encoder as its input, as well as the decoders. The first encoder takes positional information and embeddings of the input sequence as its input, rather than encodings. The positional information is necessary for the transformer to make use of the order of the sequence because no other part of the transformer makes use of this.

5.3.4 Decoder

Each decoder consists of three major components: a self-attention mechanism, an attention mechanism over the encodings, and a feed-forward neural network. The decoder functions in a similar fashion to the encoder, but an additional attention mechanism is inserted, which instead draws relevant information from the encodings generated by the encoders. Like the first encoder, the first decoder takes positional information and embeddings of the output sequence as its input, rather than encodings. Since the transformer should not use the current or future output to predict an output though the output sequence must be partially masked to prevent this reverse information flow. The last decoder is followed by a final linear transformation and Softmax layer to produce the vocabulary's output probabilities.

5.3.5 Training

Transformers typically follow a semi-supervised learning strategy involving unsupervised pretraining followed by supervised fine-tuning. In this work, we are interested in the transformer BERT that will be described in the next section.

5.4 BERT

BERT is a multi-layer bidirectional transformer encoder that uses the now ubiquitous transformer architecture [128]. The main idea is to pre-train deep bidirectional representations from unlabeled text incorporating context from both directions then fine-tune all the parameters on a downstream task. It depends on two processes, including masked language modeling and next sentence prediction.

5.4.1 Masked Language Model

MLM is a random sample of the tokens in the input sequence is selected and replaced with the special token [MASK]. The MLM objective is a cross-entropy loss on predicting the masked tokens. BERT uniformly selects 15% of the input tokens for possible replacement of the selected tokens, 80% are replaced with [MASK], 10% are left unchanged, Furthermore, 10% are replaced by a randomly selected vocabulary token. In the original implementation, random masking and replacement is performed once in the beginning and saved for the duration of the training, although in practice, data is duplicated so the mask is not always the same for every training sentence

5.4.2 Next Sentence Prediction

NSP is a binary classification loss for predicting whether two segments follow each other in the original text. Positive examples are created by taking consecutive sentences from the text corpus. Negative examples are created by pairing segments from different documents. Positive and negative examples are sampled with equal probability. The NSP objective was designed to improve performance on downstream tasks, such as Natural Language Inference, which require reasoning about the relationships between pairs of sentences.

5.5 AraBERT for Text Categorization

AraBERT, the Arabic variant of BERT, is a multi-layer bidirectional transformer encoder based on the implementation of the transformer that is described in the former section.
We propose a contextual embeddings-based method for Arabic text categorization which comprises two variants. The first one explores AraBERT pre-trained model then fine-tunes its parameters. Whereas, the second variant investigates the AraBERT as feature extractor model in conjunction with several neural network classifiers and SVM. Fig. 5.2 illustrates the overall architecture of our system that is composed of several main steps including text preprocessing, the pre-trained AraBERT fine-tuning and text features computation via the same model, and finally text category prediction.

5.5.1 Text Preprocessing

The text preprocessing consists of several steps as follows:

- Removing punctuation, diacritics, non-letters and non-Arabic words.
- Normalizing the different آ ، أ ، إ (Alef) writings with ا and replacing ة (Teh Marbuta) with ه.
- Data transformation into a format that matches the fine-tuned AraBERT model. It consists of the following stages:
    1. We first apply the tokenization to break up the input texts into tokens. Table 5.1 presents the result of AraBERTv1 and mBERT tokenizers applied to an example sentence. We observe that AraBERT tokenizer is more appropriate for Arabic since it uses Farasa Segmenter [1], which considers the characteristic of Arabic morphology;

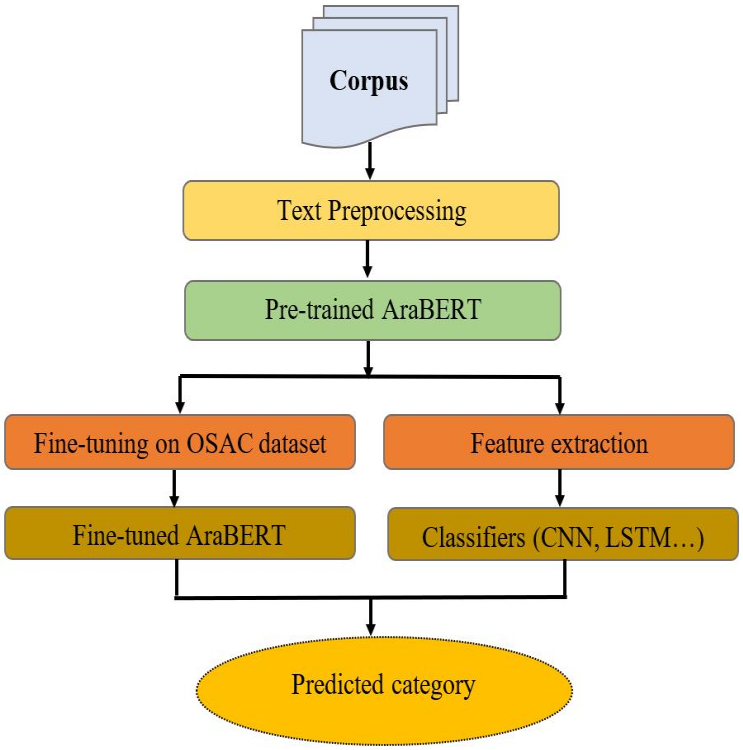


Fig. 5.2 The overall Arabic text categorization system architecture

2. Afterward, we convert each text to a BERT's format by adding the special [CLS] token in the beginning of each text, and [SEP] token between sentences and at the end;
3. We map each token to an index based on the pre-trained AraBERT's vocabulary.

Table 5.1 Example of tokenization process with the two versions of BERT

Sentence	أشار سليمان الى ان الجهاز الفني قرر تقليل الفترات التدريبية
AraBERTv1 tokenizer	'اشار', 'سليمان', 'الى', 'ان+', 'ال+', 'جهاز', 'ال+', 'فني', 'قرر', 'تقليل', 'ال+', 'فتر', 'ات+', 'ال+', 'تدريبي', 'ة+'
mBERT tokenizer	'اش', '##ار', 'سليمان', 'ال', '##ى', 'ان', 'الجهاز', 'الفن', '##ي', 'ق', '##رر', 'ت', '##قليل', 'ال', '##فت', '##رات', 'ال', '##ت', '##در', '##يب', '##ية'

5.5.2 AraBERT Fine-tuning

In this work, we explore the pre-trained AraBERT and fine-tune it for Arabic text categorization. To do so, we connect the AraBERT outputs, after the fine-tuning, to an additional layer involving the Softmax classifier to predict the text category. First, we tokenize each text to N tokens and add the [CLS] token in its beginning. Afterward, we

generate for each token i an input representation E_i that is constructed by summing the vector embeddings corresponding to the token, the segment, i.e. sentence to which it belongs, and the token position. Then, we feed the E_i vectors to AraBERT and fine-tune its parameters using the labeled data from the corpus. As shown in Fig. 5.3, we denote the final hidden vector of the special [CLS] token as h and the final hidden vector for the i-th input token as T_i. Lastly, we take the final hidden state h as the representation of the whole text and utilize it as an input for the feed-forward layer with Softmax classifier to get the probability distribution over the predicted output category c [118]:

$$p(c|h) = Softmax(Wh) \tag{5.2}$$

Where W is our task-specific parameter matrix.
During fine-tuning, all parameters from AraBERT as well as W are trained jointly to maximize the log-probability of the correct category.
In the next subsection, we describe the second variant that studies AraBERT as a feature extractor model.

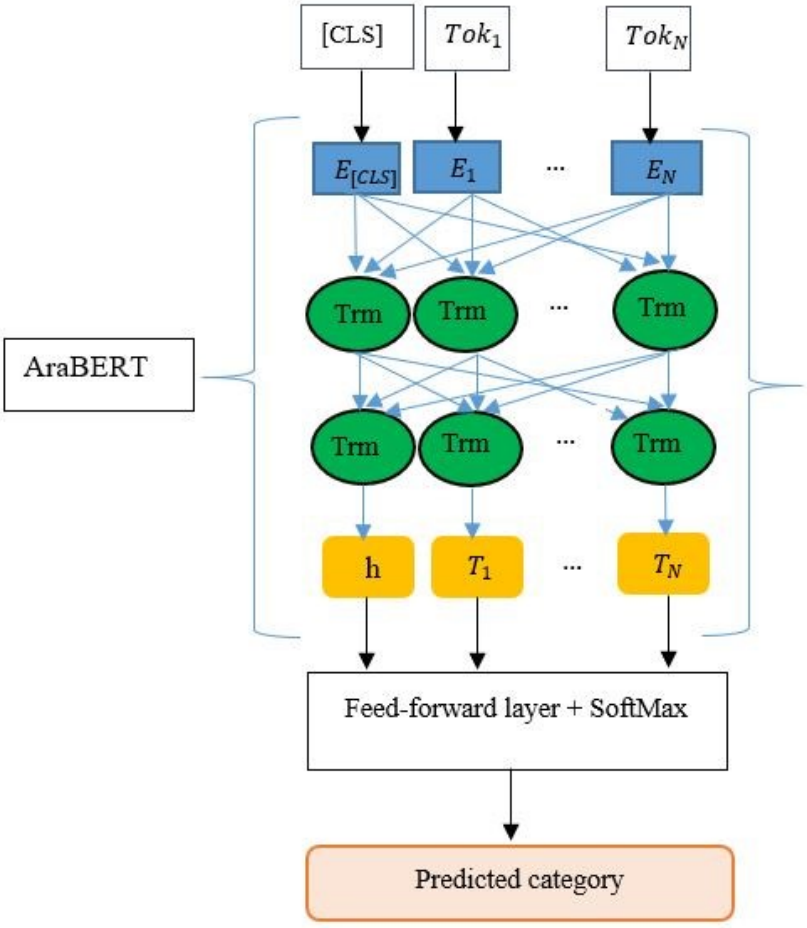


Fig. 5.3 The flowchart of fine-tuned AraBERT for ATC

5.5.3 AraBERT for Feature Extraction

The second variant of our method involves the AraBERT exploration as feature extractor. It consists of two steps: (1) Extracting fixed features from the pre-trained AraBERT and (2) Feeding the obtained features to the classifier models.

5.5.3.1 Text Representation

We compute text representation using the contextual pre-trained AraBERT. The basic idea is to represent a text into a fixed length vector. Therefore, each text is translated to 768-dimensional vector embeddings, where each element of this vector encodes implicit semantics of the original text. Inspired from [133] and as illustrated in Fig. 5.4, we explore *Bert-as-a-service* technique that extracts activations from one or more layers without fine-tuning any parameters of AraBERT. The average pooling of the second-to-last hidden layer of all tokens is computed.
The obtained representations are used as input for various classifiers such as CNN, LSTM, BiLSTM, MLP and SVM. We adopt the feature-based approach having the major advantage which is decreasing the categorization complexity algorithms by pre-computing text representations then building low-cost models on the top of the obtained vectors.

5.5.3.2 Text classification

The resulting contextual embeddings vectors from the above process are fed to several classifiers including CNN, LSTM, BiLSTM, MLP and SVM.

- CNN: We build a 1D convolutional network where the first layer is the convolutional one comprising different parameters as the filter size fixed to 64. These feature detectors are passed across the text representations matrix to identify specific features utilizing the ReLU function. The results of the convolution are pooled and then passed to a fully connected layer. Afterward, we employ dropout to prevent overfitting and increase its fraction to 0.5. Finally, the Softmax function is used in the output layer to categorize texts into predefined classes.
- LSTM: We construct an LSTM with one hidden layer containing 100 units. As input, we provide the representations matrix comprising the embedding vectors. The dropout fraction is fixed at 0.5. Then, the previous layer's resulting vectors are fed to a fully connected network. Finally, a *softmax* layer is used to get the appropriate category.

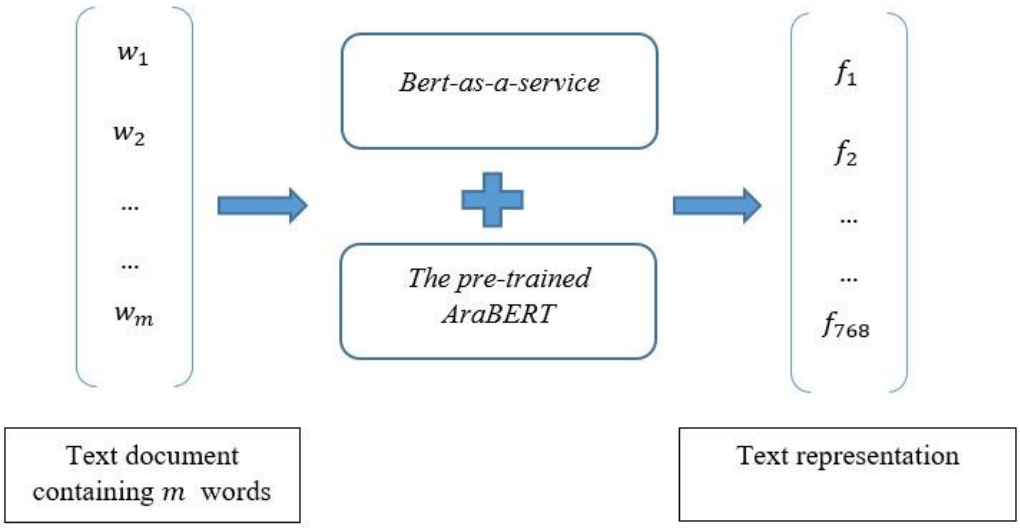


Fig. 5.4 Feature-based approach with AraBERT

- BiLSTM: We train a Bidirectional LSTM model on the obtained text representations. The first hidden layer of the constructed BiLSTM network contains 100 memory units. To avoid the overfitting issue, we adopt the dropout and fix its parameter at 0.5. Then, the outputted vectors are connected to a fully connected layer. Finally, the resulting features are fed to the output layer to predict the text category.
- MLP: In this model, we build an MLP classifier which takes as input the embeddings matrix. It relies on an underlying neural network containing 100 hidden layers with 50 nodes for each one. We perform the *Adam* optimizer and *softmax* function in the final layer receiving as input feature vectors from the previous layer to classify texts.
- SVM: We train an SVM employing the Radial Basis Function kernel to map the obtained text representations space in infinite dimensional space. The SVM based architecture predicts the text categories using a value of 0.4 for the gamma parameter and setting the value of the Penalty Parameter to 10000.

Table 5.2 OSAC dataset statistics

Datasets	Categories Number	Sample Number	Words Number	Keywords Number
CNN	6	5.070	2.241.348	144.460
BBC	7	4.763	1.860.786	106.733
OSAc	10	22.429	18.183.511	449.600

5.6 Experiments And Results

We conduct several experiments in order to evaluate the proposed Arabic text categorization method. We utilize the OSAC including three datasets including three datasets with varying size. For all experiments, 80% of the data is designated for training and 20% for testing. We make the use of the AraBERT model [12], which is pre-trained on 70 million Arabic sentences, corresponding to 24GB of text configuration. It has 12 encoder blocks, 768 hidden dimensions, 12 attention heads, 512 maximum sequence length and a total of 110M parameters. We explore two versions of the AraBERT model: AraBERTv0.1 and AraBERTv1. Furthermore, we evaluate the AraBERT against the pre-trained multilingual BERT [33]. This latter is cased and trained on 104 languages chosen from the entire Wikipedia dump for each language excluding user and talk pages. It utilizes 12-layer, 768-hidden, 12-heads and 110M parameters.

5.6.1 Dataset

The Open Source Arabic Corpora (OSAC) is a large standard dataset for text categorization [110]. It consists of different corpus including BBC, CNN and OSAc. The datasets statistics are shown in Table 5.2 and the distribution of each category according to the number of documents are illustrated in Fig. 5.5.

