Université Sultan Moulay Slimane
Faculté des Sciences et Techniques de Beni Mellal
كلية العلوم والتقنيات ببني ملال
ⵜⴰⵖⵡⴰⵍⵜ ⵏ ⵜⵙⴰⵡⴰⵍⵉⵏ ⴰ ⵜⵓⵙⵙⵏⵉ ⵏ ⵓⵣⵜ ⵅⵎⵎⴰⵎ

THESE

Présentée pour l'obtention du

DOCTORAT EN SCIENCES

Spécialité : Informatique et Traitement Automatique des Langues

Laboratoire de Traitement de l'Information et Aide à la Décision (TIAD)

Par

Otman MAAROUF

Contribution à l'amélioration des techniques de NLP pour la langue Amazigh basée sur Deep Learning

---

Soutenue publiquement, le samedi 07 janvier 2023 à 10h, devant le jury composé de :

Monsieur Mohamed FAKIR, Professeur, Faculté des Sciences et Techniques, Université Sultan Moulay Slimane, Béni Mellal, **Président** ;

Monsieur Omar BENCHAREF, Professeur Habilité, Faculté des Sciences et Techniques, Université Cadi Ayyad, Marrakech, **Rapporteur** ;

Monsieur Mohamed BASLAM, Professeur Habilité, Faculté des Sciences et Techniques, Université Sultan Moulay Slimane, Béni Mellal, **Rapporteur** ;

Monsieur Hicham ZOUGAGH, Professeur Habilité, Faculté des Sciences et Techniques, Université Sultan Moulay Slimane, Béni Mellal, **Rapporteur** ;

Monsieur Yousef EL MOURABIT, Professeur Habilité, Faculté des Sciences et Techniques, Université Sultan Moulay Slimane, Béni Mellal, **Examinateur** ;

Monsieur Mustapha OUJAOURA, Professeur Habilité, Ecole Nationale des Sciences Appliquées, Université Cadi Ayyad, Safi, **Examinateur** ;

Monsieur Mohamed BINIZ, Professeur Assistant, Faculté Polydisciplinaire, Université Sultan Moulay Slimane, Béni Mellal, **Co-encadrant** ;

Monsieur Rachid EL AYACHI, Professeur, Faculté des Sciences et Techniques, Université Sultan Moulay Slimane, Béni Mellal, **Directeur de Thèse**.

Dédicaces

À mon cher père.

À ma chère mère.

À mes frères et mes sœurs.

À ma grande famille.

À mes amis.

Liste des publications

— **Communications Orales**

1. Amazigh Part Of Speech Tagging using Gated recurrent units (GRU) ;May 2021 ; DOI : 10.1109/ICOA51614.2021.9442662 ; Conference : 2021 7th International Conference on Optimization and Applications (ICOA).
2. Part-of-Speech Tagging Using Long Short Term Memory (LSTM) : Amazigh Text Written in Tifinaghe Characters ; May 2021 ; DOI : 10.1007/978-3-030-76508-8_1 ; In book : Business Intelligence ; International Conference of Business Intelligence (CBI 27-29 May,2021).
3. Part-of-speech tagging using conditional random fields and decision tree : Amazigh text written in Tifinagh character ; December 2020 ; Conference : International Conference on Advanced Intelligent Systems for Sustainable Development (21-26 December AI2SD'2020).

— **Articles publiées**

1. Amazigh part-of-speech tagging with machine learning and deep learning ; December 2021 Indonesian Journal of Electrical Engineering and Computer Science 24(3):1814 ; DOI : 10.11591/ijeecs.v24.i3.pp1814-1822.
2. Correcting optical character recognition result via a novel approach ; April 2022 International Journal of Informatics and Communication Technology (IJ-ICT) 11(1):8 ; DOI : 10.11591/ijict.v11i1.pp8-19

— **Articles soumis**

1. MAAROUF, Otman, et EL AYACHI Rachid, BINIZ Mohamed . « Applying Deep learning techniques on Amazigh Name Entity Recognition (ANER)» Walailak Journal of science and technology.
2. MAAROUF, Otman, et EL AYACHI Rachid, BINIZ Mohamed . « Correcting OCR results with POS models» Indonesian Journal of Electrical Engineering and Computer Science .
3. MAAROUF, Otman, et EL AYACHI Rachid, BINIZ Mohamed . « Part-of speech and named entities amazigh dataset» Data in Brief Journal .
4. MAAROUF, Otman, et EL AYACHI Rachid, BINIZ Mohamed . « Automatic Translation From English To Amazigh Using Transformer Learning» Baghdad Science Journal .

Remerciements

Avant tout, je remercie Dieu qui m'a donné la force, le courage et l'espoir nécessairepour accomplir ce travail. Je tiens à remercier vivement tout d'abord mes deux directeurs de thèse :

— **Rachid EL AYACHI** : Professeur à la Faculté des Sciences et Techniques Beni Mellal, université Sultan Moulay Slimane.

— **Mohamed BINIZ** : Professeur Faculté Poly-disciplinaire, Université Sultan Moulay Slimane.

Pour leur encadrement continu, pour les remarques constructives qu'ils m'ont fournies ainsi que pour leurs précieux conseils durant toute la période de ce projet de thèse.

Je les remercie également pour la confiance qu'ils m'ont accordé et pour la grande liberté d'idées et de travail qu'ils m'ont donné. En dehors des apports scientifiques, je n'oublierai pas aussi de les remercier pour leurs qualités humaines.

Mes vifs remerciements aux membres du jury qui ont bien voulu étudier, évaluer cette thèse et pour toutes les remarques prodiguées.

Résumé

Dans la recherche scientifique, il y a quelques langues qui sont moins traitées, parmi elles on trouve la langue Amazigh, cette dernière souffre du manque d'outils intelligents de traitement. La résolution de ce problème représente l'objectif de cette thèse, qui s'intéresse à la création d'un framework de TALN (Traitement Automatique de la Langue Naturelle) regroupant un ensemble d'outils pour le traitement de la langue Amazigh. Les tâches du framework réalisées jusqu'à présent sont : la correction de la sortie d'un OCR (Optical Character Recognition), l'étiquetage, la détermination des entités nommées et la traduction. La réalisation de ce framework a nécessité les approches de TALN, Machine Learning et Deep Learning. Comme toutes les langues, un corpus est nécessaire à l'aboutissement d'un projet dans le domaine de TALN, c'est pour cela un corpus annoté de la langue Amazigh écrit en Tifinagh est élaboré.
Un OCR est un système qui permet de reconnaitre le contenu d'un document scanné, le résultat de ce système nécessite parfois une correction, alors comme première contribution, une approche de correction est proposée basée sur les n-grammes. Comme deuxième contribution, l'étiquetage morpho-syntaxique est une opération primordiale du TALN, à ce niveau, un ensemble de modèles ont été élaborés et testés afin d'avoir des résultats performants, ces modèles sont basés sur les approches de Machine Learning (Arbre de décision et Conditional Random Fields) et les approches de Deep Learning (RNN, LSTM, GRU et Bi-LSTM). En plus, dans la troisième contribution, le problème de la détermination du type d'entité nommé est résolu via la préparation des données et la proposition de cinq modèles de classification des entités nommées, ces modèles sont fondés sur l'architecture des réseaux de neurones récurrente (RNN, LSTM, GRU, Bi-LSTM et Bi-GRU). La dernière contribution concerne la traduction de la langue Amazigh en Anglais, cet outil a exigé en premier lieu un corpus parallèle Amazigh-Anglais, et en seconde lieu la proposition des modèles de traduction automatique en se basant sur GRU, LSTM et Transformateurs.
Mots clés : Langue Amazigh, TALN, Machine Learning, Deep Learning, OCR ,POS, Entités Nommées, Machine Translation.

Abstract

In scientific research, there are some languages that are less processed, among them the Amazigh language, which suffers from the lack of intelligent processing tools. The resolution of this problem represents the objective of this thesis, which is interested in the creation of an NLP (Natural Language Processing) framework gathering a set of tools for the processing of the Amazigh language. The tasks of the framework realized so far are : the correction of the output of an OCR (Optical Character Recognition), the labeling, the determination of the named entities and the translation. The realization of this framework required NLP, Machine Learning and Deep Learning approaches. Like all languages, a corpus is necessary for the success of a project in the field of NLP, that is why an annotated corpus of the Amazigh language written in Tifinagh is developed.
An OCR is a system that allows to recognize the content of a scanned document, the result of this system sometimes requires a correction, so as a first contribution, a correction approach is proposed based on n-grams. As a second contribution, morpho-syntactic labeling is a primordial operation of NLP, at this level, a set of models have been developed and tested in order to have efficient results, these models are based on Machine Learning approaches (Decision Tree and Conditional Random Fields) and Deep Learning approaches (RNN, LSTM, GRU and Bi-LSTM). In addition, in the third contribution, the problem of determining the type of named entity is solved via the preparation of data and the proposal of five classification models of named entities, these models are based on the architecture of recurrent neural networks (RNN, LSTM, GRU, Bi-LSTM and Bi-GRU). The last contribution concerns the translation of the Amazigh language into English, this tool required firstly a parallel Amazigh-English corpus, and secondly the proposal of machine translation models based on GRU, LSTM and Transformers.
Keywords : Amazigh language, TALN, Machine Learning, Deep Learning, OCR ,POS, Named Entities, Machine Translation.

ⴰⵙⴳⵣⵍ

ⴳ ⵓⵔⵣⵣⵓ ⴰⵎⴰⵙⵙⴰⵏ ⵍⵍⴰⵏⵜ ⴽⵔⴰ ⵜⵓⵜⵍⴰⵢⵉⵏ ⵏⵏⴰ ⵓⵔⵉⵙⵡ ⵓⵔ ⵉⵙⵡⵓⵔⵉⵏ, ⵣⵉ ⴷⵉⴳⵙⵏ ⵏⵓⴼⴰ ⵜⵓⵜⵍⴰⵢⵜ ⵜⴰⵎⴰⵣⵉⵖⵜ ⵜⴰⵢⵢⴰⴹ ⵜⵅⵙ ⵣⵉ ⵓⵅⵚⵚⴰ ⵏ ⵉⵎⴰⵙⵙⵏ ⵉⴳⴰⵜⴰⵔⵍⵏ ⴰⵙⵡⵓⵔⵉ ⴰⴼⵙⵙⴰⵢ ⵏ ⵜⵎⵓⴽ ⵔⵉⵙⵜ ⴰⴷ ⵉⴳⴰ ⴰⵡⵜⵜⴰⵙ ⵏ ⵜⵏⴰⵡⵜ ⴰⴷ, ⵏⵏⴰ ⵉⵔⴰⵏ ⴰⴷ ⵉⵙⴽⵔ ⵢⴰⵏ ⵓⵙⴰⵜⵉ ⵏ ⴰⵙⵉⴳⴳⵍ ⴰⵟⵓⵎⴰⵜⵉ ⵏ ⵜⵓⵜⵍⴰⵢⵜ ⵜⴰⴳⴰⵎⴰⵏⵜ (TALN) ⵉⵙⵎⵓⵏ ⵜⴰⵔⴰⴱⴱⵓⵜ ⵏ ⵉⵎⴰⵙⵙⵏ ⵉ ⵓⵙⵎⵙⴰⵙⴰ ⵏ ⵜⵓⵜⵍⴰⵢⵜ ⵜⴰⵎⴰⵣⵉⵖⵜ ⵜⵉⵡⵓⵔⵉⵡⵉⵏ ⵏ ⵓⵙⴰⵜⵉ ⵢⴰⴷ ⴰⵔ ⵖⵉⵍ ⴰⴷ ⴳⴰⵏⵜ : ⴰⵙⵉⴳⴳⵍ ⵏ ⵜⵢⴰⴼⵓⵜ ⵏ ⴰⵙⵏⵓⴱⴳ ⵏ ⵉⵙⴽⴽⵉⵍⵏ ⵏ ⵓⵥⵕⵓ (OCR), ⴰⵙⵎⴰⵍ, ⴷ ⵓⵙⵎⴰⵍ ⵏ ⵉⵙⴽⴽⵉⵍⵏ ⵉⵜⵜⵓⵙⵎⴰⴳⵍⵏ ⴷ ⵓⵙⵓⵖⵍ. ⵉⵅⵚⵚⴰ ⴰⴷ ⵢⵉⵍⵉ ⵓⵙⴰⵜⵉ ⴰⴷ ⵙ ⵜⵖⴰⵔⴰⵙⵉⵏ ⵏ TALN, ⴰⵙⵍⵎⴷ ⵏ ⵉⵎⴰⵙⵙⵏ ⴷ ⵓⵙⵍⵎⴷ ⴰⵎⵇⵔⴰⵏ. ⵣⵓⵏⴷ ⵜⵓⵜⵍⴰⵢⵉⵏ ⴰⴽⴽⵡ□ ⵉⵅⵚⵚⴰ ⴰⴷ ⵢⵉⵍⵉ ⵓⵙⵏⴼⵍⵓⵍ ⵏ ⵓⵙⵏⴼⵍⵓⵍ ⴳ ⵉⴳⵔ ⵏ TALN, ⵙ ⵎⴰⵏⴰⵢⴰ, ⵉⵜⵜⵓⵙⴽⴰⵔ ⵓⵙⵏⴼⵍⵓⵍ ⵏ ⵜⵓⵜⵍⴰⵢⵜ ⵜⴰⵎⴰⵣⵉⵖⵜ ⵉⵜⵜⵢⴰⵔⴰⵏ ⵙ ⵜⵓⵜⵍⴰⵢⵜ ⵜⵉⴼⵉⵏⴰⵖ.

OCR ⵉⴳⴰ ⵢⴰⵏ ⵓⵏⴳⵔⴰⵡ ⵏⵏⴰ ⵉⵜⵜⴰⵊⵊⴰⵏ ⴰⴷ ⵉⵜⵜⵓⵙⵙⴰⵏ ⵓⵏⴰⵙⵉⵡ ⵏ ⵢⴰⵏ ⵓⴷⵍⵉⵙ ⵉⵜⵜⵓⵙⴽⴰⵏ, ⵜⴰⵢⴰⴼⵓⵜ ⵏ ⵓⵏⴳⵔⴰⵡ ⴰⴷ ⴰⵔ ⵜ ⵉⵅⵚⵚⴰ ⴽⵔⴰ ⵏ ⵜⵉⴽⴽⴰⵍ ⵢⴰⵏ ⵓⵙⵉⴳⴳⵍ, ⴼ ⵖⵉⴽⴰⵍⵏ ⴰⵎⵎ ⵜⴰⵙⵏⵜⴰⵢⵜ ⵜⴰⵎⵣⵡⴰⵔⵓⵜ, ⵉⵜⵜⵓⵙⵓⵎⵔ ⵢⴰⵏ ⵓⴱⵔⵉⴷ ⵏ ⵓⵙⵉⴳⴳⵍ ⵉⴱⴷⴷⵏ ⵅⴼ n-gram. ⴰⵎⵎ ⵜⵉⵍⴰⵍⵜ ⵜⵉⵙⵙ ⵙⵏⴰⵜ, ⵉⴳⴰ ⵓⵙⴼⵙⵉ ⵏ ⵍⵎⵓⵔⴼⵓⵙⵉⵏⵟⴰⴳⵓ ⵢⴰⵜ ⵜⵎⵀⵍⴰ ⵜⴰⵎⵣⵡⴰⵔⵓⵜ ⵏ TALN, ⴳ ⵓⵙⵡⵉⵔ ⴰⴷ, ⵉⵜⵜⵓⵙⴽⴰⵔ ⴷ ⵉⵜⵜⵓⵙⵜⴰⵍ ⵢⴰⵏ ⵓⴳⵔⴰⵡ ⵏ ⵉⵎⴷⵢⴰⵡⵏ ⵎⵔⴰ ⴰⴷ ⵢⵉⵍⵉ ⵓⵡⵜⵜⵓ ⵏ ⵜⵢⴰⴼⵓⵜⵉⵏ ⵉⵖⵓⴷⴰⵏ, ⵉⵎⴷⵢⴰⵡⵏ ⴰⴷ ⵉⴱⴷⴷⵏ ⵅⴼ ⵉⵎⵙⴰⵙⴰ ⵏ ⵓⵙⵍⵎⴷ ⵏ ⵉⵎⴰⵙⵙⵏ (ⴰⵙⵖⵍⴼ ⴷ ⵉⴳⵔⴰⵏ ⵉⵎⵙⴰⵙⴰⵏ) ⴷ ⵉⵎⵙⴰⵙⴰ ⵏ ⵓⵙⵍⵎⴷ ⴰⵎⵇⵔⴰⵏ (RNN, LSTM, GRU ⴷ Bi-LSTM). ⵓⴳⴳⴰⵔ, ⴳ ⵜⵙⵏⵙⵉ ⵜⵉⵙⵙ ⴽⵕⴰⴹⵜ, ⴷⴰ ⵜⵜⵓⵙⴱⴷⴷ ⵜⵎⵓⴽⵔⵉⵙⵜ ⵏ ⵓⵙⵜⴰⵢ ⵏ ⵡⴰⵏⴰⵡ ⵏ ⵜⵎⵔⵔⵙⵍⵜ ⵉⵜⵜⵓⵙⵎⵔⴰⵙⵏ ⵙ ⵓⵙⵎⵓⵜⵜⴳ ⵏ ⵉⵏⵖⵎⵉⵙⵏ ⴷ ⵓⵙⵏⵓⵎⵍ ⵏ ⵙⵎⵎⵓⵙⵜ ⵏ ⵜⵎⴷⵢⴰⵣⵉⵏ ⵏ ⵓⵙⵎⵓⵜⵜⴳ ⵏ ⵜⵎⵔⵔⵙⵍⵜ ⵉⵜⵜⵓⵙⵎⵔⴰⵙⵏ, ⵜⵉⵎⴷⵢⴰⵣⵉⵏ ⴰⴷ ⴷⴰ ⵜⵜⵓⵙⵏⵎⴰⵍⴰⵏ ⵅⴼ ⵜⵓⵚⴽⵉⵡⵜ ⵏ ⵜⵊⵎⵉⵍⵉⵏ ⵏ ⵉⵏⵢⵓⵔⵓⵏ (RNN, LSTM, GRU, Bi-LSTM ⴷ Bi-GRU). ⵜⴰⵙⵏⵜⴰⵢⵜ ⵜⴰⵎⴳⴳⴰⵔⵓⵜ ⵉⵥⵍⵉⵏ ⵙ ⵓⵙⵓⵖⵍ ⵏ ⵜⵓⵜⵍⴰⵢⵜ ⵜⴰⵎⴰⵣⵉⵖⵜ ⵙ ⵜⵓⵜⵍⴰⵢⵜ ⵜⴰⵏⴳⵍⵉⵣⵉⵜ, ⵉⵙⵓⵜⵔ ⵓⵖⴰⵡⴰⵙ ⴰⴷ ⵜⴰⵎⵣⵡⴰⵔ ⵓⵜ ⵢⴰⵏ ⵓⴳⵔⴰⵡ ⵏ ⵓⵙⵓⵖⵍ ⴰⵎⵣⴳⵓⵏ ⵏ ⵜⵓⵜⵍⴰⵢⵜ ⵜⴰⵎⴰⵣⵉⵖⵜⵜⴰⵏⴳⵍⵉⵣⵉⵜ, ⴷ ⵜⵉⵙⵙ ⵙⵏⴰⵜ ⴰⵙⵏⵜⵉ ⵏ ⵉⵎⴷⵢⴰⵡⵏ ⵏ ⵓⵙⵓⵖⵍ ⴰⵜⵡⴰⵍ ⵙ ⵓⵙⴰⵜⵉ ⵏ GRU, LSTM ⴷ ⵉⵙⵏⴼⵍⵓⵍⵏ.

ⵜⵉⴳⵓⵔⵉⵡⵉ ⵜⵉⵙⵓⵔⴰ : ⵜⵓⵜⵍⴰⵢⵜ ⵜⴰⵎⴰⵣⵉⵖⵜ, TALN, ⴰⵙⵍⵎⴷ ⵏ ⵉⵎⴰⵙⵙⵏ, ⴰⵙⵍⵎⴷ ⴰⵎⵇⵔⴰⵏ, OCR,POS, ⵜⵉⵎⵔⴱⴰⵍ ⵉⵜⵜⵓⵙⵎⴰⴳⵍⵏ, ⴰⵙⵓⵖⵍ ⵏ ⵉⵎⴰⵙⵙⵏ

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

CRFs	Conditional Random Fields
HMM	Hidden Markov Model
IA	Intelligence Artificielle
IRCAM	Institut Royal de la Culture Amazighe
OOV	Out Of Vocabulary
POS	Part Of Speech
SVM	Support Vector Machines
TAL	Traitement Automatique des Langues
ML	Machine Learning
DS	decision Tree
NE	Name Entity
NER	Named Entity Recognition
HCA	Haut-Commissariat de l'Amazighité
RNN	Reccurent Neural Network
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
Bi LSTM	Bidirectional Long Short Term Memory
Bi GRU	Bidirectional Gated Recurrent Unit
NLP	Natural Language Processing
TALN	Traitement Automatique du Langage Naturel
OCR	Optical Caracters Recognition
MT	Machine Translation
SMT	Statistical Machine Translation
NMT	Neural Machine Translation

Introduction générale

La langue amazighe est principalement parlée en Afrique du Nord. D'autres communautés dans certaines régions du Niger et du Mali, ainsi que des dizaines de milliers d'immigrants du monde entier, la parlent. La langue amazighe est composée de plusieurs dialectes, dont aucun n'est considéré comme la langue nationale dans les pays où vit un peuple amazigh. Avec l'émergence de la revendication identitaire, les autochtones se battent pour la préservation et le développement de leur langue et de leur culture. Pour atteindre cet objectif, certains pays du Maghreb ont créé des organismes spécialisés, comme l'Institut Royal de la Culture Amazighe (IRCAM) au Maroc. Comme toutes les langues, il existe une discipline, nommée Traitement Automatique de la Langue Naturelle (TALN), qui s'intéresse à leurs traitements. Le TALN est un domaine qui fait partie de l'intelligence artificielle, il se concentre sur le développement des outils informatiques facilitant le traitement des langues, il admit plusieurs applications, à savoir : l'analyse syntaxique et sémantique, la traduction, la détection de plagiat, la génération des résumés, ...etc. Dans la littérature, pour quelques langues telles que : Arable, Anglais et Français, il y a un nombre important d'articles qui s'intéressent au domaine TALN. Cependant, la langue Amazigh n'a pas eu sa part dans ce domaine, ce qui justifie la pauvreté d'outils de TALN pour la langue Amazigh dans le marché. Cette pauvreté représente la problématique traitée dans cette thèse. La résolution de la problématique, citée ci-dessus, existe dans le développement d'un framework regroupant un ensemble d'outils de TALN pour la langue Amazigh. En plus des besoins matériels et logiciels, la réalisation de ce framework exige la complémentarité de plusieurs axes de recherche, en l'occurrence : les approches du Traitement de la Langue Naturelle (TLN), les approches de vectorisation, les approches de classification et la reconnaissance des caractères.

Ce rapport de thèse est un fruit des années de recherche effectuées pour répondre à la problématique soulevée précédemment, cette réponse est sous forme d'un ensemble de contributions réalisées dans le domaine de TALN, et de façon précise pour la langue Amazigh.

La première contribution servie à décrire le jeu d'étiquettes réduit utilisé pour la création du corpus de la langue amazighe, puis nous présentons les différents types des entités nommées, puis on analyse les distributions de NE et POS.

La deuxième contribution présente les différentes étapes pour entraîner et tester les approches d'apprentissage automatique (arbre de décision, CRF), et les approches d'apprentissage profond (RNN, LSTM, GRU, Bi-LSTM), pour étiqueter des phrases par les POS, puis on a faire une comparaison entre les résultats trouvé.

La troisième contribution concerne la correction des résultats fournis par un OCR utilisant

deux techniques, la premiers est une correction basée sur le calcule n-gram et une deuxième technique on utilise les résultats d'étiquiteur de POS .
La quatrième contribution est une application des différentes techniques de deep learning pour prédire les NEs dans des textes écrits en tifinagh.
La cinquième contribution contient la description des techniques de machine translation qu'on les appliquées sur un corpus parallèle contient des phrases écrit on tifinagh et des phrases anglaises, pour réaliser un traducteur de la langue anglaise vers la langue amazighe .
En plus de l'introduction générale et de la conclusion générale, le document est organisé en six chapitres ; le premier chapitre est un état d'art sur le Traitement Automatique de la Langue Naturelle et les approches d'apprentissage, le deuxième chapitre décrit le corpus annoté pour la langue Amazigh, le deuxième chapitre explique l'approche proposée pour la correction de la sortie d'un système OCR, le quatrième chapitre présente les modèles proposés pour l'étiquetage de la langue Amazigh, le cinquième chapitre est réservé à la détermination des entités nommées via un ensemble de modèles suggérés, tandis que le dernier chapitre propose quelques modèles pour la traduction.

Chapitre 1

Etat de l'art

1.1 introduction

Le traitement automatique du langage naturel (TALN) est l'approche informatisée d'analyse de texte qui repose à la fois sur un ensemble de théories et sur un ensemble de technologies. Et, étant un domaine très actif de recherche et développement, il n'y a pas une seule définition convenue qui satisferait tout le monde, mais il y a certains aspects qui feraient partie de la définition de toute personne bien informée.
L'objectif de TALN, est "d'accomplir un traitement du langage de type humain". Le choix du mot « traitement » est très délibéré et ne doit pas être remplacé par « compréhension ». Car bien que le domaine de le TALN ait été à l'origine appelé compréhension du langage naturel (NLU) aux débuts de l'IA, il est bien admis aujourd'hui que si l'objectif de TALN est une véritable NLU, cet objectif n'a pas encore été atteint.
Le TALN a des objectifs plus pratiques, dont beaucoup sont liés à l'application particulière pour laquelle elle est utilisée. Par exemple, un système IR basé sur le TALN a pour objectif de fournir des informations plus précises et complètes en réponse au besoin réel d'information d'un utilisateur. L'objectif du système TALN ici est de représenter le véritable sens et l'intention de la requête de l'utilisateur, qui peut être exprimée aussi naturellement dans le langage courant que s'il parlait à un bibliothécaire de référence. De plus, le contenu des documents recherchés sera représenté à tous leurs niveaux de signification afin qu'une véritable correspondance entre le besoin et la réponse puisse être trouvée, quelle que soit la manière dont l'un ou l'autre est exprimé dans sa forme de surface.
Dans ce chapitre, nous donnerons une brève description de la langue Amazigh, nous présenterons ensuite des différents outils de tokénisation, lemmatisation et stemming, et les approches de vectorisation et, de classifications.

1.2 Contexte Amazigh

1.2.1 Situation sociolinguistique au Maroc

Les langues les plus largement parlées au Maroc sont l'arabe marocain et l'amazigh, tous les deux sont des dialectes sémitiques de la famille des langues afro-asiatiques. L'arabe marocain, tout comme l'amazigh, possède plusieurs assortiments provinciaux, mais aussi un assortiment supra-public connu sous le nom d'arabe marocain moderne.
L'arabe standard, la langue d'autorité de l'État, est pour la plupart utilisé comme une copie papier et des discussions informelles et des annonces de nouvelles de la radio et de la télévision et est un moyen d'enseignement d'autorité.
Cependant, le Maroc est le pays qui compte le plus grand nombre de locuteurs amazighs, L'amazigh est également utilisé dans d'autres pays d'Afrique du Nord, en particulier en Algérie, en Tunisie, en Libye et en Egypte, avec deux ou trois mille locuteurs chacun. [1] situe la quantité d'amazighophones au Maroc à la moitié de la population totale, d'autres à environ 40% [2]. La plupart de ces locuteurs ont appris l'amazigh à la maison, comme première langue, et peu l'auraient appris comme langue ultérieure. Ceci est fondamentalement dû au faible statut de l'amazigh et, jusqu'à récemment, à sa non-apparition dans les établissements publics. Il existe trois assortiments primaires d'amazigh : Tarifit dans la partie nord-est du Maroc, Tamazight dans le centre du Maroc, et Tashelhit dans la partie sud-ouest du pays.
Le français reste une langue importante dans la vie métropolitaine marocaine, notamment en ce qui concerne les affaires et l'organisation actuelle qui plus est, l'organisation actuelle est concernée. Les mesures situent la quantité de Marocains qui peuvent parler ou potentiellement comprendre le français à 41,5% [3].
L'anglais a récemment acquis une plus grande importance dans le pays, principalement en raison de la diffusion du web dans la nation et de la façon dont il est utilisé parmi les deuxième et troisième âges de nombreux travailleurs marocains qui vivent leur enfance dans diverses nations en Europe, y compris lorsqu'ils retournent au Maroc en vacances.
La situation linguistique au Maroc a été décrite comme diglossique [4], ou même triglossique [5] : Les langues vernaculaires voisines, l'arabe marocain et l'amazigh, sont utilisées pour la plupart dans des contextes non formels, l'arabe standard est pour la plupart utilisé dans la composition et le discours formel.

1.2.2 Amazigh dans le contexte de la politique d'aménagement linguistique au Maroc

La première fois que la langue amazigh a été référencée dans un texte d'autorité gérant la stratégie d'aménagement linguistique dans le Maroc libre était en 1994. Dans son discours du 20 août 1994, le roi Hassan II a souligné l'importance de l'amazigh et de l'arabe marocain pour le caractère public marocain. Ce discours a largement contribué à rehausser la situation de l'amazigh, notamment lorsqu'il a évoqué ses chances de recevoir une éducation officielle.

Un second texte dans lequel l'amazigh est expressément référencé, est le Contrat National d'Education, diffusé en 1999. L'article 96 de la Charte, section 9, énonce quatre plans d'activités liés à la stratégie sur les dialectes utilisés au Maroc : (1) soutenir l'enseignement de l'arabe ; (2) différencier les dialectes utilisés dans l'enseignement des sciences et de l'innovation ; (3) ouvrir l'amazigh ; et (4) développer les capacités dans les dialectes inconnus. La sollicitation de ces quatre axes définit évidemment l'attitude de l'autorité à l'égard des dialectes du Maroc. En tout cas, vient l'arabe, la langue officielle de l'État. En deuxième position, on trouve un appel à une plus grande variété de dialectes utilisés dans la présentation de la science et de l'innovation, ce qui fait probablement allusion à l'utilisation de l'anglais à côté du français. La troisième place est occupée par "l'ouverture de l'amazigh", ce qui implique de rendre possible l'enseignement de cette langue en cas de besoin, et selon les souhaits des spécialistes de l'enseignement du voisinage. La quatrième place est accordée à la culture de la capacité dans les dialectes inconnus (une fois de plus, une référence probable à l'anglais). Le contrat stipule que les associations pour la recherche et l'amélioration de l'amazigh seront créées au même titre que les centres d'enseignement spécialisés (article 116).

Le troisième texte, et sans aucun doute le plus important, en matière de stratégie linguistique liée à l'amazigh au Maroc, est le décret royal du 17 octobre 2001. Cette déclaration a dénoté la fondation de l'Institut Royal de la Culture Amazigh (IRCAM-Institut Royal de la Culture Amazigh). La mission de l'Institut était de contribuer à "la protection et la promotion de la culture amazigh ainsi qu'à l'amélioration de son statut dans les domaines de l'enseignement public, du social et des médias [et] de lui donner un nouvel élan en tant que patrimoine public et source de fierté pour tous les Marocains".

Le décret royal mentionne également un certain nombre de motifs pour la création de l'IRCAM, parmi lesquels la mise en œuvre de " la politique linguistique déterminée par la Charte nationale de l'éducation et de la formation qui stipule l'introduction de l'amazigh dans le système éducatif ". Bien que le texte de la Charte nationale ne traite que de l'introduction de l'amazigh comme support secondaire à l'apprentissage de l'arabe, le décret trace des lignes plus larges pour la promotion de la culture amazigh comme suit : " L'Institut participe à la mise en œuvre des politiques adoptées par Notre Majesté qui permettront l'introduction de la langue et de la culture amazighs dans le système éducatif et assureront la diffusion de son influence dans les contextes social, culturel, médiatique, national, régional et local " (décret royal 2001). Deux des tâches attribuées à l'Institut au niveau de l'enseignement sont : (1) étudier l'écriture de manière à faciliter la production d'outils didactiques pour l'enseignement de l'amazigh, et élaborer le lexique général et développer des dictionnaires spécialisés ; et (2) formuler un plan d'action pédagogique dans l'enseignement général et dans les programmes liés aux affaires locales et à la vie régionale. Un autre point intéressant concernant le décret est la mention explicite de l'amazigh comme étant lié à l'identité nationale : "L'amazigh est une responsabilité nationale". C'est la première fois que l'amazigh est présenté comme un symbole unificateur. Dans le passé, les discours sur la langue et la culture amazighs comme source de conflit et de désunion avaient prédominé.2 Le décret de 2001 implique clairement que l'enseignement de l'amazigh est destiné à tous les Marocains, quelle que soit leur langue maternelle. En d'autres

termes, les cours d'amazigh sont destinés aux enfants amazighs et arabophones et peuvent être dispensés par des enseignants amazighs et arabophones. Ces tâches d'aménagement linguistique de codification, d'élaboration, de développement de programmes et de mise en œuvre ont été prises en charge par l'IRCAM dès la création de l'Institut en 2001.

1.2.3 Normalisation et élaboration de la langue amazigh

Comme indiqué précédemment, avant l'introduction de l'amazigh dans le système scolaire, un certain volume de textes dans différentes variétés d'amazigh était déjà disponible. Il s'agissait de publications scientifiques sur la grammaire de ces variétés, d'ouvrages de fiction et de non-fiction, de dictionnaires, et des listes de terminologie spécialisée (par exemple, pour les mathématiques, l'informatique, la grammaire).
La disponibilité de ces textes a fourni à l'IRCAM une base relativement solide sur laquelle pour travailler à la normalisation de l'amazigh.
La position de l'IRCAM concernant le choix entre l'enseignement des variétés locales et l'amazigh standard a été claire dès le début : ils ont décidé de choisir l'amazigh standard, mais de commencer la mise en œuvre avec les trois variétés principales et de passer progressivement à l'amazigh standard. Cette stratégie a été formulée comme suit : (1) garantir la proximité sociolinguistique qu'un géolecte offre aux locuteurs de (sous-) variétés au sein d'une région ayant un dialecte et une culture communs ; et (2) de transcender les différences dialectales superficielles afin de fournir un moyen d'intercommunication aux locuteurs natifs ainsi qu'aux arabophones ".
Le choix d'une norme nationale unique représentait un défi. Les trois principales variétés sont, dans une large mesure, mutuellement inintelligibles, en particulier au niveau lexical et au niveau de la prononciation. Cependant, les locuteurs d'autres variétés qui vivent dans des régions voisines ont moins de difficultés d'intelligibilité. En général, les gens étaient réticents à s'opposer à une norme nationale pour l'amazigh. En fait, le haut degré de similitude entre les grammaires des différentes variétés amazighs a conduit de nombreux auteurs à parler de l'amazigh comme d'une seule langue [6]. Un autre défi résidait dans l'élaboration de la terminologie. Le vocabulaire pour les termes métalinguistiques termes métalinguistiques et des matières scolaires a dû être développé, car l'amazigh était principalement utilisé dans le domaine informel et l'utilisation de l'arabe était un facteur de risque.

1.2.4 Ecriture de la langue amazigh

Comme toute langue qui passe du mode oral au mode écrit, la langue amazigh avait besoin d'un système graphique. Le Tifinagh est le nom du jeu de lettres de la langue amazigh. Le nom Tifinagh peut signifier "les lettres phéniciennes", ou potentiellement.
Des variantes du tifinagh sont utilisées pour composer les dialectes berbères au Maroc, en Algérie, au Mali et au Niger. Les jeux de lettres arabes et latines sont également utilisés. Le contenu de pointe du tifinagh est également appelé touareg, berbère ou néo-tifinagh, pour le distinguer de l'ancienne écriture berbère.

En 2003, le Tifinagh est devenu le contenu officiel de la langue tamazight au Maroc. Il est également utilisé par les Touaregs, en particulier par les femmes, pour les notes privées, les lettres d'amour et l'enrichissement. Pour les usages publics, le jeu de lettres arabes est régulièrement utilisé.
Le tifinagh s'écrit de gauche à droite et contient 33 graphèmes qui correspondent à :

— 27 consonnes dont les labiales ⵀ,ⵃ,ⴼ, les dentales ⵜ, ⵏ, ⵟ , ⴹ, ⵍ,ⵔ, ⵇ, ⵅ, les alvéolaires ⵙ ⵣ,ⵚ,ⵥ, les palatales ⵛⵊ ⵊⵉ, les vélaires ⴽ ,ⵅ, les labiovelaires ⴽⵯ,ⵅⵯ, les uvulaires ⵇ,ⵅ,ⵖ, les pharyngées ⵄ,ⵃ et les laryngées ⴰ ;

— 2 semi-consonnes :ⵢ et ⵡ ;

— voyelles : trois voyelles complètes : ⵓ,ⵉ,ⴰ et voyelle neutre (ou schwa) : ⴻ .

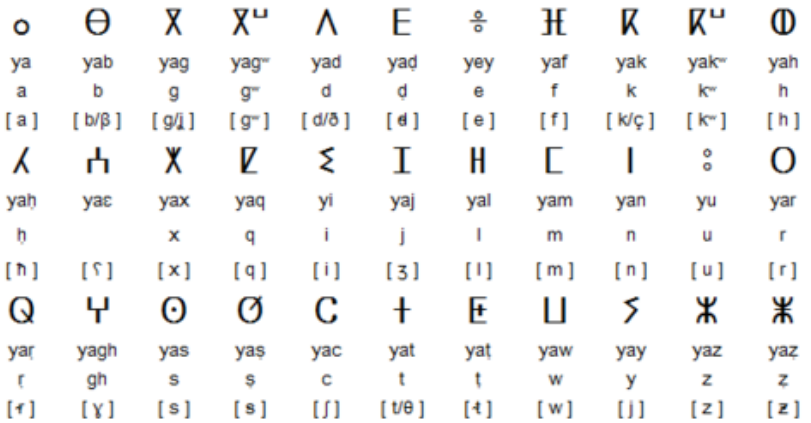

FIGURE 1.1 – Alphabet de tifinagh (la version officielle de l'IRCAM)

1.2.5 Morphologie de la langue amazigh

La langue amazigh est une langue riche en morphologie qui est agglutinante. Les classes syntaxiques les plus utilisées sont le nom, le verbe, l'adjectif et l'adverbe. Fondamentalement, les choses et les mots d'action sont la base de la morphologie amazigh et les classes les plus significatives sur lesquelles se concentrer, car d'autres peuvent être obtenues à partir d'eux. Nous allons présenter ci-dessous ces deux classes linguistiques amazighs :

1. Les noms :
   Il existe différents types de noms en amazigh : noms communs, adjectifs nominaux, emprunts arabes, noms propres et termes de parenté. Les noms communs amazighs portent des affixes de genre et de nombre. Le nom en amazigh se caractérise par le genre, le nombre et le statut. Le nom est soit masculin, soit féminin. Il est pluriel ou singulier : le pluriel commence par deux. Le nom est libre ou annexé. Les noms commençant par une voyelle sont généralement masculins, et ceux qui commencent par la consonne ⵜ sont généralement féminins :

a. ⴰⵔⴳⴰⵣ "homme", ⴰⵔⴱⴰ "garçon".
b. ⵜⴰⵎⵟⵟⵓⵜ "femme", ⵜⴰⵔⴱⴰⵜ "fille".

2. Les adjectifs :
   Les adjectifs nominaux ou adjectifs de nom sont semblables aux noms communs ; ils constituent morphologiquement une classe de mots qui a des modèles syntaxiques spéciaux. Les emprunts arabes utilisés en amazigh prennent généralement l'article arabe et ses variantes phonologiques, qui marquent le caractère définitif. Cet article obéit aux règles phono-tactiques de l'arabe et peut réaliser une consonne initiale géminée dans ce dernier est une articulation en lame de langue, tout comme en arabe.
3. Les verbes :
   L'aspect morphologique du verbe en amazigh repose fondamentalement sur l'appendice et la disposition. Quelques mots d'action sont induits par l'appendice (préfixes, postfixes) et différents mots d'action sont fondamentalement obtenus à partir de choses, soit à partir d'un mot d'action et d'une chose, soit à partir de deux mots d'action.

1.3 Etude bibliographique

1.3.1 Reconnaissance des caractères

Es Saady décrive la première version d'une base de données qui contient des caractères amazighs manuscrits (AMHCD). Consiste en 25 740 caractères manuscrits amazighs isolés et étiquetés isolés et étiquetés, produits par 60 auteurs. Cette base de données a été développée au laboratoire de l'IRF-SIC de l'université Ibn Zohr Ibn Zohr, Agadir, Maroc. Elle est destinée à l'entraînement et au et tester les systèmes de reconnaissance des caractères amazighs manuscrits [7].
Niharmine a réalisé l'extraction de caractéristiques en utilisant la technique de la direction du gradient. La nouveauté de leur approche est de générer de nouvelles caractéristiques et d'obtenir une meilleure précision en utilisant l'algorithme génétique qui produit de nouveaux vecteurs basés sur le paramètre de fitness. La phase de classification est réalisée à l'aide d'un réseau neuronal à anticipation [8].
El Wardani a exploité les avantages de l'approche d'apprentissage profond pour concevoir un système efficace de reconnaissance de l'écriture manuscrite amazigh. A cette fin, il a préparé leur jeu de données de 102 écrivains contenant chacun 33 caractères de l'IRCAM-Tifinagh. Inspiré par la base de données MNIST, l'ensemble des caractères est normalisé en taille et centré dans une image de taille fixe [9].
Oujaoura a proposé une approche pour construire un puissant système de reconnaissance automatique de caractères Tifinagh manuscrits isolés en utilisant une combinaison et une fusion de certaines méthodes d'extraction de caractéristiques. Les moments de Krawtchouk, de Chebyshev et de Hermite gaussien sont utilisés comme descripteurs dans la phase d'ex-

traction des caractéristiques en raison de leur invariance aux changements de translation, de rotation et d'échelle. Dans la phase de classification, le pouvoir discriminant des réseaux neuronaux, des SVM (Support Vector Machine) multi-classes, des classificateurs de plus proches voisins, et la nature générative des réseaux bayésiens, sont combinés ensemble afin de bénéficier de leur complémentarité [10].
SABIR a proposé un algorithme pour les alphabets Tifinagh comprend toutes les étapes traditionnelles d'un outil OCR : normalisation de l'image scannée, binarisation, segmentation, extraction de caractéristiques et un arbre de décision basé sur un réseau neuronal pour faire correspondre les alphabets Tifinagh lus avec le modèle entraîné. En outre, et afin de résoudre le problème des taux de confusion élevés, une matrice de confusion a été identifiée et, pour l'étape de classification, des classificateurs multiples ont été mis en œuvre pour l'ensemble identifié d'alphabets confus afin de sélectionner ceux qui sont reconnus. Sur la base de divers paramètres, l'algorithme proposé a été développé à l'aide de Matlab, et comparé à un outil commercial d'OCR, principalement le logiciel insight de Cognex. La méthode présentée a l'avantage d'offrir une précision de reconnaissance bien supérieure à celle d'un réseau neuronal et d'un OCR commercial qui effectue la même opération. En outre, l'utilisation de plusieurs classificateurs pour les alphabets présentant des taux de confusion élevés permet de gagner du temps [11].
NIHARMINE a met l'accent sur une méthodologie pour reconnaître les caractères Tifinagh en utilisant des caractéristiques de gradient de zonage avec une précision et un taux de reconnaissance élevés. La nouvelle méthodologie est basée sur la direction du gradient comme caractéristiques et sur les réseaux neuronaux artificiels comme classificateur [12].
El ayachi et Biniz ont construit une architecture très robuste et très rapide pour la reconnaissance des caractères manuscrits de la langue amazigh écrite en caractère Tifinagh. Ce travail comporte deux architectures optimisées, l'une pour reconnaître les images RVB et l'autre pour reconnaître les images binaires. [13].
Gounane a présenté une nouvelle approche de la reconnaissance des caractères Tifinagh en utilisant une combinaison de l'algorithme k-nearest neighbor et du modèle de langage bigramme. Après le prétraitement de l'image du texte, et la segmentation des mots, pour chaque caractère de l'image, l'algorithme k-NN propose des candidats pondérés par leur degré d'appartenance. Ensuite, il a utilisé le modèle de langage bigramme pour choisir la séquence de caractères la plus appropriée. [14].
Bennady a présenté un nouveau système de reconnaissance hors ligne basé sur des réseaux neuronaux convolutifs profonds (CNN). Il utilise les CNN pour extraire les caractéristiques des pixels bruts, ce qui rend la technique proposée plus flexible que les autres techniques conventionnelles qui nécessitent un ensemble d'étapes de prétraitement supplémentaires pour extraire les caractéristiques invariantes souhaitées. [15].

1.3.2 Reconnaissance de la parole

Zealouk a construit un système de reconnaissance automatique de la parole basé sur Sphinx4 qui permet de détecter les personnes qui ont des troubles de la voix. Ce projet de

recherche est réalisé en utilisant la langue amazigh afin de différencier les voix normales et pathologiques [16].
Hamidi a construit un système interactif de reconnaissance automatique de la parole amazigh indépendant du locuteur. Le système proposé offre une méthodologie pour extraire des données à distance d'une base de données en utilisant les technologies combinées de réponse vocale interactive (RVI) et de reconnaissance automatique de la parole (RAV). Il a décrit leur expérience dans la conception d'un système vocal interactif basé sur des modèles de Markov cachés (HMM), des modèles de mélange gaussien (GMM) et des coefficients spectraux de fréquence Mel (MFCC) basés sur dix premiers chiffres amazighs et six mots amazighs [17].
Satori a créé un système de reconnaissance automatique continue de la parole amazigh indépendant du locuteur. Le système conçu est basé sur les outils Sphinx de l'Université Carnegie Mellon. Dans la phase de l'entraînement et de test, un corpus interne Amazigh_Alphadigits a été utilisé. Ce corpus a été collecté dans le cadre de ce travail et consiste en la parole et sa transcription de 60 locuteurs marocains berbères (30 hommes et 30 femmes) natifs du berbère Tarifit. [18].
El ouahbi a étudié et réalisé un système de reconnaissance automatique de la parole, en utilisant un environnement totalement basé sur la langue AmazighTarifit. Dans ce cadre, il a d'abord construit un nouveau corpus de parole en amazightarifit, qui a été utilisé pour évaluer et faire tester les résultats de ce travail. En fait, son travail a deux objectifs : Le premier est de collecter un corpus de parole isolé de vocabulaire moyen, qui servira de corpus pour les chercheurs en parole amazigh. Le deuxième objectif est de développer un système ASR amazigh utilisant ce corpus de parole (187 mots isolés distincts). Le corpus de parole a été enregistré par 50 individus (25 hommes et 25 femmes) de langue maternelle amazightarifite. [19].

1.3.3 Marquage POS en amazigh

Outahajala a présenté les caractéristiques de la langue amazigh pour une annotation morphologique. En d'autres termes, il a vise à aborder la question de l'étiquetage de l'amazigh avec l'outil d'annotation multiniveau AnCoraPipe. Cet outil est adapté pour utiliser un jeu de balises spécifique pour annoter les corpus amazighs avec un nouveau système d'écriture défini [20].
Outahajala a présenté le premier marqueur POS en amazigh. Il a utilisé des approches d'apprentissage automatique supervisé de pointe pour construire leur modèle de marquage POS. [21].
Amri a présenté un algorithme d'apprentissage qui combine les modèles d'arbres de décision et les modèles HMM pour étiqueter les taguer les mots inconnus en amazigh. Dans le cas de méthodes statistiques telles que TreeTagger, cela aura également des avantages pratiques supplémentaires. Il a aussi présenté la création d'un corpus étiqueté POS et l'évaluation de TreeTagger sur du texte amazigh. [22].
Outahajala a présenté les résultats de son étiquetage POS basé sur deux techniques de pointe d'étiquetage de séquences, à savoir les champs aléatoires conditionnels et les ma-

chines vectorielles de support, en utilisant un petit corpus annoté manuellement de seulement 20 000 tokens. Comme la création de données étiquetées est une tâche très longue et que l'obtention de données non étiquetées l'est moins, il a décidé de rassembler un ensemble de données non étiquetées de la langue amazigh que nous avons prétraitées et tokenisées. Son travail a également pour but d'aborder l'utilisation de techniques semi-supervisées pour améliorer la précision du marquage POS. Un algorithme d'auto-formation adapté, combinant la mesure de confiance avec une fonction de mots hors vocabulaire pour sélectionner les données pour l'auto-formation, a été utilisé. [23].
Amri a développé des marqueurs POS pour la langue amazigh, une langue moins privilégiée, en utilisant le champ aléatoire conditionnel (CRF), la machine à vecteur de support (SVM) et le système TreeTagger. Il a annoté manuellement environ 75000 tokens, recueillis dans des textes écrits, avec un ensemble de 28 balises POS définies pour la langue amazigh. Les marqueurs POS utilisent les différentes caractéristiques contextuelles et orthographiques au niveau des mots. Ces caractéristiques sont indépendantes de la langue et applicables à d'autres langues également. Les étiqueteurs POS ont été entraînés et testés avec les mêmes corpus. [24].
Outhajala a créé un outil de tokenizer et un nouvel étiqueteur de parties de la parole (POS) basé sur un nouvel ensemble d'étiquettes amazighs (AMTS) composé de 28 étiquettes. Il a entraîné deux modèles de classification de séquences en utilisant des Machines à Vecteur de Support (SVMs) et des Champs Aléatoires Conditionnels (CRFs) pour construire un tokenizer et un tagger POS pour la langue amazigh. Il a utilisé la technique du 10-fold pour évaluer et valider notre approche. [25].
Amri a implémenté un nouvel outil pour l'étiquetage de la langue amazigh en utilisant des modèles de Markov et des arbres de décision. Après avoir étudié différentes approches et problèmes de marquage des parties du discours, il a implémenté un système de marquage basé sur TreeTagger - un outil générique de marquage stochastique, très populaire pour son efficacité. Il a rassemblé un corpus de travail, suffisamment large pour assurer une couverture linguistique générale. Ce corpus a été utilisé pour exécuter le processus de tokenisation, ainsi que pour entraîner TreeTagger. Ensuite, il a effectué une évaluation simple des résultats sur un petit corpus de test. [26].

1.4 Traitement de la langue naturel

Le traitement du langage naturel est l'un des domaines de la programmation où le langage naturel est traité par le logiciel. Il a de nombreuses applications comme l'analyse des sentiments, la traduction linguistique, la détection des fausses nouvelles, la détection des erreurs grammaticales, etc. L'entrée dans le traitement du langage naturel est un texte. La collecte de données pour ce texte provient de nombreuses sources. Cela nécessite beaucoup de nettoyage et de traitement avant que les données puissent être utilisées pour l'analyse. Voici quelques-unes des méthodes de traitement des données dans le traitement du langage naturel :

1.4.1 Tokénisation

1.4.1.1 Définition

La tokénisation consiste à décomposer le texte brut en petits morceaux. La tokénisation décompose le texte brut en mots, phrases appelés tokens. Ces tokens aident à comprendre le contexte ou à développer le modèle pour le NLP. La tokénisation permet d'interpréter le sens du texte en analysant la séquence des mots. Il existe différentes méthodes et bibliothèques disponibles pour effectuer la tokénisation. NLTK, Gensim, Keras sont quelques-unes des bibliothèques qui peuvent être utilisées pour accomplir cette tâche. La tokénisation peut être effectuée pour séparer des mots ou des phrases. Si le texte est divisé en mots à l'aide d'une technique de séparation, on parle de tokénisation des mots et la même séparation effectuée pour les phrases est appelée tokénisation des phrases. Les mots d'arrêt sont les mots du texte qui n'ajoutent aucune signification à la phrase et dont la suppression n'affectera pas le traitement du texte dans le but défini. Ils sont supprimés du vocabulaire pour réduire le bruit et la dimension de l'ensemble des caractéristiques. Il existe différentes techniques de tokénisation qui peut être appliquées en fonction de la langue et de l'objectif de la modélisation. Vous trouverez ci-dessous quelques-unes des techniques de tokénisation utilisées en NLP.

1.4.1.2 Raisons de la tokénisation

Comme les tokens sont les éléments constitutifs du langage naturel, la façon la plus courante de traiter le texte brut se fait au niveau des tokens. Par exemple, les modèles basés sur les transformateurs, les architectures d'apprentissage profond de pointe dans le traitement du langage naturel traitent le texte brut au niveau du token. De même, les architectures d'apprentissage profond les plus populaires pour le traitement automatique des langues, telles que RNN, GRU et LSTM, traitent également le texte brut au niveau du token.

1.4.1.3 Techniques de la tokénisation

1. Tokénisation de l'espace blanc
   Il s'agit de la technique de tokénisation la plus simple. Étant donné une phrase ou un paragraphe, elle permet la tokénisation en mots en divisant l'entrée chaque fois qu'un espace blanc est rencontré. C'est la technique de tokénisation la plus rapide, mais elle ne fonctionne que pour les langues dans lesquelles l'espace blanc divise la phrase en mots significatifs. Exemple : l'anglais.
2. Tokenisation basée sur des règles
   Dans cette technique, un ensemble de règles est créé pour le problème spécifique. La tokénisation est effectuée sur la base de ces règles. Par exemple, créer des règles basées sur la grammaire d'une langue particulière.
3. Expression régulière Tokéniser
   Cette technique utilise l'expression régulière pour contrôler la tokénisation du texte

en tokens. Les expressions régulières peuvent être simples ou complexes et parfois difficiles à comprendre. Cette technique doit être privilégiée lorsque les méthodes ci-dessus ne répondent pas à l'objectif recherché. Il s'agit d'un tokenizer basé sur des règles.

4. Tokénisation de Penn TreeBank
   La banque d'arbres est un corpus créé qui donne l'annotation sémantique et syntaxique du langage. La Penn Treebank est l'une des plus grandes banques d'arbres qui a été publiée. Cette technique de tokénisation sépare la ponctuation, les clitiques (mots qui apparaissent avec d'autres mots comme I'm, don't) et les mots avec trait d'union.

5. Tokéniseur Spacy
   Il s'agit d'une technique moderne de tokénisation, plus rapide et facilement personnalisable. Elle offre la possibilité de spécifier des tokens spéciaux qui ne doivent pas être segmentés ou qui doivent être segmentés en utilisant des règles spéciales. Supposons que vous vouliez garder $ comme un jeton séparé, il a la priorité sur les autres opérations de tokénisation. item Tokénizer Moses
   Il s'agit d'un tokenizer avancé, disponible avant l'introduction de Spacy. Il s'agit essentiellement d'un ensemble de logiques complexes de normalisation et de segmentation qui fonctionne très bien pour un langage structuré comme l'anglais.

6. Tokénisation des sous-mots
   Cette tokénisation est très utile pour des applications spécifiques où les sous-mots sont importants. Dans cette technique, les mots les plus fréquemment utilisés reçoivent un identifiant unique et les mots moins fréquents sont divisés en sous-mots qui représentent le mieux le sens indépendamment. Par exemple, si le mot "few" apparaît fréquemment dans le texte, un identifiant unique lui sera attribué, tandis que "fewer" et "fewest", qui sont des mots rares et moins fréquents dans le texte, seront divisés en sous-mots comme "few", "er" et "est". Cela aide le modèle de langage à ne pas apprendre fewer et fewest comme deux mots distincts. Cela permet d'identifier les mots inconnus dans l'ensemble de données pendant la formation. Il existe différents types de tokénisation des sous-mots, présentés ci-dessous. L'encodage par paire d'octets et WordPiece seront brièvement abordés.

7. Codage par paire d'octets (BPE)
   Cette technique est basée sur les concepts de la théorie de l'information et de la compression. Le BPE utilise le codage de Huffman pour la tokénisation, ce qui signifie qu'il utilise plus d'encastrement ou de symboles pour représenter les mots moins fréquents et moins de symboles ou d'encastrement pour les mots plus fréquents. La tokénisation BPE est une technique ascendante de tokénisation des sous-mots. Les étapes de l'algorithme BPE sont présentées ci-dessous.
   (a) Cette technique commence par diviser les mots d'entrée en caractères unicodes uniques et chacun d'entre eux correspond à un symbole dans le vocabulaire final.
   (b) Trouver la paire de symboles la plus fréquente dans le vocabulaire actuel.

(c) Ajoutez-la au vocabulaire et la taille du vocabulaire augmente d'une unité.

(d) Répétez les étapes (b) et (c) jusqu'à ce que le nombre défini de jetons soit construit ou qu'aucune nouvelle combinaison de symboles n'existe avec la fréquence requise.

8. WordPiece
WordPiece est similaire aux techniques BPE, à l'exception de la manière dont le nouveau token est ajouté au vocabulaire. BPE considère le token avec la paire de symboles la plus fréquente pour l'intégrer au vocabulaire. Alors que WordPiece prend également en compte la fréquence des symboles individuels et se base sur le nombre ci-dessous pour les intégrer au vocabulaire.

$$Compte(x, y) = frequence(x, y)/frequence(x) * frequence(y) \tag{1.1}$$

La paire de symboles ayant le compte maximum sera considérée comme étant intégrée au vocabulaire. Ainsi, il permet d'inclure les tokens rares dans le vocabulaire par rapport à BPE.

1.4.2 Stemming et lemmatisation

1.4.2.1 Définition

Stemming et la lemmatisation sont des techniques de normalisation de texte dans le domaine du traitement du langage naturel qui sont utilisées pour préparer les textes, les mots et les documents pour un traitement ultérieur, sont aussi des méthodes utilisées par les moteurs de recherche et les robots de conversation pour analyser le sens d'un mot. Stemming utilise le radical du mot, tandis que la lemmatisation utilise le contexte dans lequel le mot est utilisé. Nous reviendrons plus tard sur des explications et des exemples plus détaillés. Lorsque nous effectuons une recherche, nous voulons trouver des résultats pertinents non seulement pour l'expression exacte que nous avons tapé dans la barre de recherche, mais aussi pour les autres formes possibles des mots que nous avons utilisé.
Dans le traitement du langage naturel, on souhaite qu'un programme reconnaisse que les mots "kick" et "kicked" ne sont que des temps différents du même verbe. Il peut s'agir du concept de réduction des différents types d'un mot à une racine centrale.

1.4.2.2 Stemming

Stemming est une technique de normalisation simple, le plus souvent mise en œuvre sous la forme d'une série de règles qui sont appliquées progressivement à un mot pour produire une forme normalisée.
Les algorithmes fonctionnent en coupant la fin ou le début du mot, en prenant en compte une liste de préfixes et de suffixes communs que l'on peut trouver dans un mot infléchi. Ce découpage sans discernement peut réussir dans certaines occasions, mais pas toujours, et c'est pourquoi nous affirmons que cette approche présente certaines limites.

Les règles de normalisation varient d'une langue à l'autre et reflètent la structure morphologique de la langue en question. Par exemple, pour l'anglais, une règle possible pourrait être de supprimer le "s" à la fin d'un mot pour le transformer en sa forme singulière. Il est important de garder à l'esprit qu'il n'est pas nécessaire que le mot normalisé soit valide, mais seulement que les variations du même mot correspondent au même radical.
Nous pouvons voir ce phénomène en action en expérimentant avec le dérameur Porter, un algorithme de dérameur largement utilisé en anglais. En l'utilisant sur les mots "engine" ou "engines", on obtient "engin". Bien qu'il ne s'agisse pas d'un mot anglais valide, il nous importe seulement que les deux mots correspondent au même radical.
Bien que cela puisse sembler contre-intuitif au premier abord, cela ne représente pas un problème. L'extraction des racines est principalement utilisée pour indexer les documents dans un moteur de recherche. Ces racines, qui peuvent être des mots non valides, ne sont donc traitées qu'en interne dans les documents de recherche et ne sont jamais affichées à l'utilisateur.

1. Porter Stemmer
   The Porter stemmer algorithme est un outil largement utilisé, mais essentiel pour le traitement du langage naturel dans le domaine de l'accès à l'information. Stemming est utilisé pour supprimer les mots qui ajoutent les terminaisons morphologiques et diacritiques finales des mots en anglais à leur forme racine afin d'extraire la racine du mot, c'est-à-dire ce que l'on appelle la tige/racine dans l'étape de traitement primaire du texte. En d'autres termes, il s'agit d'un processus linguistique qui extrait simplement la partie principale qui peut être proche de la racine relative et apparentée. La classification des textes est une tâche importante pour extraire des informations pertinentes d'un grand volume de données.
   L'un des outils de stemming les plus courants - et les plus efficaces est l'algorithme de Porter, développé par Martin Porter en 1980. L'algorithme utilise cinq phases de réduction des mots, chacune ayant son propre ensemble de règles de correspondance (Polus and Abbas, 2021).
2. Snowball Stemmer
   Snowball. En 1996, Martin Porter a développé le stemmer Snowball pour l'anglais [27]. Il est devenu de loin le démembrement le plus utilisé pour l'anglais. L'équipe Snowball a développé des stemmers pour de nombreuses langues européennes, qui sont inclus dans d'importantes boîtes à outils de traitement du langage naturel telles que NLTK pour Python ou Lingua::Stem pour Perl. Snowball anglaise définit deux régions R1 et R2, où R1 "est la région après la première non-voyelle suivant une voyelle, ou est la région nulle à la fin du mot s'il n'y a pas de telle non-voyelle" et R2 est défini de la même manière, avec la différence que la définition est appliquée à l'intérieur de R1. Après avoir défini R1 et R2, Snowball supprime un certain nombre de suffixes s'ils apparaissent dans R1 ou R2. Il ne le fait pas de manière récursive mais plutôt en trois étapes, dans chacune desquelles au plus un suffixe peut être supprimé. Les deux premières étapes suppriment les suffixes les plus courants comme "ism" ou "ist", tandis que la troisième étape supprime les suffixes dérivés,

par exemple "ry" ou "al", qui sont assez rares [28].

1.4.2.3 Lemmatisation

La lemmatisation, quant à elle, prend en compte l'analyse morphologique des mots. Pour ce faire, il est nécessaire de disposer de dictionnaires détaillés que l'algorithme peut consulter pour relier la forme à son lemme. Là encore, vous pouvez voir comment cela fonctionne avec les mêmes exemples de mots. Contrairement à stemming, la lemmatisation va au-delà de la réduction des mots et prend en compte l'ensemble du vocabulaire d'une langue pour appliquer une analyse morphologique aux mots. Le lemme de " was " est " be " et le lemme de " mice " est " mouse".
La lemmatisation est généralement considérée comme beaucoup plus informative qu'un simple stemmer, c'est pourquoi Spacy a choisi de ne proposer que la lemmatisation au lieu de stemmer.
La lemmatisation examine le texte environnant pour déterminer la partie du discours d'un mot donné, elle ne catégorise pas les phrases. Pour ce faire, les algorithmes de lemmatisation dépendent de la disponibilité d'informations sur la partie du discours des mots d'entrée, car différentes règles de normalisation doivent être appliquées selon que le mot est un nom, un verbe, un adjectif ou autre.
Prenons l'exemple du mot "following". Selon le contexte, il peut s'agir d'un nom (par exemple, " he has a huge following "), d'un verbe (par exemple, " he started following the rabbit ") ou d'un adjectif (par exemple, " the following day "). S'il s'agit d'un adjectif ou d'un verbe, la lemmatisation renverra "following", tandis que s'il s'agit d'un verbe, elle renverra "follow". Un algorithme de stemming n'en sera pas conscient et supprimera le suffixe "ing" pour obtenir "follow" dans tous les cas. Les moteurs de recherche peuvent utiliser la lemmatisation pour indexer les documents de la même manière que le stemming. Cependant, étant donné sa plus grande précision, elle est utilisée dans une variété de tâches NLP où il est indispensable d'avoir des mots valides, par exemple pour la désambiguïsation du sens des mots.
Le tableau 1.1 montre la différence entre stemming et la lemmatisation sur une phrase anglaise réelle :

Original	**Stemme**	**Lemme**
New	New	New
York	York	York
is	is	be
the	the	the
most	most	most
densely	dens	densely
populated	popul	populated
city	citi	city
in	in	in

Suite à la page suivante

Table 1.1 – *Suite de la page précédente*

Original	**Stemme**	**Lemme**
the	the	the
United	Unite	United
States	State	States

TABLE 1.1 : Exemple de stemming et lemmatisation

1.5 Approches de vectorisation

1.5.1 Définition

La vectorisation est le jargon d'une approche classique consistant à convertir les données d'entrée de leur format brut (c'est-à-dire du texte) en vecteurs de nombres réels, qui est le format pris en charge par les modèles ML. Cette approche existe depuis que les ordinateurs ont été construits, elle a fonctionné à merveille dans divers domaines, et elle est maintenant utilisée dans le NLP. En apprentissage automatique, la vectorisation est une étape de l'extraction de caractéristiques. L'idée est d'extraire du texte des caractéristiques distinctes sur lesquelles le modèle pourra s'entraîner, en convertissant le texte en vecteurs numériques. Il existe de nombreuses façons d'effectuer la vectorisation, comme nous le verrons bientôt, allant des caractéristiques naïves d'occurrence de termes binaires aux représentations de caractéristiques avancées tenant compte du contexte. Selon le cas d'utilisation et le modèle, n'importe laquelle d'entre elles peut être capable d'accomplir la tâche requise. Découvrons certaines de ces techniques et voyons comment nous pouvons les utiliser.

1.5.2 Techniques de vectorisation

1.5.2.1 Sac de mots

Le modèle du sac de mots convertit le texte en vecteurs de longueur fixe en comptant combien de fois chaque mot apparaît [29]. Illustrons cela par un exemple. Considérons que nous avons les phrases suivantes :

— Text processing is necessary.
— Text processing is necessary and important.
— Text processing is easy

Nous appellerons chacune des phrases ci-dessus des documents. Si nous retirons les mots uniques de toutes ces phrases, le vocabulaire sera composé de ces 7 mots : 'Text', 'processing', 'is', 'necessary', 'and', 'important, 'easy' (voir le taleau 1.2).

Document	**Text**	**processing**	**is**	**necessary**	**and**	**important**	**easy**
1	1	1	1	1	0	0	0
2	1	1	1	1	1	1	0
3	1	1	1	0	0	0	1

TABLE 1.2 : Exemple de vectorisation utilisant sac de mots

Pour réaliser le sac de mots, nous devrons simplement compter le nombre de fois où chaque mot apparaît dans chacun des documents.
Par conséquent, nous avons les vecteurs suivants pour chacun des documents de longueur fixe 7 :
Document 1 : [1,1,1,1,0,0,0].
Document 2 : [1,1,1,1,1,1,0].
Document 3 : [1,1,1,0,0,0,1].
Cette technique de vectorisation a des limites par exemple si nous déployons le sac de mots pour générer des vecteurs pour des documents de grande taille, les vecteurs seront de grande taille et auront également trop de valeurs nulles, ce qui conduira à la création de vecteurs épars.
Le sac de mots n'apporte aucune information sur le sens du texte. Pa exemple, si nous considérons ces deux phrases " Text processing is easy but tedious." et " Text processing is tedious but easy." un modèle de sac de mots créerait les mêmes vecteurs pour les deux, même si elles ont des significations différentes.

1.5.2.2 Term Frequency Inverse Document Frequency (TFIDF)

TFIDF fonctionne en augmentant proportionnellement le nombre de fois qu'un mot apparaît dans le document, mais est contrebalancé par le nombre de documents dans lesquels il est présent [30]. Par conséquent, des mots tels que "this", "are", etc., qui sont couramment présents dans tous les documents, n'ont pas un rang très élevé. Cependant, un mot qui est présent un trop grand nombre de fois dans un petit nombre de documents se verra attribuer un rang plus élevé car il peut être indicatif du contexte du document.

1. fréquence des termes : La fréquence des termes est définie comme le nombre de fois qu'un mot (i) apparaît dans un document (j) divisé par le nombre total de mots dans le document.
$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}} \tag{1.2}$$
2. La fréquence inverse des documents : La fréquence inverse des documents correspond au logarithme du nombre total de documents divisé par le nombre de documents qui contiennent le mot. Le logarithme est ajouté pour atténuer l'importance d'une valeur très élevée d'IDF.
$$idf(w) = log\left(\frac{N}{df}\right) \tag{1.3}$$
Le TFIDF est calculé en multipliant la fréquence des termes par la fréquence inverse des documents.
$$w_{i,j} = tf_{i,j} \times log\left(\frac{N}{df_i}\right) \tag{1.4}$$

Voyons maintenant une illustration de TFIDF dans les phrases suivantes, que nous appelons des documents.
Document 1 : Text processing is necessary.
Document 2 : Text processing is necessary and important.

Mot	TF		IDF	TFIDF	
	Doc1	Doc2		Doc1	Doc2
Text	1/4	1/6	Log(2/2)=0	0	0
processing	1/4	1/6	Log(2/2)=0	0	0
is	1/4	1/6	Log(2/2)=0	0	0
necessary	1/4	1/6	Log(2/2)=0	0	0
and	0/4	1/6	Log(2/1)=0.3	0	0.05
important	0/4	1/6	Log(2/1)=0.3	0	0.05

TABLE 1.3 – :Exemple de vectorisation utilisant TFIDF

Le tableau ci-dessus montre comment le TFIDF de certains mots est nul et d'autres non, en fonction de leur fréquence dans le document et dans tous les documents.
La limite de TFIDF est encore une fois que cette vectorisation n'aide pas à apporter la signification contextuelle des mots car elle est juste basée sur la fréquence.

1.5.2.3 Word2Vec

Word2Vec - Word representations in Vector Space fondé par [31] est un groupe d'une équipe de recherche de Google qui a développé ce modèle en 2013. Word2vec est un groupe de modèles apparentés qui sont utilisés pour produire ce qu'on appelle des encastrements de mots. Ces modèles sont des réseaux neuronaux à deux couches, peu profonds, qui sont formés pour reconstruire les contextes linguistiques des mots.
La plupart des systèmes NLP traitent les mots comme des unités atomiques. Les systèmes existants sont limités par le fait qu'il n'y a pas de notion de similarité entre les mots. De plus, le système fonctionne pour des données plus petites et plus simples et est plus performant sur des données moins nombreuses, soit quelques milliards de données ou moins. Afin de s'entraîner sur un ensemble de données plus important avec des modèles complexes, les techniques modernes utilisent une architecture de réseau neuronal pour entraîner des modèles de données complexes et sont plus performantes sur des ensembles de données énormes avec des milliards de mots et un vocabulaire de millions de mots.
Cette technique permet de mesurer la qualité des représentations vectorielles résultantes. Elle fonctionne avec des mots similaires qui ont tendance à se rapprocher de mots qui peuvent avoir plusieurs degrés de similarité. Après de s'entraîner, les modèles word2vec peuvent être utilisés pour associer chaque mot à un vecteur composé généralement de plusieurs centaines d'éléments, qui représentent la relation de ce mot avec d'autres mots. Ce vecteur constitue la couche cachée du réseau neuronal. Word2vec s'appuie soit sur

les skip-grams, soit sur les sacs continus de mots (CBOW) pour créer des encastrements neuronaux de mots.