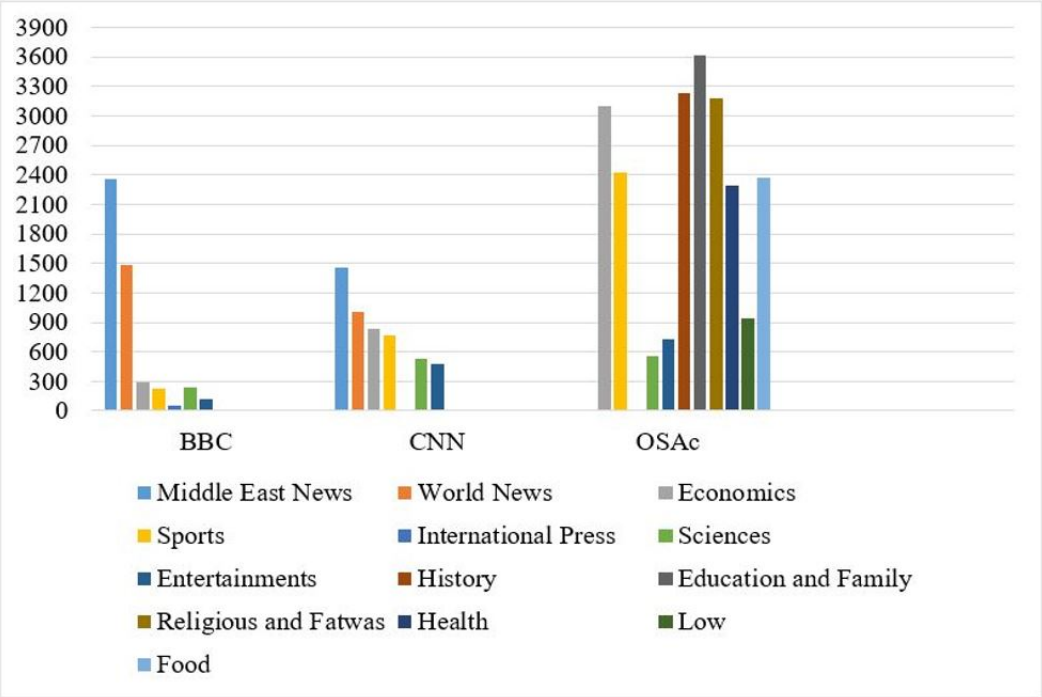


Fig. 5.5 Categories distribution in OSAC according to document number

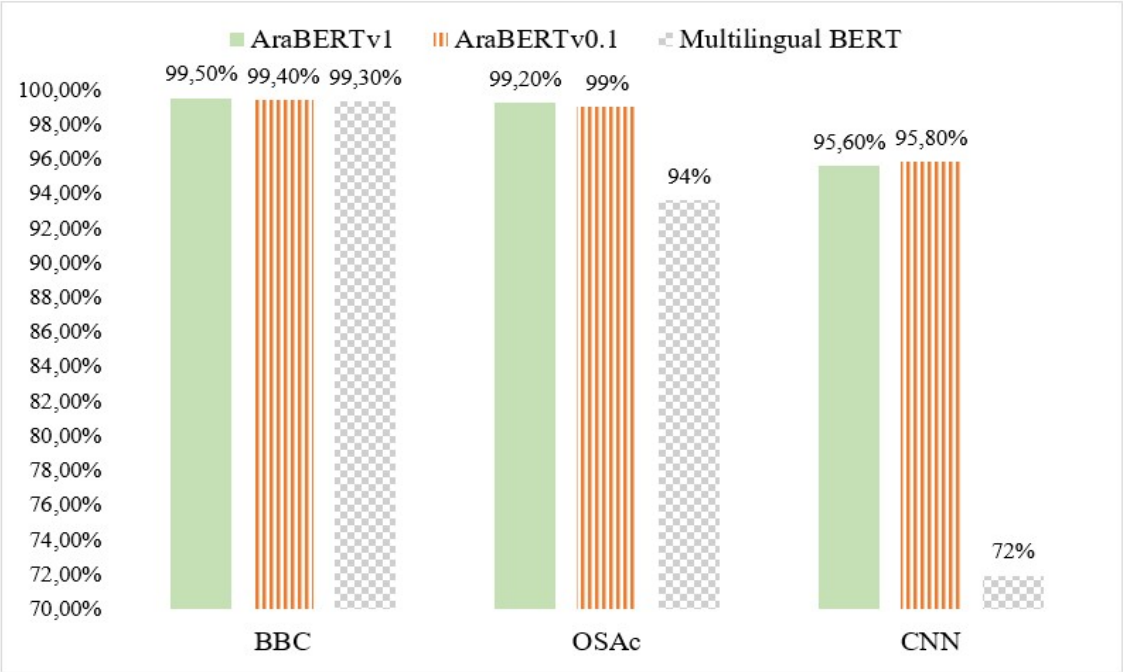


Fig. 5.6 Accuracy Result of three versions of fine-tuned BERT models on OSAC dataset

5.6.2 Discussion

We have carried out extensive experiments to show the impact of using transfer learning from the pre-trained models BERT for Arabic text categorization. Firstly, the experiments are conducted to investigate the applying of fine-tuned BERT models using different OSAC's datasets. We explore two versions of pre-trained AraBERT model including AraBERTv0.1 and AraBERTv1, and evaluate them against the mBERT model. Fig. 5.6 reports the obtained results of the fine-tuned BERT models in terms of accuracy. For BBC dataset, all models produce close results varying between 99.3% and 99.5%. For OSAc corpus, it can be noticed that the fine-tuned mBERT gains 93.6% of accuracy while the fine-tuned AraBERT achieves 99% and 99.2% based on AraBERTv0.1 and AraBERTv1, respectively. Hence, the Arabic models allow an approximately 5% improvement compared to the multilingual model. Concerning the CNN dataset, the worst and best results are reported by the fine-tuned mBERT model with accuracy of 72% and the fine-tuned AraBERT models with accuracy of 95%. In fact, the fine-tuned AraBERT allows nearly 23% improvement in the categorization results.
Overall, the fine-tuned AraBERT models outperform the fine-tuned mBERT. This is explained by the fact that the AraBERT models are pre-trained only on Arabic language. Furthermore, they employ large data and vocab in comparison with the mBERT leading to word diversity.
In the second stage of our study, we aim at answering the following question: does incorporating contextual embeddings improve the Arabic text categorization perfor-

mance? For this purpose, we evaluate the fine-tuned AraBERT models against the baselines including TFIDF_SVM, LSA_SVM, Word2vec_SVM, Doc2vec_SVM, Doc2vec_MLP, LSTM and BiLSTM.

Table 5.3 Performance results of the fine-tuned AraBERT and baseline models on OSAc

Models	Accuracy	F1
BiLSTM	93	93
LSTM	91	91
Doc2vec_MLP	90	90
LSA_SVM	90	91
Doc2vec_SVM	89	89
TFIDF_SVM	88	88
Word2vec_SVM	72	73
Fine-tuned AraBERTv0.1	99	99
Fine-tuned AraBERTv1	99.2	99

- TFIDF_SVM: We make the use of the text preprocessing and Khoja stemmer on the Bag-Of-Words. Then, we compute text representations based on TF-IDF. These latter are fed to an SVM classifier.
- LSA_SVM: We explore the LSA method to generate text features. Each text is transformed to a vector containing 1200 features. Afterward, they are classified using the SVM.
- Word2vec_SVM: We train text representations relying on the average of word embeddings. We utilize the pre-trained model presented in Ref. [19]. Then, we apply the SVM classifier on the resulting representation vectors.
- Doc2vec_SVM: We train the Doc2vec model on OSAc to produce text vectors. Each vector is represented by 300 dimensions. Afterward, these representations are fed to an SVM.
- Doc2vec_MLP: We use the model implemented in our work [36]. We build a categorization system using word sense representations, document embeddings and word sense disambiguation. The obtained representations are fed to the MLP network.
- LSTM: We build an LSTM with 100 units on the top of the Arabic word embeddings model from Ref. (Grave et al. 2018). The resulting features are passed to

a fully connected layer. Then, the generated representations are used as input for the classification process to predict the text category.

- BiLSTM: We provide text representations to a BiLSTM with 100 units trained on the top of word embeddings model [19]. These latter are connected to a fully connected layer. The resulting vectors from the previous layer are fed to the classification one. During the training, we minimize the cross-entropy error and apply the Adam optimizer.

Table 5.3 reports the accuracy and F1-score results of the fine-tuned AraBERT models and baselines. We notice that the two fine-tuned AraBERT beat all the baselines. The fine-tuned AraBERT models outperform the classical TFIDF_SVM method by 11% in terms of accuracy and F1-score. They also exceed Doc2vec_SVM by 10% and Doc2vec_MLP by 9%. Moreover, the fine-tuned AraBERT models surpass the LSA_SVM model, which reaches only 90% and 91% of accuracy and F1-score, respectively. Therefore, we note an enhancement of 9% by our models. In contrast, Word2vec _SVM gets the lowest accuracy, i.e., 73% of F1-score, compared to the other baselines. The fine-tuned AraBERT models also outperform both LSTM and BiLSTM models. This is not surprising that the two fine-tuned AraBERT models accomplish the best performances since they incorporate context from both directions.
In the third stage and in order to show the advantage of taking both left and right contexts simultaneously, we lead further experiments to compare the fine-tuned AraBERT model with ELMo and UMLFiT. Fig. 5.7 presents the obtained performances using the CNN dataset. We employ the pre-trained ELMo Representations for Many Languages [25] and the Arabic UMLFiT model from [44]. We notice that the fine-tuned AraBERT model outperforms ELMo by a score of 20%. In contrast, AraBERT surpasses ULMFiT by only 6%. This is due to the fact that AraBERT conditions on both left and right context in all layers of the Transformer model. While in ELMo contextual embeddings are generated by a bidirectional recurrent neural network that simply concatenates left and right context at the last layer. Whereas the ULMFiT is a unidirectional model.
Finally, we aim to assess AraBERT as feature-based model to show the impact of abstracting the fine-tuning approach on the categorization performance. We incorporate the previous model with several classifiers involving CNN, LSTM, BiLSTM, MLP and SVM. Table 5.4 summarizes the categorization results of AraBERT and mBERT as feature extraction models on the OSAc dataset. We observe that the AraBERTv0.1 model merged with the SVM classifier accomplishes the best results by 97% in terms of F1-score. The MLP and BiLSTM classifiers also perform good results and attain 96% in terms of F1-score, while CNN and LSTM achieve the lowest accuracy by 95%. Concerning the AraBERTv1 combined with all the classifiers, it reaches almost 96% as the highest value of F1-score. However, the mBERT performs 95% in terms of F1-

score based on the SVM classifier. The mBERT merged with MLP, LSTM, and CNN accomplishes 93% in terms of F1-score. Overall, the AraBERT models achieve good performance as feature extractors and beat mBERT. This demonstrates that AraBERT is promising for both fine-tuning and feature-based approaches.
In the main, the extensive set of experiemnts on OSAC show that the fine-tuned AraBERT models exceed all state-of-the-art methods and the fine-tuned mBERT model. Furthermore, they outperform AraBERT as feature extractor.

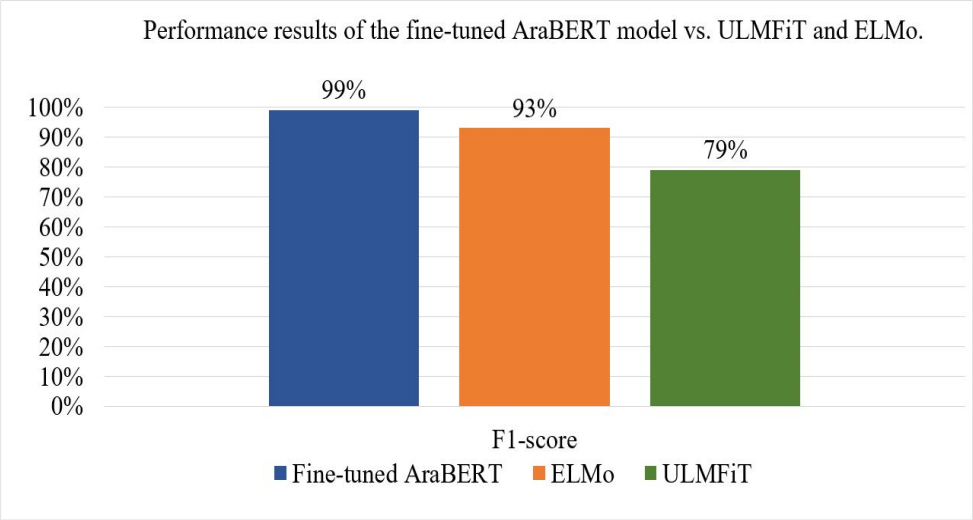


Fig. 5.7 Performance Result of three versions of fine-tuned BERT models on OSAC dataset

Table 5.4 Performance results of AraBERT vs. mBERT as feature extraction models using different classifiers on OSAc

Classifiers	mBERT	AraBERTv1	AraBERTv0.1
CNN	93	96	96
LSTM	93	96	96
BiLSTM	94	96	96
MLP	93	95	96
CNN	95	96	97

5.7 Conclusion

In chapter, before diving into the proposed method for enhancing Arabic text categorization, we have introduced and discussed the transformer architecture and BERT. We proposed a contextual semantic embeddings-based method for enhancing Arabic

text categorization. It consisted in exploring two variants of the pre-trained AraBERT model as advancement in Arabic text categorization field. The first variant relied on the pre-trained AraBERT fine-tuning, whereas the second one investigated multilingual BERT and Arabic BERT models as feature extractors combined with several classifiers. The use of contextual information allowed us to handle the Arabic language problems, such as the complexity and ambiguity, by considering a specific model (AraBERT) for Arabic. Indeed, by adopting sentence embeddings representation, our method was able to capture semantics within texts by representing relationships among multiple words and phrases in a single dense vector. To improve contextual text representations learning, we used the obtained fine-tuned BERT model to represent the sentences. To assess the effectiveness of the proposed method, we conducted a set of experiments on the OSAC dataset. The obtained results proved that the fine-tuned AraBERT model outperformed mBERT, achieved a new state-of-the-art results and reached up to 99% in terms of accuracy on the OSAC datasets.

In the next chapter, we will approach BERT but from another perspective. We will discuss our last contribution related to offensive language detection.

CHAPTER 6

A Multilingual Offensive Language Detection Method based on Transfer Learning from Transformer Fine-tuning Model

6.1 Introduction

In the most recent decade, with the rise of an interactive web and particularly popular online social media like Facebook and Twitter, there has been an exponential increment in the user-generated content being made accessible over the web. Presently any information online can arrive at billions of web users in mere seconds. This has led to not only a positive exchange of ideas but resulted in malicious and offensive content over the web. This ushers social media administrators to automatically supervise the content using NLP techniques and ban offensive messages.
In this chapter, we discuss our contribution to the field of offensive language detection in a multilingual context.
This chapter is organized as follows: in section 6.2, we present our motivations. Section 6.3 gives an overview of offensive language detection and multilingualism in the text categorization field. Section 6.4 describes the proposed method. Section 6.5 presents the experiments and the obtained results. Finally, the last section concludes the chapter.

6.2 Motivations

The offensive language detection [139] and the multilingual offensive language identification in social media [140] shared tasks in International Workshop on Semantic Evaluation (SemEval) attracted submissions from more than 100 teams. These tasks comprise three hierarchical classification sub-tasks: A) Offensive language identification, B) Automatic categorization of offense types, and C) Offense target identification. Previous works on offensive language identification deal with this problem in a monolingual context by working on specific languages like English [140], Arabic [8] or other languages. However, several languages prevail the world-wide networks conducting

to the curse of multilingualism in the NLP field. Hence, multilingual text classification is considered an important task to tackle with this issue. MTC can be formalized as follows: Given a set of texts written in different languages (Arabic, English, Spanish...) and belonging to a set of categories that is fixed across languages, we aim to categorize those texts simultaneously. Note that this problem is different from cross-language text categorization [16] when a document written in one language must be classified in a category system learned in another language. Several approaches exist to deal with the MTC problem; the first one consists of developing several monolingual classifiers where each language has a specific classification system. The second method involves one classification system for different languages; the basic idea is to feed different texts with different languages to the same classifier then perform the training on a multilingual dataset. The third method incorporates the translation phase to format all data to one language, then develop one classification system. BERT has the capability to learn how to extract complex features from raw data automatically, thus have invaded the natural language processing field and given promising performances [33]. Multilingual BERT has also pushed state-of-the-art on cross-lingual and multilingual understanding assignments by jointly pre-training large Transformer models on many languages [128]. In this work, we propose a BERT based-method to label bilingual tweets by investigating joint-multilingual and translation-based classification methods. The main contributions to the offensive language identification field can be summarized as follows:

1. We propose an effective method to label Arabic and English tweets into two categories: Offensive or not offensive.
2. We explore contextual embeddings, including BERT, mBERT, and AraBERT [12], to detect offensive language over multilingualism.
3. We present two methods to deal with the multilingualism problem, including the joint-multilingual technique and the translation-based one.