1.6 Approches de classification

1.6.1 Définition

La classification est le processus qui consiste à reconnaître, comprendre et regrouper des idées et des objets dans des catégories ou des "sous-populations" prédéfinies. À l'aide d'ensembles de données d'entraînement pré-catégorisés, les programmes d'apprentissage automatique utilisent une variété d'algorithmes pour classer les ensembles de données futurs dans des catégories.
Les algorithmes de classification de l'apprentissage automatique utilisent les données d'apprentissage en entrée pour prédire la probabilité que les données suivantes entrent dans l'une des catégories prédéterminées. L'une des utilisations les plus courantes de la classification est le filtrage des e-mails en "spam" ou "non-spam". En bref, la classification est une forme de "reconnaissance des formes", les algorithmes de classification étant appliqués aux données de formation pour trouver la même forme (mots ou sentiments similaires, séquences de chiffres, etc.) dans les futurs ensembles de données.
À l'aide d'algorithmes de classification, dont nous parlerons plus en détail ci-dessous, les logiciels d'analyse de texte peuvent effectuer des tâches telles que l'analyse des sentiments basée sur les aspects, afin de classer les textes non structurés par sujet et par polarité d'opinion (positive, négative, neutre, etc.). Essayez ce classificateur de sentiments pré-entraîné pour comprendre comment les algorithmes de classification fonctionnent en pratique, puis poursuivez votre lecture pour en savoir plus sur les différents types d'algorithmes de classification.

1.6.2 Techniques d'apprentissage automatique

1.6.2.1 Logistic Regression

La régression logistique est un calcul utilisé pour prédire un résultat binaire : soit quelque chose se produit, soit rien ne se produit. Cela peut se présenter sous la forme Oui/Non, Réussite/Echec, Vivant/Mort, etc. Les variables indépendantes sont analysées pour déterminer l'issue binaire et les résultats sont classés dans l'une des deux catégories suivantes. Les variables indépendantes peuvent être catégoriques ou numériques, mais la variable dépendante est toujours catégorique. Écrit comme ceci :

$$P\left(Y = 1 \backslash X\right) \text{où} P\left(Y = 0 \backslash X\right) \tag{1.5}$$

Elle calcule la probabilité de la variable dépendante Y, étant donné la variable indépendante X.
On peut l'utiliser pour calculer la probabilité qu'un mot ait une connotation positive ou négative (0, 1, ou sur une échelle entre les deux). On peut aussi l'utiliser pour déterminer

l'objet contenu dans une photo (arbre, fleur, herbe, etc.), en attribuant à chaque objet une probabilité comprise entre 0 et 1.

1.6.2.2 Naive Bayes

Naive Bayes calcule la possibilité qu'un point de données appartienne ou non à une certaine catégorie. Dans l'analyse de texte, il peut être utilisé pour classer des mots ou des phrases comme appartenant à une "étiquette" prédéfinie (classification) ou non.
Pour décider si une phrase doit être étiquetée par une "étiquette" prédéfinie ou non, il doit faire des calculs :

$$P\left(A \backslash B\right) = \frac{P\left(B \backslash A\right) \times P\left(A\right)}{P\left(B\right)} \tag{1.6}$$

Où $P\left(A \backslash B\right)$la probabilité de A, si B est vrai, est égale à la probabilité de B, si A est vrai, multipliée par la probabilité de A étant vrai, divisée par la probabilité de B étant vrai.

1.6.2.3 K-nearest Neighbors

K-nearest neighbors (k-NN) est un algorithme de reconnaissance des formes qui utilise des ensembles de données de formation pour trouver les k plus proches parents dans les exemples futurs.
Lorsque le k-NN est utilisé dans la classification, vous calculez pour placer les données dans la catégorie de leur plus proche voisin. Si k = 1, elle sera placée dans la classe la plus proche de 1. K est classé par une pluralité de ses voisins.

1.6.2.4 Arbre de décision

Un arbre de décision est un algorithme d'apprentissage supervisé qui est parfait pour les problèmes de classification, car il est capable d'ordonner les classes à un niveau précis. Il fonctionne comme un organigramme, séparant les points de données en deux catégories similaires à la fois, du "tronc de l'arbre" aux "branches", puis aux "feuilles", où les catégories deviennent plus finement similaires. Cela crée des catégories dans les catégories, permettant une classification organique avec une supervision humaine limitée.

1.6.2.5 Random Forest

L'algorithme de la forêt aléatoire est une extension de l'arbre de décision, en ce sens que vous construisez d'abord une multitude d'arbres de décision avec des données d'entraînement, puis vous adaptez vos nouvelles données à l'un des arbres en tant que "forêt aléatoire".
Il établit essentiellement la moyenne de vos données pour les relier à l'arbre le plus proche sur l'échelle des données. Les modèles de forêt aléatoire sont utiles car ils remédient au

problème de l'arbre de décision qui consiste à "forcer" inutilement les points de données dans une catégorie.

1.6.2.6 Support Vector Machine

"Support Vector Machine" (SVM) est un algorithme d'apprentissage automatique supervisé qui peut être utilisé pour les problèmes de classification ou de régression. Cependant, il est surtout utilisé dans les problèmes de classification. Dans l'algorithme SVM, il représente chaque élément de données comme un point dans un espace à n dimensions (où n est le nombre de caractéristiques que vous avez), la valeur de chaque caractéristique étant la valeur d'une coordonnée particulière. Ensuite, il effectue la classification en trouvant l'hyperplan qui différencie très bien les deux classes.

1.6.2.7 Champ aléatoire conditionnel

Les CRFs [32] sont des modèles graphiques non dirigés qui cherchent à représenter la distribution des probabilités d'annotation (ou étiquettes ou labels) et conditionnellement à l'observation x d'exemples étiquetés (exemples avec étiquettes attendues). Ce sont donc des modèles obtenus par apprentissage supervisé, très utilisés notamment dans les problèmes d'étiquetage de séquences. Dans le cas séquentiel, c'est-à-dire l'étiquetage des observations xi par des étiquettes yi, la fonction potentielle au cœur des CRF s'écrit :

$$p(x \mid y) = \frac{1}{Z(x)} \exp\left(\sum_{k=1}^{k_1}\sum_{i=1}^{n} \lambda_k f_k\left(y_i, x\right) \sum_{k=1}^{k_2}\sum_{i=1}^{n} \mu_k g_k\left(y_{i-1}, y_i, x\right)\right) \tag{1.7}$$

Où :

- $Z(x)$ un facteur de normalisation ;
- Les fonctions caractéristiques locales et globales f et g : les fonctions f caractérisent les relations locales entre l'étiquette courante en position i et les observations ; les fonctions g caractérisent les transitions entre les noeuds du graphe, c'est-à-dire entre chaque paire d'étiquettes i et $i-1$, et la séquence d'observations.
- Les valeurs k_1 ,k_2 et n sont respectivement le nombre de fonctions f le nombre de fonctions g, et la taille de la séquence d'étiquettes à prédire. Les fonctions f et g sont généralement des fonctions binaires qui vérifient une certaine combinaison d'étiquettes et d'attributs décrivant les observations et appliquées à chaque position de la séquence.

1.6.3 Techniques d'apprentissage profond

1.6.3.1 Recurrent Neural Network (RNN)

Un réseau neuronal récurrent (RNN) est une classe de réseaux neuronaux artificiels où les associations entre les nœuds structurent un schéma coordonné le long d'une disposition transitoire. Cela lui permet de présenter un comportement unique et fugace. Comme obtenu à partir du réseau neuronal à anticipation, les RNN peuvent utiliser leur état interne

(mémoire) pour traiter des groupements de sources d'information de longueur variable. Cela les rend matériels pour différentes tâches, par exemple, la reconnaissance d'écriture non segmentée et connectée ou la reconnaissance vocale [33].

L'équation suivante montre les RNNs déroulés, le rendement h_t étape de temps spécifique t.

$$h_t = \sum_{t=0}^{T} X_t W_t + U h_{t-1} + b \tag{1.8}$$

h_{t-1} est la sortie précédente et X_t est l'entrée actuelle et W_t représente le poids au pas de temps t, également U représente le poids associé à la sortie h_{t-1} et b représente les termes de biais.

1.6.3.2 Bi-Directional Recurrent Neural Network (RNN)

Dans un RNN bidirectionnel, nous considérons 2 séquences distinctes. L'une de droite à gauche et l'autre dans l'ordre inverse. Mais, maintenant vient la question de savoir comment combiner les deux RNN ensemble. Regardez la figure 1.2 ci-dessous pour bien comprendre. Considérons la séquence de mots « ⵉⵅⴰ ⵜⵓⵅⵅⵏⴰⵓ ⴰⵍⴰⵣⵢ ». La couche avant alimenterait la séquence comme telle. Mais, la couche arrière alimentera la séquence dans l'ordre inverse « ⴰⵍⴰⵣⵢ ⵜⵓⵅⵅⵏⴰⵓ ⵉⵅⴰ ». Maintenant, les sorties seraient générées en concaténant les séquences de mots à chaque fois et en générant des poids en conséquence. Cette méthode peut également être utilisée pour les problèmes de marquage POS.

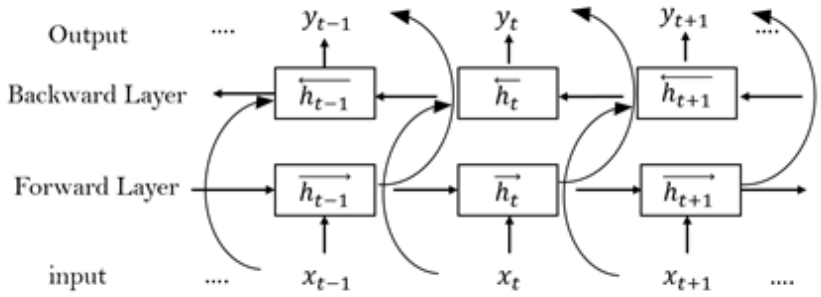

FIGURE 1.2 – Bi-Directional RNN pour la séquence de mots

1.6.3.3 Long short-Term Memory Networks (LSTM)

[34] a proposé la cellule de mémoire à long terme (LSTM) pour sa position favorable dans l'union plus rapide et l'identification et la mémorisation des dépendances à long terme. Nonobstant l'état à court terme, la cellule LSTM ht a un état à long terme. Les deux états sont des éléments de la sortie actuelle et des états passés tels qu'ils sont résumés dans les

fonctions C_t correspondantes. Le rendement de la cellule n'est que l'état à court terme.

$$\begin{cases} f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}C_{t-1}\right) \\ i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}C_{t-1}\right) \\ o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}C_{t-1}\right) \\ C_t = f_t \odot C_{t-1} + i_t \odot \tanh\left(W_{xc}x_t + W_{hc}h_{t-1}\right) \\ h_t = o_t \odot \tanh\left(C_t\right) \end{cases} \tag{1.9}$$

x est le vecteur d'entrée, h est le vecteur de sortie et c est l'état de la cellule. L'indice t représente l'état actuel et $t-1$ est le dernier état. σ est la function sigmoid , $\odot$ est le produit de Hadamard et W représente des paramètres indéterminés. Dans l'équation (1.9), f est la forget gate qui décide quelles informations doivent être éliminées de l'état de la cellule. i est input gate qui décide quelle information doit être stockée dans l'état de la cellule. O C'est output gate qui décide de l'information à transmettre. [35].

1.6.3.4 Gated Recurrent Units (GRU)

Les unités récurrentes à grille ont été proposées par [36]. Il s'agit d'une version simplifiée du LSTM qui nécessite moins de temps d'apprentissage et améliore les performances du réseau. Le fonctionnement d'une cellule GRU est similaire à celui d'une cellule LSTM mais la cellule GRU utilise un état caché qui fusionne la porte d'oubli et la porte d'entrée en une seule porte de mise à jour. De plus, elle combine l'état de la cellule et l'état caché en un seul état. Par conséquent, le nombre total de portes dans la cellule GRU est la moitié (portes de mise à jour et de réinitialisation) du nombre total de portes dans le LSTM.

- Dans GRU, il n'y a pas d'unité de mémoire explicite. L'unité de mémoire est combinée avec le réseau.
- Il n'y a pas de porte d'oubli et de porte de mise à jour dans GRU. Elles sont toutes les deux combinées ensemble et ainsi le nombre de paramètres est réduit.
- Lorsque l'on compare GRU avec LSTM, il fonctionne bien mais peut avoir une légère baisse de la précision. Mais nous avons toujours moins de paramètres à former, ce qui rend son utilisation avantageuse.

$$\begin{cases} r_t = \sigma\left(W_r x_t + U_r h_{t-1}\right) \\ z_t = \sigma\left(W_z x_t + U_z h_{t-1}\right) \\ \tilde{h}_t = \tanh\left(W_h x_t + U\left(r_t \odot h_{t-1}\right)\right) \\ h_t = \left(1 - z_t\right) h_{t-1} + z_t \widetilde{h_t} \end{cases} \tag{1.10}$$

1.6.3.5 Bidirectional Long Short Term Memory (Bi-LSTM)

Un Bidirectional LSTM ou biLSTM est une augmentation des modèles LSTM représentés dans lesquels deux LSTM sont appliqués aux informations d'information. Dans le premier cycle, un LSTM est appliqué au groupement d'informations (c'est-à-dire la couche avant). Dans le deuxième cycle, le type opposé de succession d'informations est pris en

charge dans le modèle LSTM (c'est-à-dire dans la couche inverse). Le fait d'appliquer deux fois le LSTM permet d'améliorer l'apprentissage des conditions de longue durée et, par conséquent, d'améliorer l'exactitude du modèle .
Le Bi-LSTM peut saisir le cadre autour des mots sources jusqu'à des arrangements extrêmement longs dans les deux sens (passé et direct) [37]. En outre, il n'a pas besoin de s'embarrasser de faits saillants créés à la main pour fonctionner admirablement. Ces attributs en font un outil approprié pour l'étiquetage POS de l'amazigh. La structure Bi-LSTM diffère du RNN exemplaire en ce qu'elle ajoute une cellule de mémoire à la conception de l'organisation neuronale qui détermine comment se souvenir des données sur un groupe pendant de longues périodes (voir la figure 1.3).

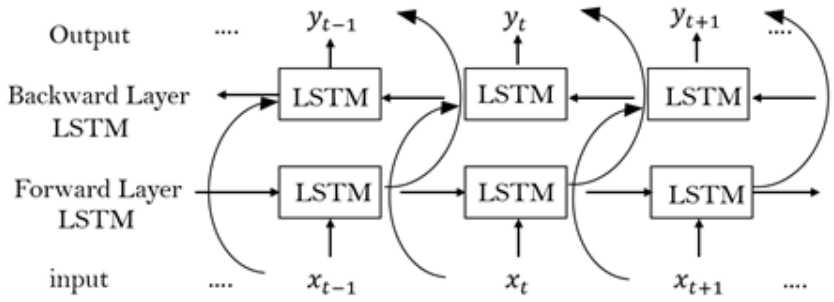

FIGURE 1.3 – Bidirectionnel LSTM pour la séquence de mots

1.7 Conclusion

Dans ce chapitre, nous avons présenté une brève description de la langue amazigh, la normalisation et l'élaboration de la langue amazigh, les méthodes de saisie, le classement des chaines de caractères amazighs et autres renseignements linguistiques de cette langue. Ensuite, nous avons présenté quelques travaux appliqués sur la langue amazigh tels que la reconnaissance des caractères tifinagh, la reconnaissance de voix, et l'étiquetage POS. Ensuite, nous avons décrit des techniques de tokénisation , en l'occurrence : MOSES, sausmots, BPE, lemmatisation et stemming. Puis nous avons présenté quelques approches de vectorisation (Sac des mots, TF-IDF, word-2-Vec). Enfin , nous avons cité des techniques de classifications, des techniques d'apprentissage automatique (SVM, CRF, Arbre de décision, KNN,…), et des techniques d'apprentissage profond (RNN, LSTM, GRU, BiLSTM,…). Dans le chapitre suivant, nous présenterons la démarche que nous avons suivie afin de construire un corpus annoté qui contient des balises de POS et les entités nommées pour la langue amazigh.

Chapitre 2

Construction d'un corpus annoté de la langue amazigh

2.1 Introduction

Les connaissances sont générées statistiquement dans le cas de la linguistique computationnelle. Chomsky a critiqué l'idée d'utiliser les chaines de Markov en tant qu'approche statistique en mettant le point sur le fait qu'elles ne sont pas capables de modéliser les structures récursives.
La linguistique computationnelle est restée influencée par cet avis et les approches symboliques à base de règles prédéfinies étaient les plus utilisées jusqu'aux années 80.
Par ailleurs, les attentes des utilisateurs sont devenues plus réalistes. Par exemple, en traduction automatique, les utilisateurs acceptent les traductions même approximatives, sur lesquelles ensuite, ils introduisent, le cas échéant, les changements qu'ils jugent nécessaires.
La plupart des langues du monde ne peuvent pas bénéficier des méthodes du traitement du langage naturel (NLP) en raison d'un manque de ressources. Étant donné que le besoin de technologies linguistiques est réparti de manière égale entre les locuteurs de chaque langue, l'étude de la construction des ressources linguistiques est nécessaire.
La langue amazigh, comme la plupart des langues qui n'ont que récemment commencé la recherche en TAL, souffre de la pénurie d'outils et de ressources pour son traitement automatique. Dans cette optique, et vu que les corpus constituent la base de la recherche dans le domaine des technologies langagières de l'homme, nous avons fixé comme objectif stratégique la construction d'un grand corpus annoté pour la langue amazigh. Dans ce chapitre nous allons décrire la méthode de construction du corpus utilisé pour étiqueter les textes écrits en tifinagh par POS et NER.

2.2 Collection des données pour la langue amazigh

2.2.1 Ingénierie des langues

Plusieurs méthodes statistiques novatrices en relation avec le traitement du langage naturel ont vu le jour et ont marqué ce domaine de recherche. Citons, à titre d'exemples, les domaines suivants : la recherche d'information textuelle [38], la reconnaissance automatique de la parole [39], l'étiquetage morphosyntaxique [40], l'analyse syntaxique [41], la traduction automatique [42] et la compréhension de la parole [43]. Les recherches en TAL ont toujours eu pour finalité la recherche d'une robustesse optimale sur des données ou des applications réelles. Elles se donnent pour objectif prioritaire la réalisation de systèmes efficaces et opérationnels dont les performances sont estimées au cours de campagnes d'évaluation significatives portant sur des situations proches du réel. Pour développer les outils de traitement informatique du langage, les chercheurs et les industriels ont besoin de recueillir de grandes quantités de textes écrits et oraux numérisés.

2.2.2 Propriétés du corpus

Dans cette partie, nous avons préparé un jeu de données qui a été collecté à partir d'un grand dépôt de textes amazighs. Nous avons réécrit le corpus annoté manuellement de Outahajala [1] du latin au tifinagh, et nous avons collecté les autres mots du site MAP [2] que nous avons annoté manuellement.
La collecte des données a été faite en utilisant un script Python qui scrape le site web pour trouver le texte amazigh. Le jeu de données peut être très utile pour des recherches ultérieures afin d'améliorer l'analyse de la parole amazigh.

2.2.3 Valeurs des données

— La langue amazigh est la deuxième langue officielle du Maroc. L'étiquetage des parties du discours est l'un des domaines de traitement du langage le plus naturel. Ce jeu de données peut être utilisé pour l'automatisation de ce processus ;
— Le jeu de données peut être considéré comme une référence pour la partie du discours amazigh ;
— Les chercheurs peuvent utiliser ce jeu de données pour améliorer le domaine du traitement du langage naturel pour la langue amazigh ;
— A notre connaissance, il n'existe pas de jeu de données public pour le langage amazigh (texte écrit en Tifinagh) ;
— Les données de corpus peuvent être utilisées pour classifier les textes Amazigh en entités nommées ;

1. M. Outahajal.,P. Rosso, L. Zenkor, Building an annotated corpus for Amazigh', Proceedings of 4th International Conference on Amazigh and ICT , Rabat Morocco, 2011.
2. MAP, MAP NEWS LETTERS. https://www.map.ma/am/home, (accessed May 08, 2021).

— Le premier corpus qui contient une annotation des entités nommées pour la langue Amazigh.

2.3 Corpus pour étiqueter les mots en POS

2.3.1 Amazigh POS tagset

La définition d'un ensemble d'étiquettes adéquat est une tâche essentielle dans la construction d'un étiqueteur POS automatique. Ce dernier vise à définir un ensemble d'étiquettes avec le niveau de granularité approprié. Par exemple ni très fin ni excessivement peu profond pour les cadres unitaires potentiels qui l'utiliseront. Le corpus d'origine est constitué d'un ensemble d'écrits séparés d'un assortiment de sources, par exemple, quelques livres, tout comme certains écrits du site de l'IRCAM. Nous avions la possibilité d'arriver à un nombre de mots supérieur à 60k tokens. Ce corpus est commenté morphologiquement en utilisant le jeu d'étiquettes présenté dans [21].

N°	**POS**	**Designation**
1	NN	Common noun
2	NNK	Kinship noun
3	NNP	Proper noun
4	ROT	Residual, other
5	S	Preposition
6	FW	Foreign word
7	NUM	Numeral
8	ROT	Residual, other
9	PUNC	Punctuation
10	SYM	Symbol
11	VBP	Verb, participle
12	ADJ	Adjective
13	ADV	Adverbe
14	C	Conjunction
15	DT	Determiner
16	FOC	Focalize
17	IN	Interjection
18	NEG	Particle, negative
19	VOC	Vocative
20	PRED	Particle, predicate
21	PROR	Particle, orientation
22	PRPR	Particle, preverbal
23	PROT	Particle, other
24	PDEM	Demonstrative pronoun
25	PP	Personal pronoun

Suite à la page suivante

Table 2.1 – *Suite de la page précédente*

N°	**POS**	**Designation**
26	PPOS	Possessive pronoun
27	INT	Interrogative
28	REL	Relative

TABLE 2.1 : Les balises de POS pour la langue Amazigh.

Comme le montre le tableau 2.1, nous avons mis 28 balises (nom, verbe, adjectif, adverbe,…) et quelques balises ont des sous classes comme mentionné dans le tableau ci-dessous tableau 2.2 ; par exemple

- Nom a comme sous classe : nom propre, nom commun…
- Verbe a comme sous classe : verbe infinitif, participe,…
- Pronom a comme sous classes : Pronom démonstratif, Pronom personnel, Pronom possessif
- Pratique a comme sous classe : négatif prédicat, orientation, préverbal, autre,

Balises	**Sous classes**
Noun	Common noun, Kinship noun, proper noun
verb	Verb base form, Verb participle
Pronoun	Demonstrative pronoun, Personal pronoun, Possessive pronoun
Practicle	negative, predicate, orientation, preverbal, other

TABLE 2.2 : Sous classes des balises POS pour la langue Amazigh

2.3.2 Corpus de POS pour la langue Amazigh

Un corpus est une collection de données linguistiques qui sont sélectionnées et organisées selon des critères linguistiques explicites critères pour servir d'échantillon d'emplois déterminés d'une langue.
En général, un corpus contient jusqu'à quelques millions de mots et peut être lemmatisé et annoté avec des informations sur POS. Parmi les corpus, il y a le British National Corpus (100 millions de mots) et l'American National Corpus (20 millions de mots). Un corpus équilibré fournirait une large sélection de différents types de textes et provenant de sources

diverses telles que des journaux, des livres encyclopédies ou le web. Pour la langue amazigh marocaine, il a été difficile de trouver des ressources prêtes à l'emploi. Nous pouvons simplement mentionner le corpus annoté manuellement de [44]. Ce corpus contient 20 k mots c'est pourquoi nous avons décidé d'utiliser son corpus dans cette étude mais nous avons fait quelques modifications dans ce corpus :

— Nous avons utilisé les caractères Tifinagh au lieu de l'écriture latine. L'avantage d'utiliser le caractère Tifinagh est d'optimiser le programme en évitant l'utilisation du transcodage. Dans cette étape, nous avons rencontré quelques erreurs dans l'écriture des mots en latin ; il existe des caractères dans le corpus de Outahajala sont écrits d'une manière erronée par exemple (ⴹ=T), (ⵅ=H), (ⵣ=Z,) (ⵄ=E),(ⴹ=D), (ⵇ=R), (ⵙ=S),(ⵖ=G), (ⵡ=Gw...), mais l'équivalence correcte de ces caractères sont ((ⴹ=ṭ),(ⵅ=ḥ), (ⵣ=ẓ), (ⵄ=ϵ), (ⴹ=ḍ), (ⵇ=ṛ), (ⵙ=ṣ), (ⵖ=γ), (ⵡ =gw...), après le transcodage en caractères Tifinagh nous donne des mots sans signification ou des mots avec une signification différente à l'origine. le tableau 2.3 cite quelques exemples rencontrés :

Mot dans le corpus	**Mot transcodeur**	**Mot correct en tifinagh**	**Mot correct en Latin**
amaziG	ⵄⵎⴰⵣⵡ	ⵄⵎⴰⵣⵖ	amaziγ
nG	ⵏⵡ	ⵏⵖ	nγ
Akkw	ⵄⵇⵇⵍ	ⵄⵇⵇⵓ	akkwn
tdggwat	ⵜⴷⴳⴳⵍⵄⵜ	ⵜⵄⴷⴳⵓⵄ	tdggwater
iD	ⵉⴷ	ⵉⴹ	iḍ
iZlmD	ⵉⵣⵍⵎⴷ	ⵉⵣⵍⵎⴹ	iẓlmḍ
iRaZaGn	ⵉⵔⴰⵣⴰⵖⵏ	ⵉⵕⴰⵣⴰⵣⵖⵏ	iṛaẓaγn
lmasiH	ⵍⵎⴰⵙⵉⵀ	ⵍⵎⴰⵙⵉⵅ	lmasiḥ
miSR	ⵎⵉⵙⵔ	ⵎⵉⵙⵕ	miṣṛ
Eica	ⵄⵉⵛⴰ	ⵄⵉⵛⴰ	ica
iHyaWi	ⵉⵀⵢⴰⵍⵉ	ⵉⵅⵢⴰⵍⵉ	iḥyawi
uTumatik	ⵓⵜⵓⵎⴰⵜⵉⵇ	ⵍⵓⵓⵎⴰⵜⵉⵇ	uṭumatik

Table 2.3 : Exemple des mots erronés dans le corpus

— Nous avons ajouté le type « symbole » dans les types des balises et qui signifie les caractères spéciaux qui sont considérés comme des ponctuations dans ce corpus.
— Nous avons éliminé le type date car il s'agit d'une entité.

Après la phase de transcodage et de correction des données nous avons changé la structure de corpus. Le corpus sous forme d'un fichier texte contient des lignes formatées comme suit : Chaque ligne contient une phrase commencée par un caractère spécial, '#' et le mot 'text', après chaque phrase, nous commençons les lignes suivantes avec les mots de la phrase, chaque mot nous mettons son balise séparé par une tabulation. Ce premier corpus qui contient plus que 20K mots et 2k phrases a été utilisé dans une étude intitulée « Part-of-Speech Tagging Using Long Short Term Memory (LSTM) : Amazigh Text Written in

tifinagh Characters » [45].

Nous avons enrichi notre corpus par d'autres phrases pour atteint presque 60K mots, Nous avons collecté à partir du site web MAP. Le site web dispose d'une grande collection de textes amazighs. Le nombre d'étiquettes part-of-speech est de 28 au total. Cependant, toutes les balises POS ne sont pas couramment utilisées dans ces données. Certaines balises POS sont très utilisées alors que d'autres sont moins courants. Certaines balises sont extrêmement rares dans cet ensemble de données, par exemple ROT, IN, PDEM,… etc. La distribution des balises dans l'ensemble de données est illustrée dans la figure 2.1. Neuf balises ont une fréquence de données inférieure à 800, tandis que la plupart des balises ont une fréquence supérieure à 2000.

Le site web à partir duquel nous avons collecté le jeu de données classe les textes amazighs en différents domaines. Tous les textes extraits du site Web ont été segmentés en phrases. Nous avons gratté le site web en accédant à chaque catégorie, et nous obtenons les paragraphes. Le langage de programmation utilisé pour cette tâche est Python avec les bibliothèques Beautifulsoup et Selenium.

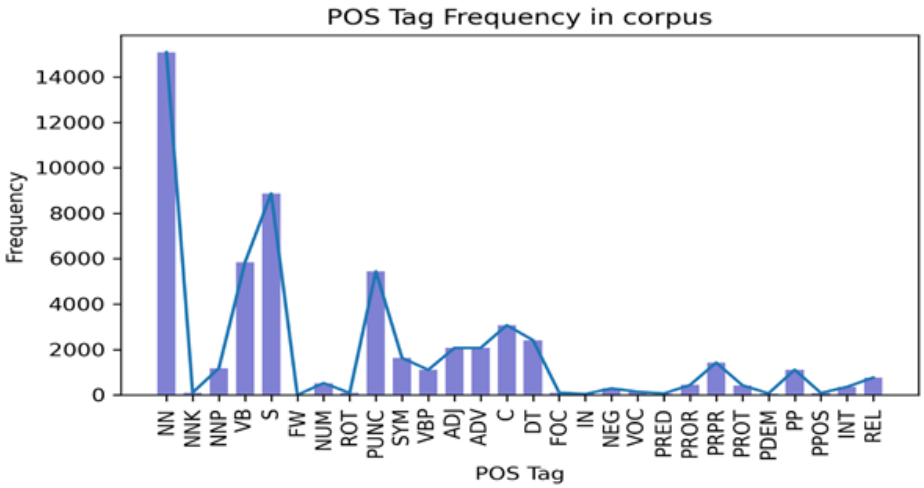

FIGURE 2.1 – Distribution des balises de la langue amazigh dans le corpus.

2.4 Corpus pour classifier les mots amazigh en entités nommées

2.4.1 Type des entités nommées

Dans l'exploration de données, une entité nommée est une phrase qui identifie clairement un élément parmi un ensemble d'autres éléments ayant des attributs similaires. Des exemples d'entités nommées sont les noms et prénoms, les lieux géographiques, les noms

des personnes, les adresses, les numéros de téléphone, les sociétés et les adresses. Les entités nommées sont souvent exploitées pour des initiatives de marketing.

Entité nommée	**description**
PERSON	Personnes, y compris les personnages fictifs
DATE	Dates ou périodes absolues ou relatives
CARDINAL	Les nombres qui ne relèvent d'aucun autre type
NORP	Nationalités, groupes religieux ou politiques
LOC	Emplacements non-GPE, chaînes de montagnes, plans d'eau
GPE	pays et villes
ORG	Entreprises, agences, institutions, etc.
LANGUAGE	Langues
ORDINAL	premier, "deuxième", etc.
EVENT	Batailles d'ouragans, batailles, guerres, événements sportifs, etc.
ANIMAL	les animaux
TIME	Le temps : heur, minute....
PLANT	les plantes
BODY	les organes de corps
FAC	Bâtiments, aéroports, autoroutes, ponts, etc.
COLOR	Couleurs
PRODUCT	Objets, véhicules, aliments, etc. (pas de services)
WORKOFART	Titres de livres, chansons, etc.
LAW	LAW
QUANTITY	Les mesures sont en poids ou en distance.
PERCENT	Pourcentage, y compris "%".
INS	insectes
MONEY	Valeurs monétaires, y compris l'unité.

TABLE 2.4 : Les types des entités nommées

Le tableau 2.4 décrit les différents types des entités nommées utilisés dans le corpus de la langue Amazigh, par exemple l'entité PERSON signifie les Personnes, y compris

les personnages fictifs, ORG pour les entreprises, agences, institutions, NORP pour les nationalités, groupes religieux ou politiques,…etc

Entité nommée	**Exemple**
PERSON	Otman
DATE	juin 2020
CARDINAL	5
NORP	Morocain
LOC	himalayan
GPE	Maroc
ORG	Google
LANGUAGE	Englais
ORDINAL	premier
EVENT	Coup du monde
ANIMAL	Chat
TIME	10:30
PLANT	fleur
BODY	pied
FAC	airport Mohamed 5
COLOR	blue
PRODUCT	Pizza
WORKOFART	Monaliza
LAW	Constitution marocaine
QUANTITY	2Kg
PERCENT	98%
INS	Abeilles
MONEY	100Dh

TABLE 2.5 : Exemple pour chaque type d'entité nommée

Le tableau 2.5 présente quelques exemples pour chaque type d'entité nommée, il existe des types qui représentés sous forme d'une phrase par exemple la date « 8 juin 2022 », c'est pour cela nous avons besoin d'intégrer d'autre schémas avec les types des entités nommées pour signifier le début et la fin d'une entité nommée, nous avons deux schémas existe « IOB » et « BILUO » (Voir les tableaux : 2.6 et 2.7).

Schema IOB	**Description**
I	Le token est à l'intérieur d'une entité
O	Le token est à l'extérieur d'une entité
B	Le token est le début d'une entité

Schema IOB	**Description**

TABLE 2.6 : Description du schéma IOB

Schéma BILUO	**Description**
B	Le token est le début d'une entité multi-token
I	Le token est à l'intérieur d'une entité multi-token
L	Le token est le dernier token d'une entité multi-token
U	Le token est une entité unitaire à un seul token
O	Le token est à l'extérieur d'une entité

TABLE 2.7 : Description du schéma BILUO

Le mot	**IOB**	**BILUO**
Yassin	B-PERSON	U-PERSON
is	O	O
from	O	O
faculty	B-ORG	B-ORG
of	I-ORG	I-ORG
science	I-ORG	I-ORG
and	I-ORG	I-ORG
technology	I-ORG	L-ORG
in	O	O
Beni	B-LOC	B-LOC
Mella	I-LOC	L-LOC

TABLE 2.8 : Exemple décrit la différence entre les deux schémas

Le schéma IOB (abréviation de inside, outside, beginning) est un schéma de marquage commun pour le marquage de tokens dans une tâche de chunking en linguistique informatique (ex. reconnaissance d'entités nommées), il a été présenté par [46]. Comme il indique le tableau 2.6 le préfixe I- situé devant une balise indique que la balise se trouve à l'intérieur d'un chunk. Une balise O indique qu'un token n'appartient à aucun chunk, et le préfixe B- devant une balise indique que la balise est le début d'un chunk qui suit immédiatement un autre chunk sans balise O entre eux. Il n'est utilisé que dans le cas que un chunk vient après une balise O, le premier token du chunk prend le préfixe I-.

Le deuxième schéma est BILUO (abréviation de beginning, inside, last, unit, outside), la différence entre les deux schémas est les deux préfixes L et U ; tel que le préfixe L situé devant une balise qui est la dernière balise d'un chunk qui contient plus qu'une balise, et U situé devant une balise qui est unique dans le chunk.

Les experts considèrent que le schéma BILUO est le meilleur pour rendre les modèles de reconnaissance des entités nommées aussi précis le plus possible. En effet, grâce au schéma à préfixes multiples, ils spécifient des informations plus détaillées dans le texte, ce qui améliore les capacités d'apprentissage de l'algorithme d'apprentissage automatique utilisé. Le tableau 2.8 montre la différence entre les deux schémas, on observe que BILUO schémas plus précis que le schéma IOB.

2.4.2 Distribution des entités nommées dans le corpus

Le corpus crée pour les entités nommés de la langue Amazigh est le même que nous avons créé pour POS, nous avons ajouté une troisième colonne qui contient NE, séparée avec la deuxième par le symbole « + », le type de NE a attaché avec des préfixe de schéma BILUO, la figure 2.2 montre un extrait de corpus utilisé pour POS et NE.

```
#text= ⵓⵙⵎⵓⵔ ⵏ ⵜⵙⵔⴷⵓⵜ ⵓⵎⵣⵢⴰⵏ ⵙ ⵎⵍⵍⵉ ⵏ ⵜⵙⵔⴷⵓⵜ ⵓⵎⵣⵢⴰⵏ ⵓⵎⵙⵉ ,
ⵓⵙⵎⵓⵔ    NN+O
ⵏ          S+O
ⵜⵙⵔⴷⵓⵜ    NN+B-DATE
ⵓⵎⵣⵢⴰⵏ    ADJ+L-DATE
ⵙ          S+O
ⵎⵍⵍⵉ       NN+O
ⵏ          S+O
ⵜⵙⵔⴷⵓⵜ    NN+B-DATE
ⵓⵎⵣⵢⴰⵏ    ADJ+I-DATE
ⵓⵎⵙⵉ       ADJ+L-DATE
,          PUNC+O
```

FIGURE 2.2 – Un extrait de corpus

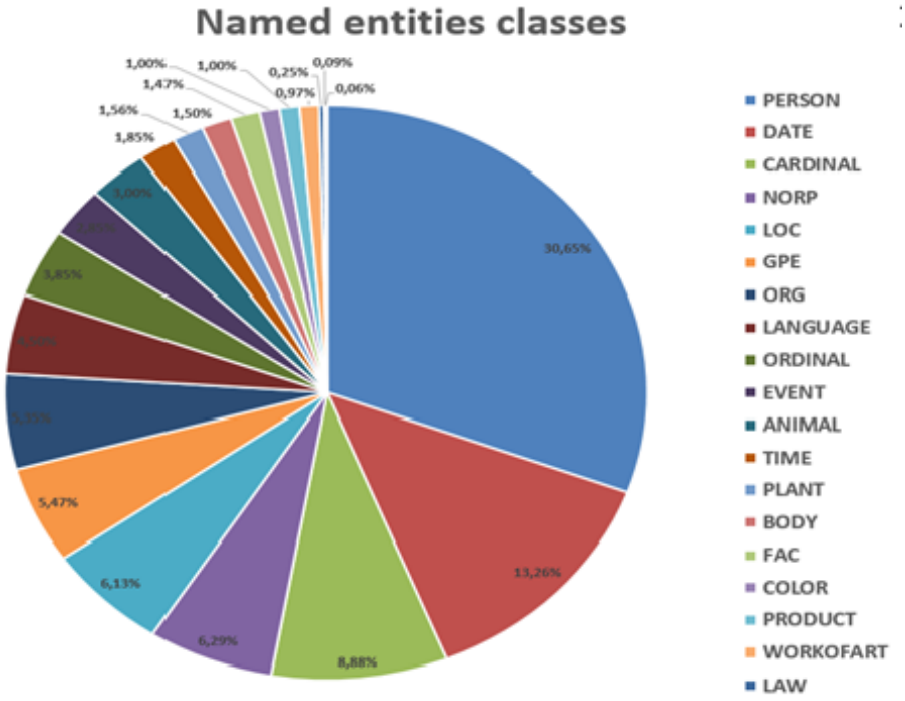

FIGURE 2.3 – Distribution des types d'entité nommée dans le corpus

La figure 2.3 présente la fréquence de chaque type d'entité nommée dans le corpus, nous observons que le type « PERSON » est le plus fréquent par rapport aux autres types, il représente un pourcentage de 30.65% de corpus, le type « DATE » aussi est fréquent il représente un pourcentage de 13.26% de corpus, nous trouvons dans le troisième type fréquent « CARDINAL » qui représente un pourcentage de 8.88% de corpus, les moins fréquents sont « LOW », « WORKOFART », « « PRODUCT »,….etc. Alors notre corpus a besoin de plus des phrases qui contiennent ces types des entités nommées.

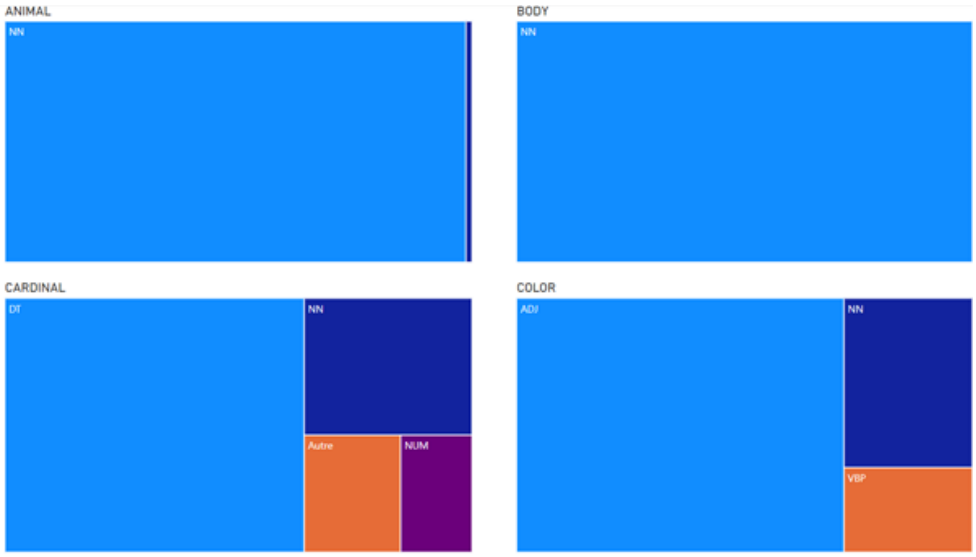

FIGURE 2.4 – Distribution des balises POS pour les entités ANIMAL, CARDINAL, BODY, COLOR

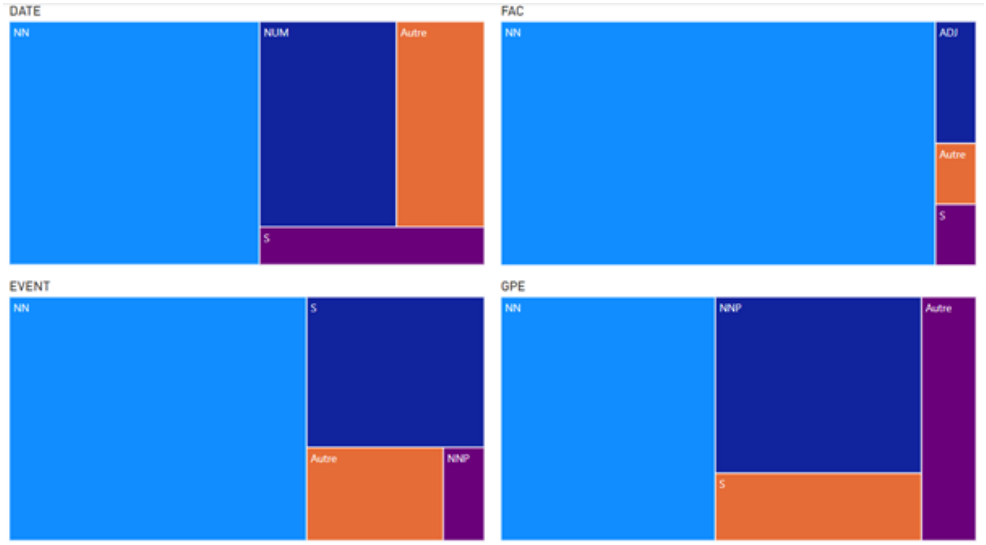

FIGURE 2.5 – Distribution des balises POS pour les entités DATE, FAC, EVENT, GPE

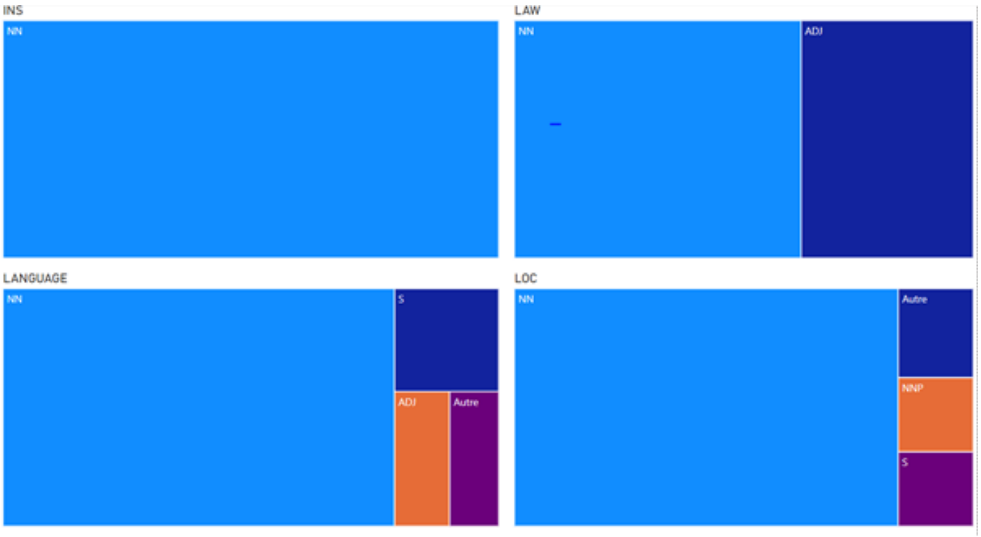

FIGURE 2.6 – Distribution des balises POS pour les entités INS, LAW, LANGUAGE, LOC

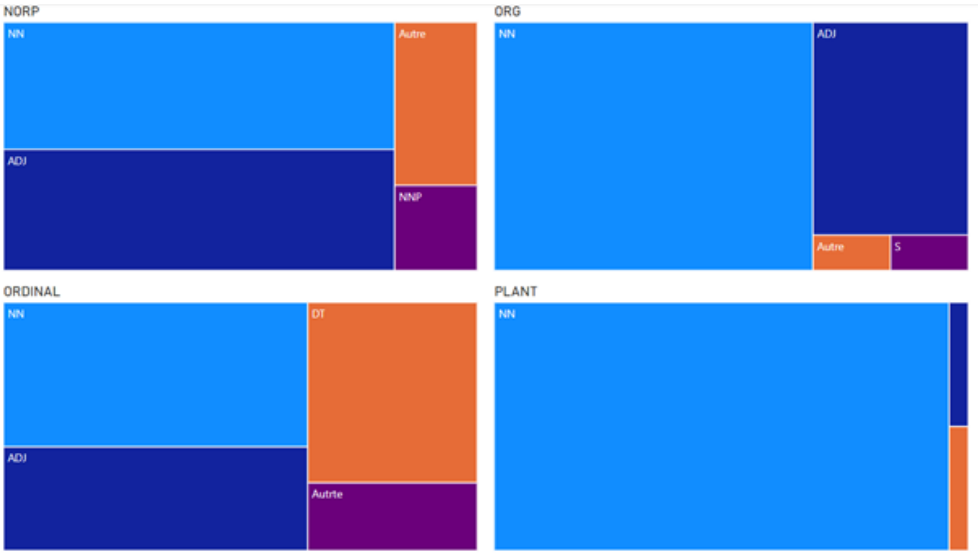

FIGURE 2.7 – Distribution des balises POS pour les entités NORP, ORG, ORDINAL, PLANT

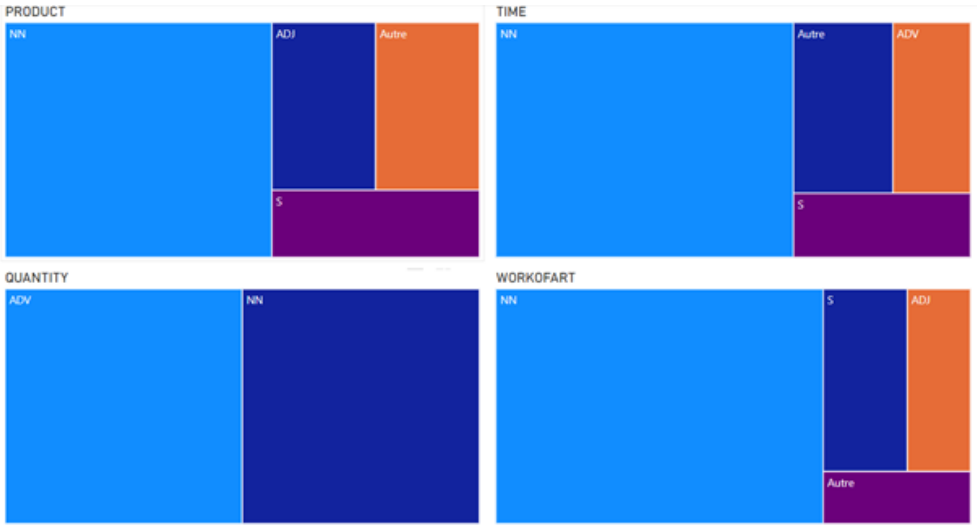

FIGURE 2.8 – Distribution des balises POS pour les entités PRODUCT, TIME, QUANTITY, WORKOFART

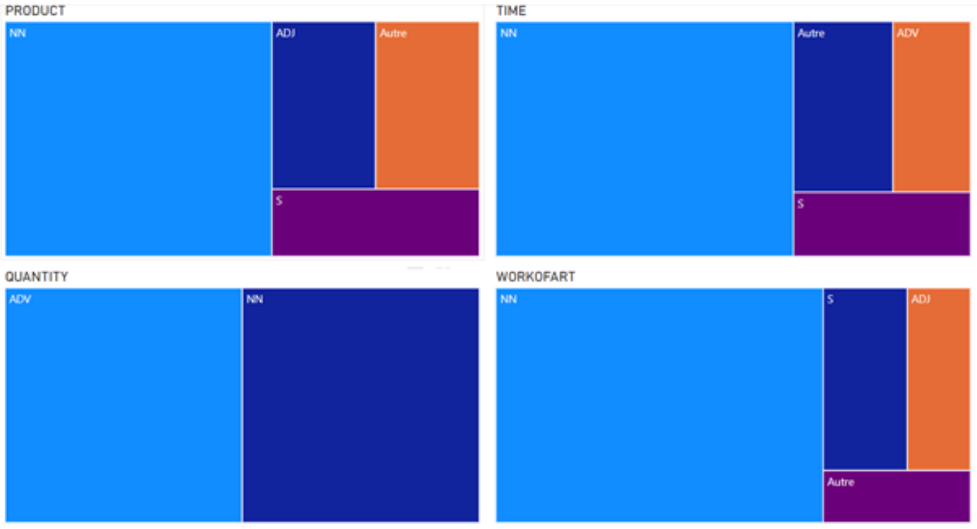

FIGURE 2.9 – Distribution des balises POS pour l'entité PERSON

Les figures 2.4, 2.5, 2.6, 2.7,2.8 et 2.9 représentent la distribution de chaque balise pour chaque type des entités nommées, par exemple :

— L'entité « PERSON » contient plus de 50% des mots de type NN, plus que 25% des mots de type NNP, plus que 8% des mots de type NNK, et tous les types restés représentent moins de 17%.

— L'entité « PLANT » contient 96% des mots de type NN, 2% des mots de type NNP, et 2% des mots de type ADV.

— L'entité « PRODUCT » contient plus de 56% des mots de type NN, plus que 15% des mots de type ADJ, plus que 12% des mots de type S, et tous les types restés représentent moins de 17
— L'entité « TIME » contient plus de 62% des mots de type NN, plus que 11% des mots de type ADV, plus que 10% des mots de type S, et tous les types restés représentent moins de 17%.
— L'entité « WORKOFART » contient plus de 68% des mots de type NN, plus que 13% des mots de type S, plus que 10% des mots de type ADJ, et tous les types restés représentent moins de 9%.
— L'entité « ORG » contient plus de 67% des mots de type NN, plus que 28% des mots de type ADJ et tous les types restés représentent moins de 5%.
— L'entité « QUANTITY » contient 50% des mots de type NN, et 50% des mots de type ADV.
— L'entité « ANIMAL » contient plus de 98% des mots de type NN, plus que 1% des mots de type NNP.
— L'entité « ORDINAL » contient plus de 37% des mots de type NN, plus que 26% des mots de type ADJ, plus que 26% des mots de type DT, et tous les types restés représentent moins de 11%.
— L'entité « LOC » contient plus de 83% des mots de type NN, plus que 5% des mots de type NNP, plus que 5% des mots de type S, et tous les types restés représentent moins de 7%.
— L'entité « NORP » contient plus de 42% des mots de type NN, plus que 40% des mots de type ADJ, plus que 5% des mots de type NNP et tous les types restés représentent moins de 13%.
— L'entité « LAW » contient plus de 62% des mots de type NN, plus que 37% des mots de type ADJ.
— L'entité « LANGUAGE » contient plus de 79% des mots de type NN, plus que 9% des mots de type S, plus que 6% des mots de type ADJ, et tous les types restés représentent moins de 6%.
— L'entité « GPE » contient plus de 45% des mots de type NN, plus que 31% des mots de type NNP, plus que 12% des mots de type S, et tous les types restés représentent moins de 12%.
— L'entité « FAC » contient plus de 91% des mots de type NN, plus que 4% des mots de type ADJ, plus que 2% des mots de type S, et tous les types restés représentent moins de 3%.
— L'entité « EVENT » contient plus de 62% des mots de type NN, plus que 23% des mots de type S, plus que 3% des mots de type NNP, et tous les types restés représentent moins de 12%.
— L'entité « DATE » contient plus de 52% des mots de type NN, plus que 24% des mots de type NUM, plus que 7% des mots de type S, et tous les types restés représentent moins de 17%.
— L'entité « COLOR » contient plus de 71% des mots de type ADJ, plus que 18% des mots de type NN, plus que 9% des mots de type VBP.

— L'entité « CARDINAL » contient plus de 64% des mots de type DT, plus que 19% des mots de type NN, plus que 7% des mots de type NUM, et tous les types restés représentent moins de 10%.
— L'entité « BODY » contient plus de 97% des mots de type NN, plus que 2% des mots de type VB.
— L'entité « INS » contient plus de 100% des mots de type NN.

On observe que les balises de type NN plus dominantes que les autres balises, l'existence des balises de type S peut expliquer par l'existence des entités nommées composées par plusieurs mots qui contient des balises de types S.

2.5 Conclusion

Les modèles complexes tels que les réseaux neuronaux profonds ont fait progresser un large éventail d'applications d'apprentissage automatique et ont permis de résoudre des tâches difficiles. Cependant, de grandes quantités de données annotées par l'homme sont nécessaires pour entraîner ces modèles dans le cadre de l'apprentissage supervisé, et restent le goulot d'étranglement dans des applications importantes telles que la reconnaissance d'entités nommées (NER) et l'étiquetage des mots en POS. La ressource présentée ci-dessus est constituée de plus de 60.000 mots de l'amazigh marocain écrit en caractère tifinagh, basée sur un existe corpus [44] , des article extrait du site MAP, et à partir de textes variés respectant les règles orthographiques adoptées au Maroc.Dans le chapitre suivant, nous décrirons des techniques d'apprentissage automatique et d'apprentissage profond pour étiqueter les phrases écrit en tifinagh.

Chapitre 3

Étiquetage de la langue Amazigh Part-Of-Speech

3.1 Introduction

L'étiquetage de POS est le moyen d'attribuer des marqueurs de caractéristiques grammaticales à chaque mot d'un texte informatif dans différentes langues et dialectes [47]. La contribution à un calcul d'étiquetage est une succession de mots (tokenisés) et un ensemble d'étiquettes, et le résultat est un arrangement d'étiquettes, une pour chaque token[48].
Récemment, les réseaux de neurones ont gagné en popularité dans le domaine de l'intelligence artificielle. Les progrès sont dus à la percée dans les algorithmes qui apprennent et reconnaissent des modèles très complexes en utilisant des couches profondes de réseaux de neurones ou communément appelés les réseaux de neurones profonds (DNN) [49], et l'introduction de différents types de réseaux de neurones tels que le réseau de neurones convolutif et réseau neuronal récurrent (RNN). Par exemple, les réseaux de neurones convolutifs, qui sont un type spécial de réseaux de neurones à rétroaction avec des réseaux à deux dimensions, ont montré une grande précision dans la classification des images à travers les champs récepteurs locaux, les poids partagés, la mise en commun, de la simple reconnaissance des chiffres manuscrits à la reconnaissance faciale plus complexe. Dans la modélisation de modèles séquentiels, tels que la reconnaissance de phonèmes, la reconnaissance automatique de la parole [50], la synthèse vocale, la traduction vocale [51], le chatbot et bien d'autres, RNN ou le type plus spécialisé de RNN, les réseaux à mémoire longue et courte durée (LSTM) se sont révélés meilleurs que la plupart des approches traditionnelles. Ce chapitre présente une étude comparative de trois méthodes pour résoudre le problème de l'étiquetage de la partie du discours (POS) amazigh. Ces méthodes sont les réseaux LSTM, les champs aléatoires conditionnels (CRF) et l'arbre de décision. L'objectif est d'examiner les performances de l'état actuel des réseaux LSTM par rapport au CRF et à l'arbre de décision dans le marquage POS. Le balisage POS est une tâche de traitement du langage qui attribue une balise POS (par exemple, nom, verbe, adjectif, etc.) à chaque mot d'une phrase. Dans ce chapitre nous présenterons les différentes étapes pour entraîner et tester les approches d'apprentissage automatique (arbre de décision, CRF), et les approches d'apprentissage

profond (RNN, LSTM, GRU, Bi-LSTM), puis nous allons établir une comparaison entre les résultats trouvés.

3.2 Les techniques d'étiquetage

3.2.1 Performance des modèles

Le texte annoté POS d'Amazigh utilisé contient plus que 60K mots, qui ont été étiquetés en utilisant 28 étiquettes POS. Ce corpus est étiqueté phrase par phrase, chaque phrase divisée en mots, chaque mots est étiqueté par un POS. Sur le total des mots étiquetés, environ 80 % des mots ont été utilisés pour l'entraînement, et les 20 % de phrases restantes ont été utilisées respectivement pour le test et le développement.
Les performances sont estimées en calculant la moyenne de la précision (P), du rappel (R) et du F-mesure(F1) sur le nombre total de tests. [52] Ces derniers sont calculés pour chaque étiquette l comme suit :

$$P_l = \frac{nombre\ desitems\ tiquet\ correctement\ par\ l}{nombre\ desitems\ tiquet\ par\ l} \tag{3.1}$$

$$R_l = \frac{nombre\ desitems\ tiquet\ correctement\ par\ l}{nombre\,des\ items\ tiquet\ manuellement\ par\ l} \tag{3.2}$$

$$F1_l = 2 \times \frac{P_l \times R_l}{P_l + R_l} \tag{3.3}$$

3.2.2 Techniques d'apprentissage automatique

3.2.2.1 Arbre décision

Une technique courante pour prédire les étiquettes de parties du discours est l'apprentissage par arbre de décision. L'apprentissage par arbre de décision est un algorithme d'apprentissage automatique qui partitionne séquentiellement l'ensemble des données d'apprentissage en fonction des valeurs d'attributs, séquentiellement l'ensemble des données d'apprentissage en fonction des valeurs des attributs, en partitionnant récursivement les données jusqu'à ce que tous les attributs d'un nœud puissent être classés avec la même valeur. Dans le même ordre d'idées, TreeTagger est un système de marquage de textes avec des informations sur les parties de la parole et les lemmes [22]
TreeTagger a été utilisé avec succès pour baliser l'allemand, l'anglais, le français, l'italien, le néerlandais, l'espagnol, le russe, etc. Il est généralement adaptable à d'autres langues si un lexique et un corpus d'entraînement étiqueté manuellement sont disponibles. Le cœur de TreeTagger, comme son nom l'indique, est " l'estimation des probabilités de transition avec un arbre de décision binaire ". L'étape initiale de construction de l'arbre de décision a lieu pendant la phase d'apprentissage. Le site analysera le texte et les trigrammes, en insérant chaque unigramme dans l'arbre.
Les probabilités de l'étiquette à utiliser sont déterminées pour un nœud donné de l'arbre

en fonction des informations obtenues des deux nœuds précédents. Une fois que l'arbre est créé, ses nœuds sont élagués. Si le gain d'information d'un nœud particulier est inférieur à un seuil défini, ses nœuds enfants sont élagués. La figure 3.1 représente une version simplifiée d'un arbre de décision utilisant un exemple de phrase Amazigh.

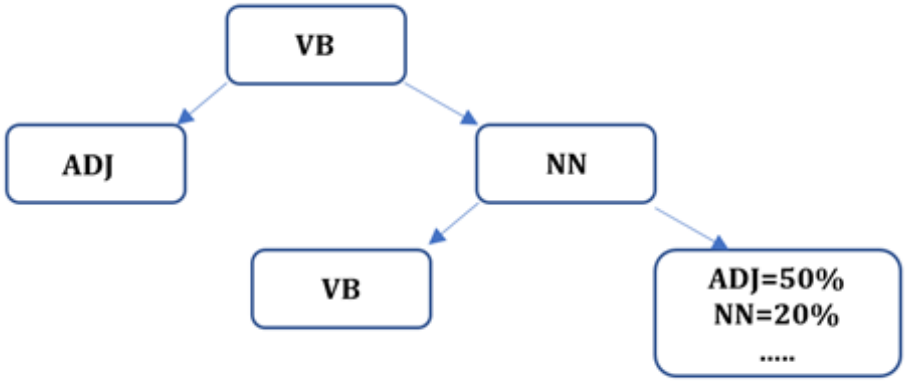

FIGURE 3.1 – Version simplifiée d'un arbre de décision.

Nous avons entrainé et testé l'approche de l'arbre de décision en utilisant la bibliothèque Scikit-Leran. Pour chaque attribut de corpus, l'algorithme de l'arbre de décision forme un nœud, où l'attribut le plus important est placé comme un nœud racine. Pour l'évaluation, nous commençons par le nœud racine et descendons dans l'arbre en suivant le nœud correspondant qui répond à notre condition. Ce processus se poursuit jusqu'à ce que l'on atteigne un nœud feuille, qui contient la prédiction [53]. Le tableau suivant contient les différents résultats de l'entraînement de base .

POS	**Précision**	**Rappel**	**F-mesure**
S	0,893	0,99	0,897
ADJ	0,751	0,781	0,686
PUNC	1	1	1
VB	0,848	0,899	0,848
NN	0,822	0,916	0,844
ADV	0,706	0,701	0,65
DT	0,793	0,805	0,778
VBP	0,772	0,804	0,684
C	0,716	0,812	0,75
REL	0,854	0,902	0,771
PROR	0,804	0,809	0,811
PP	0,954	0,98	0,931
NNP	0,821	0,899	0,802

Suite à la page suivante

Table 3.1 – *Suite de la page précédente*

POS	**Précision**	**Rappel**	**F-mesure**
PRPR	0,874	0,913	0,879
SYM	1	1	1
INT	0,65	0,815	0,729
NEG	0,997	1	0,988
PDEM	0,545	0,675	0,375
PRED	0,9	0,67	0,587
IN	0,9	0,834	0,753
PROT	0,47	0,475	0,409
NUM	0,993	0,993	0,993
VOC	0,873	0,882	0,846
NNK	0,867	0,907	0,82
PPOS	0,82	0,77	0,606
FOC	0,642	0,704	0,623
ROT	0,253	0,22	0,124
Moyenne pondérée	0,876	0.842	0,832

TABLE 3.1 : Résultats de classification du modèle d'arbre de décision

Selon les résultats indiqués dans le tableau 3.1, nous avons trouvé la moyenne pondérée 0.876, ce résultat est justifié par le manque des phrases qui contiennent ROT, FOC et ADV c'est pourquoi il faut enrichir notre corpus par des phrases qui contiennent ces types. Maintenant, nous allons présenter les différents résultats obtenus après l'entraînement du modèle d'arbre de décision.

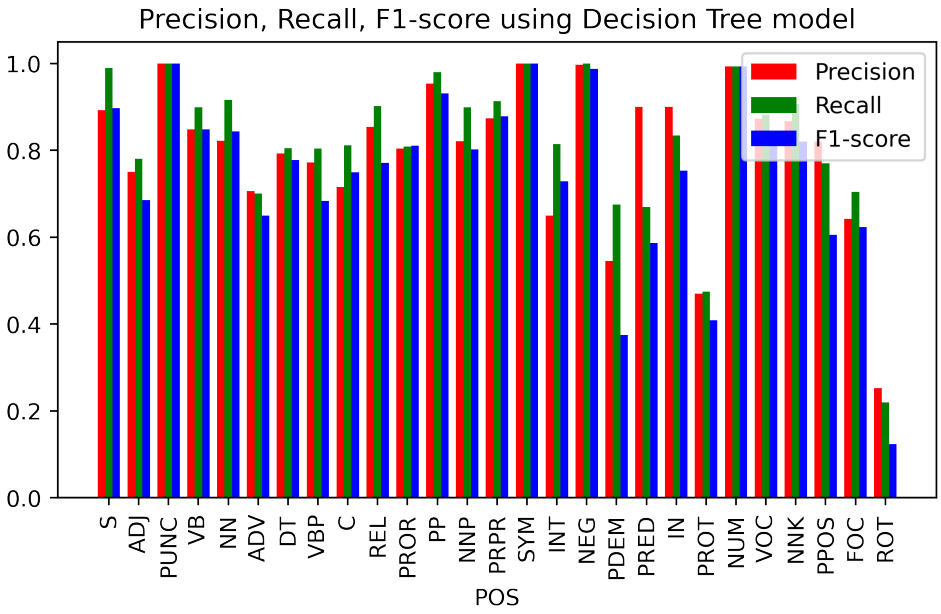

FIGURE 3.2 – Précision, rappel est F-mesure du modèle d'arbre de décision.

La figure 3.2 indique les variations de la précision, du rappel et du F-mesure après l'entraînement du modèle d'algorithme d'arbre de décision qui donne la même variation pour les trois valeurs.

3.2.2.2 Les champs aléatoires conditionnels

L'application des CRF à de nouvelles données consiste à trouver la séquence d'étiquettes la plus probable étant donné une séquence d'observations non observées. Comme pour les autres méthodes stochastiques, celle-ci est généralement obtenue avec un algorithme de Viterbi [54].
Le modèle CRF (non normalisé) pour une phrase $x = (x_1, ..., x_{|x|})$ et une séquence POS $y = (y_1, ..., y_{|x|})$ définie comme suit :

$$p(y/x; w) = \prod_{i=n}^{|x|} = exp\left(w.\emptyset\left(y_{i-n}, ..., y_i, x, i\right)\right) \tag{3.4}$$

Où n désignent l'ordre du modèle, w le vecteur de paramètres du modèle, et $\emptyset$ la fonction d'extraction de caractéristiques. Nous désignons l'ensemble des balises par Y, c'est-à-dire, $y_i \in Y$ pour $i \in 1...|x|$.[55].
Pour tester l'algorithme CRF, nous utilisons la bibliothèque Scikit-Leran (sklearn-crfsuite) qui fournit cinq algorithmes :

— Gradient descent using the L-BFGS method 'lbfgs'
— Stochastic Gradient Desc with L2 regularization term 'l2sgd'
— Averaged Perceptron 'ap'

— Passive Aggressive 'pa'
— Adaptive Regularization Of Weight Vector 'arow'

Paramétres de CRF :

— all_possible_transitions (bool, optional (default=False)) : Spécifie si CRFsuite génère des caractéristiques de transition qui n'apparaissent même pas dans les données d'apprentissage (c'est-à-dire des caractéristiques de transition négatives). Si True, CRFsuite génère des caractéristiques de transition qui associent toutes les paires d'étiquettes possibles. Supposons que le nombre d'étiquettes dans les données d'apprentissage soit L, cette fonction générera (L * L) caractéristiques de transition. Cette fonction est désactivée par défaut.
— c1 (variable flottante, facultative (par défaut=0)) : Le coefficient pour la régularisation L1. Si une valeur non nulle est spécifiée, CRFsuite passe à la méthode Orthant-Wise Limited-memory Quasi-Newton (OWL-QN). La valeur par défaut est zéro (pas de régularisation L1). Algorithmes d'apprentissage pris en charge : lbfgs
— c_2 (variable flottante, facultative (valeur par défaut = 1,0)) : Le coefficient de régularisation L2. Algorithmes d'apprentissage pris en charge : l2sgd, lbfgs
— max_iterations (int, facultatif (par défaut=None)) : Le nombre maximum d'itérations pour les algorithmes d'optimisation. La valeur par défaut dépend de l'algorithme d'apprentissage : (lbfgs illimité ; l2sgd 1000 ;ap 100 ;pa 100)
Où n désignent l'ordre du modèle, w le vecteur de paramètres du modèle, et $\emptyset$ la fonction d'extraction de caractéristiques. Nous désignons l'ensemble des balises par Y, c'est-à-dire, $y_i \in Y$ pour $i \in 1...|x|$[55].

Dans notre cas, nous avons choisi lbfgs comme algorithme, nous avons mis tous les transitions possibles vrai pour générer tous les caractéristiques de transition qui associent les paires de POS comme il est indiqué dans le tableau 3.2, nous avons fixé la valeur de c_1 à 0.01, la valeur de c_2 à 0.1 et max_etteration à 100. Le tableau 3.3 indique les différentes valeurs de précision, rappel et F-mesure [53].

étiquette 1	**étiquette 2**	**CRF transition**
PRPR	VB	2.564851
NNK	PP	2.407193
NN	C	2.376361
S	PP	2.374117
VOC	NNK	2.320469
VOC	NN	2.277056
VOC	NNP	2.216471
NNP	NNP	2.126588
PRPR	PP	2.073439
ADJ	C	2.069595
PROP	VB	1.989484

Suite à la page suivante

Table 3.2 – *Suite de la page précédente*

étiquette 1	**étiquette 2**	**CRF transition**
NN	NUM	1.824346
IN	PUNC	1.692434
S	NN	1.682533
REL	FOC	1.662785
ADV	PRED	1.661423
REL	SYM	1.611566
VOC	REL	1.573148
REL	VBP	1.571219
NN	DT	1.511112

TABLE 3.2 : : Les 20 premières paires des étiquettes qui ont des fortes caractéristiques de transition

POS	**Précision**	**Rappel**	**F-mesure**
S	0.973	0.980	0.977
ADJ	0.831	0.711	0.766
PUNC	1.000	1.000	1.000
VB	0.928	0.929	0.928
NN	0.902	0.946	0.924
ADV	0.786	0.681	0.730
DT	0.833	0.885	0.858
VBP	0.822	0.714	0.764
C	0.776	0.892	0.830
REL	0.914	0.762	0.831
PROR	0.914	0.779	0.841
PP	0.984	0.940	0.961
NNP	0.848	0.779	0.812
PRPR	0.894	0.923	0.909
SYM	1.000	1.000	1.000
INT	0.630	0.895	0.739
NEG	0.977	1.000	0.988
PDEM	0.625	0.375	0.455
PRED	1.000	0.500	0.667
IN	1.000	0.714	0.833
PROT	0.500	0.375	0.429
NUM	0.973	0.973	0.973
VOC	0.893	0.962	0.926
NNK	0.947	0.857	0.900

Suite à la page suivante

Table 3.3 – *Suite de la page précédente*

POS	**Précision**	**Rappel**	**F-mesure**
PPOS	0.800	0.600	0.686
FOC	0.722	0.684	0.703
ROT	0.333	0.100	0.154
Moyenne pondérée	0.914	0.915	0.913

TABLE 3.3 : Résultats de classification du modèle CRF

Selon les résultats indiqués dans le tableau 3.3, nous avons trouvé la moyenne pondérée 0.934. Ces résultats peuvent améliorer si nous avons enrichit notre corpus par d'autres phrases pour l'équilibrer.

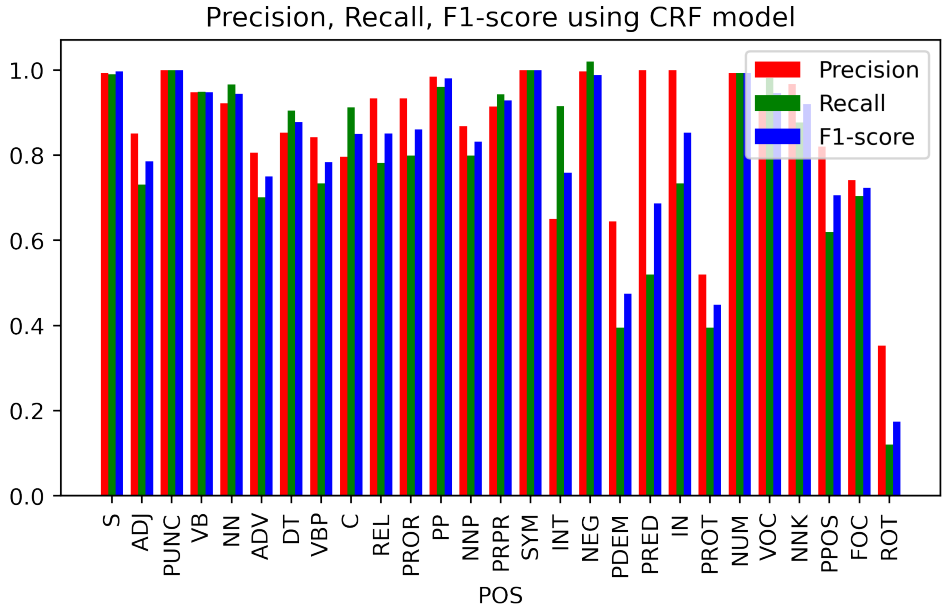

FIGURE 3.3 – Précision, rappel est F-mesure du modèle CRF

La figure 3.3 montre les variations de la précision, du rappel et du score F1 après l'entraînement du modèle des champs aléatoires conditionnels (CRF) qui donne la même variation pour les trois valeurs.