6.3 Multilingual Offensive Language Detection

6.3.1 Offensive Language Detection

As of late, the identification of cyberbullying, aggression, hate Speech, toxic comments, and offensive language in Social Media is of great importance and receives much attention from the researcher's community. Several public datasets are available to train machine classifiers for those assignments. However, there are no standard benchmark corpora or training sets that can be combined to obtain more robust classification sys-

tems. For aggression identification, [69] presented the Shared Task on Aggression Identification report and findings. The provided dataset contains 15,000 annotated Facebook posts and comments in English and Hindi. The goal was to discriminate between three classes: non-aggressive, covertly aggressive, and overly aggressive. The toxic comment classification was an open competition at Kaggle. Various methods were evaluated for this task on a dataset containing user with comments from Wikipedia. These comments are organized into six classes: toxic, severe toxic, obscene, threat, insult, identity hate. Concerning the hate speech identification, [31] presented a recent hate speech detection dataset with over 24,000 English tweets belonging to three categories: non-offensive, hate speech, and profanity. [85] reported the shared tasks on offensive language identification where three datasets were developed from Twitter and Facebook and made available for Hindi, German, and English. Moreover, [139, 140] presented several offensive language detection results in several languages obtained by SemEval competition teams.

6.3.2 Multilingual Text Categorization

Multilingual Text Categorization is an emerging field in machine learning that enables global knowledge discovery within international information networks and allows retrieval, discovery, and categorizing documents in whatever language. This leads to the convergence of various interests from diverse research communities focusing on the issues related to multilingual text categorization. Several methods were proposed to deal with MTC, including multiple monolingual classifications, joint multilingual classification, and joint translated monolingual classification.

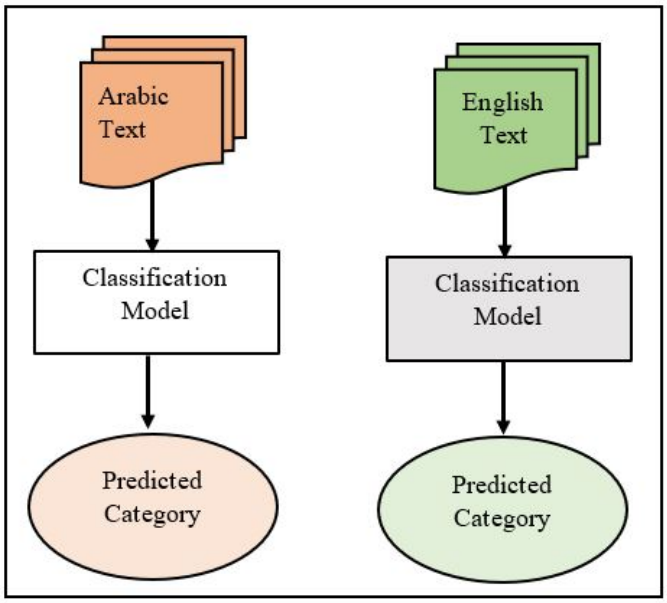


Fig. 6.1 Multiple monolingual classification

6.3.2.1 Multiple Monolingual Classification

Multiple monolingual classification methods consist of developing a classifier for each language. Considering the two languages Arabic and English, we will use a separate classification system for each one. We train each classifier in one language. We choose to classify the multilingual documents using the same classifiers but independently. Fig. 6.1 shows the process of the first variant.

6.3.2.2 Joint-Multilingual Classification

The joint multilingual method incorporates all the languages to classify them together with a single classification system. We generate a multilingual classification system that merges both English and Arabic documents into a single document. Then, we apply specific classifiers. Fig. 6.2 shows the process of the joint-multilingual method.

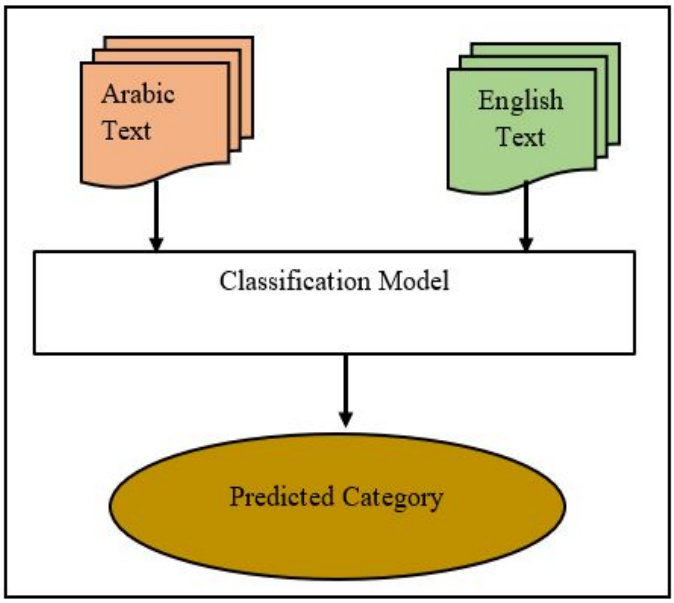


Fig. 6.2 Joint-multilingual classification

6.3.2.3 Joint-Translated Monolingual Classification

We proceed by translating English documents to a universal language using google translator API. Then, we merge and unify all texts. Afterward, we feed all the documents to the classification model. Fig. 6.3 describes the joint-translated monolingual method steps.

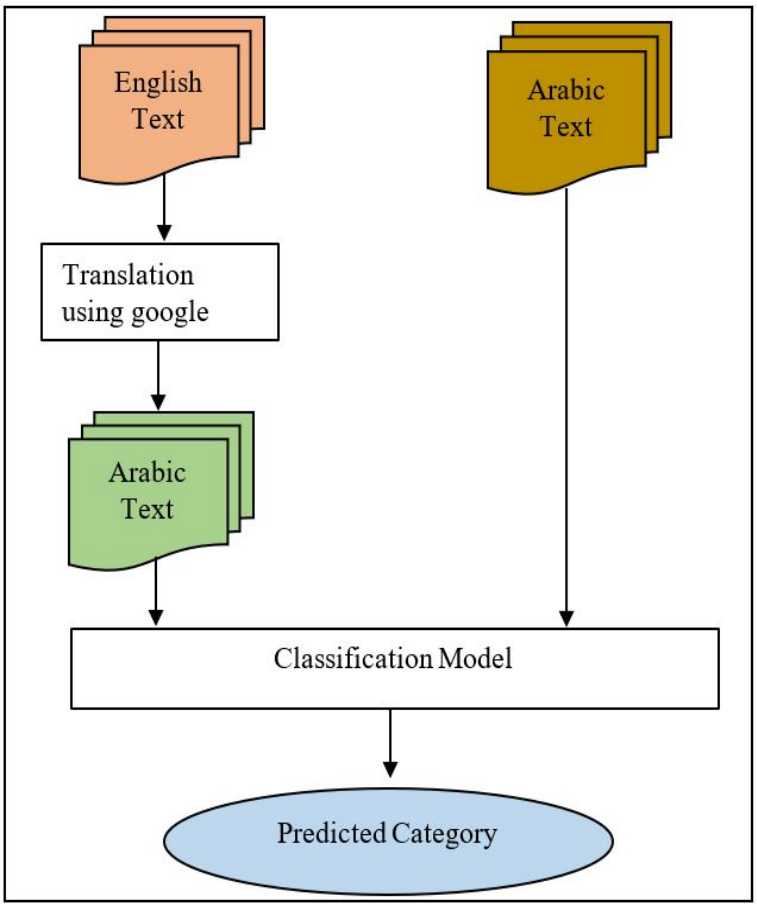


Fig. 6.3 Joint-translated monolingual classification

6.4 Methodology

In this section, we present our method that concerns offensive language detection using transfer learning. In this contribution, we work in a bilingual context by considering the classification of Arabic and English tweets. As illustrated in Fig. 6.4, our method comprises several phases, including tweet preprocessing, translation, BERT tokenization, tweet representation, and classification.

6.4.1 Preprocessing

The first step in our model is preprocessing the tweets in order to keep the pertinent information only. To do so, we follow the upcoming steps:

1. We proceed by removing all the hashtags, HTML tags, mentions, and URLs.
2. For English text, we further replace contractions with their full forms, fix misspelled words, and convert text to lowercase.

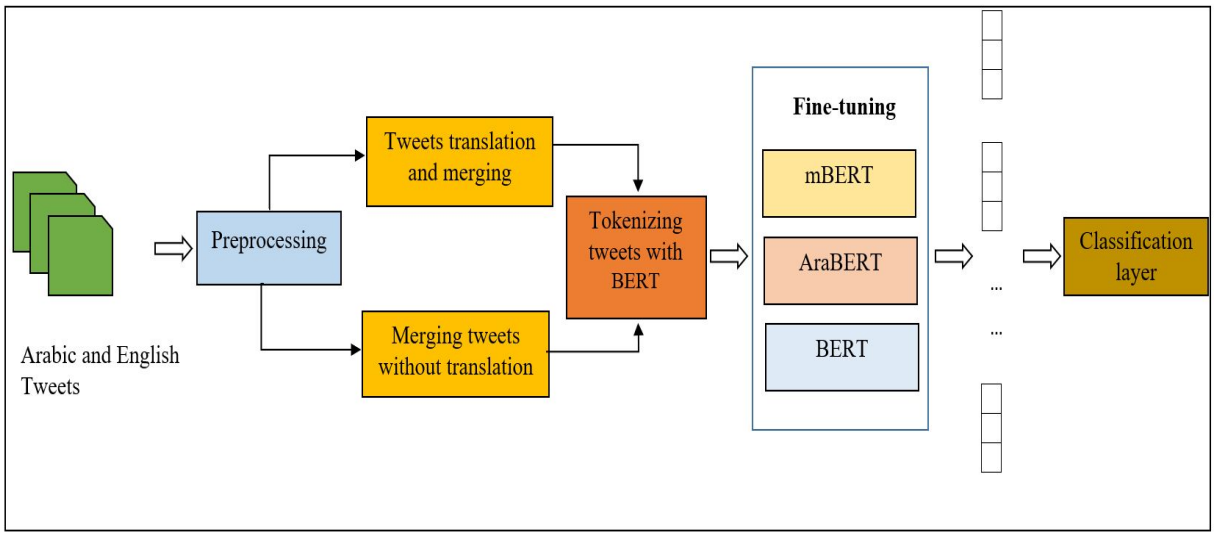


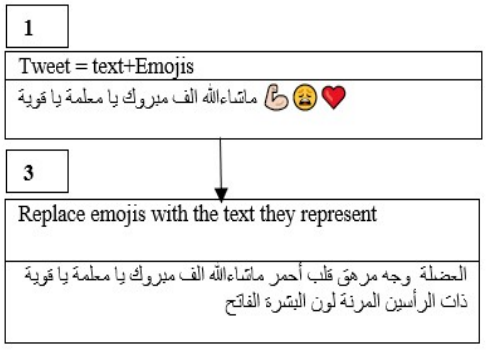
Fig. 6.4 The multilingual offensive language detection method flowchart

3. We replace emojis if they existed with the text they represent since emojis or emoticons play an essential role in representing a sentiment. We have two cases:

   - For Arabic tweets, we first detect emojis then translate emojis meanings from English to Arabic. Afterward, we substitute emojis by their Arabic meanings in the tweet. Fig. 6.5 illustrates the flowchart of Arabic tweet substitution if emojis exist.

   - For English tweets, we replace emojis by their meaning if they exist.

After the preprocessing step and depending on the chosen strategy to deal with the multilingualism, we either merge tweets without translation or after translation using the translation process described in the next section.



2

Emojis	English meaning	Arabic meaning
❤️	red_heart_selector	محدد القلب الأحمر
😫	weary_face	وجه مرهق
💪	flexed_biceps_light_skin_tone	العضلة ذات الرأسين المرنة لون البشرة الفاتح

Fig. 6.5 Emojis preprocessing in Arabic tweets

6.4.2 Translation

In this stage, we generate two corpora as follows:

- To generate the first corpus, we translate English tweets to Arabic using google translator API. Then, we merge Arabic and translated tweets into one document. Afterward, we apply the AraBERT tokenizer.
- To produce the second corpus, we translate Arabic tweets into English based on the same API. Then, we combine English and translated tweets into one file. Next, we feed all tweets to the BERT tokenizer.

6.4.3 BERT Tokenization

We first add the special token [CLS] to each tweet's head and the special token [SEP] between sentences and at the end during the tokenization. For Arabic, all the words tokens are segmented by Farasa segmenter [1] and then tokenized with AraBERT tokenizer. For English and bilingual tweets, we use the BERT tokenizer. Finally, we map each token to an index based on the pre-trained BERT model vocabulary.

6.4.4 Tweet Representation

In this work, we explore three models of the transformer, including BERT, mBERT [33], and AraBERT [12], according to the multilingual approach. Uncased BERT base (L=12-layer, H=768, A=110M params), mBERT (L = 12, H = 768, A = 110M params) whereas the AraBERT (L = 12, H = 768, A = 110M params) is pre-trained on 70 million Arabic sentences. To fine-tune BERT models on the multilingual text classification, we tokenize the input texts, add [CLS] and [SEP] tokens and then generate for each token an input representation that is constructed by summing the vector embeddings corresponding to the token, the segment to which it belongs, and the token position. Then, we feed these representation vectors to BERT and fine-tune its parameters on the specific task offensive language identification on Twitter using a pooling strategy.

6.4.5 Text Classification

After the transformer fine-tuning and in order to detect the tweet label, we feed tweets representations to a sigmoid function and train the model to optimize the binary cross-entropy loss. During fine-tuning, we define the weights for the classification layer and calculate a standard classification loss.

6.5 Experimental Results

6.5.1 The Multilingual Corpora

The English dataset is extracted from the semi-supervised offensive language identification dataset in the SemEval'2020 competition. This dataset was proposed by [106], containing over 9,000,000 annotated tweets following OLID's three-level taxonomy: Offensive Language Detection, Categorization of Offensive Language, and Offensive Language Target Identification. We take 5994 English tweets from the Offensive Language Detection dataset covering two categories: Offensive and not offensive.
For the Arabic dataset, we extract 8000 tweets from the dataset proposed by [92] in SemEval'2020. It is a large dataset that contains 10000 labeled tweets with OFF or NOT, for offensive and not offensive.

6.5.2 Experimental Setup

We aim at exploring the success of the transformer BERT to classify bilingual tweets into two categories: Offensive and not offensive. Therefore, we conduct a set of experiments to evaluate our method. All the experiments are performed on the corpus described in the previous subsection. We split the dataset into 80% for training and the rest 20% for testing. As measure evaluation, we utilize two metrics, including Accuracy and F-measure.

6.5.3 Comparison Methods

We compare BERT models against various other deep learning models, including CNN, RNN, and bidirectional RNN. For these neural networks, the preprocessing phase was required; the principal hierarchy consists of the embedding layer, hidden layer, and output layer. The Embedding layer is considered the first hidden layer of deep learning networks. The input size of each vector is 200. We use the dropout to avoid the overfitting problem; its parameter is set to 0.5. The vector size for each text is 300 dimensions. The output layer utilizes the Sigmoid activation function and the cross-entropy loss to predict tweet labels. Concerning the parameters setting, every deep neural network has its own adaptation.

- CNN
  We build a CNN model consisting of a 1D convolutional layer with a kernel size of 5 and 128 filters. The next layer is the max-pooling with default values

followed by the dropout layer. Finally, the output layer is responsible for affecting a category to each text.

- RNN
  We make the use of LSTM and GRU models. The hierarchy of our LSTM model consists of one LSTM layer containing 100 units, followed by a dropout layer. The final layer is the classification one that predicts the tweet category. We keep the same architecture for the GRU model, and we change the LSTM layer by a GRU layer. This architecture gives the best performances after trying different models. Both models are designed to tackle the problem of the Vanishing Gradient of the basic RNN.