3.2.3 Techniques d'apprentissage profond

3.2.3.1 Description des modèles

Dans le domaine de TALN nous trouvons plusieurs techniques de classification à savoir les techniques d'apprentissage profond. Dans notre cas nous avons utilisé quatre modèles

basés sur l'apprentissage profond ; RNN, LSTM, GRU et Bi-LSTM ; pour étiqueter les mots dans une phrase écrit en Tifinagh.

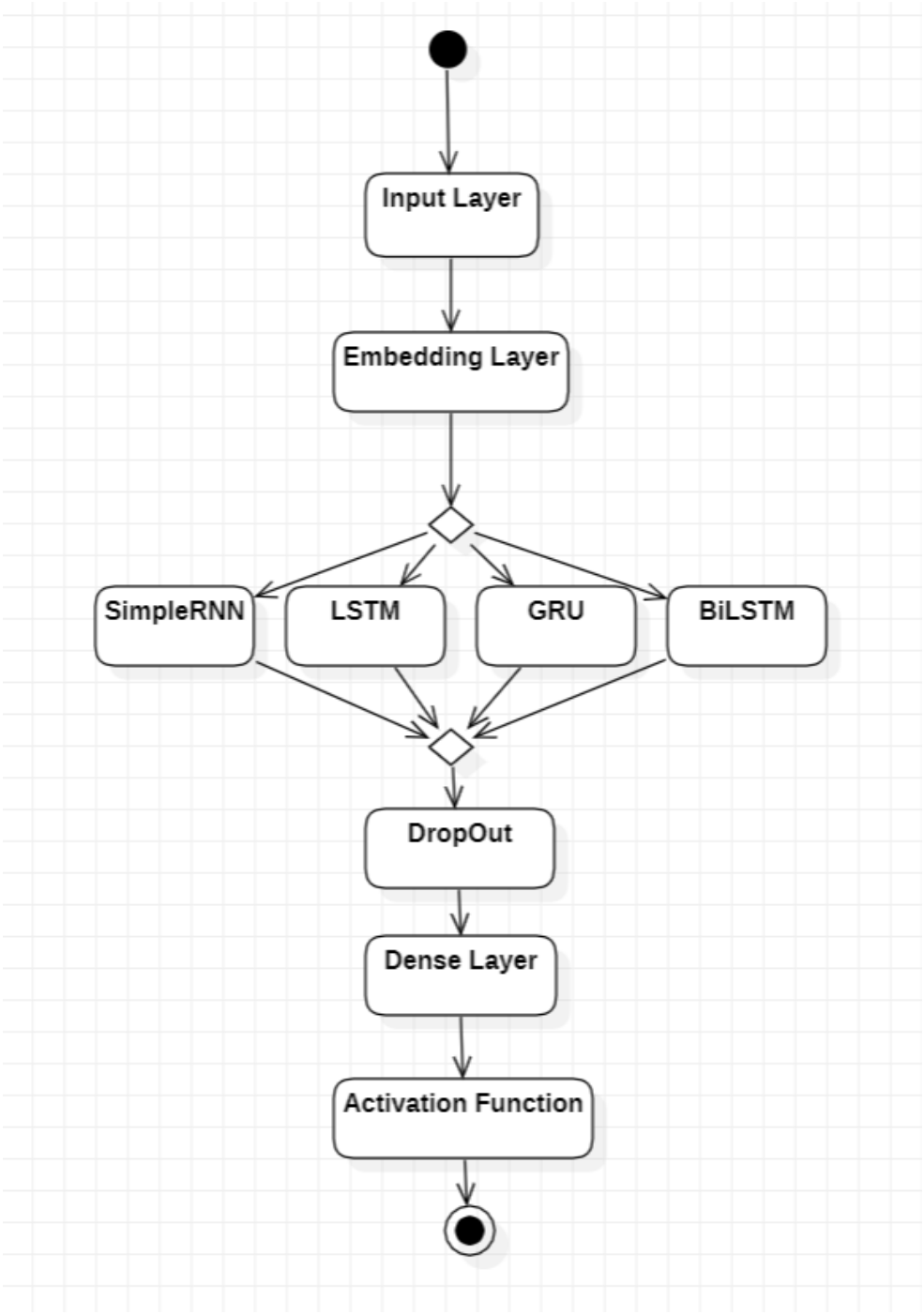

FIGURE 3.4 – Architecture des modèles de POS

Tous ces modèles utilisent la même architecture de la figure 3.4 utilisé pour schématiser les différentes couches des modèles. On peut y voir la structure de chaque couche et sa configuration. Définir le modèle comme séquentiel signifie qu'il est construit par une pile

de couches. Le modèle est composé des couches suivantes :

- Input Layer : elle accepte des vecteurs de forme.
- Embedding Layer : Cette couche est destinée à compresser le dictionnaire de mots et aide également à trouver des relations entre les mots. Cette couche vise à représenter les mots à l'aide d'une représentation vectorielle dense avec leur signification relative.
- RNN/LSTM/GRU/Bi-LSTM : Dans cette étape nous avons utilisé SimpleRNN qui souffre du problème du gradient évanescent, et nous avons choisi comme solutions les cellules LSTM, les GRU visent à résoudre le problème du gradient évanescent. Avec ces cellules récurrentes, il est capable de se souvenir des informations des itérations passées. Dans ce modèle, ce type de réseau devrait être capable de se souvenir des informations des mots traités dans le passé, puis de les utiliser dans les mots suivants. Le modèle utilise des GRU car ils nécessitent moins de puissance de calcul que les LSTM, ce qui conduit à un moins de temps de formation. Cette couche utilise 64 neurones dans chaque direction. Grâce à la couche bidirectionnelle, le modèle traite les séquences d'entrée dans l'ordre avant et arrière, en fusionnant les connaissances des deux directions à la sortie.
- Dropout Layer : Cette couche a pour but d'éviter overfitting et la dépendance entre les neurones en coupant 30% (ratio de 0,3) des connexions choisies au hasard entre les neurones à chaque époque.
- Dense Layer : Cette couche peut être comprise comme l'application d'un ensemble différent de balises pour chaque élément de la séquence d'entrée, afin qu'elle puisse être classée dans l'une des classes.
- Fonction d'activation : Le modèle utilise la fonction d'activation softmax en raison de la nature du problème : classification multi-classes.

3.2.3.2 Résultats et discussion

Input Layer est la couche d'entrée, elle accepte les vecteurs de (28) éléments et correspond à notre variable X (nous avons 28 tokens dans chacune de nos séquences train et test). Ensuite, nous avons embedding layer, cette couche va prendre chacun de nos tokens/mots et le transformer en un vecteur dense de taille 100. Il s'agit d'une table de consultation géante (ou d'un dictionnaire) avec les mots-clés comme clés et les vecteurs réels comme valeurs. Cette table de consultation peut être entraînée, c'est-à-dire qu'à chaque époque de l'apprentissage du modèle, nous mettons à jour ces vecteurs pour qu'ils correspondent à l'entrée. Après embedding layer, notre entrée passe d'un vecteur de longueur 28 à une matrice de taille (28, 100). Chacun des 28 token a maintenant un vecteur de taille 100. Une fois que nous avons cela, nous pouvons utiliser la couche (RNN, LSTM [45], GRU [56] ou Bi-LSTM). La sortie de cette couche est une matrice de taille (28, 64).

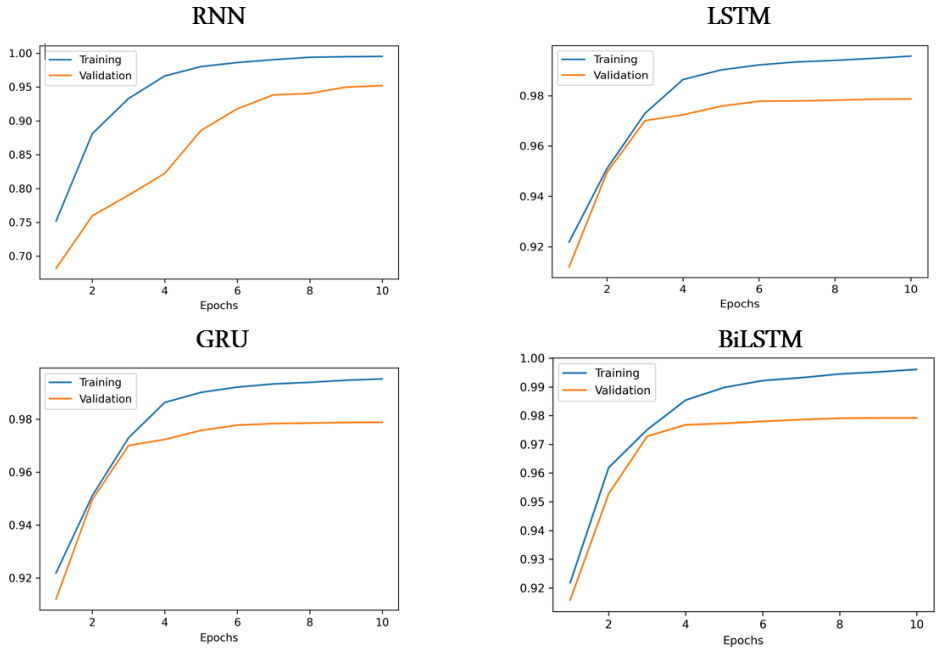

FIGURE 3.5 – La précision de l'entraînement et du validation dans les différents modèles

La figure 3.5 montre l'évolution de la précision à chaque époque lors de l'apprentissage du chaque modèle. Elle augmente rapidement au début du processus d'apprentissage, puis se stabilise progressivement jusqu'à ce qu'il n'y ait plus d'amélioration insignifiante de la précision. Le processus d'entraînement a été répété un nombre différent d'époques, en essayant de trouver un équilibre entre le temps d'entraînement et les résultats obtenus, en évitant overfitting. Le nombre optimal d'époques a été trouvé à 10. La précision obtenue pour l'ensemble d'entraînement est de 99.99% en utilisant les couches RNN ou LSTM mais pour la couche GRU nous avons trouvé 99.53% et Bi-LSTM 99.50%. Cependant pour l'ensemble de test, nous obtenons une précision de 95.20% pour la couche RNN, 97.88% pour la couche LSTM, 97.89% pour la couche GRU et 97.92% pour la couche Bi-LSTM, ces valeurs sont inférieures à celles obtenues avec l'ensemble d'entraînement. Nous pensons que ces résultats sont acceptables car ils dépassent les 90% dans tous les modèles, mais le modèle qui contient la couche Bi-LSTM est plus performant que les autres modèles.

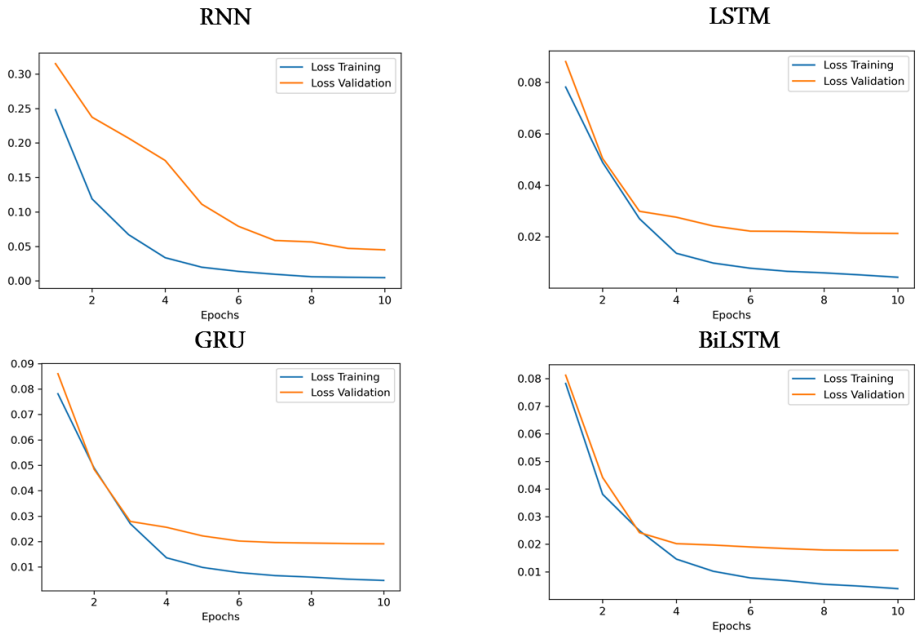

FIGURE 3.6 – La perte de l'entraînement et du validation dans les différents modèles

La perte obtenue pour l'ensemble d'entraînement est de 0,01% en utilisant les couches RNN et LSTM, pour la couche GRU nous avons obtenu 0.47%, dans le cas de la couche Bi-LSTM nous avons trouvé 0.5%, pour l'ensemble de test nous avons trouvé 4.80% par la couche RNN, 2.12% par LSTM, 2.11% par GRU et 2.08% par la couche Bi-LSTM.

Modèles	**Nombre des paramètres**	**Temps d'entraînement (min)**
RNN	821 740	36.34
LSTM	853 420	52.22
GRU	844 009	53.38
Bi-LSTM	898 665	75.65

TABLE 3.4 : Comparaison des modèles en termes de nombre des paramètres et le temps d'entraînement

Ce qui concerne le temps d'exécution pendant l'entraînement, le tableau 3.4 indique l'efficacité du modèle RNN. Dans la phase d'entraînement, le modèle de RNN est plus rapide par rapport au modèle LSTM par 30.40%, par rapport au modèle GRU par 31.92%, et par rapport au modèle Bi-LSTM par 51.96%. nous observons que le modèle LSTM est rapide par rapport au modèle GRU par 2.17%, par rapport au modèle Bi-LSTM par 30.97%. Le modèle GRU est rapide par rapport au modèle Bi-LSTM par 29.43%.
Cette différence entre les modèles dans le temps d'entraînement est expliquée par la différence dans le nombre des paramètres utilisés dans chaque modèles ; dans le modèle RNN,

nous avons utilisé 821,740 paramètres, dans le modèle LSTM, nous avons utilisé 853,420 paramètres, dans le modèle GRU, nous avons utilisé 844,009 paramètres, et dans le modèle Bi-LSTM, nous avons utilisé 898,665 paramètres.

Modèles	**DT**	**CRF**	**RNN**	**LSTM**	**GRU**	**Bi-LSTM**
Accuracy	87.60%	93.40%	95.20%	97.88%	97.89%	97.92%

TABLE 3.5 : Comparaison des modèles en termes de précision(Accuracy)

Comparant les modèles d'aprentissage automatique et d'apprentissage profond en terme d'accuracy on observe que les modèles d'apprentissage profond atteint des meilleurs résultats, comme il indique le tableau 3.5, le modèle de l'arbre de décision atteint 87.60%, 93.40% pour CRF, ce qui concerne RNN on a obtenu 95.20% dans l'accuracy, 97.88% pour LTSM, 97.89% pour GRU et Bi-LSTM atteint 97.92%. Bi-LSTM surpasse les autres structures basées sur les RNN, a cause de déploiement du mécanisme bidirectionnel.

3.3 Conclusion

L'étiquetage de langue Amazigh (POS) est l'un des sujets les plus intéressants traités par plusieurs chercheurs, en particulier les chercheurs marocains. Dans ce chapitre, nous avons utilisé des algorithmes d'apprentissage automatique et profond pour étiqueter les mots écrits en caractères tifinagh. Nous avons comparé tous les modèles entre eux (les méthodes d'apprentissage profond et les méthodes d'apprentissage automatique), nous avons constaté que le modèle RNN (95.20%) est meilleur que CRF (93.40%) et arbre de décision (87.60%) (algorithmes d'apprentissage automatique) , ces modèles sont testés sur les mêmes données et dans les mêmes conditions.
En comparant les algorithmes d'apprentissage profond entre eux, nous constatons que le modèle RNN (95.20%) est faible par rapport au modèle LSTM (97.88%), par rapport au modèle GRU (97.89%), et par rapport au modèle Bi-LSTM (97.92%) en termes de précision, mais que le temps d'entraînement RNN est le meilleur. Ceci est dû au fait que le gradient de la fonction de perte décroît exponentiellement avec le temps (le problème du gradient évanescent). Cela montre que les réseaux LSTM et GRU peuvent capturer la connaissance d'une langue car les unités LSTM comprennent une "cellule mémoire" qui peut maintenir l'information en mémoire pendant de longues périodes. Dans le chapitre suivant, nous appliquerons les modèles de prédictions de POS pour corriger les erreurs d'un OCR.

Chapitre 4

Correction de la sortie d'un OCR

4.1 Introduction

Avec les progrès de l'innovation et de la vitesse de traitement, un nombre croissant de calculs compliqués pour les cadres de reconnaissance optique de caractères (OCR), y compris l'IA et les organisations neuronales, sont proposés. [57]
La reconnaissance optique de caractères (OCR) permet de transformer une image en un texte éditable. Il s'agit d'une technique de numérisation de textes imprimés et écrits à la main [58]. De nombreuses applications, dont la reconnaissance des plaques minéralogiques, la vérification de livres et la transformation continue de textes transcrits, bénéficient de l'OCR. [59] Malheureusement, les résultats donnés par l'OCR ne sont pas toujours satisfaisants ; ils contiennent des erreurs influençant le sens des phrases. Ces erreurs sont divisées en plusieurs types : (i) Caractères manquants : le résultat de l'OCR contient plusieurs caractères de moins que le nombre de caractères de l'image à reconnaître. (ii) Ajout de caractères : le résultat de l'OCR contient plusieurs caractères supérieurs au nombre de caractères figurant dans l'image à reconnaître. (iii) Modification de caractères : le résultat de l'OCR contient plusieurs caractères égaux au nombre de caractères de l'image à reconnaître, mais certains sont modifiés par d'autres caractères différents de l'origine. Pour résoudre ce problème, un post-traitement doit être ajouté. A ce niveau, nous proposons d'utiliser une nouvelle approche qui s'applique en deux étapes ; elle commence par la détection des erreurs, puis elle s'attaque à la correction syntaxique et sémantique de la sortie OCR, cette approche se base sur la fréquence de deux mots corrects dans la phrase et sur une technique récursive. Cette approche commence par le calcul de la fréquence de chaque deux mots successifs dans les corpus, les mots qui ont la plus grande fréquence construisent un centre de correction, puis elle commence à corriger tous les mots en utilisant la technique récursive qui sera décrite dans la section suivante. Cette approche appartient au domaine du traitement du langage naturel (NLP), c'est-à-dire une branche de l'intelligence artificielle qui analyse, comprend et génère les langages naturels utilisés par les humains pour interagir avec les ordinateurs dans des contextes écrits et parlés en utilisant des langages humains naturels plutôt que des langages informatiques [60]. Dans la littérature, nous constatons que le domaine NLP est utilisé dans plusieurs langues, à savoir : L'anglais [61] Le français

[62] L'arabe [63]. Par contre, la langue amazighe, qui utilise les caractères tifinagh, n'a pas bénéficié des avantages offerts par ce domaine. Ce manque nous a motivé à aborder ce domaine pour améliorer les résultats de l'OCR pour les caractères Tifinagh. Le Tifinagh [64] est l'ensemble des alphabets utilisés par la population amazighe. L'Institut Royal de la Culture Amazighe (IRCAM) a normalisé l'alphabet Tifinagh de trente-trois caractères comme le montre la figure 4.1.

FIGURE 4.1 – Caractères de Tifinagh (IRCAM)

4.2 Système de reconnaissance optique de caractères

Expliquer le déroulement de la recherche de manière chronologique, y compris la conception de la recherche, la procédure de recherche (sous forme d'algorithmes, de Pseudocode ou autre), la façon de tester et l'acquisition des données [61]. La description du déroulement de la recherche doit être étayée de références, afin que l'explication puisse être acceptée scientifiquement [62]. L'avancement de la reconnaissance des formes s'est accéléré récemment en raison des nombreuses applications émergentes qui sont non seulement difficiles, mais aussi plus exigeantes sur le plan informatique, comme évidentes dans l'OCR, [65] la classification des documents, la vision par ordinateur [66], l'exploration de données, la reconnaissance des formes et l'authentification biométrique, par exemple. L'espace de l'OCR est en train de devenir une pièce indispensable des scanners d'archives et est utilisé dans de nombreuses applications comme le traitement postal, l'accusé de réception de script, la banque, la sécurité (par exemple la confirmation de visa) et la preuve de distinction de langue. L'exploration dans ce domaine a progressé pendant plus de 50 ans et les résultats ont été surprenants avec des taux de reconnaissance fructueux pour les caractères imprimés dépassant près de 100%, avec d'énormes améliorations dans l'exécution pour la reconnaissance des personnes écrites à la main en cursive où les taux de reconnaissance ont dépassé l'empreinte de 90%. De nos jours, de nombreuses associations s'appuient sur les

cadres OCR pour se passer des connexions humaines afin d'améliorer l'exécution et la compétence [67]. Dans le cadre du travail actuel, un système de reconnaissance de caractères est présenté pour reconnaître les caractères Tifinagh extraits d'images/dessins implantés dans des enregistrements de texte, par exemple, des images de cartes de visite [68]. La Figure 4.2 décrit les étapes de notre OCR, il prend comme image d'entrée, il commence par l'extraction du texte, puis la phase de binarisation, la phase de segmentation et la dernière est la reconnaissance enfin il donne comme sortie un fichier texte.

4.2.1 Image acquisition

L'OCR proposé commence par le processus d'acquisition d'image [59], c'est la première étape de l'OCR, qui consiste à obtenir une image numérique et à la convertir dans une forme appropriée qui peut être facilement manipulée par un ordinateur. Cela peut inclure la quantification ainsi que la compression de l'image [69]. Un cas particulier de quantification est la binarisation, qui n'implique que deux niveaux d'image [70]. Dans la plupart des cas, l'image binaire est suffisante pour caractériser l'image. La compression elle-même peut être avec ou sans perte. Un aperçu des différentes techniques de compression d'image dans [71]. Cet OCR prend n'importe quelle image dactylographiée contenant des caractères Tifinagh au format ". png" ou ". jpg".

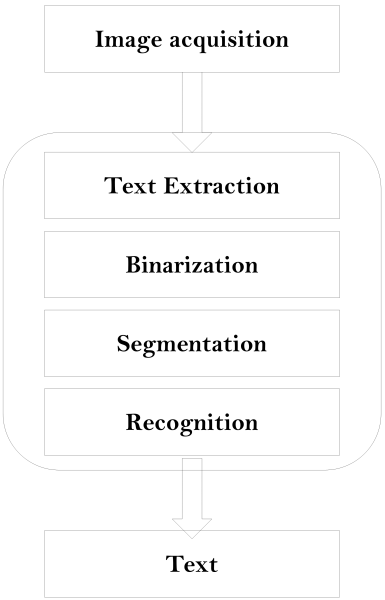

FIGURE 4.2 – Schéma du système OCR

4.2.2 Extraction du texte

Le système de filtrage permet d'obtenir une image avancée du premier enregistrement. Dans l'OCR, on utilise des scanners optiques qui, pour la plupart, comprennent un instrument de véhicule ainsi qu'un gadget de détection qui convertit l'énergie lumineuse en

niveaux d'obscurité. [72] Les documents imprimés sont généralement composés de caractères foncés sur un fond blanc. Par conséquent, lors de l'exécution de l'OCR, il n'est pas rare que l'image décalée soit transformée en une image à deux niveaux de contraste élevé. Ce cycle, connu sous le nom de seuillage, est régulièrement effectué sur le scanner pour économiser de l'espace mémoire et des efforts de calcul. Les problèmes liés au seuillage sont illustrés à la figure 4.2.
Le système de seuillage est important car les effets de l'accusé de réception qui l'accompagne dépendent entièrement de la nature de l'image à deux niveaux. En tout état de cause, le seuillage effectué sur le scanner est normalement exceptionnellement simple. [73] Une limite fixée est utilisée, où les niveaux de dim sous ce bord sont censés être sombres et les niveaux au-dessus sont censés être blancs. Pour une archive à haut équilibre avec une fondation uniforme, une limite fixée préchoisie peut être sufficient. Dans tous les cas, une tonne d'archives expérimentées pratiquement parlant ont une portée quelque peu énorme intéressant. Dans ces cas, des stratégies plus modernes de seuillage sont nécessaires pour obtenir un résultat décent [74].
Les meilleures techniques de seuillage sont généralement celles qui peuvent différer la limite sur l'enregistrement en s'ajustant aux propriétés du voisinage comme la différenciation et la splendeur. Quoi qu'il en soit, ces techniques reposent généralement sur un filtrage échelonné de l'enregistrement, ce qui nécessite davantage de mémoire et de limites de calcul. C'est pourquoi ces procédures sont rarement utilisées dans les cadres d'OCR, bien qu'elles permettent d'obtenir de meilleures images [67].

4.2.3 Binarisation

Une locale de texte révisée en biais est binarisée en utilisant une technique de binarisation simple mais efficace que nous avons créée avant de la diviser [68]. Essentiellement, il s'agit d'une forme plus développée de la stratégie de binarisation de Bernsen. Dans sa technique, la méthode de jonglage des nombres pour les niveaux d'obscurité les plus extrêmes (Gmax) et les plus bas (Gmin) autour d'un pixel est prise comme limite pour binariser le pixel. Dans le calcul actuel, les huit voisins rapides autour du pixel soumis à la binarisation sont également pris comme considération centrale pour la binarisation. Ce type d'approche est particulièrement utile pour interfacer les pixels frontaux détachés d'une personne [75].

4.2.4 Segmentation

La segmentation est la séparation de caractères ou de mots. La plupart des calculs de reconnaissance optique des personnes fragmentent les mots en caractères séparés, qui sont perçus séparément [67]. Typiquement, cette division est effectuée en séparant chaque partie associée, c'est-à-dire chaque région sombre associée. Cette procédure n'est pas difficile à réaliser, mais des problèmes surviennent si les caractères se touchent ou encore si les caractères sont divisés et comprennent plusieurs sections. Les principaux problèmes liés à la division peuvent être divisés en quatre groupes :

— Extraction des caractères qui se touchent et des caractères fragmentés.
— Distinguer le bruit du texte.
— Confondre les graphiques ou la géométrie avec le texte.
— Confondre le texte avec les graphiques.

Dans notre cas, le processus de segmentation est basé sur le seuillage par histogramme. Nous signifierons l'histogramme des estimes de pixels par $h_0, h_1, ..., h_N$ où h_K spécifie la quantité de pixels dans une image avec l'estime de niveau de gris k et N est la plus grande estime de pixel (régulièrement 255).
Tout d'abord, il faut supposer une incitation potentielle pour le bord. À partir de là, on détermine les côtés supérieurs moyens des pixels dans les deux classes créées en utilisant cette limite. La limite est repositionnée pour se situer précisément entre les deux méthodes. Les qualités moyennes sont déterminées une fois de plus, et une autre limite est obtenue, jusqu'à ce que le bord cesse d'évoluer dans l'estime [76].

4.2.5 Reconnaissance

La reconnaissance est la dernière phase du système OCR qui est utilisée pour identifier le contenu segmenté. Dans cette étape, les coefficients de corrélation sont utilisés dans la classification. Le coefficient de corrélation est traité à partir des informations de l'exemple ; il estime la force et la portée d'un lien entre deux facteurs. Un coefficient de relation est un nombre compris entre 0 et 1. S'il n'y a pas de relation entre les qualités prévues et les qualités réelles, le coefficient de relation est égal à 0 ou exceptionnellement faible (les qualités prévues ne sont pas plus excellentes que des chiffres irréguliers). Au fur et à mesure que la force du lien entre les qualités anticipées et la valeur réelle augmente, le coefficient de liaison augmente également. Une correspondance idéale donne un coefficient de 1,0. Ainsi, un résultat supérieur est comparé au coefficient de corrélation le plus élevé [77]. Corr2 calcule le coefficient de corrélation en utilisant :

$$r = \frac{\sum_m \sum_n \left(A_{mn} - \bar{A}\right)\left(B_{mn} - \bar{B}\right)}{\sqrt{\left(\sum_m \sum_n \left(A_{mn} - \bar{A}\right)^2\right)\left(\sum_m \sum_n \left(B_{mn} - \bar{B}\right)^2\right)}} \tag{4.1}$$

où

— A et B sont deux matrices
— $\bar{A}$ et $\bar{B}$ sont les moyennes des valeurs de A et B.
— m : nombre de lignes
— n : nombre de colonnes

4.3 Le modèle du canal bruyant

Nous présentons ici le modèle du canal bruyant et conseillons la manière la plus idéale de l'appliquer à la tâche de reconnaissance et de correction des fautes d'orthographe. Le modèle du canal bruyant a été appliqué à la tâche de correction orthographique à peu près à

la même époque par des recherches artificielles AT et T Bell [78] et IBM Watson Research [79]. La nature du modèle du canal fort de la figure 4.3 est de traiter le mot du canal bruyant incorrectement orthographié comme un mot correctement orthographié avait été "ravagé" en étant utilisé une procédure de correspondance clamorante. Ce canal présente les "uproar" sous forme de remplacements ou de modifications diverses des lettres, ce qui rend difficile de voir le mot "certifié". L'objectif, d'ici là, est de développer un modèle du canal tapageur. Étant donné ce modèle, nous trouvons alors le mot authentique en faisant passer chaque déclaration de la langue par le modèle du canal bruyant et nous voyons laquelle se rapproche le plus du mot mal orthographié [80].
Dans le modèle du canal bruyant, nous envisageons que la structure des surfaces que nous voyons est en réalité un type "mal formé" d'un mot unique passé par un canal bruyant. Le décodeur passe chaque théorie par un modèle de ce canal et choisit le mot qui correspond le mieux au mot bruyant de surface [81].

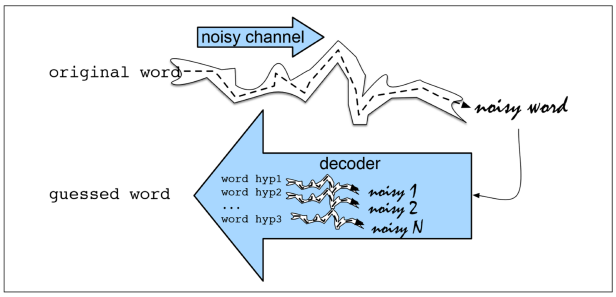

FIGURE 4.3 – Modèle du canal bruyant

4.3.1 Extraction des candidats

Ce modèle du canal bruyant est une sorte d'inférence bayésienne. Nous voyons une perception x (un mot mal orthographié) et le travail consiste à trouver le mot w qui a créé ce mot mal orthographié. Parmi tous les mots potentiels du jargon V, nous devons trouver le mot w dans une mesure telle que $P(w|x)$ est le plus élevé.

$$\widehat{w} = \text{argmax}_{w \in V} P(w \mid x) \tag{4.2}$$

La fonction $argmax_x f(x)$ signifie " x tel que f(x) est maximisé ". L'équation 4.2 de cette manière implique que parmi tous les mots du jargon, nous avons besoin du mot spécifique qui augmente le côté droit $P(w|x)$.

4.3.2 Classification bayésienne

L'instinct de classification bayésienne consiste à utiliser la règle de Bayes pour transformer l'équation 4.2 en un ensemble de probabilités différentes. La règle de Bayes est introduite dans l'équation 4.3 ; elle nous offre un moyen de ramener toute probabilité contingente

$P(a|b)$ en trois probabilités différentes :

$$P(a \mid b) = \frac{P(b \mid a)P(a)}{P(b)} \tag{4.3}$$

on peut alors substituer l'équation 4.3 dans l'équation 4.2 pour obtenir l'équation 4.4 :

$$w = \text{argmax}_{w \in V} \frac{P(x \mid W)P(w)}{P(x)} \tag{4.4}$$

Il peut utilement rationaliser l'équation 4.4 en abandonnant le dénominateur $P(x)$. Pour quelle raison ? Puisque nous choisissons un mot d'ajustement potentiel, toutes choses étant égales par ailleurs, nous traiterons $\frac{P(x|w)P(w)}{P(x)}$ pour chaque mot. De toute façon, P(x) ne change pas pour chaque mot ; nous nous rapprochons continuellement du mot sans doute pour l'erreur surveillée équivalente, qui doit avoir une même probabilité P(x). Dans cette optique, nous pouvons choisir le mot qui augmente la difficulté [80].

$$w = \text{argmax}_{w \in V} P(x \mid w)P(w) \tag{4.5}$$

Pour résumer, le modèle du canal bruyant dit que nous avons un certain mot de base évident w, et que nous avons un canal bruyant qui modifies le mot en une certaine structure de surface remarquée incorrectement orthographiée concevable. La probabilité où le modèle du canal bruyant délivrant un groupement de perception spécifique x est démontrée par $P(x|w)$. La probabilité antérieure d'un mot secret est démontrée par $P(w)$. Nous pouvons traiter le mot plausible le plus ancien $\widehat{w}$ en tenant compte du fait que nous avons remarqué une orthographe incorrecte x en augmentant l'antériorité $P(w)$ et la vraisemblance $P(x|w)$ et en choisissant le mot pour lequel cet élément est le plus important.
L'approche par le canal bruyant permet de rectifier les fautes d'orthographe sans mot en prenant n'importe quel mot qui n'est pas dans la référence du mot d'orthographe, en créant une liste de mots qui se rapprochent, en les positionnant selon l'équation 4.5 et en choisissant le mot positionné le plus remarquable. Nous pouvons ajuster l'équation 4.5 pour faire allusion à cette liste de mots concurrents plutôt qu'au jargon complet V comme par [80].

$$\widehat{W} = \text{argmax}_{w \in C} \overbrace{P(x \mid w)}\overbrace{P(w)} \tag{4.6}$$

4.4 L'approche proposée pour corriger les erreurs d'orthographe

4.4.1 Génération des candidats

La correction d'un mot erroné nécessite une liste de candidats à sélectionner parmi eux. Dans cette partie, nous traiterons la correction de mots sur la base d'un dictionnaire de mots de Tifinagh, en calculant la ressemblance entre les mots afin d'avoir la liste de

candidats prenant les mots dont la ressemblance est maximale. La ressemblance entre deux mots est le nombre de lettres communes entre les deux lettres dans la même position, qui est calculé par l'équation 4.7.

$$\begin{aligned}\text{ress}\,(w_1, w_2) =| \{c_i : (c_i \in W_1^k \wedge c_i \in w_2^k) \vee (c_i \in W_1^k \wedge c_i \in w_2^{k\pm1}) \\ \vee (c_i \in w_1^{k\pm1} \wedge c_i \in W_2^k) \vee (c_i \in w_1^{k\pm1} \wedge c_i \in w_2^{k\pm1}) \vee\} |\end{aligned} \tag{4.7}$$

avec

— c_i : Le caractère d'indice i dans le mot .

— w_j^k : La position k dans le mot j.

En OCR, nous pouvons trouver certains types d'erreurs, des mots erronés par suppression, insertion, transposition ou substitution. Le tableau 4.1 nous montre que les deux types de mots erronés (par transposition et substitution), ont le même nombre de caractères du mot correct, par contre, les mots effacés par suppression ont un nombre de caractères inférieur aux caractères du mot correct, et les mots effacés par l'insertion ont un nombre de caractères supérieur au nombre de caractères du mot correct.

Mots erronés	**Mots corrects**	**Types d'erreurs**
₀O₀O	₀Λ O₀O	suppression
₀ΛΛO₀O	₀Λ O₀O	insertion
₀OΛ₀O	₀Λ O₀O	transposition
₀ΛΘ₀O	₀Λ O₀O	substitution

TABLE 4.1 : Types des mots erronés

Pour corriger un mot erroné, on calculera sa ressemblance avec tous les mots du dictionnaire qui ont la même taille ou une taille ±1.

Nous pouvons retrouver tous les mots qui ont une ressemblance maximale avec le mot erroné, en utilisant le maximum de la ressemblance par l'équation 4.8.

$$words\_corrects(w) = \{w_i : max\,\{ress_i(w, w_i)\}\} \tag{4.8}$$

avec :

— w le mot corrigé

— w_i : sont les mots du dictionnaire qui ont une taille égale à la taille ou à la taille ±1 de w

4.4.2 Correction des erreurs à l'aide de la technique des n-grammes

4.4.2.1 Fréquence des mots

Pour corriger une phrase, nous devons commencer la correction par des mots corrects. Par conséquent, nous déterminerons la position du mot appartenant au dictionnaire, de

telle sorte que le mot qui suit appartient également au dictionnaire, par l'équation 4.9.

$$\text{posw}_{\ i} = \{i : w_i \in \text{ dictionnaire } \wedge w_{i+1} \in \text{ dictionnaire } \} \tag{4.9}$$

avec w_i le mot de la phrase a la position i. Pour $posw_i = \emptyset$, on va recalculer $posw_i$ par l'équation 4.10.

$$\text{posw}_{\ i} = \{i : w_i \in \text{ dictionnaire } \} \tag{4.10}$$

Nous calculons la fréquence des mots w_i dans le corpus sachant que w_{i+1} est consécutif à w_i par l'équation 4.11.

$$\text{freqw}_i = \left\{n_{i/i+1}/N : i \in \text{posw}\right\} \tag{4.11}$$

Avec :

— $n_{i/i+1}$ est le nombre d'occurrences de w_i dans le corpus suivi par w_{i+1}

— N est le nombre total de mots du corpus

si nous utilisons l'équation 4.10, nous calculerons la fréquence des mots par l'équation 4.12.

$$\text{freqw}_i = \{n_i/N : i \in \text{posw}\} \tag{4.12}$$

avec :

— n_i : le nombre d'occurrences des mots dans le corpus

— N : nombre total de mots du corpus

Après le calcul des fréquences des mots, nous allons rechercher la fréquence maximale. Ensuite, nous retournerons la position du mot qui a une fréquence maximale par l'équation 4.13. Par conséquent, le processus de correction se concentre sur cette position (appelée : le centre de correction).

$$\text{pos} = \{i : argmax(freqw_i)\} \tag{4.13}$$

4.4.2.2 Correction de la phrase

La fréquence des mots que nous avons calculé dans la section précédente, nous a permis de corriger chaque mot de la phrase, sur la base des mots existants dans le dictionnaire. Maintenant nous allons calculer la fréquence des mots qui sont proches du mot erroné, en utilisant le résultat de l'équation 4.13, par un calcul qui prend en considération n mots corrects après le mot erroné ou avant le mot erroné par l'équation 4.14 et l'équation 4.15.

$$\text{freqw}\ {}^{L}_{i/\ \text{pos}} = \left\{\frac{n_{i/\ \text{pos}}}{N} : w_i \in words\_corrects(\bar{w})\right\} \tag{4.14}$$

Avec :

— $n_{i/pos}$ le nombre d'occurrences du mot w_i dans le corpus sachant qu'il est suivi par $w_{i+1}...w_{pos}$.

— N : Le nombre total de mots du corpus

— $\bar{w}$: Le mot à corriger

$$\text{freqw}\ {}^{R}_{i/\ \text{pos}} = \left\{\frac{n_{i/\ \text{pos}}}{N} : w_i \in words\_corrects(\bar{w})\right\} \tag{4.15}$$

Avec :

- $n_{i/pos}$ le nombre d'occurrences du mot w_i dans le corpus sachant qu'il qu'il a précédé par $w_{pos}...w_{i-1}$.
- N : Le nombre total de mots du corpus
- $\bar{w}$: Le mot à corriger

Pour les équations 4.10 et 4.11, nous calculons le maximum des fréquences à gauche ou à droite comme par l'équation 4.16 et l'équation 4.17.

$$\text{maxfreq}\ {}^{L}_{i/\ \text{pos}} = \left\{ max(\ \text{freqw}\ {}^{L}_{i/\ \text{pos}}\ ) \right\} \tag{4.16}$$

$$\text{maxfreq}\ {}^{R}_{i/\ \text{pos}} = \left\{ max(\ \text{freqw}\ {}^{R}_{i/\ \text{pos}}\ ) \right\} \tag{4.17}$$

Si nous avons $\text{maxfreq}\ {}^{R}_{i/\ \text{pos}} = 0$, nous allons appliquer la technique récursive : l'incrémentation de pos ($pos = pos + 1$) jusqu'à ce que $\text{maxfreq}\ {}^{R}_{i/\ \text{pos}} > 0$. De même, si nous avons $\text{maxfreq}\ {}^{L}_{i/\ \text{pos}} = 0$, nous appliquerons la technique récursive : la décrémentation de $pos(pos = pos - 1)$ jusqu'à ce que $\text{maxfreq}\ {}^{L}_{i/\ \text{pos}} > 0$. Pour corriger un mot erroné, nous allons utiliser deux formules en fonction de sa position par rapport au centre de correction. si nous avons la position du mot erroné supérieur au centre de correction : :

$$correct^{R}(\bar{w}) = \left\{ w_i : \text{maxfreq}^{R}_{i/pos} = \ \text{freqw}\ {}^{R}_{i/pos}\ \text{and}\ w_i \in\ \text{words\_corrects}\ {}^{R}\bar{w} \right\} \tag{4.18}$$

Si nous avons la position du mot erroné inférieur au centre de correction :

$$correct^{L}(\bar{w}) = \left\{ w_i : \text{maxfreq}^{L}_{i/pos} = \ \text{freqw}\ {}^{L}_{i/pos}\ \text{and}\ w_i \in\ \text{words\_corrects}\ {}^{L}\bar{w}) \right\} \tag{4.19}$$

avec $\bar{w}$ est le mot erroné. Finalement, nous pouvons corriger une phrase, en utilisant la correction du mot à gauche ou à droite et la technique récursive, par l'équation 4.20.

$$correct(\ \text{sentence}\ ) = \left\{ \text{correct}^{L}\left(w_{i<\ \text{pos}}\right) ; \text{correct}^{R}\left(w_{i>\ \text{pos}}\right) \right\} \tag{4.20}$$

- $w_{i<pos}$: Les mots de la phrase se situent avant que le mot existe dans le centre de correction
- $w_{i>pos}$: Les mots de la phrase se trouvent après que le mot existe dans le centre de correction.

4.4.3 Correction des erreurs à l'aide d'étiquetage POS

La correction des erreurs d'orthographe en utilisant POS est une technique basée sur la prédiction de part-of-speech des mots dans la phrase à corriger.

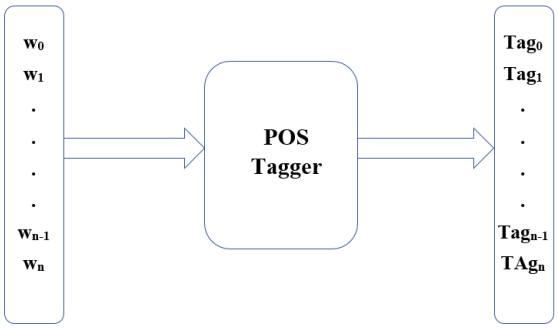

FIGURE 4.4 – Schéma de prédiction des tags POS

Soit un phrase P contient n mots $P = w_0, w_1, ..., w_{n-1}, w_n$, pour corriger un mot erroné existe à la position pos dans la phrase, on va utiliser le meilleur modèle de prédiction des Part-Of-Speech réalisé dans le chapitre 3 pour prédire le POS du mot w_{pos} noté $Tag(w_{pos})$. Puis on va utiliser l'équation 4.8 pour générer les candidats de la correction du mot erroné $\bar{w}$. Après l'obtention du $Tagw_{pos}$ est les candidats de la correction $correct\_word(\bar{w})$, on va extraire les mots qui ont le même POS de $\bar{w}$ par l'équation 4.21.

$$\text{W } = \{w_i : Tag(w_i) = Tag(w_{pos}) \text{ tel que } w_i \in correct\_word(\bar{w})\} \tag{4.21}$$

— w_i : le candidat de la correction ;
— $\bar{w}$: le mot à corriger.

Si $|W| = 1$ alors $correct(\bar{w})$, sinon on va utiliser une autre formule pour bien préciser le mot correct.

$$\begin{aligned}\text{W } = \big\{&w_i : Tag(w_i) = Tag(w_{pos}) \text{ et } Tag(w_i^N) = Tag(w_{pos+1}) \\ &\text{tel que } w_i \in W, \text{ Si pos=0 } \}\end{aligned} \tag{4.22}$$

$$\begin{aligned}\text{W } = \big\{&w_i : Tag(w_i) = Tag(w_{pos}) \text{ et } Tag(w_i^P) = Tag(w_{pos-1}) \\ &\text{tel que } w_i \in W, \text{ Si pos=n } \}\end{aligned} \tag{4.23}$$

$$\begin{aligned}\text{W } = \big\{&w_i : Tag(w_i) = Tag(w_{pos}) \text{ et } Tag(w_i^N) = Tag(w_{pos+1}) \\ &\text{et } Tag(w_i^P) = Tag(w_{pos-1}) \text{ tel que } w_i \in W)\big\}\end{aligned} \tag{4.24}$$

— w_i^P : le mot existe avant le candidat de la correction dans le corpus ;
— w_i^N : le mot existe après le candidat de la correction dans le corpus ;
— w_{pos-1} : le mot existe avant le mot à corriger ;
— w_{pos+1} : le mot existe après le mot à corriger.

Les équations 4.22, 4.23 et 4.24 permettent de trouver le mot correct en déterminant le candidat de la correction et les mots qui se suivent et se précèdent qui ont les même POS des mots qui se suivent et se précèdent le mot à corriger. L'équation 4.22 pour la correction du mot s'il existe au début de la phrase, l'équation 4.23 pour la correction du mot s'il existe à la fin de la phrase et l'équation 4.24 pour la correction du mot s'il existe au milieu de la

phrase.

Si $|W| = 1$ alors $correct(\bar{w})$ sinon on va choisir le candidat de la correction le plus fréquent dans le corpus.

4.4.4 Résultats expérimentaux

4.4.4.1 Résultats d'OCR

Le système OCR réalisé commence par lire une image contenant un texte écrit en Tifinagh, puis effectue la reconnaissance du contenu, enfin génère un fichier contenant le résultat de la reconnaissance. La figure 4.5 est un exemple de l'image, contenant une phrase écrite en Tifinagh, utilisée pour tester notre OCR. Le résultat de l'OCR obtenu est illustré dans la Figure 4.6. L'observation de ce résultat montre qu'il y a deux erreurs, ces erreurs sont reportées dans le tableau 4.2. Dans le premier mot, l'erreur existe dans le troisième caractère. Par contre, dans le deuxième mot, l'erreur apparaît dans le premier caractère.

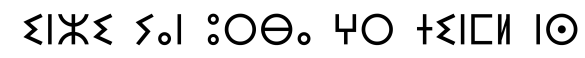

FIGURE 4.5 – Exemple d'image présentée à l'entrée de l'OCR

FIGURE 4.6 – Exemple de texte récupérée à la sortie de l'OCR

Entré	Sortie
ⵣⵉⵣ	ⵣⵎⵣ
ⵢⵓ	ⵃⵓ

TABLE 4.2 : Erreurs extraites

4.4.4.2 Correction par la méthode n-grammes

La correction des erreurs sera réalisée en utilisant l'approche n-grammes. En utilisant 4.9, nous avons trouvé les résultats suivants pour la correction de □ⵣⵎⵣ□ : Les mots candidats sont : ⵣⵉⵙⵣ, ⵣⵉⵣⵣ, ⵣⵉⵣⵣ. Pour la correction de ⵃⵓ, nous avons trois mots candidats : ⵅⵓ, ⵢⵓ,ⵆⵓ . Nous allons maintenant chercher la position du mot le plus fréquemment parmi les mots corrects dans la phrase.

Mot	Fréquence
ⵙⵏ ⵗⵔⵜⵏ	6.895301738404073E-4
ⵜⵣⵍⵛⵣ ⵍⵔ	9.85043105486296E-5

TABLE 4.3 : Fréquence de deux mots corrects successifs

D'après les résultats du tableau 4.3, on peut voir que ⵙⵏ ⵗⵔⵜⵏ est la combinaison la plus existe, donc la position renvoyée par l'équation 4.13 est "1". C'est le centre de correction et la position du mot ⵙⵏ. En utilisant la correction de gauche, nous allons corriger les mots qui sont avant le centre de correction. Le mot ⵣⵎⵎⵣ n'existe pas dans le dictionnaire, nous allons calculer la fréquence de tous ces candidats de la correction en utilisant l'équation 4.18 avec pos=1 (c'est-à-dire la fréquence des mots proches de ⵣⵎⵎⵣ suivi de ⵙⵏ ⵗⵔⵜⵏ) comme indiqué dans le tableau 4.4.

Candidats de la correction	Fréquence
ⵣⵍⵔⵣ	0.0
ⵣⵍⵟⵣ	6.85584650296E-9
ⵣⵍⵟⵣ	8.82545056296E-7

TABLE 4.4 : Fréquence de chaque mot parmi les mots candidats de ⵣⵎⵎⵣ

Selon les résultats indiqués dans le tableau 4.4, nous concluons que le mot correct est ⵣⵍⵟⵣ , au lieu de mettre ⵣⵎⵎⵣ, nous allons le remplacer par ⵣⵍⵟⵣ. Maintenant nous allons passer à la bonne correction, nous avons le mot ⵃⵔ n'existe pas dans le dictionnaire, nous allons calculer la fréquence de tous les mots qui lui sont proches en utilisant l'équation 4.19 avec pos=1 (c'est-à-dire, la fréquence des mots proches de ⵃⵔ précédé de ⵣⵍⵟⵣ ⵙⵏ ⵗⵔⵜⵏ voir le tableau 4.5).

Candidats de la correction	Fréquence
ⵖⵔ	8.31651623326E-8
ⵅⵔ	0.0
ⵆⵔ	0.0

TABLE 4.5 : Fréquence de chaque mot parmi les mots candidats de " ⵃⵔ"

D'après les résultats, figurés dans le tableau 4.5, nous pouvons conclure que le mot correct est ⵖⵔ, au lieu de mettre ⵃⵔ, nous allons le remplacer par ⵖⵔ. Nous avons le mot ⵜⵣⵍⵛⵣ existe dans le dictionnaire, sa fréquence sachant qu'il est précédé de ⵣⵍⵟⵣ ⵙⵏ ⵗⵔⵜⵏ ⵖⵔ est 1.3558880795743596E-6 non nulle, donc le mot ⵜⵣⵍⵛⵣ est correct. De même, nous avons le mot ⵍⵔ existe dans le dictionnaire, sa fréquence sachant qu'il est précédé de ⵣⵍⵟⵣ ⵙⵏ ⵗⵔⵜⵏ ⵖⵔ ⵜⵣⵍⵛⵣ est 1,3558880795743596E-6 non nulle, donc le mot ⵍⵔ est correct. De même, nous avons le mot ⵍⵔ existe dans le dictionnaire sa fréquence sachant qu'il est

précédé de ⵙⵉⵣⵙ ⵏⵍ ⵓⵔⵜⵡ ⵢⵔ ⵜⵙⵉⵎⵏ est 1,3558880795743596E-6 non nulle, donc le mot ⵍⵔ est correct.

4.4.4.3 Correction par la méthode d'étiquetage POS

Pour corriger les erreurs d'orthographe figurées dans la sortie de l'OCR (ⵙⵎⵙ ⵀⵔ) par la méthode d'étiquetage POS, on va utiliser les candidats de la correction trouvés par l'équation 4.8 ; (ⵙⵉⵔⵙ ⵙⵉⵣⵙ ⵙⵉⵣⵙ ⵙⵉⵟⵙ)et (ⵟⵔ ⵢⵔ ⵔ) ; on va prédire POS des deux mots erronés en utilisant le modèle du Bi-LSTM. Le tableau 4.6 contient les résultats du prédiction du POS des deux mots.

Mot	**POS**
ⵙⵎⵙ	VB
ⵀⵔ	S

Table 4.6 : Prédictions de POS des mots erronés

D'après tableau 4.6, on conclure que le premier mot erroné doit être un verbe et le deuxième mot erroné doit être une préposition.Parmi les candidats de la correction on va garder les mots qui ont le même POS prédit. Le tableau 4.7 contient POS des différents candidats de la correction du premier mot.

Candidats de la correction	**POS**
ⵙⵉⵔⵙ	NN
ⵙⵉⵣⵙ	VB
ⵙⵉⵣⵙ	VB

Table 4.7 : POS des mots candidats de ⵙⵎⵙ

D'après le tableau 4.7 on observe que les deux mots ⵙⵉⵣⵙ et ⵙⵉⵣⵙ ont le même POS de mot erroné, tant que le mot erroné existe au début de la phrase alors on va garder juste le candidat qui suivi par un mot de même POS du mot existe après le mot erroné, sachant que le POS du ce dernier est DT. Après une recherche dans le corpus on trouve que les deux candidats peuvent être suivis par un mot de type DT, donc on va chercher le mot le plus fréquent qui est suivi par un DT parmi les deux candidats.

Candidats de la correction	**Fréquence**
ⵙⵉⵣⵙ	8.1548852E-6
ⵙⵉⵣⵙ	5.5472215E-4

Table 4.8 : Fréquence des mots candidats de ⵙⵎⵙ

Le tableau 4.8 montre que le mot ⵙⵉⵣⵙ est le plus fréquent, donc on peut le considère comme le mot correct. Pour le deuxième mot erroné, on va extraire POS de ces candidats. Le tableau 4.9 contient POS des différents candidats du deuxième mot.

Candidats de la correction	**POS**
ⵢⵓ	S
ⵅⵓ	VB
ⵆⵓ	C

TABLE 4.9 : POS des mots candidats de ⵀⵓ

Comme le tableau 4.9 montre le mot ⵢⵓ qui a le même POS du deuxième mot, contre les deux autres candidats, alors le candidat qui peut remplacer le mot erroné est ⵢⵓ. Après avoir corrigé les résultats de l'OCR par les deux méthodes, nous pouvons schématiser un processus global comme suit (Figure 24).

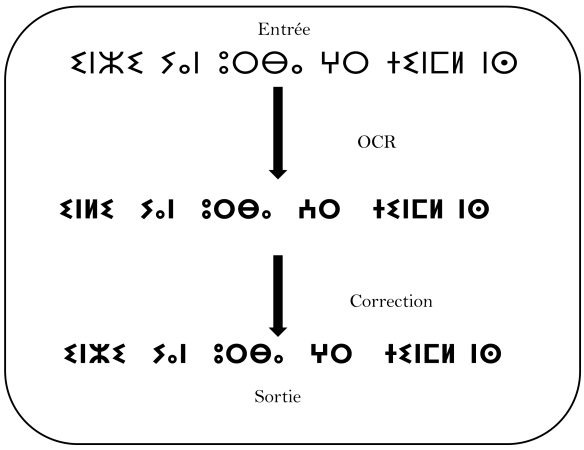

FIGURE 4.7 – Schéma Final d'exemple de correction

Après plusieurs tests, on trouve que la correction des erreurs d'orthographe par la méthode d'étiquetage POS est plus mieux par rapport à la méthode n-grammes en terme de précision et de temps d'exécution, Cela est dû à l'élimination de certains candidats à la correction qui ne sont pas adaptés au POS du mot erroné, c'est à dire moins de calcul, et aussi à la deuxième technique (correction par étiquetage POS) qui n'utilise pas le calcul récursif.

Techniques	**Taux de correction**	**Taux d'erreur**	**Temps d'exécution (s)**
OCR	86%	14%	20s
OCR+Canal bruyant	96%	4%	75s
OCR+Approche n-gramme	98%	2%	52s
OCR+Approche POS	98.94%	1.06%	40s

TABLE 4.10 : Taux de correction et temps d'exécution

Après plusieurs tests, nous constatons que l'approche n-gramme et l'approche POS

a amélioré les résultats donnés par l'OCR. Les résultats du tableau 4.10 montrent que le taux de reconnaissance de l'OCR est de 86%, après avoir utilisé l'approche n-gramme, nous constatons que la reconnaissance est augmentée à 98% et à 98.94% en utilisant l'approche POS. Meilleur que le canal bruyant Malgré la légère augmentation du temps.

4.5 Conclusion

Les résultats d'un OCR ne sont pas toujours corrects, ce qui nécessite un post-traitement permettant la détection et la correction des erreurs. Dans ce chapitre, nous avons élaboré deux systèmes de correction des erreurs d'un OCR utilisé pour reconnaître des documents écrits en caractères Tifinagh : (i) Le premier est basé sur la méthode n-grammes, qui commence la correction à partir d'un centre déterminé, puis le calcul de fréquences des chaque candidats en utilisant une correction à gauche et à droite et une technique recursive. (ii) Le deuxième est basé sur la prédiction de Part-Of-Speech des mots erronés pour éliminer des candidats qui n'ont pas le même POS des mots à corriger, puis il corrige le mot erroné en gardant les candidats qui ont dans leurs entourages les mêmes POS de l'entourage du mots erroné selon ça position dans la phrase. L'efficacité de la méthode basée sur la prédiction de Part-Of-Speech est justifiée par les résultats expérimentaux obtenus et le temps d'exécution.

Chapitre 5

Les entités nommées pour la langue Amazigh

5.1 Introduction

Les entités nommées sont des mots ou des phrases, qui sont nommés où catégorisés dans un certain sujet. Elles portent généralement des informations clés dans une phrase qui servent de cibles importantes pour la plupart des systèmes de traitement du langage. La reconnaissance précise des entités nommées peut être utilisée comme une source d'information utile pour différentes applications NLP. Par exemple, les performances d'applications telles que la réponse aux questions, la traduction automatique ou la recherche d'informations ont été améliorées par les informations sur les entités nommées. Les trois classes intuitives de personne (PER), de lieu (LOC), d'organisation (ORG) ainsi que la classe diverse (MIS) vaguement définie sont utilisées dans la plupart des systèmes de NER. Ces classes sont surtout pertinentes pour les corpus liés à l'actualité. Pour les autres domaines, les systèmes de NER doivent être formés et testés avec d'autres étiquettes de classe pertinentes. La reconnaissance d'entités nommées (NER) est un sous-problème de l'extraction d'information et implique le traitement de documents structurés et non structurés et l'identification d'expressions faisant référence à des personnes, des lieux, des organisations et des entreprises. La reconnaissance des entités nommées est une tâche fondamentale et constitue le cœur du système de traitement du langage naturel (NLP). Le NER implique deux tâches, à savoir l'identification des noms propres dans le texte et la classification de ces noms dans un ensemble de catégories d'intérêt prédéfinies, telles que les noms de personnes, les organisations (entreprises, organisations gouvernementales, comités, etc.), les lieux (villes, pays, rivières, etc.) et les expressions de date et d'heure. Dans le domaine général, le NER se concentre sur l'identification des noms de personnes, de lieux et d'organisations, ... dans les articles de presse. Grâce à la disponibilité de corpus annotés, les méthodes d'apprentissage supervisé ont été largement adoptées dans de nombreuses langues et l'emportent sur les méthodes non supervisées. Les systèmes de NER les plus avancés ont atteint des performances aussi élevées que celles des annotateurs humains. De leur côté, les systèmes de NER s'améliorent avec l'arrivée d'un plus grand nombre de corpus annotés pour l'apprentissage.

Les méthodes traditionnelles de traitement des NER vont de la mise en correspondance de dictionnaires et de règles heuristiques à l'étiquetage de séquences basées sur des modèles de Markov cachés (HMM) ou des champs aléatoires conditionnels (CRF) supervisés. Ces approches ne nécessitent pas de données d'apprentissage, mais impliquent généralement des règles et des hypothèses qui peuvent limiter le type d'entités et de textes auxquels elles pourraient s'appliquer [82]. Nous avons adopté la méthodologie standard qui utilise des procédures basées sur le langage sémantique pour favoriser l'ANER (Amazigh Named Entity Recognation). La méthodologie est conduite par les attributs et les excentricités de la langue Amazigh. L'interaction de l'accusé de réception prend deux cycles, en utilisant la partie de la liste blanche et en appliquant ensuite les règles de structure de la phrase. Nous créons nos propres conceptions d'algorithmes d'apprentissage profond en utilisant les techniques RNN, LSTM, GRU, BiLSTM et BiGRU. Cette approche de conception ouverte souligne l'adaptabilité et la polyvalence de notre cadre. Nous présentons les résultats de notre tentative de reconnaissance et d'extraction des 23 entités nommées les plus importantes dans le script amazigh, à savoir PERSON, GPE, FAC, DATE, TIME, etc. Le système ANER est évalué en utilisant un corpus de référence qui est étiqueté avec des noms d'une manière semi-automatique. Les résultats obtenus sont satisfaisants lorsqu'ils sont évalués par rapport à la mesure standard : précision, rappel et f-mesure.

5.2 Tâche de reconnaissance des entités nommées

5.2.1 Entités nommées

Les entités nommées (EN) jouent un rôle central dans la transmission d'informations spécifiques à un domaine important dans un texte, et de bons systèmes de reconnaissance d'entités nommées sont souvent nécessaires pour construire des systèmes d'extraction d'informations pratiques. Il n'existe pas de types généraux d'entités nommées qui soient communément utilisés dans toutes les langues. Par conséquent, un système de NER peut reconnaître des différences d'une langue à l'autre ou d'un domaine à l'autre. Cette caractéristique est assez variable en raison de l'ambiguïté de l'utilisation du terme "entité nommée" en fonction des différents forums ou événements. Il existe des conférences ou des concours qui sont organisés pour définir les types d'EN et évaluer les performances d'un système de NER développé pour une langue donnée. Ces conférences sont, par exemple, Message Understanding Conference (MUC) pour l'anglais, Conferences on Natural Language Learning (CoNLL), une tâche NER indépendante de la langue et Information Retrieval and Extraction Exercise (IREX) pour le japonais.

5.2.2 Défis de la reconnaissance des entités nommées

La reconnaissance d'entités nommées se compose des deux sous-problèmes suivants : (1) la reconnaissance des limites des entités nommées ; (2) la reconnaissance des catégories (classes) d'entités nommées. Ces problèmes sont généralement (mais pas nécessairement) traités simultanément. Comme pour la plupart des problèmes de traitement de la langue,

il existe des ambiguïtés dans la langue qui ajoutent au défi de la tâche. Dans ce qui suit, nous présentons des exemples d'ambiguïtés dans la reconnaissance et la catégorisation d'entités nommées. En outre, Washington peut faire référence à une personne, un lieu ou une organisation (gouvernement américain). La plupart des défis du NER résident dans sa nature fortement lexicalisée et dépendante du domaine. Les noms occupent une grande partie d'une langue et évoluent constamment dans différents domaines. Afin de disposer d'un système de NER robuste pour un domaine donné (par exemple, le tourisme), nous avons besoin de corpus et de lexiques étiquetés (par exemple, les noms de monuments). La création et la mise à jour de telles ressources pour divers sujets est une tâche coûteuse qui nécessite des connaissances linguistiques et spécialisées. Dans ce qui suit, nous passerons en revue deux cadres de NER basés sur des règles et des statistiques et nous discuterons de leurs exigences en matière de données et de leur robustesse.

5.2.3 Reconnaissance d'entités nommées basée sur des règles

Les premières approches de la reconnaissance des entités nommées étaient principalement basées sur des règles. La plupart des systèmes basés sur des règles utilisaient trois composantes principales : (1) un ensemble de règles d'extraction d'entités nommées, (2) des lexiques pour différents types de classes d'entités nommées, et (3) le moteur d'extraction qui applique les règles et les lexiques au texte. L'ensemble de règles et les lexiques ont été soit entièrement créés à la main par des humains, soit créés à partir de quelques exemples créés à la main. Un exemple réussi du cadre basé sur les règles est le système d'extraction d'information Auto-Slog. Le système commence avec un ensemble de règles simples pour certaines entités connues. Dans un cadre d'amorçage itératif, les règles sont appliquées et étendues pour extraire de nouvelles entités. Les systèmes basés sur des règles sont relativement précis mais ont généralement une faible couverture et fonctionnent bien dans des domaines étroits. Leurs performances dépendent généralement de l'exhaustivité des règles et des lexiques. Les cadres d'amorçage, comme les systèmes à base de règles, sont toujours limités au domaine des règles et du lexique d'origine. En outre, l'incorporation de connaissances plus approfondies, au-delà des mots et lexiques de surface, dans un système à base de règles nécessite un effort manuel coûteux. En revanche, les cadres statistiques sont plus flexibles pour incorporer des connaissances linguistiques plus riches (par exemple, la syntaxe), ce qui donne des systèmes plus robustes.

5.3 Algorithmes et résultats

5.3.1 Description des modèles

Nous avons le privilège d'appliquer les techniques d'apprentissage profond sur la langue Amazigh pour prédire les entités nommées des mots écrits en caractères tifinagh en utilisant la bibliothèque open-source Keras. Dans ce chapitre, nous avons utilisé cinq algorithmes d'apprentissage profond (RNN, LSTM, GRU, BiLSTM, BiGRU) qui ont les mêmes couches. La figure 5.1 décrit l'architecture des modèles.

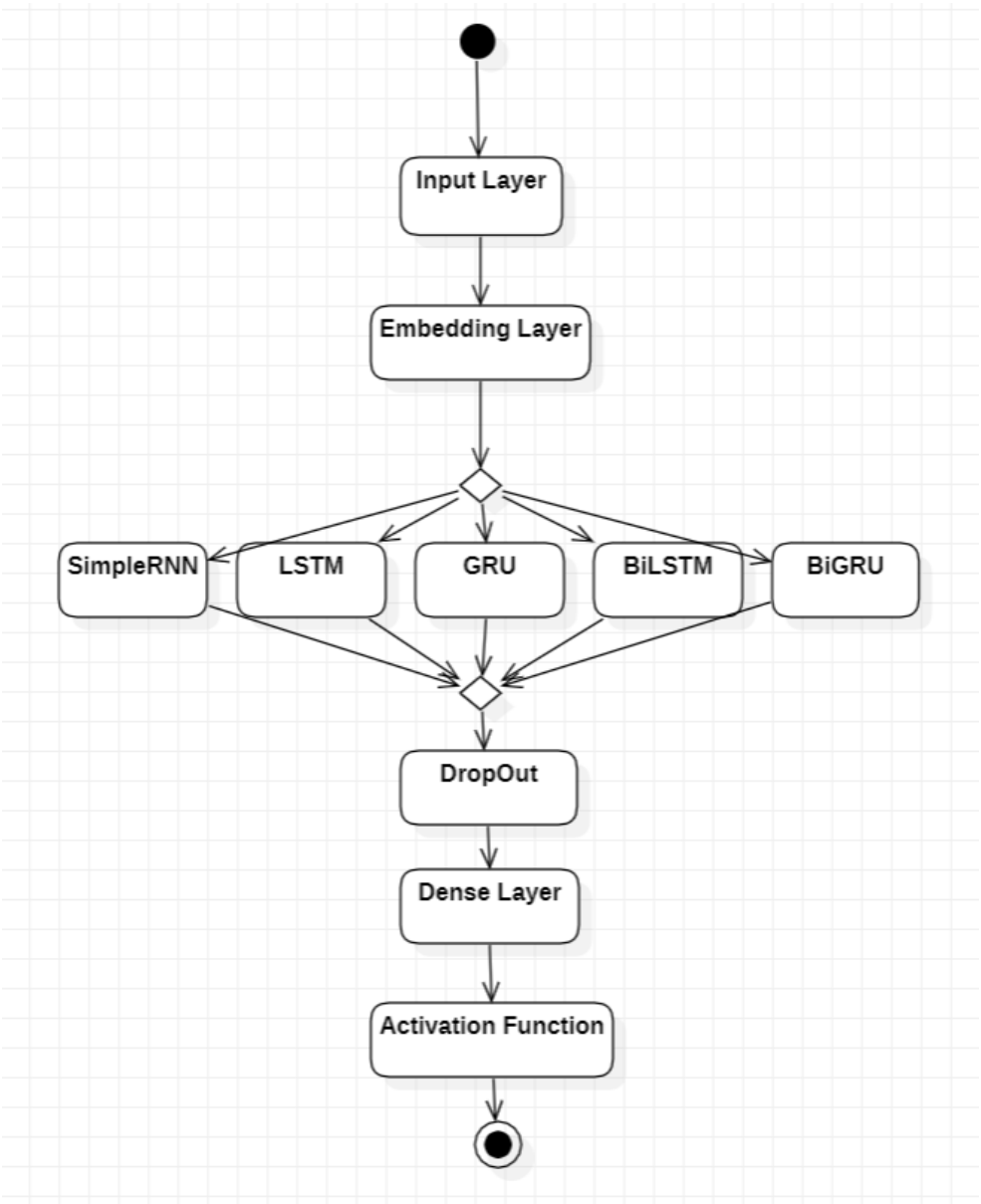

FIGURE 5.1 – Architecture des modèles de NER

La première couche est la couche d'entrée, elle accepte des vecteurs de (89) valeurs et correspond à notre variable X, (nous avons 89 tokens dans chacune de nos séquences train et test). Ensuite, nous avons la couche d'incorporation. Cette couche va prendre chacun de nos tokens/mots et le transformer en un vecteur dense de taille 200. Il s'agit d'un tableau de consultation géante (ou d'un dictionnaire) avec les mots-clés comme clés et les vecteurs réels comme valeurs. Ce tableau de consultation peut être entraînée, c'est-à-dire qu'à chaque époque de l'apprentissage du modèle, nous mettons à jour ces vecteurs pour qu'ils correspondent à l'entrée. Après la couche d'incorporation, notre entrée passe d'un vecteur de longueur 89 à une matrice de taille (89, 200). Chacun des 89 tokens a maintenant un vecteur de taille 200. Une fois que nous avons cela, nous pouvons utiliser la couche LSTM (couche RNN simple, couche GRU, couche BiLSTM, ou Bi-GRU) qui, pour chaque token, regardera dans les deux sens dans la phrase. La sortie de cette couche est une matrice de taille (89, 128). Ensuite, nous avons la couche de Dropout qui est une matrice de taille (89, 128). Enfin, nous avons une couche dense distribuée dans le temps. Elle prend la matrice (89, 128) de la sortie de la couche LSTM (RNN simple, couche GRU , couche BiLSTM ou couche BiLSTM). Le tableau 5.1, le tableau 5.2, le tableau 5.3, le tableau 5.4 et le tableau 5.5 montrent les couches de notre modèle. Dans le modèle, nous avons choisi d'utiliser la couche de normalisation par lots, la fonction ReLU dans la couche cachée, et la fonction softtmax dans la couche de sortie, nous avons utilisé les méthodes Adam comme optimiseur.

Couches	**Forme de la sortie**	**Nombre des paramètres**
InputLayer	[(None,89)]	0
Embedding	(None,89,200)	686800
SimpleRNN	(None,128)	16960
Dropout	(None,89,128)	0
Dense	(None,89,32)	2080
Dense 1	(None,89,76)	2508

TABLE 5.1 : Architecture proposée pour le modèle RNN.