- Bidirectional RNN
  We construct a bidirectional LSTM composed of one BiLSTM layer with 100 hidden units. The outputted vectors are flattened then fed to the classification layer. While the Bidirectional GRU is built using a BiGRU layer with the same configuration of BiLSTM.

- Model combinations
  We combine several models differently. First, we merge CNN and RNN layers to produce CNN-LSTM and CNN-GRU. Both models comprise one CNN layer followed by one RNN layer. Then, we use the global max pooling and the dropout layer, and finally, the prediction layer to attribute category to tweet. Concerning the LSTM-CNN combination, we use one LSTM layer with 100 hidden units followed by a 1D convolutional layer. Moreover, we add the global max pooling and dropout layers. The CNN-BiLSTM model combines CNN with Bi-LSTM. The networks' hierarchy contains one CNN layer and one BiLSTM layer with 100 hidden units followed by global max pooling and a dropout layer.

6.5.4 Joint-Multilingual Classification Results

The first experiment is designed to perform the mBERT model for offensive language detection. We evaluate mBERT for joint multilingual classification against CNN, LSTM, GRU, BiLSTM, and CNN-BiLSTM models. Furthermore, we compare the joint multilingual approach with the monolingual one using only the Arabic tweets. For the monolingual text classification, we perform the Arabic version of BERT, CNN, and GRU on the Arabic dataset only. Table 6.1 summarizes the obtained accuracy and F-measure. It can be observed that mBERT improves the overall performance metrics by 91% of accuracy and F-measure and beats all the baselines. It exceeds CNN by 2% and 1% for accuracy and F-measure, respectively.

Table 6.1 Performance results of joint-multilingual classification method on the bilingual tweets dataset

Models	Accuracy(%)	F-measure(%)
CNN	89.74	90
LSTM	66.43	80
GRU	90.06	90
BiLSTM	90.42	89
CNN-BiLSTM	88.36	89
Monolingual AraBERT	92.65	79
Monolingual_CNN_ AraBERT	87.69	89
Monolingual_GRU_ AraBERT	79	89
mBERT	91.36	91

Moreover, mBERT outperforms the LSTM model by 10% in terms of F-measure. GRU and BiLSTM also perform well and achieve 90% of accuracy. However, they do not surpass mBERT performances. It is also reported that mBERT exceeds the AraBERT model for monolingual text classification by 11% of accuracy. Hence, the joint multilingual text classification approach outperforms the monolingual one.

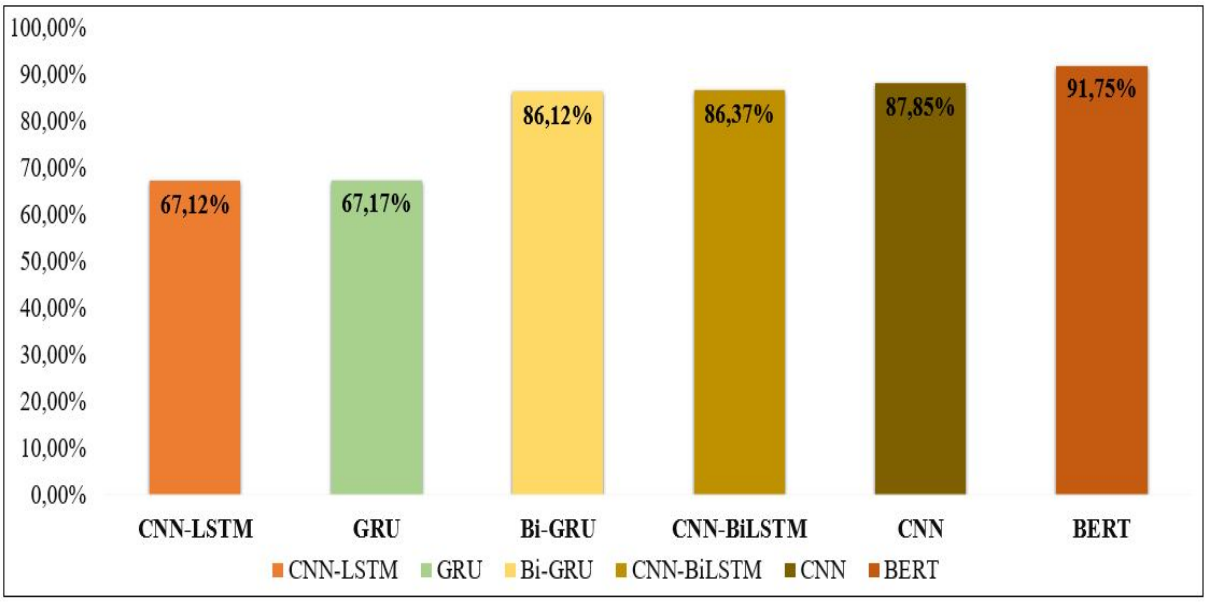


Fig. 6.6 Performance results of the joint-translated monolingual method on tweet dataset

Table 6.2 Performance results of joint-translated classification using the Arabic language

Models	Accuracy(%)	F-measure(%)
CNN	85	85
LSTM	67.7	81
GRU	85.28	86
BiLSTM	83.9	84
ArabBERT	90.7	93

6.5.5 Joint-Translated Monolingual Results

6.5.5.1 English

The second experiment concerns the joint-translated monolingual method in which we translated Arabic tweets to English using google translator. Then, we merged the two datasets and fed them to the BERT classifier. The findings are illustrated in Fig. 6.6. It can be noticed that BERT achieves the best accuracy results by almost 92% of accuracy. At the same time, CNN-LSTM and GRU models are stigmatically worse than the other models by 67% of accuracy. Moreover, CNN, CNN-LSTM, and BiGRU attain mostly the same accuracy ranging from 86% to 87%.
We further evaluate BERT against other versions like RoBERTa [83] and DistilBERT [113]. The obtained results are reported in Fig. 6.7. The obtained results indicate that BERT outperforms RoBERTa and distilBERT in terms of accuracy. Nevertheless, in terms of loss, RoBERTa performs the lowest value equals to 0.29.

6.5.5.2 Arabic

The third experiment is designed to perform AraBERT to detect offensive tweets using a translation method. In this experiment, we translate English tweets to Arabic. Then, we merge them with Arabic tweets. Table 6.2 summarizes the accuracy and F-measure results on the merged dataset. It can be observed that AraBERT beats all the deep neural networks and achieves 93% in terms of F-measure. Besides, it can be pointed out that the GRU model outperforms the other neural networks by a value of 86% for F-measure. CNN reported the same accuracy as GRU, which equals to 85%. Moreover, BiLSTM achieves 84% of accuracy and F-measure. Whereas the lowest performances are given by LSTM that affords 67% in terms of accuracy.
In total, we can notice clearly that the BERT models beat all the other deep learning models. Fig. 6.8 illustrates the comparison between the multilingualism approaches.

Based on the obtained results, we adopt the AraBERT with the join-translated method since this combination achieves the best F-measure by a value of 93%.

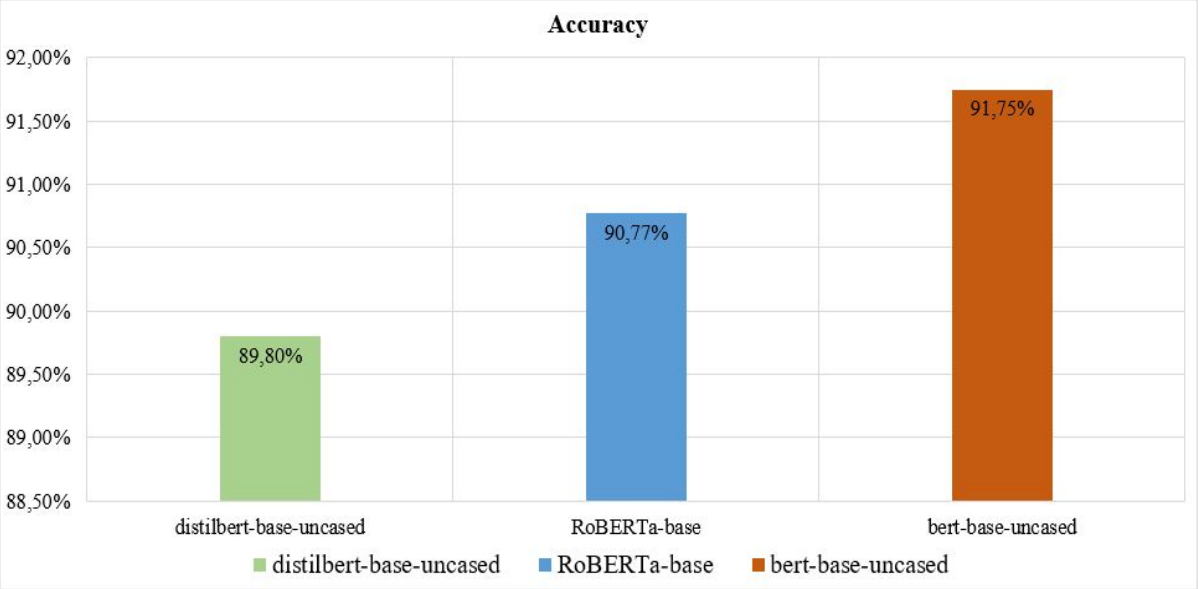


Fig. 6.7 Accuracy results of the joint-translated method using English language

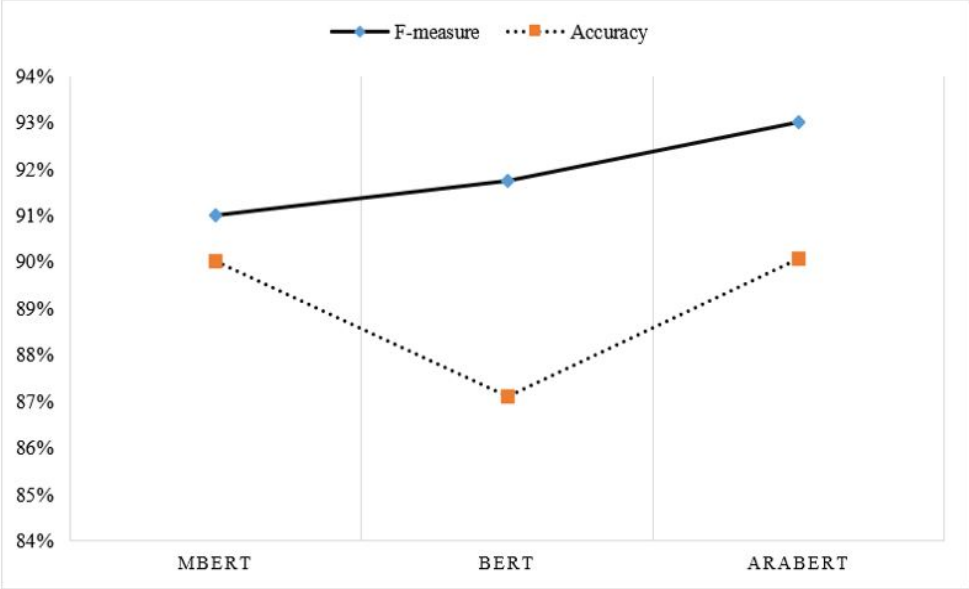


Fig. 6.8 Joint-translated monolingual results vs. Join-multilingual classification

6.6 Conclusion

In this chapter, we presented a transfer learning method for identifying the offensive language in a bilingual context where we simultaneously consider Arabic and English tweets. Our method is twofold. First, we tackle the multilingualism problem. Second, we cope with offensive language detection. Our method relies on BERT models, including BERT, mBERT, and AraBERT, that are fine-tuned on offensive detection task. Our system consists of several modules: (1) Preprocessing tweets by taking into consideration emojis substitution, (2) Tweet translation for joint translated classification, (3) Tokenizing tweets using BERT tokenizer according to the used multilingualism approach, (4) BERT fine-tuning to produce tweets representation and (5) tweets prediction applying a Sigmoid function on tweets representation. We carried out a set of experiments on a bilingual dataset from the SemEval competition. The experimental findings confirmed that the AraBERT model using the joint-translated classification method achieved over 93% of F-measure. Importantly, our results provide evidence for BERT robustness in the field of multilingual text classification.

Conclusion and Future Work

In this thesis, we dealed with the Arabic text categorization problem from a deep learning perspective to involve syntactic and semantic dependencies within texts. In fact, we were interested in tackling issues related to ATC, including the language ambiguity, lack of semantics, and disparity. First, the literature related to the ANLP, TC, and deep learning techniques was considered in Chapters 1 and 2. Then, we presented the main contributions in the ATC field that are structured in two important sections:

- Text representation: Improving the representation of documents' content by considering different deep learning architectures and different processes as Retrofitting and fine-tuning. The aim was to produce a textual document of high quality taking into account the complex morphology of Arabic to remedy the problem of ambiguity;
- Categorization: Choosing the most appropriate deep algorithms to categorize Arabic texts by considering the Arabic text's properties.

In our first contribution, we proposed the use of deep Autoencoder for Arabic text representation. To enhance Autoencoder performance, we used RBM training. The RBM Autoencoder representation is efficient since it reduces the feature space dimensionality and reflects text semantics. To further enhance the text representations, we incorporated the explicit semantic using Arabic WordNet and several feature selection techniques. In the second phase, we explored different deep neural architectures, including the MLP and the Autoencoder. We carried out an exhaustive set of experiments on the OSAc dataset. The obtained results have shown that using Autoencoder as text representation model and classifier outperformed state-of-the-art techniques and achieved the best results with Chi-Square by 94% of F-measure.
To handle the morphological variety of Arabic words, we proposed a deep neural network-based method that explores the word embeddings model using position weights and subword information. To further enhance the text semantically and guarantee the nearest vector representations for related words, our method adopts the Retrofitting strategy. We studied the efficiency of different neural networks for Arabic text categorization. Several experiments have been conducted on the OSAc dataset. The obtained results

approve the vigor of our method.
Contextual embedding models have invaded the NLP fields. Aiming to show their effectiveness for Arabic text categorization and solve the above issues, we proposed our third contribution. This latter concerns the usage of AraBERT for text categorization from two perspectives. First, we fine-tuned the AraBERT model on several corpora from OSAC dataset, and then we perform the categorization task. Secondly, we used AraBERT as a feature extractor and combined it with other deep neural architectures. The two variants give promising performances in the Arabic text categorization field.
To conclude our research, we proposed an offensive language detection application. This contribution is two folds, we dealt first with the offensive language in tweets, and secondly, we approached the multilingualism problem from a deep learning prospect. Our method relies on transfer learning. We explored different pre-trained models, including BERT, mBERT, and AraBERT, after fine-tuning their parameters. To tackle multilingualism, we studied two approaches, including joint-multilingual and translation-based classification methods. We performed a set of experiments on a bilingual dataset from the SemEval competition. The experimental findings confirmed that the AraBERT model using the joint-translated classification method accomplished over 93% of F-measure.
These works are promising. They can lead to new perspectives.

- Extending method capabilities to other domains as text clustering and sentiment analysis. This research task had been briefly unveiled by experiments presented in the fourth contribution. The obtained results suggest the robustness and flexibility of the method introduced.
- New deep learning algorithms appear continuously. Some of the current proposals may be promising within our domain. It would also be possible to replace the BERT with further transfer learning models such as XLNet and ULMFiT.
- It would be interesting to experiment transfer learning techniques in coss-lingual text classification field, that could lead to significant improvements. However, in our last contribution, we dealt only with multilingualism.
- A deeper analysis of ANLP may lead to improved ATC system performance.