Couches	**Forme de la sortie**	**Nombre des paramètres**
InputLayer	[(None,89)]	0
Embedding	(None,89,200)	686800
LSTM	(None,128)	93440
Dropout	(None,89,128)	0
Dense	(None,89,32)	2080
Dense 1	(None,89,76)	2508

Couches	**Forme de la sortie**	**Nombre des paramètres**

TABLE 5.2 : Architecture proposée pour le modèle LSTM.

Couches	**Forme de la sortie**	**Nombre des paramètres**
InputLayer	[(None,89)]	0
Embedding	(None,89,200)	686800
GRU	(None,128)	51072
Dropout	(None,89,128)	0
Dense	(None,89,32)	2080
Dense 1	(None,89,76)	3663

TABLE 5.3 : Architecture proposée pour le modèle GRU.

Couches	**Forme de la sortie**	**Nombre des paramètres**
InputLayer	[(None,89)]	0
Embedding	(None,89,200)	686800
BiLSTM	(None,128)	186880
Dropout	(None,89,128)	0
Dense	(None,89,32)	2080
Dense 1	(None,89,76)	3663

TABLE 5.4 : Architecture proposée pour le modèle BiLSTM.

Couches	**Forme de la sortie**	**Nombre des paramètres**
InputLayer	[(None,89)]	0
Embedding	(None,89,200)	686800
Bi-GRU	(None,128)	102144
Dropout	(None,89,128)	0
Dense	(None,89,32)	2080
Dense 1	(None,89,76)	2508

TABLE 5.5 : Architecture proposée pour le modèle Bi-GRU.

La couche "Embedding", reçoit en entrée un vecteur de longueur 89 et se transforme en une matrice de taille (89, 200), présente 686 800 paramètres dans tous les modèles, 23 360 pour la couche RNN simple, 93 440 pour la couche LSTM, 51 072 pour la couche GRU, 186 880 pour la couche BiLSTM et 102 144 pour la couche Bi-GRU. La première couche dense compte 4 128 paramètres et la deuxième couche dense compte 2 508. Par conséquent, le total des paramètres pour SimpleRNN est de 708 348 paramètres entraînables : 708 348 ; et Paramètres non entraînables : 0. Pour LSTM le total des paramètres est : 784 828 les paramètres traitables : 784 828 ; et les paramètres non-formables : 0. Pour GRU, le total des paramètres est de : 743 615 ; les paramètres pouvant être formés : 743 615 ; et les paramètres non entraînables : 0. Pour BiLSTM, le total des paramètres est de : 880 316 Paramètres entraînables : 880 316 ; et paramètres non entraînables : 0. Pour Bi-GRU, le total des paramètres est de : 795 580. Paramètres entraînables : 795 580 et paramètres non entraînables : 0. Tous les modèles ont été entraînés à l'aide de l'optimiseur Adam [83]. La taille des unités cachées et la taille des lots de chaque modèle ont été fixées à 64 et 32, respectivement. Nous avons implémenté une couche « dropout layer » après la couche RNN, (LSTM, GRU, BiLSTM, Bi-GRU) avec une valeur de 0,3 pour réduire l'excès à l'overfitting et pour améliorer la précision du modèle.

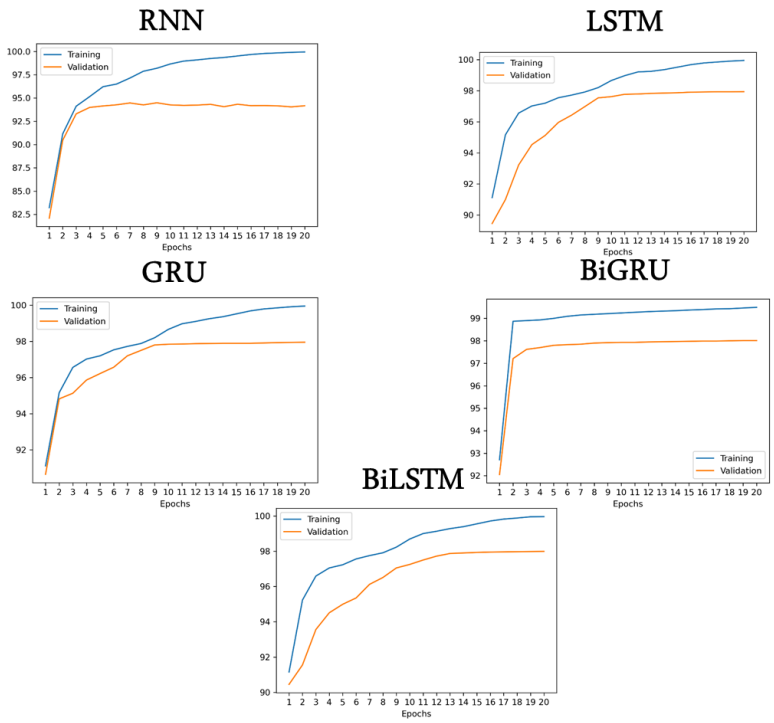

FIGURE 5.2 – La précision de l'entraînement et du validation pour les modèles d'apprentissage profond.

Les figures 5.3 montre l'évolution de la précision à chaque époque lors de l'apprentissage de chaque modèle. Elle augmente rapidement au début du processus d'apprentissage, puis se stabilise progressivement jusqu'à ce qu'il n'y ait plus d'amélioration insignifiante de la

précision. Le processus d'entraînement a été répété un nombre différents d'époques, en essayant de trouver un équilibre entre le temps d'entraînement et les résultats obtenus, en évitant l'overfitting. Le nombre optimal d'époques a été trouvé à 20.

À partir de ces figures, nous arrivons aux conclusions décrites ci-dessous : Bi-GRU surpasse les autres structures basées sur les RNN, et le déploiement du mécanisme bidirectionnel entraîne une nette amélioration de la précision. Dans la comparaison des différentes valeurs de la précision, nous avons trouvé la précision obtenue pour l'ensemble d'entraînement est de 99.37% en utilisant la couche RNN, 99.52% par LSTM, par la couche GRU nous avons trouvé 99.32%, Bi-LSTM 99.27% et Bi-GRU attient 99.50%. Lorsque comparaison avec l'ensemble de test, nous obtenons une précision de 94.49% pour la couche RNN, 97.95% pour la couche LSTM, 97.96% pour la couche GRU, 97.99% pour la couche Bi-LSTM et 98.01% pour la couche Bi-GRU, ces valeurs sont inférieures à celles obtenues avec l'ensemble d'entraînement.

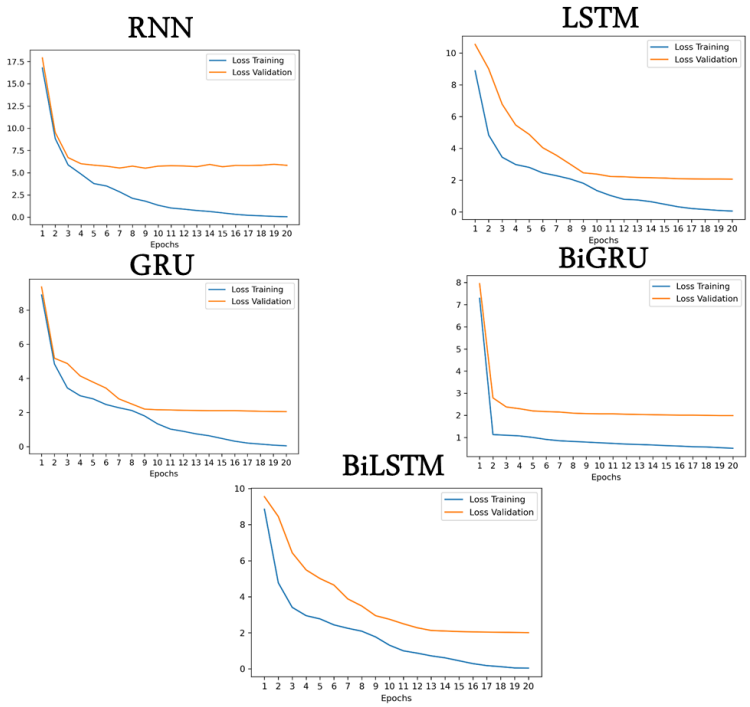

FIGURE 5.3 – La perte de l'entraînement et du validation pour les modèles d'apprentissage profond.

L'erreur obtenue pour l'ensemble d'entraînement est de 0,63% en utilisant la couche RNN, 0.47% pour la couche LSTM, en utilisant la couche GRU nous avons obtenue 0.68%, dans le cas de la couche Bi-LSTM nous avons trouvé 0.73% et pour la couche Bi-GRU l'erreur trouvée est 0.5%. Pour l'ensemble de test, nous avons calculé 5.51% par la couche RNN, 2.05% par LSTM, 2.04% par GRU, 2.01% par Bi-LSTM, et 1.99% par la couche Bi-GRU.

Modèles	**Nombre des paramètres**	**Temps d'entraînement (min)**
RNN	708348	72.211
LSTM	784828	150.851
GRU	743615	149.512
Bi-LSTM	829116	232.241
BiGRU	795580	241.135

TABLE 5.6 : Comparaison des modèles en termes de nombre des paramètres et du temps d'entraînement.

Ce qui concerne le temps d'exécution pendant l'entraînement, le tableau 5.6 indique que le modèle de RNN justifie leur efficacité. Dans la phase d'entraînement, le modèle de RNN est plus rapide par rapport aux autres modèles par un pourcentage supérieur à 49%. Nous observons que les modèles LSTM et GRU ont presque la même vitesse lors de l'entraînement, ils sont plus rapides par rapport aux modèles Bidirectionnel par 38%. Cette différence entre les modèles RNN, LSTM et GRU dans le temps d'entraînement, est expliquée par la différence dans le nombre des paramètres utilisés dans chaque modèles ; dans le modèle RNN, nous avons utilisé 708 348 paramètres, dans le modèle LSTM, nous avons utilisé 784 828 paramètres, dans le modèle GRU, nous avons utilisé 743 615 paramètres ; par contre les modèles Bidirectionnels sont plus lents par rapport aux autres modèles, cela est expliquée par la caractéristique de traitement dans les deux sens avant et arrière, en fusionnant les connaissances des deux directions à la sortie.

5.4 Conclusion

Dans ce chapitre, nous avons évalué brièvement cinq types d'approches utilisées pour la reconnaissance des entités nommées sur la langue Amazigh. Chacune des techniques et chacun des modèles proposés ont tenté d'améliorer la précision du module de reconnaissance et la compacité de la zone de reconnaissance. Comme nous l'avons mentionné précédemment, le problème le plus important de la reconnaissance d'entités nommées est peut-être de changer et de transformer l'espace d'information en un autre espace, ce que l'on appelle la polyvalence. La méthodologie basée sur les normes utilisée avec une maîtrise phonétique extraordinaire a permis une exécution fructueuse du cadre NER pour la langue Amazigh en surmontant les difficultés présentées par la langue Amazigh. Un ensemble de règles de structure de la langue est déterminé en disséquant le cadre lexical de voisinage d'un grand nombre d'informations assorties. Les règles sont équipées pour percevoir les structures arquées en les séparant en tiges et en jointures. En outre, l'utilisation judicieuse des règles du canal permet de gérer l'imprécision de la reconnaissance entre les éléments nommés. Nous avons évalué l'exécution de notre cadre en utilisant nos propres corpus étiquetés de manière semi-informatique. En outre, ces corpus pourraient être utilisés comme un ensemble de données d'évaluation standard pour les approches NER Amazighs.

Chapitre 6

Traduction automatique de la langue amazigh

6.1 Introduction

La traduction automatique est l'un des domaines les plus anciens et les plus fascinants du traitement du langage naturel. L'objectif premier est d'éliminer les barrières linguistiques en développant un système de traduction automatique capable de traduire une langue humaine dans une autre. La traduction automatique est un sous-domaine de l'intelligence artificielle qui traduit une langue naturelle en une autre langue naturelle à l'aide d'ordinateurs [84].Il s'agit d'un domaine de recherche interdisciplinaire qui intègre des idées provenant de différents domaines tels que les langues, l'intelligence artificielle, les statistiques et les mathématiques [85].

Déjà lors de la dernière vague de recherche sur les réseaux neuronaux dans les années 1980 et 1990, la traduction automatique était dans le viseur des chercheurs explorant ces méthodes [86]. En fait, les modèles proposés par [87] ressemblent de façon frappante aux les approches de traduction automatique neuronale actuellement dominantes. Cependant, aucun de ces modèles n'a été entraîné sur des données de taille suffisante pour produire des résultats raisonnables pour tout autre chose que des exemples jouets. La complexité informatique impliquée dépassait de loin les ressources informatiques de l'époque.

L'idée de la traduction automatique remonte à l'époque où les ordinateurs ont vu le jour. En 1949, le domaine de la traduction automatique est apparu dans le mémorandum de Warren Weaver, l'un des pionniers dans le domaine de la traduction automatique [88]. À l'ère du numérique, diverses communautés du monde entier sont reliées entre elles et partagent d'immenses ressources. Les différentes langues constituent un obstacle à la communication dans ce type d'environnement numérique. Des chercheurs de plusieurs pays et de grandes entreprises travaillent à la construction de systèmes de traduction automatique afin de surmonter cet obstacle. Avant le 20e siècle, la réalisation du processus de traduction requis était un rêve. Au XXe siècle, il est devenu réalité lorsque des programmes informa-

tiques, bien que limités à des domaines spécifiques, ont été utilisés pour le processus de traduction [89].L'idée de la traduction automatique remonte à l'époque où les ordinateurs ont vu le jour. En 1949, le domaine de la traduction automatique est apparu dans le mémorandum de Warren Weaver, l'un des pionniers dans le domaine de la traduction automatique [88].
À l'ère du numérique, diverses communautés du monde entier sont reliées entre elles et partagent d'immenses ressources. Les différentes langues constituent un obstacle à la communication dans ce type d'environnement numérique. Des chercheurs de plusieurs pays et de grandes entreprises travaillent à la construction de systèmes de traduction automatique afin de surmonter cet obstacle. Avant le 20e siècle, la réalisation du processus de traduction requis était un rêve. Au XXe siècle, il est devenu réalité lorsque des programmes informatiques, bien que limités à des domaines spécifiques, ont été utilisés pour le processus de traduction [89].
Le résultat du système de traduction automatique était affiché pour produire une traduction de haute qualité. La traduction automatique s'est avérée être un bon outil pour la traduction de grands textes de documents scientifiques, de rapports de journaux et d'autres documents [90]. L'augmentation de la croissance industrielle et de l'échange d'informations entre plusieurs langues régionales au cours de la dernière décennie a eu un grand impact sur le marché de la traduction automatique, qui exige que l'accès à l'information soit disponible dans toutes les langues régionales.

Au cours des premières années de recherche, des systèmes de traduction automatique ont été construits à l'aide de dictionnaires bilingues et de quelques règles manuscrites ; cependant, avec ces règles manuscrites, il s'est avéré difficile de traiter toutes les anomalies linguistiques [90]. L'augmentation des capacités de traitement dans les années 1980 a permis de passer d'une méthode basée sur des règles à une traduction automatique statistique. La disponibilité d'énormes corpus parallèles et les développements de l'apprentissage profond ont entraîné un changement de paradigme des modèles statistiques aux modèles neuronaux. Dans ce chapitre nous allons appliquer les techniques de traduction automatique pour traduire des phrases anglaises au phrases amazighs utilisant un modèle de traduction automatique statistique et des modèles de traduction automatique neuronale.

6.2 Traduction automatique statistique

La traduction automatique statistique (SMT) est un paradigme de traduction automatique dans lequel les traductions sont générées sur la base de modèles statistiques dont les paramètres sont dérivés de l'analyse de corpus de textes bilingues. L'approche statistique s'oppose aux approches de la traduction automatique basées sur des règles ainsi qu'à la traduction automatique basée sur des exemples [91].

Les premières idées de la traduction automatique statistique ont été présentées par [92],[42] y compris les idées d'application de la théorie de l'information de Claude Shannon.

La traduction automatique statistique a été réintroduite à la fin des années 1980 et au début des années 1990 par des chercheurs du centre de recherche Thomas J. Watson d'IBM[42] et a contribué à l'important regain d'intérêt pour la traduction automatique ces dernières années. Avant l'introduction de la traduction automatique neuronale, celle-ci était de loin la méthode de traduction automatique la plus étudiée.

6.2.1 Moses

Moses est un logiciel libre, un moteur de traduction automatique statistique qui peut être utilisé pour former des modèles statistiques de traduction de textes d'une langue source vers une langue cible. Moses permet ensuite de décoder un nouveau texte en langue source en utilisant ces modèles pour produire des traductions automatiques dans la langue cible. L'entraînement nécessite un corpus parallèle de passages dans les deux langues, généralement des paires de phrases traduites manuellement. Moses est publié sous la licence LGPL et disponible sous forme de code source et de binaires pour Windows [91] et Linux. Son développement est principalement soutenu par le projet EuroMatrix, avec un financement de la Commission européenne.

Parmi ses caractéristiques, citons :

— Un algorithme de recherche par faisceau qui trouve rapidement la traduction la plus probable parmi un certain nombre de choix.
— Traduction par phrase de courts morceaux de texte.
— Traitement des mots avec des représentations factorisées multiples pour permettre l'intégration d'informations linguistiques et autres (par exemple, forme de surface, lemme et morphologie, partie du discours, classe de mots).
— Décode les formes ambiguës d'une phrase source, représentée sous la forme d'un réseau de confusion, afin de permettre l'intégration avec des outils en amont tels que les systèmes de reconnaissance vocale.
— Prise en charge de grands modèles de langage (LM) tels que l'IRSTLM (un LM exact utilisant le mappage de la mémoire) et le RandLM (un LM inexact basé sur des filtres Bloom).

6.2.2 Travail du système statistique

Nous devons traduire une phrase anglaise en phrase amazigh. Nous disposons d'un modèle $p(P|E)$ qui estime la probabilité conditionnelle de toute phrase P en amazigh étant donné la phrase anglaise E. Nous utilisons le corpus d'entraînement pour définir les paramètres. Le problème de la traduction automatique est divisé en trois étapes :

— Modèle de langue $p(P)$, ce modèle peut être l'estimation du modèle de trigramme à partir de n'importe quelle donnée. Pour ce modèle de langue, nous n'avons pas besoin de corpus parallèle [91].
— $p(P|E)$ modèle de traduction, ce modèle doit être entraîné à partir de corpus parallèles.
— $p(P|E)p(P)$ Recherche de P maximisant le produit.

A partir de ces deux modèles, nous pouvons développer un système de traduction.

6.2.2.1 Modèle linguistique :

Supposons que nous décomposions une phrase amazigh P en mots $P = P_1, P_2, P_3, ..., P_m$. Nous pouvons alors écrire la probabilité de P comme produit de probabilités conditionnelles :

$$p(P) = \prod_{j=1}^{m} p(P_j|P_1, P_2, ..., P_{m-1}) \tag{6.1}$$

6.2.2.2 Modèle de traduction

Le modèle de traduction est construit pour calculer la probabilité $p(P|E)$. Le modèle de traduction estime les paramètres à partir de corpus parallèles et renvoie l'alignement des mots de la langue source à la langue cible.

$$p(P|E) = \frac{p(E,P)}{p(P)} = \frac{p(P)p(E \vee P)}{\sum p(P)p(E \vee P)} \tag{6.2}$$

Considérons la paire de phrases alignées suivante (Yessefk ad d-tegrem leɛmeṛ nnes deg leḥsab.) You(1) must(1) take(2)(3) his(5) age(4) into(6) account(7). Alignement de cette phrase parallèle : $a = \{1, 2, 3, 4, 5, 6, 7\}$ La figure 6.1 montre un exemple d'alignement de deux phrases (ablais-amazigh). Nous définirons le modèle pour $p(P|E)$ et $p(E|a, P)$

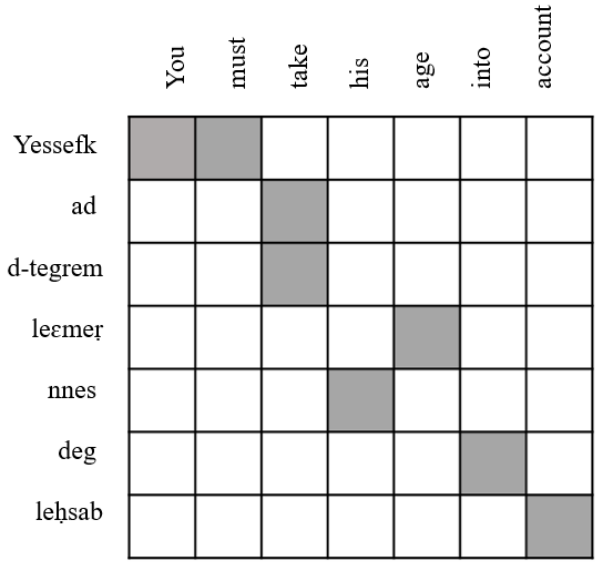

FIGURE 6.1 – Alignement de deux phrases (anglais-amazigh)

donnant comme :

$$p(E|a, P) = p(a|P)p(E|a, P) \tag{6.3}$$

6.2.2.3 L'alignement

Dans des corpus parallèles, des phrases uniques dans une langue peuvent être trouvées traduites en plusieurs phrases dans l'autre et vice versa. Les phrases longues peuvent être fractionnées, les phrases courtes peuvent être fusionnées. Il existe même certaines langues qui utilisent des systèmes d'écriture sans indication claire d'une fin de phrase. L'alignement des phrases peut être effectué via l'algorithme d'alignement [93]. Grâce à cela et à d'autres modèles mathématiques, une recherche et une récupération efficaces de l'alignement de phrase le plus élevé sont possibles.

Alignement des mots L'alignement des phrases est généralement soit fourni par le corpus, soit obtenu par l'algorithme d'alignement Gale-Church susmentionné. Pour apprendre par ex. le modèle de traduction, cependant, nous devons savoir quels mots s'alignent dans une paire de phrases source-cible. Les solutions sont les modèles IBM ou l'approche HMM.

L'un des problèmes posés est celui des mots fonctionnels qui n'ont pas d'équivalent clair dans la langue cible.

Les figures 6.2, 6.3 et 6.4 nous montrent plusieurs alignements qui sont tous acceptables avec des probabilités différentes.

- La figure 6.2 présente le type one to many alignement par exemple le mot 'call' est l'équivalent de 'ⵖⵓ ⴰⵙ' ;
- La figure 6.3 présente le type many to one, alignement par exemple les mots 'can i' sont l'équivalent de 'ⵣⵎⵙ' ;
- La figure 6.4 présente le type many to many, alignement par exemple les mots 'see you' sont l'équivalent de 'ⴰⵣⵍ '.

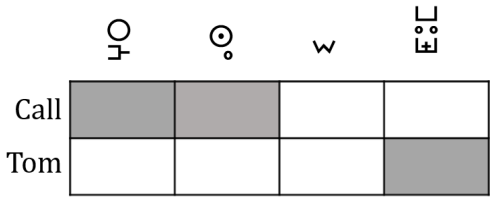

Figure 6.2 – Exemple d'alignement One To Many

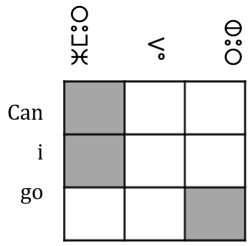

Figure 6.3 – Exemple d'alignement Many To One

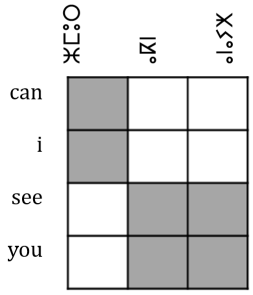

FIGURE 6.4 – Exemple d'alignement Many To Many

6.2.2.4 Décodeur

Le but du décodage est de maximiser la probabilité du texte traduit. La traduction de l'anglais en amazigh, peut être vue comme le problème de trouver P, qui maximise $p(P|E)P(E,P) = argmaxp(P)p(E|P)$ Selon le modèle de traduction ci-dessus basé sur l'alignement, le problème est reformulé comme suit :

Recherche d'une paire (P, E) qui maximise $p(P, a|E)$ En utilisant le théorème de Bayes, ceci est équivalent à trouver (P, a) qui maximise $p(P, a|E) = p(E, a|P)p(P)/p(E)$.

6.2.3 Méthodologie

Cette section comprend la collecte du corpus, la préparation des données, le développement du modèle linguistique, du modèle de traduction et la formation du décodeur à l'aide de l'outil Moses.

6.2.3.1 Préparation du corpus

Le manque de corpus parallèles est un obstacle majeur dans le développement de système de traduction automatique statistique. Le corpus parallèle est la collection de phrases des langues source et cible. Pour cela, nous avons utilisé un corpus parallèle anglais-amazigh contenant environ 137 000 phrases. Ce corpus contient des phrases de différents domaines. On a utilisé 80% de corpus pour l'entraînement et 20% pour le teste.

6.2.3.2 Entraînement du modèle linguistique

Le modèle est construit avec la langue amazigh qui est la langue cible du système proposé. Il est important que le système connaisse la langue, afin que les sorties soient structurées. Le manuel d'utilisation Moses [91] donne une explication complète de l'option de la ligne de commande.

6.2.3.3 Entraînement du système de traduction

Enfin, le processus d'entraînement du modèle de traduction en fonction de notre paire de langues. L'outil d'alignement de mots GIZA++ est utilisé pour aligner les phrases du

corpus parallèle, des règles de ré-ordonnancement lexicalisées sont créées pour le fichier de configuration de Moses. Nous avons créé un répertoire pour exécuter la formation journaux de commande.

6.2.3.4 Teste

Maintenant nous lançons moses par la commande suivante : « ∼/Amaz_moses/bin/moses -f \∼/work/mart-work/moses.ini » Nous pouvons taper une phrase en anglais et obtenir la sortie en amazigh. Nous utilisons « echo » pour afficher la sortie comme ceci : « Echo ”i can go.” » « ∼/ Amaz_moses/bin/moses -f ∼/work/mert-work/moses.ini » Ceci donnera la sortie : « Zemre ad ṛuḥe . ».

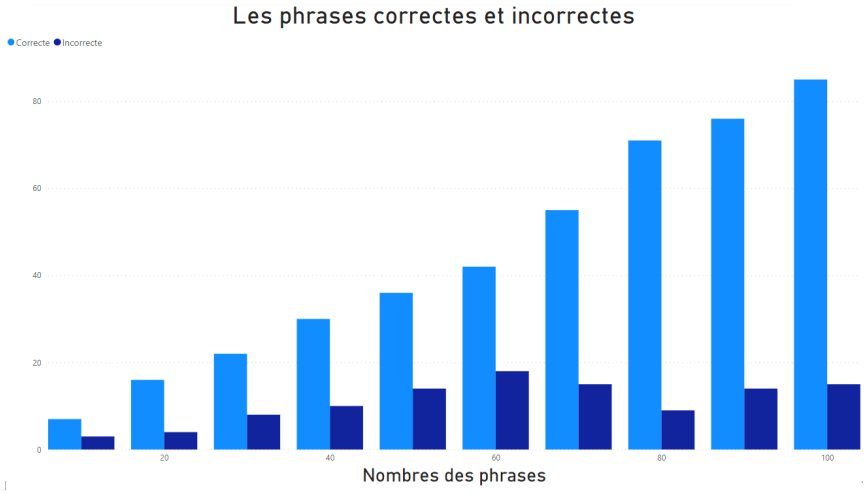

FIGURE 6.5 – L'analyse graphique pour l'observation des phrases

Maintenant nous pouvons mesurer la précision de notre système proposé. Le décodeur est également testé par cet ensemble de test, puis nous exécutons les scripts BLEU sur celui-ci.

Pour les expériences, nous prenons plusieurs ensembles de données qui contiennent des phrases de 10 jusqu'à 100. Nous observons la réussite et l'échec de la traduction des phrases. Nous représentons ces résultats graphiquement dans la figure 6.5.

Le résultat a été évalué en utilisant le BLEU (Bilingual Evaluation Understudy). La boîte à outils BLEU est utilisée pour calculer la qualité de la traduction produite par le système. La précision moyenne est de 80,7%.

6.3 Traduction automatique neuronale

La traduction automatique neuronale est une forme d'apprentissage de bout en bout qui peut être utilisée pour automatiser la traduction. Dans la traduction automatique neuronale, c'est le réseau neuronal du programme qui se charge de coder et de décoder le texte source, au lieu d'exécuter un ensemble de règles prédéfinies dès le départ.

La traduction automatique neuronale a donc le potentiel de résoudre de nombreux problèmes des systèmes de traduction traditionnels basés sur des phrases et il a été démontré qu'elle produit des traductions de meilleure qualité. Une réalisation de la consiste à utiliser un modèle puissant pour les données séquentielles, à savoir le réseau neuronal récurrent (RNN), à la fois pour le codeur et le décodeur [94].

Une forme de base de NMT consiste en deux composants : un encodeur qui calcule une représentation de la phrase source S et un décodeur qui génère un mot cible à la fois et décompose ainsi la probabilité conditionnelle. L'architecture encodeur-décodeur basée sur l'attention utilisée pour le NMT anglais-amazigh est présentée dans la figure 6.6

En général, le modèle proposé fonctionne comme suit :

— Lire les mots d'entrée un par un pour obtenir une représentation vectorielle à partir de chaque étape temporelle du codeur en utilisant un codeur basé sur GRU.
— Fournir la représentation de l'encodeur au décodeur.
— Extraire les mots de sortie un par un en utilisant un autre décodeur basé sur GRU qui est conditionné par les entrées sélectionnées à partir de l'état caché de l'encodeur à chaque pas de temps.

Avec cette configuration, le modèle est capable de se concentrer sélectivement sur les parties utiles de la séquence d'entrée et donc d'apprendre l'alignement (mise en correspondance des segments du texte original avec les segments correspondants de la traduction).

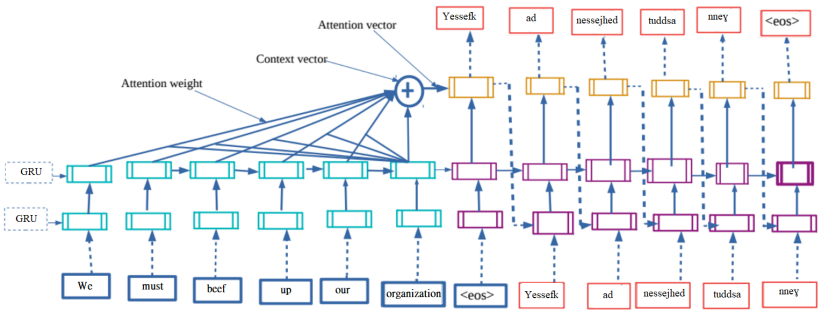

FIGURE 6.6 – L'analyse graphique pour l'observation des phrases

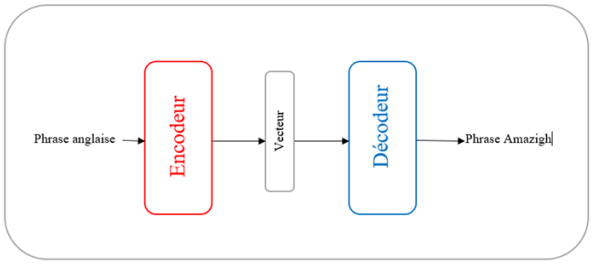

FIGURE 6.7 – L'architecture de l'encodeur et du décodeur

La figure 6.7 décrit l'architecture de l'approche générale pour NMT. Un encodeur

convertit une phrase source en un vecteur de sens qui est transmis à un décodeur pour produire une traduction.

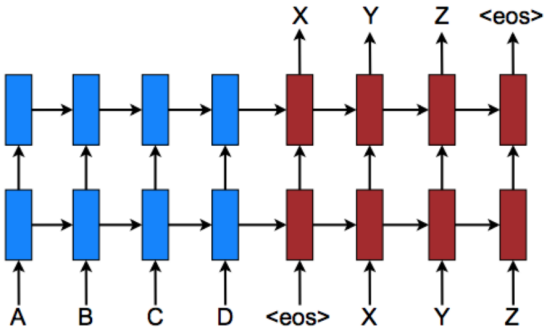

FIGURE 6.8 – La traduction automatique neuronale en tant qu'architecture récurrente à empilement pour traduire une séquence source A B C D en une séquence cible X Y Z.

La NMT modélise directement la probabilité conditionnelle $p(y|x)$ de traduire une phrase source $(x_1, x_2, ..., x_n)$ en une phrase cible $(y_1, y_2, ..., y_n)$. La NMT se compose de deux éléments :

— Un encodeur qui calcule une représentation S pour chaque phrase source.
— Un décodeur qui génère la traduction d'un mot à la fois et qui décompose donc la probabilité conditionnelle comme suit :

$$log(p(y|x)) = \sum_{j=1}^{n} log(p(y_j|y < j, s)) \tag{6.4}$$

On pourrait paramétrer la probabilité de décodage de chaque mot y_j comme suit :

$$p(y_i|y < j, s) = softmax(g(h_i)) \tag{6.5}$$

Où h_j peut être modélisé comme suit :

$$h_j = f(h_{j-1}, s) \tag{6.6}$$

Où
— g : une fonction de transformation qui produit un vecteur de taille de vocabulaire.
— h : unité cachée du RNN.
— f : calcule l'état caché actuel en fonction de l'état caché précédent.
L'objectif de l'entraînement pour le processus de traduction pourrait être formulé comme suit :

$$J_t = \sum_{(x,y)\in D} -log(p(y, x)) \tag{6.7}$$

Dans la plupart des architectures RNN non basées sur l'attention, la représentation de la source, S, n'est utilisée qu'une seule fois pour initialiser l'état caché du décodeur. Dans

la figure 6.8 le décodeur n'a accès qu'à la dernière couche de l'encodeur.

D'autre part, les réseaux basés sur l'attention se réfèrent à un ensemble d'états cachés source S tout au long du processus de traduction. Dans la figure 6.9 le décodeur a accès à tous les états cachés de l'encodeur.

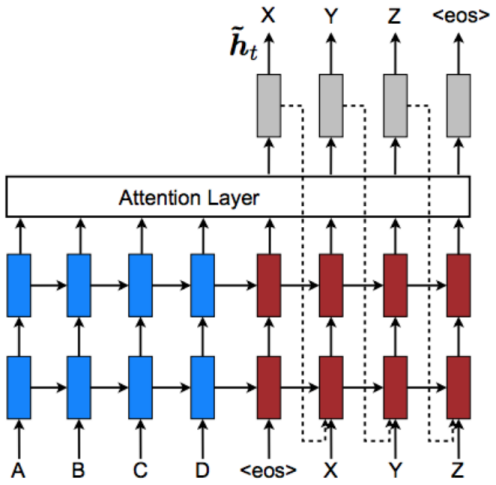

FIGURE 6.9 – NMT avec attention.

La figure 6.9 décrit une architecture d'encodeur-décodeur basée sur un RNN avec attention. Comme nous l'avons expliqué précédemment, l'attention peut être différenciée en deux types :

— L'attention globale : L'attention est placée sur toutes les positions de la source.
— L'attention locale : L'attention est placée uniquement sur quelques positions de la source.

Les deux modèles basés sur l'attention ne diffèrent de l'architecture normale de l'encodeur-décodeur que dans la phase de décodage. Ces méthodes basées sur l'attention diffèrent dans la manière dont elles calculent le vecteur de contexte c_t.

— h_t : État cible caché.
— c_t : Vecteur de contexte côté source.
— y_t : Mot cible actuel.
— $\tilde{h}_t$: Etat caché attentionnel.
— a_t : Vecteur d'alignement.

Les deux méthodes basées sur l'attention ont les étapes communes suivantes :

— Les deux approches prennent d'abord en entrée l'état caché de la couche supérieure d'un RNN à empilement.
— Déterminer c_t pour capturer les informations pertinentes du côté source afin d'aider à prédire y_t (cellule bleue supérieure). c_t est essentiellement le contexte que vous avez construit pour chaque mot en fonction de ses poids d'alignement et de l'état caché des encodeurs.