Appendices

Appendix A

Résumé Détaillé de la Thèse en Français

Modèles d'apprentissage profond pour la catégorisation des documents textes en langue Arabe: Application à la détection du langage offensif

A.1 Introduction

Le développement du web et des supports de stockage ont permis d'archiver des énormes quantités de données textuelles ce qui a conduit à une explosion d'information. Par conséquent, la capacité d'extraire automatiquement et efficacement l'information pertinente à partir de ces données devient cruciale. Ces constats ont donné une naissance naturelle au domaine de la catégorisation de texte. La fouille de textes ou l'extraction de connaissances dans les textes est une spécialisation de la fouille de données et fait partie du domaine de l'intelligence artificielle. Les disciplines impliquées dans le processus de la fouille de textes sont donc la linguistique calculatoire, l'ingénierie des langues, l'apprentissage artificiel, les statistiques et l'informatique.
La fouille de textes [61] se compose généralement de plusieurs phases. La première phase consiste à préparer des données textuelles pour la phase d'apprentissage, cette phase inclus différent processus comme le prétraitement, la sélection de caractéristiques et la représentation. Le but est de représenter les données textuelles dans un format compatible avec le modèle d'apprentissage, et puis appliquez différents modèles qui améliorent l'efficacité de l'application de la fouille de textes. Le prétraitement du texte comprend différentes techniques telles que la tokenisation, la suppression de mots vides, la racinisation, la lemmatisation et l'étiquetage morphosyntaxique. La représentation de texte est une étape fondamentale dans la fouille de textes où les documents textuels sont représentés selon un modèle de représentation, par exemple, le modèle sac des mots et le modèle n-gramme. En outre, la pondération des termes est également une étape importante qui attribue des scores aux termes en utilisant différentes techniques telles que TF-IDF [141] et ses variantes ou d'autres méthodes plus récentes telles que les représentations distribuées et contextuelles des mots et des documents [87, 77, 103, 33],

FastText [19], etc. La sélection des caractéristiques est une autre technique importante qui tend à sélectionner les termes les plus utiles dans un texte. Les techniques de sélection des caractéristiques sont adoptées pour simplifier la construction d'un modèle et réduire les temps d'apprentissage. La deuxième phase dans le processus de la fouille de textes consiste à choisir une des tâches de la fouille de textes. Celles-ci se divisent en deux groupes : les tâches d'apprentissage supervisé et les tâches d'apprentissage non supervisé. Chaque groupe a un type d'algorithme différent. Enfin, la dernière phase est l'évaluation en vue de mesurer la performance de l'application de la fouille.
La catégorisation de textes est une tâche primitive de la fouille de textes qui consiste à chercher une liaison fonctionnelle entre un ensemble de textes et un ensemble de catégories (étiquettes, classes). Elle requiert les méthodes de traitement morphologiques et morphosyntaxiques des langages naturels, plusieurs outils informatiques, et les techniques de l'Intelligence Artificielle comme les algorithmes d'apprentissage automatique. De nombreux algorithmes ont été proposés et mis en œuvre pour répondre au problème de catégorisation de textes en langues latines (anglaise, française, etc.). Cependant, les études effectuées pour catégoriser les textes en langue arabe restent limitées. Dans ce contexte, nous distinguons deux types d'approches : l'approche classique qui se base sur des représentations classiques comme TF-IDF et n-gramme. Néanmoins, la catégorisation s'apparente au problème de l'extraction de la sémantique d'un texte puisque l'appartenance d'un document à une catégorie est étroitement liée à la signification de ce texte. Pour cela, plusieurs travaux ont été proposés, ils sont basés sur des méthodes d'analyse comme LSA, PLSA, et LDA [32, 60, 18], qui révèlent la sémantique d'une manière implicite. Cependant la taille très élevée du vocabulaire peut s'avérer un obstacle à l'utilisation de ces méthodes car la complexité de celles-ci augmente. Des méthodes plus récentes comme les représentations distribuées des mots et des textes ont été proposées et bien appliquées au domaine de la catégorisation de textes. Cependant, ces méthodes ignorent les informations intégrées dans les bases de données lexicales comme WordNet et ne prisent pas en compte les relations entre plusieurs mots et le contexte des phrases. D'une autre part, la richesse et la structure morphologique complexe de la langue arabe rendent la catégorisation une tâche extrêmement ardue.
Cette thèse s'inscrit dans le cadre des approches fondées sur l'apprentissage profond pour la catégorisation des documents textuels. Notre objectif principal est d'apporter des solutions pertinentes aux défis de la catégorisation de textes en langue arabe notamment la prise en compte de la sémantique, la réduction de la dimensionnalité de l'espace des caractéristiques, et l'ambiguïté et la complexité de la langue arabe. Pour aboutir à un tel objectif nous avons apporté quatre contributions dans le cadre de ce travail de thèse :

- Dans la première contribution, nous avons proposé une méthode de représentation

des textes qui se base sur l'Autoencodeur, celle-ci est exploitée dans le cadre d'un système de catégorisation. Nous cherchons à obtenir des textes représentés par des vecteurs de dimension réduite, une représentation vectorielle qui reflète la sémantique implicite de texte, un temps de calcul plus abordable et une bonne précision de classification. De plus, nous explorons les techniques de sélection de caractéristiques en vue de sélectionner l'information pertinente dans le texte. Pour pallier le problème d'ambigüité de la langue arabe, nous optons pour la représentation sac des concepts en utilisant l'ontologie AWN. Le but est d'intégrer la sémantique explicite pour enrichir les présentations des textes. Dans la phase de catégorisation, nous développons plusieurs classificateurs tels que l'Autoencodeur et le perceptron multi-couches.

- La deuxième contribution consiste à suggérer une méthode de catégorisation de textes en langue arabe. Celle-ci explore les réseaux neuronaux tels que les réseaux convolutifs et récurrents. En effet, ces réseaux ont montré des bonnes performances pour les tâches de classification des images et des textes. De plus pour booster les performances de ces réseaux, nous intégrons une technique qui s'appelle Retrofitting pour capturer davantage la sémantique du texte.
- La troisième contribution propose l'utilisation de l'apprentissage par transfert basé sur le modèle BERT pour la catégorisation de textes en langue arabe. L'idée sous-jacente consiste à représenter le texte arabe à l'aide de AraBERT, puis affiner les représentations obtenues sur la tâche de la catégorisation. Nous étudions aussi l'utilité de BERT en tant que un extracteur de caractéristiques en combinaison avec différents réseaux neuronaux.
- La quatrième contribution consiste à développer une application de détection du langage offensif dans un contexte multilingue (MOLD). L'idée principale consiste à représenter les tweets bilingues à l'aide des modèles BERT et par la suite affiner les représntations obtenues sur la tâche MOLD. De plus, pour gérer l'aspect multilingue, nous explorons différentes approches, telles que l'approche multilingue conjointe et celle basée sur la traduction.

A.2 Méthodes Proposées

Nous proposons dans ce travail de thèse un ensemble de méthodes pour remédier le probleme de catégorisation de textes en langue arabe. Dans ce qui suit, nous résumons et soulignons les principales contributions et les résultats de ce travail de thèse.

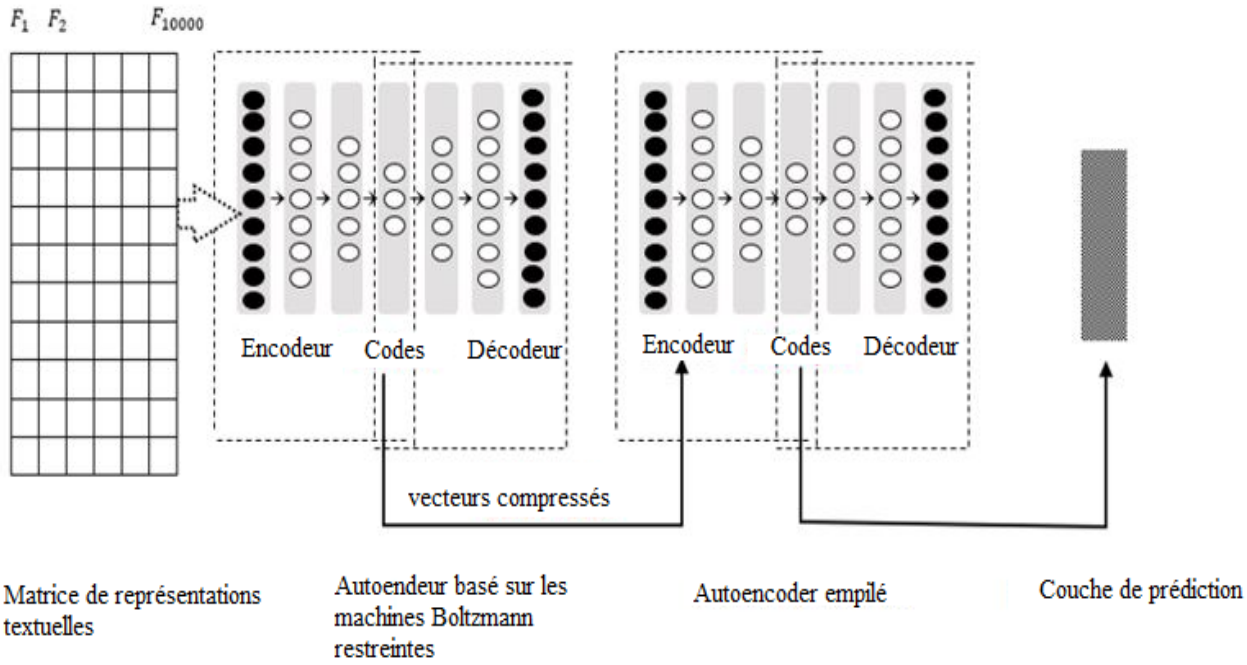


Fig. A.1 L'architecture du système de catégorisation de textes en utilisant l'Autoencodeur

A.2.1 Autoencodeur Profond pour la Catégorisation de Textes en Langue Arabe

Dans le chapitre 3, nous avons présenté une méthode de catégorisation de textes en langue arabe à base de l'apprentissage profond. Nous avons proposé une représentation textuelle basée sur l'Autoencodeur. Notre méthode se compose de trois étapes : (1) L'extraction des concepts les plus pertinents en utilisant AWN et en se basant sur le processus de sélection des caractéristiques (2) L'apprentissage des caractéristiques via un algorithme non supervisé pour la représentation de textes (3) La catégorisation de textes à l'aide des réseaux neuronaux comme l'Autoencodeur, et le perceptron multi-couches. Notre méthode permet de considérer la sémantique du document en combinant à la fois la sémantique implicite et explicite, et de réduire la dimensionnalité de l'espace des caractéristiques. La figure A.1 illustre l'architecture globale de la méthode proposée.

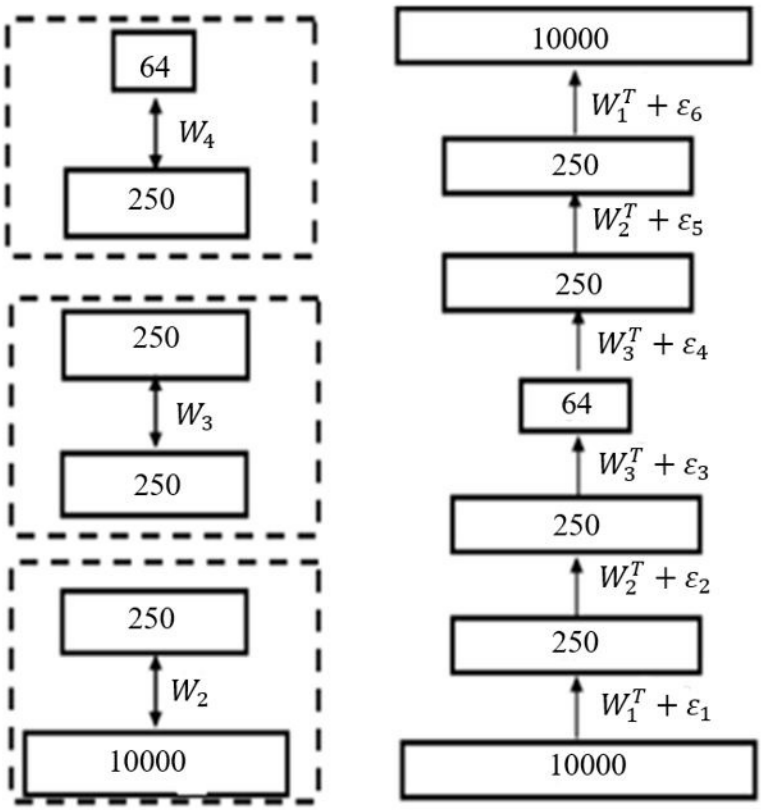


Fig. A.2 Panneau de gauche: entraînement par une pile de machines Boltzmann restreintes. Panneau de droite: Ajustement

Nous commençons par la phase du prétraitement et la sélection des caractéristiques. Celle-ci comprend trois étapes : la tokenisation, la suppression des mots vides, des signes de ponctuation, des nombres et des mots écrits dans différentes langues et la normalisation des mots en remplaçant les lettres (”أ إ آ”) par (”ا”), les lettres (”ءؤ”) par (”ا”) et (”ا”) par (”ى”).

Après le prétraitement du texte, nous transformons les sacs de mots en sacs de concepts en utilisant AWN pour intégrer la sémantique et enrichir les représentations textuelles. Nous ajoutons seulement des concepts des fréquences importantes dans le texte. Ensuite, nous appliquons des techniques de sélection de caractéristiques tels que le chi carré et le seuil de variance afin de sélectionner les meilleurs mots en éliminant les termes non discriminants. Le chi carré mesure le degré d'indépendance entre un terme et une catégorie. Nous avons fixé le nombre de caractéristiques à 10000. Le seuil de variance est une technique standard pour sélectionner des entités. Elle expulse toutes les caractéristiques dont la variance n'atteint pas un certain seuil. Elle rejette chacune des entités à variance nulle, par exemple les entités qui ont une valeur similaire dans tous les textes. Concernant nos travaux, nous avons fixé le seuil à 0.001.

Table A.1 Résultats de catégorisation de textes en langue arabe en utilisant différentes représentations textuelles

Méthodes	F-measure	Précision	Rappel
La méthode proposée	0.93	0.94	0.91
sac des mots	0.88	0.91	0.88
Doc2vec	0.89	0.91	0.88
L'analyse sémantique latente	0.89	0.91	0.89

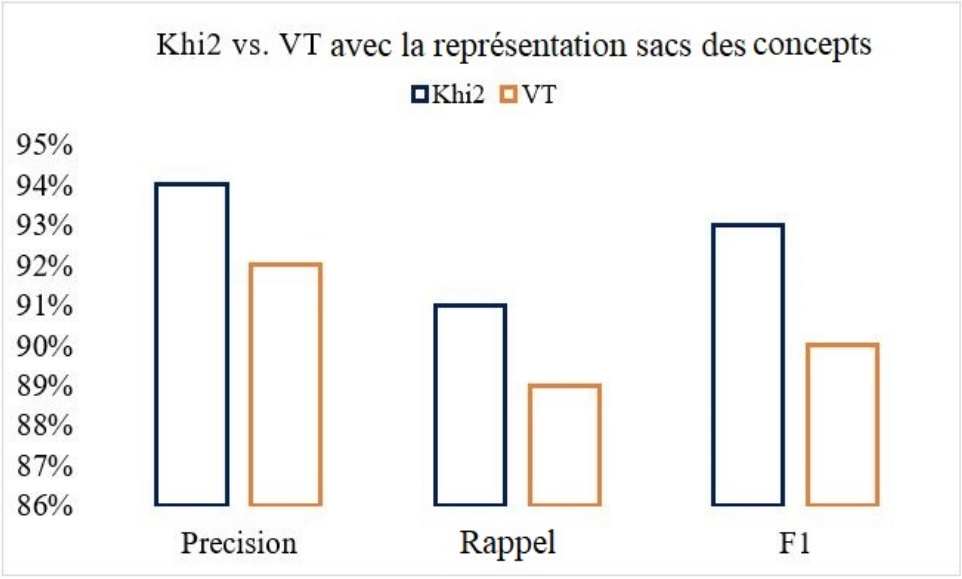
Après avoir sélectionné les meilleures fonctionnalités à l'aide du chi carré ou du seuil de variance, les vecteurs obtenus sont transmis au module d'Autoencodeur. Pour faire l'apprentissage de l'Autoencodeur profond, nous avons utilisé les machines de Boltzmann restreintes. Au cours de cette étape, une pile de machines de Boltzmann restreintes est générée. Ensuite, chaque Machine de Boltzmann restreinte est dépliée pour avoir une architecture symétrique. Puis, nous avons affiné les poids à l'aide de la fonction d'erreur carrée pour réduire l'erreur de reconstruction entre les données d'entrée et de sortie. Nous avons construit un Autoencodeur profond avec 7 couches. Nous avons évalué différentes architectures et adopté l'architecture 10000-250-250-64-250-250-10000 (voir la figure A.2) en se basant sur les bons résultats obtenus. Après avoir gelé la couche décodeur, nous avons obtenu des vecteurs de faible dimension (64 caractéristiques).

Après avoir produit des représentations textuelles en se basant sur l'Autoencodeur profond, celles-ci sont transmises à un module de catégorisation. Nous avons exploré différents réseaux neuronaux comme l'Autoencodeur et le perceptron multi-couches. Nous avons adopté l'Autoencodeur profond en raison de sa capacité à reconstruire les entrées sans avoir besoin d'étiquettes et son efficacité dans les applications de classification. Nous avons entraîné notre Autoencodeur en utilisant une pile de couches. Après son apprentissage, nous avons supprimé la couche de décodeur et connecté une couche softmax à la couche des caractéristiques de l'encodeur pour attribuer les classe appropriées aux représentations vectorielles. Après plusieurs expérimentations, nous avons utilisé un empilement de couches (trois couches pour l'encodeur et trois couches pour le décodeur : 64-700-64-64-64-700-64). Nous avons appliqué la fonction d'activation *Relu* et l'optimiseur *Adam*. Nous avons aussi construit un perceptron mutli-couches contenant quatre couches avec 1000 nœuds pour chaque couche.

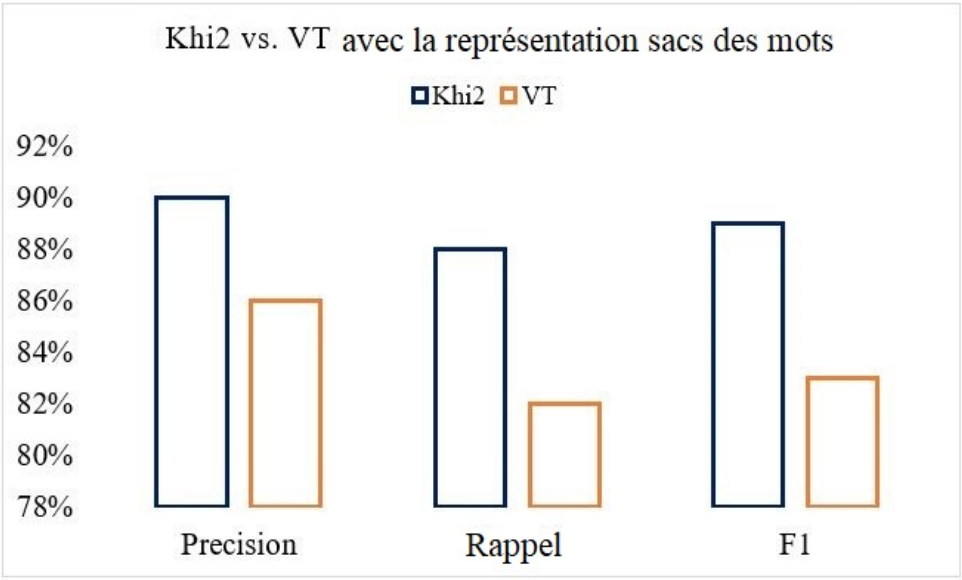
Le tableau A.1 présente les principaux résultats obtenus sur le corpus standard de catégorisation de textes OSAc.

Les résultats obtenus montrent que la méthode proposée améliore significativement les systèmes de catégorisation de l'état de l'art. Notre modèle en se basant sur la représentation du sac de concepts et le chi carré a surpassé toutes les méthodes de l'état

de l'art et atteint respectivement 93 % et 94 % en termes de F-mesure et de précision. De plus, une réduction significative de la dimensionnalité de l'espace des caractéristiques a été dérivée (64 vecteurs de longueur fixe) par rapport aux autres méthodes.



(a)



(b)

Fig. A.3 Performances de khi2 vs. VT. (a) Utilisation de la représentation des sacs de concepts. (b) Utilisation de la représentation des sacs de mots

Table A.2 Comparaison entre les classificateurs Autoencodeur, Perceptron multi-couches et SVM en utilisant les sacs des concepts et chi-carré

Méthodes	**Autoencodeur**	**Perceptron multi-couches**	**SVM**
Précision	0.94	0.93	0.93
Rappel	0.91	0.90	0.91
F-measure	0.93	0.91	0.92

Pour étudier l'impact de sélection des caractéristiques sur l'efficacité de notre système, nous avons testé plusieurs techniques. La figure A.3 présente les différents résultats. Nous observons que le chi carré (Khi2) était plus efficace que le seuil de variance (VT) pour les représentations à la fois du sac de mots et du sac de concepts. Le chi carré en conjonction avec les sacs des concepts a réalisé respectivement 94 %, 91 % et 93 % pour la précision, le rappel et la F-mesure.
Pour évaluer l'impact du choix du classificateur, nous avons comparé différents classificateurs tels que l'Autoencodeur, le perceptron multi-couches et SVM. Les résultats sont illustrés dans le tableau A.2. Nous observons que le l'Autoencodeur a surpassé légèrement les autres classificateurs et réalisé les meilleures performances par 93 % de F-measure.

A.2.2 Modèles Neuronaux Profonds et rétrofit pour la Catégorisation de Texte en Langue Arabe

Dans chapitre 4, nous avons proposé une méthode de catégorisation de textes en langue arabe en utilisant des modèles neuronaux profonds et la technique du rétrofit. Notre système se compose de plusieurs étapes : (1) Le prétraitement du texte ; (2) L'incorporation des représentations de mots basé sur des informations de sous-mots ; (3) Le rétrofit du modèle à l'aide de la ressource lexicale arabe WordNet; (4) La combinaison des modèles CNN, LSTM pour la catégorisation.
La première étape consiste en un processus de prétraitement au cours duquel nous avons supprimé les signes de ponctuation, les nombres, les mots écrits dans différentes langues et les mots vides. Chaque texte est ensuite divisé en un ensemble de mots. Après cette étape, nous avons construit une matrice de taille $x \times r$ où r est la longueur des vecteurs de mots et x est la longueur maximale des textes. Soit maxlen la longueur maximale des textes. Chaque texte contient un nombre de termes inférieur à maxlen sera complété pour avoir la même longueur que la longueur maximale. Cependant, pour les textes d'une longueur supérieure à maxlen, nous coupons simplement les mots supplémentaires. Ensuite, les représentations textuelles sont envoyées à l'un des modèles (LSTM, CNN, LSTM-CNN ou CNN-LSTM).

Table A.3 Comparaisons de modèles proposés avec des modèles de l'état de l'art sur OSAc

Modèles	F-measure
TFIDF_SVM	0.89
TFIDF+Khi2_SVM	0.96
LDA_SVM	0.96
Skip-gram_SVM	0.96
Glove_SVM	0.95
Doc2vec_SVM	0.94
CNN-32	0.98
CNN-64	0.98
CNN-128	0.99
LSTM	0.93
CNN-LSTM	0.99
LSTM-CNN	0.92

Tous les modèles ont été construits sur un modèle distribué et réajusté en utilisant la technique du rétrofit et en considérant les informations des sous-mots. Pour valider notre méthode, nous avons mené plusieurs expérimentations sur le corpus OSAc. Le tableau A.3 montre les résultats obtenus. Nous observons que le modèle CNN donne les meilleures performances dans le cas d'un nombre de filtres égal à 128. De plus, CNN-128 et CNN-LSTM obtiennent les meilleurs résultats pour la catégorisation de textes.

Pour montrer l'effet du Retrofitting sur la catégorisation, nous avons implémenté nos modèles sans et avec Retrofitting, les résultats obtenus sont présentés dans le tableau A.4. Nous observons que l'utilisation de Retrofitting améliore les résultats de catégorisation pour tous les modèles.

Table A.4 Résultats de la catégorisation pour différents modèles proposés sans et avec Retrofitting

	Sans Retrofitting		**Avec Retrofitting**	
Modèles	**Entraînement**	**Test**	**Entraînement**	**Test**
CNN-128	0.989	0.9743	0.9995	0.9868
CNN-64	0.9994	0.9816	0.9996	0.987
CNN-32	0.9975	0.9832	0.9992	0.9822
LSTM	0.9229	0.9196	0.9304	0.9263
CNN-LSTM	0.9977	0.9898	0.9988	0.9923
LSTM-CNN	0.8055	0.789	0.9256	0.919

A.2.3 Embeddings Sémantiques et Contextuels basés sur le Modèle AraBERT Affiné pour la Catégorisation Multi-classes de Textes Arabe

Dans chapitre 5, nous avons présenté notre troisième contribution qui concerne l'utilisation du modèle AraBERT pour la catégorisation de textes en langue arabe. Nous avons étudié deux variantes. D'une part, nous avons exploité le modèle AraBERT après avoir affiné ses paramètres sur l'ensemble de données OSAC pour augmenter l'efficacité de la catégorisation. D'autre part, nous avons utilisé AraBERT, en tant que modèle d'extraction de caractéristiques, en le combinant avec plusieurs classificateurs dont CNN, LSTM, Bi-LSTM, MLP et SVM. L'architecture globale de notre système se compose de plusieurs étapes principales, notamment le prétraitement du texte, l'ajustement du modèle AraBERT et le calcul des caractéristiques de texte via le même modèle, et enfin la prédiction de la catégorie.

Nous avons mené plusieurs expérimentations afin d'évaluer notre méthode de catégorisation de textes arabe. Nous avons utilisé le corpus OSAC comprenant trois ensembles de données avec un nombre variable de textes. Pour toutes les expérimentations, 80 % des données sont destinées à l'apprentissage et 20 % aux tests. Nous avons employé le modèle AraBERT [12] en explorant deux versions de ce modèle : AraBERTv0.1 et AraBERTv1. De plus, nous avons évalué l'AraBERT par rapport au BERT multilingue [33].

La figure. A.4 présente les résultats obtenus des modèles AraBERT affinés en termes de précision. En fait, l'AraBERT affiné permet une amélioration de près de 23 % pour CNN. Le tableau A.5 montre les résultats de précision des modèles AraBERT affinés et les modèles de l'état de l'art. Nous remarquons que les deux modèles AraBERT affinés ont battu toutes les modèles de l'état de l'art.

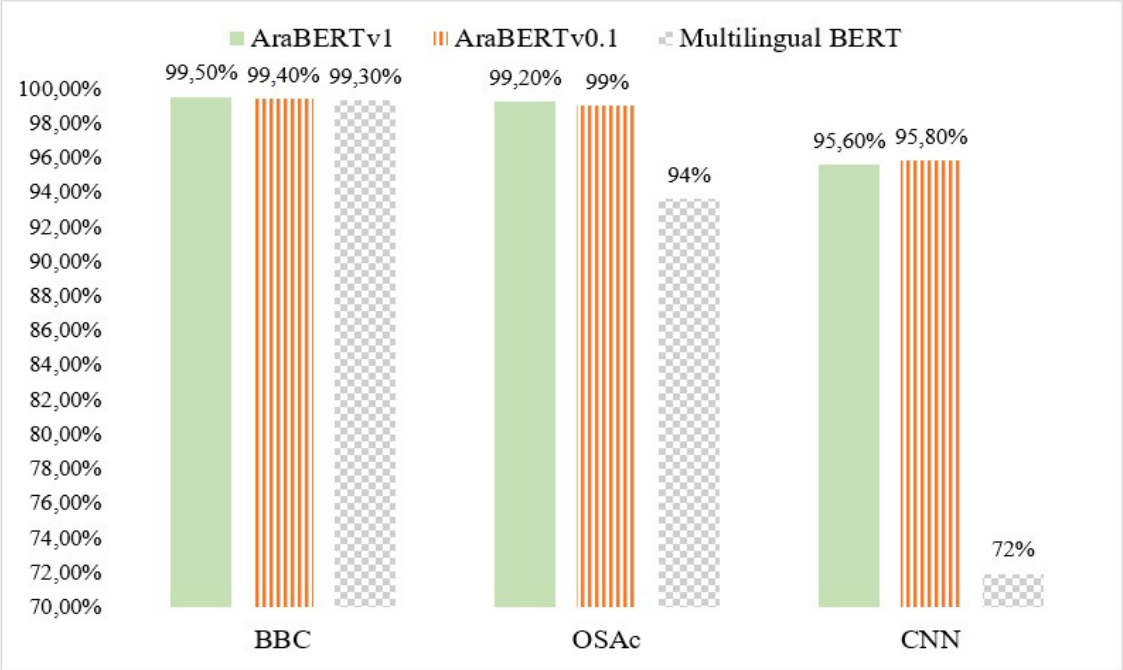


Fig. A.4 Performances de trois versions de modèles BERT affinés sur l'ensemble de données OSAC

Table A.5 Performances des modèles AraBERT en comparaison avec l'état de l'art

Modèles	Accuracy(%)
BiLSTM	93
LSTM	91
Doc2vec_MLP	90
LSA_SVM	90
Doc2vec_SVM	89
TFIDF_SVM	0.88
Word2vec_SVM	72
AraBERTv0.1 Affiné	99
AraBERTv1 Affiné	99.2

Table A.6 Performances des modèles AraBERT par rapport à mBERT en tant que modèles d'extraction de caractéristiques en utilisant différents classificateurs sur OSAc

Classificateurs	mBERT	AraBERTv1	AraBERTv0.1
CNN	93	96	96
LSTM	93	96	96
BiLSTM	94	96	96
MLP	93	95	96
CNN	95	96	97

Enfin, nous avons évalué AraBERT en tant qu'extracteur de caractéristiques pour montrer l'impact de l'abstraction de l'approche d'ajustement sur les performances de la catégorisation. Nous avons intégré le modèle précédent avec plusieurs classificateurs: CNN, LSTM, BiLSTM, MLP et SVM. Le tableau A.6 résume les performances en termes de F1 des modèles AraBERT et mBERT en tant qu'extracteurs de caractéristiques sur l'ensemble de données OSAc. Nous observons que le modèle AraBERTv0.1 fusionné avec le classificateur SVM a donné les meilleurs résultats.

A.2.4 Une Méthode de Détection du Langage Offensif dans un Contexte Multilingue en se Basant sur l'Apprentissage par Transfert

Dans chapitre 6, nous avons présenté notre quatrième contribution qui concerne la détection du langage offensant par l'apprentissage par transfert. Dans cette contribution, nous avons travaillé dans un contexte bilingue en considérant la classification des tweets en arabe et anglais. Comme c'est illustré dans la figure A.5, notre méthode se compose de plusieurs phases, y compris le prétraitement du tweet, la traduction, la tokenisation avec BERT, la représentation du tweet et la classification.

La première étape de notre modèle consiste à prétraiter les tweets afin de ne conserver que les informations pertinentes. Nous avons commencé par supprimer tous les hashtags, balises HTML, mentions et URL. Pour le texte anglais, nous avons remplacé les contractions par leurs formes complètes, nous avons corrigé les mots mal orthographiés. Nous avons remplacé les émojis s'ils existaient par le texte qu'ils représentent car les émojis ou les émoticônes jouent un rôle essentiel dans la représentation d'un tweet.
Dans l'étape de traduction, nous avons généré deux corpus comme suit :

- Pour générer le premier corpus, nous traduisons les tweets anglais en arabe à l'aide de l'API Google Traduction. Ensuite, nous fusionnons tous les tweets en un seul document. Puis, nous appliquons l'analyseur lexicale AraBERT.

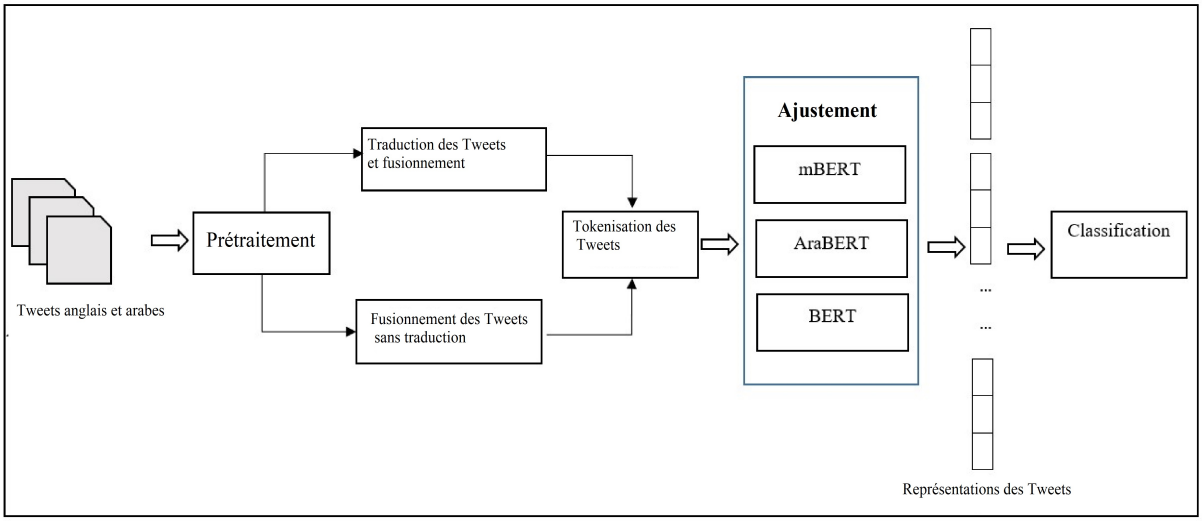


Fig. A.5 Organigramme de la méthode de la détection du langage offensif dans un contexte multilingue

- Pour produire le deuxième corpus, nous traduisons les tweets arabes en anglais avec la même API. Ensuite, nous combinons tous les tweets dans un seul fichier. Puis, nous transmettons tous les tweets à l'analyseur lexicale BERT.

Ensuite, nous appliquons la tokenisation des tweets en utilisant le processus de tokenisation du modèle BERT. Après, ces vecteurs de représentation sont introduits dans BERT. Par la suite les paramètres de BERT sont affinés sur la tâche de l'identification du langage offensant. Après l'étape d'ajustement et afin de détecter l'étiquette du tweet, nous avons transmis les représentations de tweets à une fonction sigmoïde et entraîné le modèle pour optimiser la fonction d'entropie croisée binaire.

A.2.4.1 Résultats de la classification multilingue conjointe

La table A.7 montre les résultats de l'approche de classification multilingue conjointe. Nous remarquons que celle-ci surpasse la méthode monolingue.

A.2.4.2 Résultats de performance de la méthode de traduction sur l'ensemble des tweets

La deuxième expérimentation concerne la méthode de traduction monolingue dans laquelle nous avons traduit les tweets arabes en anglais. Ensuite, nous avons fusionnié les deux ensembles de données et les transmis au classificateur BERT. Les résultats sont illustrés dans la figure A.6. Nous remarquons que BERT obtient les meilleurs résultats de précision avec 92% de précision.

Table A.7 Résultats de performance de la méthode de classification conjointe multilingue

Modèles	Accuracy(%)	F-measure(%)
CNN	89.74	90
LSTM	66.43	80
GRU	90.06	90
BiLSTM	90.42	89
CNN-BiLSTM	88.36	89
AraBERT Monolingue	92.65	79
CNN_ AraBERT Monolingue	87.69	89
GRU_ AraBERT Monolingue	79	89
mBERT	91.36	91

La troisième expérimentation est conçue pour exécuter AraBERT enfin de détecter les tweets offensants. Dans celle-ci, nous avons traduit les tweets anglais en arabe. Ensuite, nous avons les fusionné avec des tweets arabes. Le tableau A.8 résume les résultats de précision et de F-mesure sur l'ensemble de données. AraBERT bat tous les réseaux de neurones profonds et atteint 93 % en termes de F-mesure.
Au total, nous pouvons clairement remarquer que les modèles BERT battent tous les autres modèles d'apprentissage profond.

Table A.8 Résultats de performance de la catégorisation en se basant sur la méthode de la traduction conjointe en langue arabe

Modèles	Accuracy(%)	F-measure(%)
CNN	85	85
LSTM	67.7	81
GRU	85.28	86
BiLSTM	83.9	84
ArabBERT	90.7	93

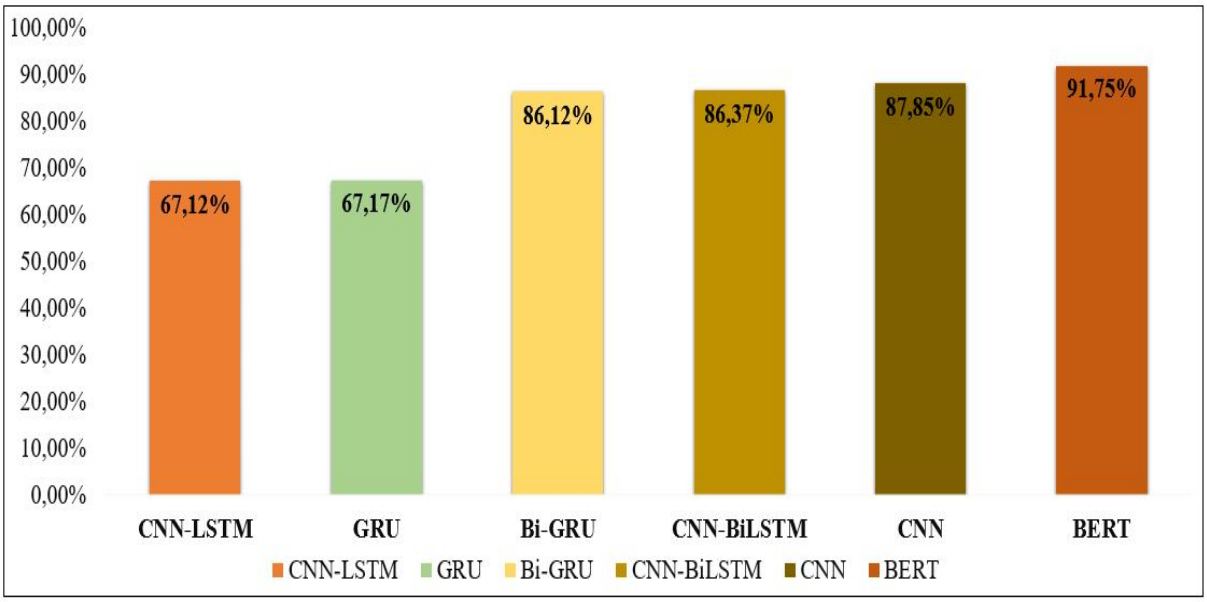


Fig. A.6 Résultats de performance pour la classification traduite conjointe en anglais

A.3 Conclusion et Perspectives

Dans ce travail de thèse, nous nous sommes penchés sur l'étude et l'amélioration des méthodes permettant la catégorisation automatique de textes en langue arabe. Les améliorations que nous avons apportées concernent la représentation textuelle et la catégorisation. En effet, nous avons proposé un ensemble de méthodes qui se base sur les techniques d'apprentissage en profondeur.
Tout d'abord, la littérature relative à l'ANLP, au TC et aux techniques d'apprentissage profond a été décrite dans les chapitres 1 et 2. Ensuite, nous avons présenté les principales contributions dans le domaine ATC qui sont structurées en deux sections importantes :

- La représentation textuelle : L'amélioration de la représentation du contenu des documents en considérant différentes architectures d'apprentissage en profondeur et différentes techniques comme le Retrofit et l'ajustement. L'objectif était de produire une représentation textuelle de haute qualité tenant compte la morphologie complexe de l'arabe pour remédier au problème de l'ambiguïté;
- La catégorisation : Choisir les algorithmes d'apprentissage profonds qui sont plus appropriés pour catégoriser les textes en langue arabe.

Dans notre première contribution, nous avons proposé l'utilisation de l'Autoencodeur profond pour la représentation de textes en langue arabe. Pour améliorer les performances de l'Autoencodeur, nous avons utilisé le pré-entrainement en se basant sur les machines de Boltzmann restreintes. Pour améliorer encore les représentations textuelles,

nous avons incorporé la sémantique explicite à l'aide d'AWN et de plusieurs techniques de sélection de caractéristiques. Dans la deuxième phase, nous avons exploré différentes architectures neuronales profondes, notamment MLP et l'Autoencodeur. Nous avons réalisé un ensemble exhaustif d'expérimentations sur l'ensemble de données OSAC. Les résultats obtenus ont montré que l'utilisation d'Autoencodeur comme modèle de représentation de texte et classificateur surpassait les techniques de l'état de l'art et obtenait les meilleurs résultats.

Dans notre dexième contribution, nous avons proposé une méthode basée sur les réseaux neuronaux profonds pour la catégorisation de textes en langue arabe. Celle-ci explore le modèle de représentation de mots en utilisant les informations de sous-mots pour gérer la variété morphologique des mots arabes. Pour améliorer encore le texte sémantiquement et garantir les représentations vectorielles les plus proches pour les mots associés, notre méthode adopte la stratégie du Rétrofit. Nous avons étudié l'efficacité de différents réseaux de neurones pour la catégorisation de texte arabe. Plusieurs expérimentations ont été menées sur l'ensemble de données OSAC. Les résultats obtenus approuvent le potentiel de notre méthode.

Les modèles d'apprentissage par transfert ont envahi le domaine de NLP. Dans le but de montrer leur efficacité pour la catégorisation de textes en langue arabe, nous avons proposé notre troisième contribution. Cette dernière concerne l'utilisation d'AraBERT pour la catégorisation de textes; Deux variantes ont été proposées. La première variante consiste à affiner le modèle AraBERT sur plusieurs corpus, puis effectuer la tâche de catégorisation. Pour la deuxième variante, nous avons utilisé AraBERT en tant qu'un extracteur de caractéristiques en le combinant avec d'autres architectures neuronales profondes. Les deux variantes donnent des performances prometteuses dans le domaine de la catégorisation de textes en langue arabe.

Pour conclure nos recherches, nous avons proposé une application de détection du langage offensif. Cette contribution a traité deux problématiques. D'une part, nous avons abordé le langage offensant dans les tweets, et d'autre part, nous avons abordé l'aspect multilingue dans une perspective d'apprentissage profond. Notre méthode repose sur l'apprentissage par transfert. Nous avons exploré différents modèles, notamment BERT, mBERT et AraBERT, après avoir affiné leurs paramètres. Pour s'attaquer à l'aspect multilingue, nous avons étudié deux méthodes, dont la méthode de classification multilingue conjointe et la méthode basée sur la traduction. Nous avons réalisé une série d'expérimentations sur un ensemble de données bilingues. Les résultats expérimentaux ont confirmé que le modèle AraBERT en conjonction avec la méthode de traduction accomplissait plus de 93 % de F-mesure.

Les contributions apportées dans ce travail de thèse permettent d'améliorer la catégorisation de textes en langue arabe. De plus, elles ouvrent plusieurs directions pour des futurs

travaux de recherche. Ces perspectives concernent différents volets :

- L'extension des méthodes proposées aux autres domaines comme la classification non supervisée de textes et l'analyse des sentiments. L'analyse des sentiments est une tâche de recherche qui a été brièvement dévoilée par les expérimentations présentées dans la quatrième contribution. Les résultats obtenus suggèrent la robustesse et la flexibilité de la méthode introduite.
- De nouveaux algorithmes de l'apprentissage profond continuent à apparaitre. Certaines des propositions actuelles peuvent être prometteuses dans notre domaine. Il serait également possible de remplacer le BERT par d'autres modèles tels que XLNet et ULMFiT.
- Il serait intéressant d'expérimenter des techniques d'apprentissage par transfert pour la classification cos-lingual, cela pourrait conduire à des améliorations significatives. Cependant, dans notre dernière contribution, nous n'avons traité que l'aspect multilingue.
- Une analyse plus approfondie de l'ANLP peut conduire à une amélioration des performances du système ATC.

LIST OF PUBLICATIONS

The research works carried out for the completion of this thesis work have resulted in several publications in international conferences, book chapter, and specialized journals. We present these publications next, sorting them by publication type.

International journals

1. Fatima-Zahra EL-ALAMI, Abdelkader EL MAHDAOUY, Said OUATIK EL ALAOUI, and Noureddine EN NAHNAHI. A Deep Autoencoder-Based Representation for Arabic Text Categorization, Journal of Information and Communication Technology (JICT), vol. 19, No. 3, pp. 381-398, 2020.
2. Fatima-Zahra EL-ALAMI, Said OUATIK EL ALAOUI, and Noureddine EN NAHNAHI. Deep Neural Models and Retrofitting for Arabic Text Categorization. International Journal of Intelligent Information Technologies (IJIIT), vol. 16, No. 2, pp. 74-86, 2020.
3. Fatima-Zahra EL-ALAMI, Said OUATIK EL ALAOUI, and Noureddine EN NAHNAHI. Contextual Semantic Embeddings based on Fine-tuned AraBERT Model for Arabic Text Multi-class Categorization, Journal of King Saud University - Computer and Information Sciences, 2021.

International conferences

1. Fatima-Zahra EL-ALAMI and Said OUATIK EL ALAOUI. An Efficient Method based on Deep Learning Approach for Arabic Text Categorization, International Arab Conference on Information Technologie: ACIT, Beni Mellal, Morocco, November 2016.
2. Fatima-Zahra EL-ALAMI and Said OUATIK EL ALAOUI. An ontology-based method for word sense representation and Arabic text categorization, International Symposium on Signal, Image, Video: ISIVC, Rabat, Morocco, November 2018.
3. Fatima-Zahra EL-ALAMI and Said OUATIK EL ALAOUI. Combining and Merging Deep Neural Networks for Arabic Text Categorization, International Conference on Advanced Intelligent Systems for Sustainable Development AI2SD'2020, Tangier, Morocco, December 2020.

Book chapter

1. Fatima-Zahra EL-ALAMI and Said OUATIK EL ALAOUI. Combining and Merging Deep Neural Networks for Arabic Text Categorization, As a chapter in Advanced Intelligent Systems for Sustainable Development Applied to Education and Intelligent Learning System, AI2SD'2020, 2021.

REFERENCES

[1] A. Abdelali, K. Darwish, N. Durrani, and H. Mubarak. Farasa: A fast and furious segmenter for arabic. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Demonstrations*, pages 11–16, 2016.

[2] A. Abu-Errub. Arabic text classification algorithm using tfidf and chi square measurements. *International Journal of Computer Applications*, 93(6), 2014.

[3] M. A. M. E. Ahmed. *A large-scale computational processor of the Arabic morphology, and applications*. PhD thesis, 2000.

[4] F. S. Al-Anzi and D. AbuZeina. Toward an enhanced arabic text classification using cosine similarity and latent semantic indexing. *Journal of King Saud University-Computer and Information Sciences*, 29(2):189–195, 2017.

[5] S. Al-Fedaghi and F. Al-Anzi. A new algorithm to generate arabic root-pattern forms. In *proceedings of the 11th national Computer Conference and Exhibition*, pages 391–400. King Fahd University of Petroleum and Minerals Dhahran, Saudi Arabia, 1989.

[6] B. Al-Salemi and M. J. A. Aziz. Statistical bayesian learning for automatic arabic text categorization. *Journal of computer Science*, 7(1):39, 2011.

[7] R. Al-Shalabi and M. Evens. A computational morphology system for arabic. In *Computational Approaches to Semitic Languages*, 1998.

[8] E.-n. Alami, Hamza, B. Noureddine, E. A. Abdessamad, and S. Ouatik. Overcoming arabert's pretrain-finetune discrepancy for arabic offensive language identification(offenseval 2020). 2020.

[9] A. M. Alayba, V. Palade, M. England, and R. Iqbal. A combined cnn and lstm model for arabic sentiment analysis. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 179–191. Springer, 2018.

[10] H. A. Almuzaini and A. M. Azmi. Impact of stemming and word embedding on

deep learning-based arabic text categorization. *IEEE Access*, 8:127913–127928, 2020.

[11] A. Alosaimy and E. Atwell. Tagging classical arabic text using available morphological analysers and part of speech taggers. *Journal for Language Technology and Computational Linguistics*, 32(1/2017):1–26, 2017.

[12] W. Antoun, F. Baly, and H. Hajj. Arabert: Transformer-based model for arabic language understanding. *arXiv preprint arXiv:2003.00104*, 2020.

[13] M. Attia. Arabic tokenization system. In *Proceedings of the 2007 workshop on computational approaches to semitic languages: Common issues and resources*, pages 65–72, 2007.

[14] M. Attia. *Ambiguity in Arabic computational morphology and syntax: A study within the lexical functional grammar framework*. LAP LAMBERT Academic Publishing, 2012.

[15] K. Beesley. Finite-state morphological analysis of arabic at xerox research: Status and plans in 2001. In *the proceedings of the workshop on Arabic natural language processing*, pages 1–8.

[16] N. Bel, C. H. Koster, and M. Villegas. Cross-lingual text categorization. In *International Conference on Theory and Practice of Digital Libraries*, pages 126–139. Springer, 2003.

[17] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[18] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[19] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[20] M. M. Boudabous, N. C. Kammoun, N. Khedher, L. H. Belguith, and F. Sadat. Arabic wordnet semantic relations enrichment through morpho-lexical patterns. In *2013 1st International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, pages 1–6. IEEE, 2013.

[21] M. Boudchiche, A. Mazroui, M. O. A. O. Bebah, A. Lakhouaja, and A. Boudlal. Alkhalil morpho sys 2: A robust arabic morpho-syntactic analyzer. *Journal*

of King Saud University-Computer and Information Sciences, 29(2):141–146, 2017.

[22] K. Bouzoubaa, H. Baidouri, T. Loukili, and T. El Yazidi. Arabic stop words: Towards a generalisation and standardisation. In *the 13th International Business Information Management Association Conference IBIMA*, 2009.

[23] L. Cai and T. Hofmann. Text categorization by boosting automatically extracted concepts. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 182–189, 2003.

[24] M. A. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning. In *Aistats*, volume 10, pages 33–40. Citeseer, 2005.

[25] W. Che, Y. Liu, Y. Wang, B. Zheng, and T. Liu. Towards better ud parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. *arXiv preprint arXiv:1807.03121*, 2018.

[26] A. Chen and F. Gey. Building an arabic stemmer for information retrieval. In *TREC*, volume 2002, pages 631–639, 2002.

[27] G. Chen, D. Ye, Z. Xing, J. Chen, and E. Cambria. Ensemble application of convolutional and recurrent neural networks for multi-label text categorization. In *2017 international joint conference on neural networks (IJCNN)*, pages 2377–2383. IEEE, 2017.

[28] J. Cheng, L. Dong, and M. Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016.

[29] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*, 2016.

[30] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[31] T. Davidson, D. Warmsley, M. Macy, and I. Weber. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*, 2017.

[32] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[33] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of

deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[34] M. Diab. Second generation amira tools for arabic processing: Fast and robust tokenization, pos tagging, and base phrase chunking. In *2nd International Conference on Arabic Language Resources and Tools*, volume 110, page 198, 2009.

[35] F.-Z. El-Alami and S. O. El Alaoui. An efficient method based on deep learning approach for arabic text categorization. In *International Arab Conference on Information Technology*, 2016.

[36] F.-Z. El-Alami and S. O. El Alaoui. Word sense representation based-method for arabic text categorization. In *2018 9th International Symposium on Signal, Image, Video and Communications (ISIVC)*, pages 141–146. IEEE, 2018.

[37] F.-Z. El-Alami and S. O. El Alaoui. Combining and merging deep neural networks for arabic text categorization. In *International Conference on Advanced Intelligent Systems for Sustainable Development AI2SD'2020*. Springer, 2020.

[38] F.-Z. El-Alami, S. O. El Alaoui, and N. En-Nahnahi. A combining and merging deep neural networks for arabic text categorization. *Advanced Intelligent Systems Applied to Health*, 2020.

[39] F.-Z. El-Alami, S. O. El Alaoui, and N. En-Nahnahi. Contextual semantic embeddings based on fine-tuned arabert model for arabic text multi-class categorization. *Journal of King Saud University - Computer and Information Sciences*, 2020.

[40] F.-Z. El-Alami, S. O. El Alaoui, and N. En-Nahnahi. Deep neural models and retrofitting for arabic text categorization. *International Journal of Intelligent Information Technologies (IJIIT)*, 16(2):74–86, 2020.

[41] F.-Z. El-Alami, A. El Mahdaouy, S. O. El Alaoui, and N. En-Nahnahi. A deep autoencoder-based representation for arabic text categorization. *Journal of Information and Communication Technology*, 19(3):381–398, 2020.

[42] I. A. El-Khair. Effects of stop words elimination for arabic information retrieval: a comparative study. *International Journal of Computing and Information Sciences*, 4(3):119–133, 2006.

[43] A. El Mahdaouy, E. Gaussier, and S. O. El Alaoui. Arabic text classification based on word and document embeddings. In *International Conference on Advanced Intelligent Systems and Informatics*, pages 32–41. Springer, 2016.

[44] O. ElJundi, W. Antoun, N. El Droubi, H. Hajj, W. El-Hajj, and K. Shaban. hul-

mona: The universal language model in arabic. In *Proceedings of the Fourth Arabic Natural Language Processing Workshop*, pages 68–77, 2019.

[45] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[46] A. Elnagar, R. Al-Debsi, and O. Einea. Arabic text classification using deep learning models. *Information Processing and Management*, 57(1):102121, 2020.

[47] A. Farghaly and K. Shaalan. Arabic natural language processing: Challenges and solutions. *ACM Transactions on Asian Language Information Processing (TALIP)*, 8(4):1–22, 2009.

[48] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*, 2014.

[49] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305, 2003.

[50] Y. Freund. An adaptive version of the boost by majority algorithm. *Machine learning*, 43(3):293–318, 2001.

[51] N. Friedman, D. Geiger, M. Goldszmidt, G. Provan, P. Langley, and P. Smyth. Bayesian network classifiers, machine learning29 (1997), 131-163. *Google Scholar Google Scholar Digital Library Digital Library*, 1997.

[52] C. Gravelines. Deep learning via stacked sparse autoencoders for automated voxel-wise brain parcellation based on functional connectivity. 2014.

[53] B. Guo, C. Zhang, J. Liu, and X. Ma. Improving text classification with weighted word embeddings via a multi-channel textcnn model. *Neurocomputing*, 363:366–374, 2019.

[54] N. Y. Habash. Introduction to arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187, 2010.

[55] M. Hadni, S. E. A. Ouatik, and A. Lachkar. Word sense disambiguation for arabic text categorization. *Int. Arab J. Inf. Technol.*, 13(1A):215–222, 2016.

[56] S. Hingmire, S. Chougule, G. K. Palshikar, and S. Chakraborti. Document classification by topic labeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 877–880, 2013.

[57] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[58] I. Hmeidi, M. Al-Ayyoub, N. A. Abdulla, A. A. Almodawar, R. Abooraig, and N. A. Mahyoub. Automatic arabic text categorization: A comprehensive comparative study. *Journal of Information Science*, 41(1):114–124, 2015.

[59] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[60] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.

[61] A. Hotho, A. Nürnberger, and G. Paaß. A brief survey of text mining. In *Ldv Forum*, volume 20, pages 19–62. Citeseer, 2005.

[62] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[63] R. Johnson and T. Zhang. Supervised and semi-supervised text categorization using lstm for region embeddings. *arXiv preprint arXiv:1602.02373*, 2016.

[64] Y. Kadri and J.-Y. Nie. Effective stemming for arabic information retrieval. In *proceedings of the Challenge of Arabic for NLP/MT Conference, Londres, Royaume-Uni*, pages 68–74, 2006.

[65] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[66] S. Khan, H. Rahmani, S. A. A. Shah, and M. Bennamoun. A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, 8(1):1–207, 2018.

[67] S. Khoja and R. Garside. Stemming arabic text. *Lancaster, UK, Computing Department, Lancaster University*, 1999.

[68] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[69] R. Kumar, A. K. Ojha, S. Malmasi, and M. Zampieri. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11, 2018.

[70] M. Lan, C. L. Tan, J. Su, and Y. Lu. Supervised and traditional term weighting methods for automatic text categorization. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):721–735, 2008.

[71] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[72] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[73] C. Largeron, C. Moulin, and M. Géry. Entropy based feature selection for text categorization. In *Proceedings of the 2011 ACM symposium on applied computing*, pages 924–928, 2011.

[74] L. S. Larkey, L. Ballesteros, and M. E. Connell. Improving stemming for arabic information retrieval: light stemming and co-occurrence analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–282, 2002.

[75] L. S. Larkey, L. Ballesteros, and M. E. Connell. Light stemming for arabic information retrieval. In *Arabic computational morphology*, pages 221–243. Springer, 2007.

[76] L. S. Larkey and M. E. Connell. Arabic information retrieval at umass in trec-10. In *TREC*. Citeseer, 2001.

[77] Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[78] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[79] D. D. Lewis. Evaluating text categorization i. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*, 1991.

[80] D. D. Lewis. Representation and learning in information retrieval. 1992.

[81] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[82] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124, 2017.

[83] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[84] F. H. Mahyoub, M. A. Siddiqui, and M. Y. Dahab. Building an arabic sentiment lexicon using semi-supervised learning. *Journal of King Saud University-Computer and Information Sciences*, 26(4):417–424, 2014.

[85] T. Mandl, S. Modha, P. Majumder, D. Patel, M. Dave, C. Mandlia, and A. Patel. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In *Proceedings of the 11th Forum for Information Retrieval Evaluation*, pages 14–17, 2019.

[86] O. Melamud, J. Goldberger, and I. Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 51–61, 2016.

[87] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[88] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[89] A. H. Mohammad, T. Alwada'n, and O. Al-Momani. Arabic text categorization using support vector machine, naïve bayes and neural network. *GSTF Journal on Computing (JoC)*, 5(1):108, 2016.

[90] A. Moh'd Mesleh. Feature sub-set selection metrics for arabic text classification. *Pattern Recognition Letters*, 32(14):1922–1929, 2011.

[91] A. Montejo Ráez et al. Automatic text categorization of documents in the high energy physics domain. 2005.

[92] H. Mubarak, A. Rashed, K. Darwish, Y. Samih, and A. Abdelali. Arabic offensive language on twitter: Analysis and experiments. *arXiv preprint arXiv:2004.02192*, 2020.

[93] J. Nowak, A. Taspinar, and R. Scherer. Lstm recurrent neural networks for short text and sentiment classification. In *International Conference on Artificial Intelligence and Soft Computing*, pages 553–562. Springer, 2017.

[94] A. Nwesri. Effective retrieval techniques for arabic text. 2008.

[95] A. Odeh, A. Abu-Errub, Q. Shambour, and N. Turab. Arabic text categorization

algorithm using vector evaluation method. *arXiv preprint arXiv:1501.01318*, 2015.

[96] C. Olah. Understanding lstm networks. 2015.

[97] E. Omara, M. Mosa, and N. Ismail. Deep convolutional network for arabic sentiment analysis. In *2018 International Japan-Africa Conference on Electronics, Communications and Computations (JAC-ECC)*, pages 155–159. IEEE, 2018.

[98] C. D. Paice. Method for evaluation of stemming algorithms based on error counting. *Journal of the American Society for Information Science*, 47(8):632–649, 1996.

[99] A. Pasha, M. Al-Badrashiny, M. T. Diab, A. El Kholy, R. Eskander, N. Habash, M. Pooleery, O. Rambow, and R. Roth. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *Lrec*, volume 14, pages 1094–1101, 2014.

[100] D. Patel and M. Bhatnagar. Mobile sms classification. *International Journal of Soft Computing and Engineering (IJSCE) ISSN*, pages 2231–2307, 2011.

[101] A. Patra and D. Singh. Neural network approach for text classification using relevance factor as term weighing method. *International Journal of Computer Applications*, 68(17), 2013.

[102] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[103] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[104] T. Pranckevičius and V. Marcinkevičius. Application of logistic regression with part-of-the-speech tagging for multi-class text classification. In *2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE)*, pages 1–5. IEEE, 2016.

[105] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[106] S. Rosenthal, P. Atanasova, G. Karadzhov, M. Zampieri, and P. Nakov. A large-scale semi-supervised dataset for offensive language identification. *arXiv preprint arXiv:2004.14454*, 2020.

[107] F. Rossenblatt. The perceptron, a perceiving and recognizing automation. cornell aeronautical laboratory, 1957. Technical report, Report, 1957.

[108] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[109] K. C. Ryding. *A reference grammar of modern standard Arabic*. Cambridge university press, 2005.

[110] M. K. Saad and W. M. Ashour. Osac: Open source arabic corpora. *Osac: Open source arabic corpora*, 10, 2010.

[111] G. Salton. Automatic text processing: The transformation, analysis, and retrieval of. *Reading: Addison-Wesley*, 169, 1989.

[112] S. L. Salzberg. C4. 5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993, 1994.

[113] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[114] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168, 2000.

[115] R. A. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4(Sep):633–648, 2003.

[116] T. B. Shahi, T. N. Dhamala, and B. Balami. Support vector machines based part of speech tagging for nepali text. *International Journal of Computer Applications*, 70(24), 2013.

[117] D. Steinberg and P. Colla. Cart: classification and regression trees. *The top ten algorithms in data mining*, 9:179, 2009.

[118] C. Sun, X. Qiu, Y. Xu, and X. Huang. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019.

[119] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[120] H. Taira and M. Haruno. Feature selection in svm text categorization. In *AAAI/IAAI*, pages 480–486, 1999.

[121] C. C. Tan and C. Eswaran. Performance comparison of three types of autoen-

coder neural networks. In *2008 Second Asia International Conference on Modelling and Simulation (AMS)*, pages 213–218. IEEE, 2008.

[122] D. Tang, B. Qin, and T. Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432, 2015.

[123] Q. Tang, J. Chen, H. Lu, Y. Du, K. Yang, et al. Full attention-based bi-gru neural network for news text classification. In *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, pages 1970–1974. IEEE, 2019.

[124] X. Tian, R. Hérault, G. Gasso, S. Canu, and L. EA4108. Pré-apprentissage supervisé pour les réseaux profonds. In *Proceedings of Rfia*, volume 2010, page 36, 2010.

[125] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 conference of the North American chapter of the association for computational linguistics on human language technology-volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[126] B. Uria, M.-A. Côté, K. Gregor, I. Murray, and H. Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016.

[127] V. N. Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

[128] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[129] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, and L. Carin. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*, 2018.

[130] J. Wang, Z. Wang, D. Zhang, and J. Yan. Combining knowledge with deep convolutional neural networks for short text classification. In *IJCAI*, volume 350, 2017.

[131] J. C. Watson. *The phonology and morphology of Arabic*. Oxford University Press on Demand, 2002.

[132] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[133] H. Xiao. bert-as-service. https://github.com/hanxiao/bert-as-service, 2018.

[134] A. B. Yagoub. *Mausooat Annaho wa Assa*. Dar Alilm Lilmalayn, 1988.

[135] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019.

[136] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.

[137] S. A. Yousif, V. W. Samawi, I. Elkabani, and R. Zantout. The effect of combining different semantic relations on arabic text classification. *World Comput. Sci. Inform. Technol. J*, 5(1):12–118, 2015.

[138] S. A. Yousif, V. W. Samawi, I. Elkabani, and R. Zantout. Enhancement of arabic text classification using semantic relations with part of speech tagger. *W transactions Advances In Electrical And Computer Engineering*, pages 195–201, 2015.

[139] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*, 2019.

[140] M. Zampieri, P. Nakov, S. Rosenthal, P. Atanasova, G. Karadzhov, H. Mubarak, L. Derczynski, Z. Pitenis, and Ç. Çöltekin. Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020). *arXiv preprint arXiv:2006.07235*, 2020.

[141] W. Zhang, T. Yoshida, and X. Tang. A comparative study of tf* idf, lsi and multi-words for text classification. *Expert Systems with Applications*, 38(3):2758–2765, 2011.

[142] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[143] Y. Zhang, C. Li, R. Barzilay, and K. Darwish. Randomized greedy inference for joint segmentation, pos tagging and dependency parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 42–52, 2015.

[144] Y. Zhang and B. Wallace. A sensitivity analysis of (and practitioners' guide

to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.

[145] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*, 2016.

[146] X. Zhou, X. Wan, and J. Xiao. Attention-based lstm network for cross-lingual sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 247–256, 2016.

[147] X. Zhu, P. Sobihani, and H. Guo. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, pages 1604–1612, 2015.

[148] M. Zrigui, R. Ayadi, M. Mars, and M. Maraoui. Arabic text classification framework based on latent dirichlet allocation. *Journal of computing and information technology*, 20(2):125–140, 2012.

[149] M. Zulqarnain, R. Ghazali, M. G. Ghouse, and M. F. Mushtaq. Efficient processing of gru based on word embedding for text classification. *JOIV: International Journal on Informatics Visualization*, 3(4):377–383, 2019.