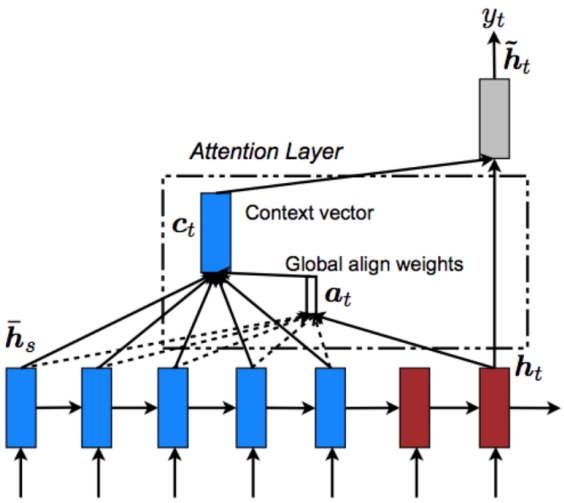

FIGURE 6.10 – État caché de l'architecture NMT avec attention globale.

— Calculez $\tilde{h}_t$ à partir d'une simple concaténation de h_t et c_t (cellule grise supérieure).

$$\tilde{h}_t = tanh(W_c)[c_t; h_t]) \tag{6.8}$$

Contrairement aux architectures non basées sur l'attention où seule la sortie finale de l'encodeur est fournie au décodeur, $\tilde{h}_t$ a accès à tous les états, des états cachés de l'encodeur qui fournit une vue informative de la phrase source.

— Le vecteur attentionnel est transformé en utilisant la couche $softmax$ pour produire la distribution prédictive. Il utilise la couche $softmax$ car il doit trouver le mot le plus probable parmi tous les mots disponibles dans le vocabulaire.

$$p(y_t|y < t, x) = softmax(W_s\tilde{h}_t) \tag{6.9}$$

Le paragraphe ci-dessus explique l'architecture de base des réseaux basés sur l'attention. Dans le paragraphe suivant, nous allons comprendre comment le vecteur de contexte c_t est calculé différemment dans l'attention locale et globale et quelles en sont les répercussions.

6.3.1 Attention globale

L'attention globale prend en considération tous les états cachés de l'encodeur pour dériver le vecteur de contexte c_t. Afin de calculer c_t, il calcule a_t qui est un vecteur d'alignement de longueur variable. Le vecteur d'alignement est dérivé en calculant une mesure de similarité entre h_t et $\bar{h}_s$ où h_t est l'état caché source et $\bar{h}_s$ l'état caché cible. Les états similaires dans le codeur et le décodeur renvoient en fait à la même signification.

Le vecteur d'alignement $a_t(s)$ est défini comme suit :

$$a_t(s) = align(h_t, \bar{h}_s) = \frac{exp(score(h_t, \bar{h}_s))}{\sum_{s'} exp(score(h_t, \bar{h}_{s'}))} \tag{6.10}$$

Le score est une fonction basée sur le contenu pour laquelle l'une des alternatives suivantes aurait pu être utilisée :

$$score(h_t, \bar{h}_s) = \begin{cases} h_t^T \tilde{h}_s \\ h_t^T W_a \tilde{h}_s \\ v_a^T tanh(W_a[h_t; \bar{h}_s]) \end{cases} \tag{6.11}$$

Grâce à la fonction de score, il essai de calculer la similarité entre les états cachés de la cible et de la source. Intuitivement, des états similaires dans le caché et la source font référence à la même signification mais dans des langues différentes.

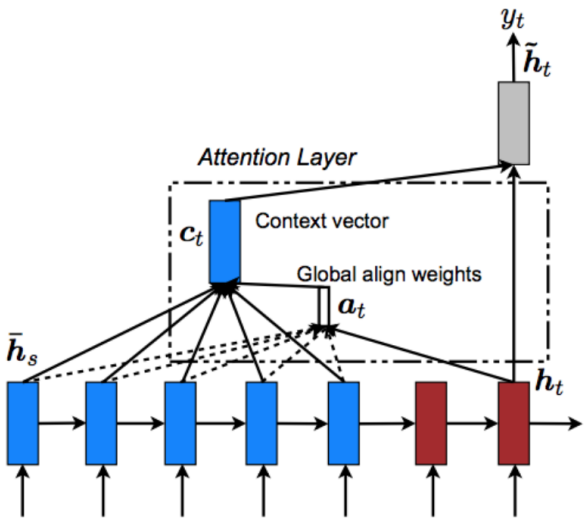

FIGURE 6.11 – Modèle de l'attention globale.

À chaque pas de temps t, le modèle infère un vecteur de poids d'alignement de longueur variable a_t basé sur l'état cible actuel h_t et tous les états sources $\bar{h}_s$. Un vecteur de contexte global, c_t est ensuite calculé comme la moyenne pondérée, selon a_t, de tous les états sources.

Les lignes de connexion de la figure 6.11 représentent les variables interdépendantes.

— a_t dépend de h_t et de $\bar{h}_s$.
— c_t dépend de a_t et de $\bar{h}_s$.
— $\tilde{h}_t$ dépend de c_t et h_t.

6.3.2 Attention locale

Comme l'attention globale se concentre sur tous les mots latéraux de la source pour tous les mots cibles, elle est très coûteuse en termes de calcul et n'est pas pratique pour la traduction de longues phrases. Pour surmonter ce problème, l'attention locale choisit de se concentrer uniquement sur un petit sous-ensemble d'états cachés de l'encodeur par mot cible.

L'attention locale comporte les étapes suivantes, qui sont différentes de celles de l'attention globale :

— Le modèle génère d'abord une position alignée p_t pour chaque mot cible au moment t. Contrairement au modèle d'attention globale où nous supposons un alignement monotone, nous apprenons les positions alignées dans l'attention locale. En d'autres termes, en plus d'apprendre les traductions, vous apprenez également si l'ordre de la traduction est différent de celui de la phrase source (le mot 1 de la phrase source pourrait être le mot 4 dans la phrase traduite, nous devons donc le calculer, sinon notre score de similarité sera erroné car notre attention sera focalisée sur un mot de la phrase source qui n'est pas lié au mot 1 de la phrase source).
— Le vecteur de contexte c_t est dérivé comme une moyenne pondérée sur l'ensemble des états cachés de la source dans la fenêtre $[p_t - D, p_t + D]$; D est choisi empiriquement. Par rapport au vecteur d'alignement global, le vecteur d'alignement local a_t est maintenant de dimension fixe.

Jusqu'à présent, nous avons supposé que la phrase traduite et la phrase source sont alignées de manière monotone. Sur cette base, nous avons une différenciation supplémentaire de l'attention locale qui est la suivante :

6.3.2.1 Alignement monotone

Définir $p_t = t$, ce qui signifie que nous supposons que les séquences source et cible sont grossièrement alignées de façon monotone. Le vecteur d'alignement est le même que l'alignement global de l'équation 6.10.

6.3.2.2 Alignement prédictif

Au lieu de supposer des alignements monotones, le modèle prédit une position alignée comme suit :

$$p_t = S.sigmoid(v_P^T tanh(W_p h_t)) \tag{6.12}$$

W_p et v_p sont des paramètres de modèles qui seront appris pour prédire les positions. Avec S est la longueur de la phrase source et p_t entre 0 et S. Pour favoriser la position d'alignement p_t, nous plaçons une distribution gaussienne centrée autour de p_t. Cela donne plus de poids à la position p_t. Nous modifions nos poids d'alignement comme suit pour capturer la même chose :

$$a_t(s) = align(h_t, \bar{h}_s) exp(-\frac{(s - p_t)^2}{2\sigma^2}) \tag{6.13}$$

Pour résumer, l'attention globale est plus coûteuse en calcul et est inutile pour les phrases longues, tandis que l'attention locale se concentre sur les états cachés D des deux côtés de $p(t)$ pour surmonter cela. L'attention locale a deux saveurs : local-m (l'alignement de la source et de la cible est supposé être le même) et local-p (où nous calculons le $p(t)$).

6.4 Traduction automatique neuronale avec transformateurs

Les architectures basées sur les RNN ont démontré les meilleures performances dans les difficultés de traduction automatique ces dernières années, mais elles ont encore des problèmes à résoudre. Tout d'abord, elles ont du mal à gérer la dépendance à long terme (tout comme le LSTM lorsqu'il doit traiter des phrases très longues). Deuxièmement, chaque état dissimulé dépend de celui qui le précède, ce qui rend le processus difficile à paralléliser et inefficace sur les GPU.

Ces modèles sont constitués de deux composants : un encodeur et un décodeur. Au niveau du traitement, l'encodeur condense toute la sémantique de la séquence en un seul vecteur, qu'il fournit ensuite au décodeur. Le décodeur analyse les données et génère des prédictions. Le problème est que le décodeur a besoin de différentes informations à différents moments, et qu'il reçoit toutes les informations sur les dépendances de la séquence dans un seul vecteur traité. Par conséquent, il est extrêmement difficile pour le décodeur de percer et d'extraire les informations secrètes, ce qui constitue le principal défi et l'objectif de la méthode de l'attention ajoutée ultérieurement.

La méthode de l'attention permet au décodeur de revenir sur l'ensemble de la séquence d'entrée et de récupérer les données dont il a besoin pendant le traitement. Le LSTM était utile pour traiter des séquences étendues, mais il ne parvenait pas à sauvegarder les informations globales de la phrase originale. Par exemple, le mot "like" correspond au mot "dancing" dans l'énoncé "I prefer dancing over swimming". Pendant la lecture de la phrase source, cette relation doit être stockée dans la mémoire à long terme et utilisée pour générer la phrase cible.

Après avoir traduit le mot "dancing", il devra savoir à quoi le terme "dancing" est comparé, mais il n'aura plus besoin de se souvenir de l'expression "dancing".

Une autre différence est que ces réseaux (RNN, LSTM, GRU) analysent les séquences de représentation des symboles clé par clé, plutôt que de travailler sur la séquence complète en une seule fois.

6.4.1 Définition de transformer

Le Transformer est une nouvelle approche pour résoudre le problème de la traduction automatique qui dépend fortement du mécanisme d'attention pour dessiner les dépendances. Par rapport aux conceptions précédentes, le Transformer a obtenu les meilleurs scores BLEU sur les tâches de traduction automatique évaluées par l'équipe de recherche de Google, nécessitant beaucoup moins de calculs et de temps d'apprentissage.

Le transformateur utilise un mécanisme d'auto-attention à chaque phase, qui modélise directement les relations entre tous les mots d'une phrase, indépendamment de leur position. Le transformateur, comme les RNN, est une architecture permettant de convertir une séquence en une autre en utilisant le processus d'encodage-décodage, mais il diffère des modèles Seq2Seq précédents en ce qu'il ne nécessite pas l'utilisation d'un réseau récurrent (GRU, LSTM, etc.). Contrairement aux RNN, cependant, le Transformer traite la séquence d'entrée complète en une seule fois plutôt que d'itérer mot par mot.

Regardons de plus près l'architecture du transformateur et voyons comment il fonctionne. L'architecture du transformateur est représentée dans la figure 6.12. L'encodeur est à gauche, tandis que le décodeur est à droite.

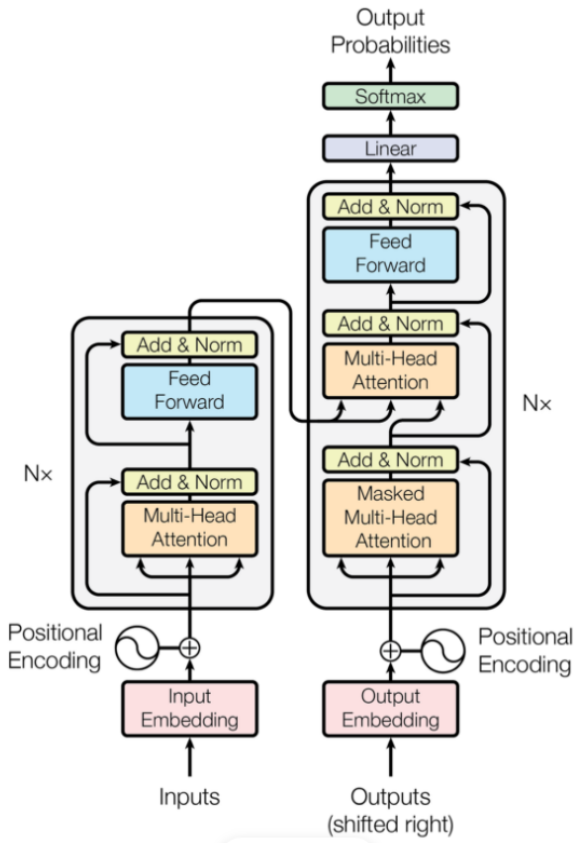

FIGURE 6.12 – Architecture de transformateur.

6.4.2 Encodeur

Chaque mot de la phrase d'entrée est pris par l'encodeur, qui le convertit en une représentation intermédiaire et le compare à tous les autres mots de la phrase. Le résultat de cette comparaison est un score d'attention qui évalue la contribution de chaque mot de la phrase au mot clé. Les notes d'attention sont ensuite utilisées comme poids pour les représentations des mots, qui sont fournies au réseau entièrement connecté, qui développe

une nouvelle représentation pour le mot clé. Il le fait pour tous les mots de la phrase et envoie la nouvelle représentation au décodeur, qui peut utiliser ces informations pour créer toutes les dépendances dont il a besoin pour faire des prédictions.

6.4.3 La structure de l'encodeur

L'encodeur est composé de deux pièces qui sont reproduites six fois (un nombre arbitraire) comme le montre le schéma ci-dessus :

— Mécanisme d'auto-attention multi-têtes

— Réseau Feed-Forward entièrement connecté

Les vecteurs d'intégration sont envoyés à l'encodeur sous la forme d'une liste de vecteurs, chacun ayant une dimension de 512 (qui peut être ajustée comme un hyperparamètre). L'encodeur et le décodeur ajoutent tous deux un codage positionnel à leur entrée (qui sera décrit plus loin). Tous les deux emploient une dérivation de connexion résiduelle, qui est suivie par l'inclusion de l'entrée de la couche sous-originale et d'une autre couche de normalisation (qui est également connue sous le nom de normalisation par lots).

6.4.4 Le décodeur

Tous les états secrets de l'encodeur sont accessibles au décodeur, qui les utilise à chaque étape pour anticiper le mot suivant. Comme tous les états cachés ne sont pas significatifs à chaque étape, l'état de chaque décodeur est pondéré différemment, et le modèle apprend à se "concentrer" sur les bits pertinents de l'entrée en fonction de la cible. Le décodeur reçoit l'entrée du codeur et la sortie précédente du décodeur à chaque itération et utilise les deux pour l'étape suivante.

6.4.5 La structure du décodeur

Le décodeur est composé de trois pièces qui sont dupliquées six fois chacune :

— Mécanisme d'auto-attention masqué à plusieurs têtes

— Mécanisme d'auto-attention multi-têtes

— Réseau Feed-Forward entièrement connecté

Contrairement au codeur, le décodeur utilise le masquage en plus de l'attention multi-têtes (Multi-Head). Le but de cette procédure est d'empêcher le décodeur de divulguer des informations postérieures. Cela implique qu'au niveau de l'entraînement, le décodeur n'a pas accès aux tokens de la phrase cible qui indiqueraient la bonne réponse, ce qui perturbe le processus d'apprentissage. Il s'agit d'un composant essentiel du décodeur, car sans lui, le modèle n'apprendrait rien et se contenterait de répéter le texte cible.

6.4.6 Encodage par position (Positional Encoding)

Le transformateur diffère des modèles de réseaux récurrents en ce qu'il traite les composants comme une seule unité, ce qui entraîne la perte de l'ordre des éléments de la séquence.

Comme ces informations sont importantes pour les étapes ultérieures, l'attention utilise une fonction avant l'encodeur qui ajoute une partie supplémentaire au vecteur d'encastrement appelée Encodage par Position (PE). Sur la base des calculs de sinus et de cosinus, cette fonction génère un indice qui révèle l'emplacement du mot spécifique dans le texte.

6.4.7 Connexions résiduelles (Residual Connections)

Le Transformer, comme ResNet, ajoute l'entrée qui est entrée dans chaque sous-couche avant le traitement à la sous-couche de chaque sortie (indiquée par "Add & Norm" dans les cases jaunes de la figure 6.12). La liaison restante a été créée pour simplifier l'optimisation. Il enregistre les données avant chaque opération pour permettre un apprentissage plus rapide dans la phase de rétropropagation, qui compare la sortie du modèle à l'objectif que nous visons.

L'entrée avant chaque sous-couche permet à la technique de rétropropagation de mettre à jour les poids de manière simple et efficace, ce qui est important puisque nous voulons les mettre à jour avec précision en cours de route. Le réseau sera capable de mettre à jour plus rapidement les derniers niveaux qui sont les plus proches de la sortie réelle ; mais, les couches des premiers stades s'estomperont avec le temps en raison des processus mathématiques. Au lieu d'essayer de faire face à une sortie compliquée si quelque chose ne va pas le long de la route, l'ajout de l'entrée d'origine facilite l'apprentissage des poids. Il s'agit d'une sorte d'étape intermédiaire de comparaison entrée-sortie après chaque opération qui empêche la méthode de décodage de produire un résultat monstrueux. Sans cette action, le mécanisme d'attention aura du mal à trouver comment ajuster chaque sous-couche dans la phase d'apprentissage pour améliorer les résultats.

6.4.8 Réseau feed-forward entièrement connecté(Fully Connected Feed-Forward Network)

L'encodeur et le décodeur ont tous deux deux couches entièrement connectées, suivies d'une fonction d'activation (ReLU). L'objectif de ces couches est d'améliorer le processus d'apprentissage et de lui permettre d'apprendre de nouvelles dépendances de séquence par lui-même.

6.4.9 Multi-Têtes-Automatisme (Multi-Head-Self-Attention)

Vérifier comment chaque mot d'une phrase est relié aux autres mots est le principe de l'auto-attention. La technique du produit scalaire est utilisée par l'attention pour vérifier la similitude de deux mots représentés par des vecteurs. Prenons l'exemple des phrases suivantes :

— " the cat didn't eat the fish because it is not hungry ".
— " the cat didn't eat the fish because it is sick ".

Le mot « it » fait référence à « the cat » dans la première, alors que le mot « it » fait référence à « the fish » dans la seconde. Lorsque le modèle analyse le mot « it », l'auto-attention découvre des corrélations entre « it » et les mots « the cat » dans le premier exemple et « the fish » dans le second.

Un seul aspect d'une phrase peut être capté par l'attention, et les phrases en comportent généralement plus d'un. L'encodage du mot « it », par exemple, fera qu'une tête d'attention créera une relation avec le terme « cat », tandis que l'autre trouvera une relation avec le mot « hungry » ; d'où l'appellation "Multi-Têtes". Le transformateur emploie de nombreuses attentions, chacune d'entre elles travaillant avec un composant différent de la phrase, et elles travaillent toutes ensemble pour préserver toutes les relations dans la phrase.

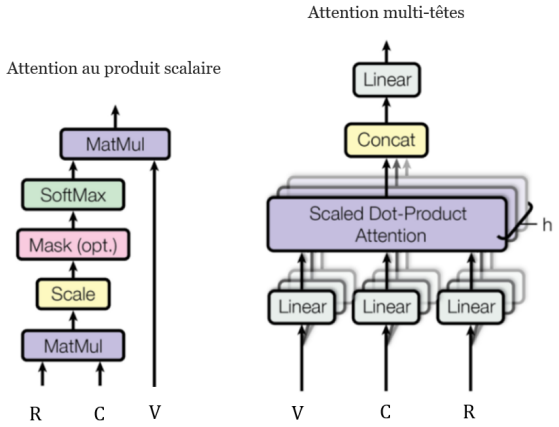

FIGURE 6.13 – Etapes des attentions au produit scalaire et multi-têtes.

Nous allons maintenant examiner la formule du produit scalaire de l'attention, qui est calculé à l'aide de l'équation suivante :

$$Attention(R, C, V) = softmax(\frac{RC^T}{\sqrt{d_k}})V \tag{6.14}$$

6.4.9.1 Création d'une clé, une valeur et une requête

Si nous regardons attentivement la figure 6.13 nous pouvons voir trois flèches qui entrent dans la boîte orange du système d'auto-attribution multi-têtes. La requête, la clé et la valeur sont les trois vecteurs d'entrée représentés par les trois flèches. Les vecteurs d'incorporation sont multipliés par trois matrices qui sont formées tout au long de la procédure de formation pour former ces vecteurs pour chaque mot.

L'opération d'attention est une procédure du système de recherche qui utilise les principes clé/valeur/requête. Par exemple, si vous tapez une requête dans un moteur de recherche pour trouver un article sur google, le moteur de recherche va essayer d'identifier un ensemble de clés (titre de l'article, description, etc.) sur la base des restrictions de la

requête. Les articles seront présentés par ordre décroissant dans une liste des meilleures correspondances. Les valeurs pondérées calculées dans l'attention sont mises en parallèle avec ce classement.

- Vecteur de Requêtes : L'état caché du décodeur est défini par un vecteur mot dans la séquence.
- Vecteur de clés : L'état caché de l'encodeur est défini par tous les mots de la séquence.
- Vecteur de valeurs : Les poids d'attention des états cachés du codeur sont définis par tous les mots de la séquence.

L'auto-attention est appliquée aux séquences source et cible individuellement par le codeur et le décodeur, comme le montre la figure 6.13. En outre, le décodeur accorde une attention particulière à R, qui est dérivée de la séquence cible, et à C, V, qui est dérivée de la séquence source.

Nous obtiendrons ces vecteurs en multipliant le vecteur d'intégration de chaque mot (par exemple, "X_1" représente le mot "programming") par les matrices de poids $W_r, W_c et W_c$. Cette multiplication donnera les vecteurs de requête, de clé et de valeur $r_1, c_1 et v_1$ (vecteurs de requête, de clé et de valeur) associés au mot "programming" et $r_1, c_1 et v_1$ (vecteurs de requête, de clé et de valeur) associés au mot "Machines".

6.4.9.2 Calcul des scores

Les scores sont calculés dans la deuxième phase. Si nous évaluons l'auto-attention pour le premier mot, dans ce cas "Programming", le score de chaque mot de la phrase d'entrée sera comparé à ce mot.

Nous ferons le produit scalaire du vecteur de requête r_1 et du vecteur clé du mot que nous évaluons pour obtenir le score. Par exemple, pour évaluer l'expression "Machines" tout en calculant l'auto-attention pour " Programming ", nous produirons le produit scalaire de r_1 par c_1.

6.4.9.3 Normalisation

La dimension des requêtes et des clés est représentée par d_k dans l'équation 6.15, et elle est utilisée pour standardiser les scores.

6.4.9.4 Softmax

Les scores fractionnés sont ensuite envoyés à l'algorithme $softmax$. Cette fonction calcule la probabilité et s'assure que les scores sont tous positifs et que leur somme est égale à un.

6.4.9.5 Multiplication par la valeur

Ensuite, le score de chaque $softmax$ sera multiplié par le vecteur de valeur de chaque mot.

6.4.9.6 Vecteur de synthèse

Enfin, la sortie de la couche d'auto-attention pour le mot actuel est donnée par la somme des vecteurs de valeur pondérés.

6.5 Résultats des modèles basés sur les réseaux de neurones

6.5.1 Évaluation des modèles linguistiques

Des mesures d'évaluation automatiques ont été utilisées pour évaluer la qualité du système de traduction automatique.

Les données de l'entraînement sont utilisées pour faire des choix entre différents modèles, ou pour ajuster les hyperparamètres du modèle. Les hyperparamètres dans les modèles ci-dessus peuvent inclure la longueur maximale de n dans le modèle n-gram ou le type de méthode de lissage.

Les données de test sont utilisées pour mesurer la notre précision finale et rapporter les résultats.

Les paramètres des modèles sont les suivants :

— batch_size : la taille du lot doit être choisie avec soin car la traduction automatique neuronale nécessite une quantité importante de mémoire lors de son exécution.

— num_nodes : cela représente le nombre de nœuds cachés dans le LSTM. Un grand nombre de nœuds se traduira par de meilleures performances et un coût de calcul plus élevé.

— embedding_size : c'est la dimensionnalité des vecteurs. En général, une taille d'intégration de 100 à 300 est adéquate pour la plupart des problèmes du monde réel qui utilisent des vecteurs de mots.

Les paramètres d'évaluation utilisés sont les suivants.

6.5.1.1 BLEU score

Il s'agit d'une métrique d'évaluation automatique qui signifie "Bilingual Evaluation Understudy".

Le BLEU est calculé en comptant les mots du résultat de la traduction automatique qui correspondent à la traduction de référence. Le score BLEU va de 0 à 1 ou (0 à 100), 0 indiquant l'absence de correspondance et 1 indiquant toutes les correspondances ce qui

n'est pas possible pour toutes les phrases de test.

Le score BLEU est calculé comme suit :

$$precision = \frac{P}{T} \tag{6.15}$$

Où

— P : phrase candidate mots en reference.

— T : total des mots de la phrase de reference.

Les petites phrases sont préférées par la précision. Dans l'évaluation, cela ouvre la possibilité que la traduction automatique puisse construire des phrases minuscules pour des références longues tout en maintenant une précision élevée. Pour contourner ce problème, *Brevitypenalty* a été mise en œuvre. La précision $n-gram$ modifiée P_n a un poids de W_n.

$$Brevity\ penalty(bp) = \begin{cases} 1, & if\ \ c > r \\ e^{1-(\frac{r}{c})}, & if\ \ c \leq r \end{cases} \tag{6.16}$$

Où c est la longueur de la phrase candidate et est la longueur de la phrase de référence.

$$BLEU = bp.exp(\sum_{n=1}^{N} w_n log(p_n)) \tag{6.17}$$

6.5.1.2 Perplexité

Pour les modèles de la langue, nous voulons essentiellement savoir si le modèle est un modèle précis de la langue, et il y a plusieurs façons de le définir. La façon la plus directe de définir la précision est la vraisemblance du modèle par rapport aux données de développement ou de test. La vraisemblance des paramètres θ par rapport à ces données est égale à la probabilité que le modèle attribue aux paramètres θ. Par exemple, si nous avons un ensemble de données de test ϵ_{test}, c'est $P(\epsilon_{test}; \theta)$.

Nous supposons souvent que ces données sont constituées de plusieurs phrases ou documents indépendants E, ce qui nous donne :

$$P(\epsilon_{test}; \theta) = \prod_{E \in \epsilon_{test}} P(E; \theta) \tag{6.18}$$

Une autre mesure couramment utilisée est la vraisemblance logarithmique.

$$log(P(\epsilon_{test}; \theta)) = \sum_{E \in \epsilon_{test}} log(P(E; \theta)) \tag{6.19}$$

La vraisemblance logarithmique est utilisée pour plusieurs raisons. La première est que la probabilité d'une phrase particulière selon le modèle de langage peut être un très petit nombre, et que le produit de ces petits nombres peut devenir un nombre très petit qui posera des problèmes de précision numérique sur le matériel informatique standard.

La deuxième raison est qu'il est parfois plus pratique mathématiquement de traiter dans l'espace logarithmique. Par exemple, lorsque l'on prend la dérivée dans les méthodes basées sur le gradient pour optimiser les paramètres, il est plus pratique de traiter la somme dans l'équation 6.19 que le produit dans l'équation 6.18.

Il est également courant de diviser la vraisemblance logarithmique par le nombre de mots du corpus.

$$length(\epsilon_{test}) = \sum_{E \in \epsilon_{test}} |E| \tag{6.20}$$

Il est ainsi plus facile de comparer et de contraster les résultats entre des corpus de longueurs différentes.

La dernière mesure commune de la précision d'un modèle de langage est la perplexité, qui est définie comme étant comme l'exposant du logarithme négatif moyen de la vraisemblance par mot.

$$ppl(\epsilon_{test}; \theta) = e^{-\frac{log(P(\epsilon_{test};\theta)}{length(\epsilon_{test})}} \tag{6.21}$$

Une explication intuitive de la perplexité est "à quel point le modèle est-il confus quant à sa décision ?". Plus précisément, elle exprime la valeur "si nous choisissons au hasard des mots dans la distribution de probabilité calculée par le modèle de langage à chaque pas de temps, en moyenne combien de mots devrait-il choisir pour obtenir le mot correct ?" Une des raisons pour lesquelles il est courant de voir des perplexités dans les documents de recherche est que les nombres calculés par perplexité sont plus grands, ce qui rend les différences entre les modèles plus facilement perceptibles par l'œil humain.

6.5.2 Préparation des données et prétraitement du texte

6.5.2.1 Télécharger et préparer les données

Un corpus parallèle de 137322 phrases est utilisé. Ce corpus contient des phrases tifinagh scraper du web et, des nouvelles phrases couramment utilisées dans la vie quotidienne écrit en Latin. Le corpus a été nettoyé et divisé en phrases. Après ces opérations, une validation manuelle a été effectuée pour vérifier l'absence d'erreurs.

Nous avons importé les données à partir du drive sous forme des fichiers textes, après nous avons préparé ces données, voici les étapes à suivre :

— Ajouter un jeton START et END de chaque phrase.
— Nettoyer les phrases en supprimant les caractères spéciaux.
— Créer un index de mots et un index de mots inversé (mappage de dictionnaires de mot → id et id → mot).
— Compléter chaque phrase jusqu'à une longueur maximale.

À partir de ces tableaux de chaînes, nous avons créé un tf.data.Dataset de chaînes et les mettre en lots de manière efficace.

6.5.2.2 Prétraitement du texte - Normalisation

Le modèle NMT est un modèle qui peut être exporté comme tf.saved_model. Pour ce modèle, il devrait prendre des entrées tf.string, et des sorties tf.string , tout le traitement du texte se fait à l'intérieur du modèle.

Le modèle traite du texte bilingue avec un vocabulaire limité. Il sera donc important de normaliser le texte d'entrée.

La première étape est la normalisation Unicode pour séparer les caractères accentués et remplacer les caractères de compatibilité par leurs équivalents ASCII. Le paquage tensorflow_text contient une opération de normalisation Unicode.

6.5.2.3 Modèle de NMT Tansorflow

Cette méthode est mise en œuvre à l'aide de six couches d'encodeurs et de six couches de décodeurs, ainsi que d'un corpus de 137322 phrases parallèles. On a utilisé Google Colab pour tester le modèle.

Il s'agit de valeurs ou de configurations qui ne peuvent pas être déduites des données mais qui sont utilisées pour estimer les paramètres du modèle et qui sont externes au modèle.

BLEU	**PRESICION**	**RECALL**	**F-measue**
38,2	60,5	29	39,2
39,2	84,72	28,6	42,76
38,5	63,5	26,5	37,39
48	84,56	28,14	42,22
45,6	85,3	29,3	43,61
29	67,3	27,5	39,04
75,5	61,5	22,54	32,98
54,3	63,14	20,5	30,95
29,3	81,6	28,6	42,35
38,4	79,65	29,6	43,16
37,9	54,6	23,5	32,85
65	58,2	26,3	36,22
35,1	59,6	27,4	37,54
25,3	63,5	20,6	31,10
56,7	78,5	29,3	42,67
45,5	85,2	25,9	39,72
48,6	82,63	28,6	42,49
55	65,2	23,12	34,13
42,3	84,65	25,4	39,07
47,9	83,5	27,8	41,71

BLEU	PRESICION	RECALL	F-measue

TABLE 6.1 : Les valeurs métriques d'évaluation du modèle de tensorflow

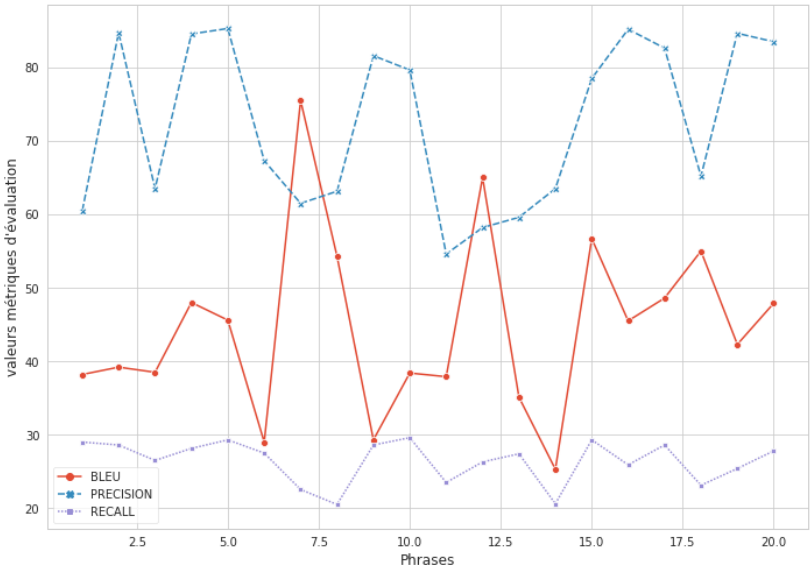

FIGURE 6.14 – Mesures BLEU, PRECISION, et RECALL du modèle de tensorflow.

Le tableau 6.1 montre les différentes valeurs obtenues pour de nombreuses métriques d'évaluation après des simulations étendues. Le score BLEU moyen obtenu est de 44.76.

La figure 6.14 illustre les représentations graphiques des résultats du tableau 3. Le graphique montre clairement que plus le taux d'erreurs de mots augmente, plus le score BLEU diminue, et plus le taux d'erreurs de mots diminue, plus le score BLEU augmente. Cela s'explique par le fait que plus le taux de mots erronés est élevé, plus le score BLEU est faible ; lorsque le taux de mots erronés est faible, cela signifie que la qualité de la traduction est bonne, et donc que le score BLEU est bon.

6.5.3 Traduction automatique neuronale utilisé PyTorch (OpenNMT)

Pour construire le modèle NMT et traduire le texte de l'amazigh à l'anglais, le système OpenNMT, qui est une boîte à outils open-source pour NMT, a été utilisé. La performance du système NMT est grandement influencée par le prétraitement des caractères tifinagh et anglais. Les phrases sont séparées et alignées manuellement. Après des tests approfondis, la longueur maximale des séquences source et cible a été fixée à 44, la taille maximale des lots pour la formation et la validation a été fixée à 128, et le taux de Dropout a été fixé à

0,3 pour les types de RNN LSTM avec optimisation Adam.

Les autres paramètres sont définis sur leurs valeurs par défaut. Le système enregistre le modèle pour chacune 10 époques, puis calcule la précision et la perplexité 10 fois pour chaque modèle. La perplexité fait référence à la facilité avec laquelle une distribution de probabilité (le modèle) peut anticiper la séquence suivante. Le modèle de traduction est bon pour prédire/traduire l'ensemble de test si la perplexité est faible.

epochs	**PPL**	**Accuracy**	**Val PPL**	**Val Accuracy**
10	6944,479	32,85	40,15	38,81
20	20,99	48,74	18,52	50,43
30	11,64	55,98	14,13	50,81
40	9,47	52,88	14,89	48,38
50	10,25	53,56	9,17	57
60	6,12	64,23	8,55	58,3
70	5,46	63,96	6,65	61,46
80	6,5	64,11	7,24	59,01
90	5,25	68,54	5,65	64,21
100	4,92	72,5	5,33	68,5
110	4,23	75,63	4,95	71,53
120	4,12	75,89	4,65	71,62
130	3,86	78,23	4,3	74,33
140	3,75	79,95	3,95	75,26
150	3,22	81,45	3,78	77,32
160	3,12	83,5	3,56	79,14
170	2,96	85,4	3,1	81,23
180	2,31	86,87	2,85	82,88
190	2,1	88,56	2,65	84,26
200	1,95	91,3	2,5	87,9

TABLE 6.2 : Les valeurs métriques du modèle OPENNMT

La figure 6.15 et le tableau 6.2 montre l'évaluation de la NMT basée sur LSTM, où "Val ppl", "Val Acc", "PPL" et "Accuracy" représentent respectivement la perplexité de validation, la précision de validation, le score moyen de prédiction et la perplexité de prédiction. Le résultat indique que la NMT basée sur LSTM donne des bons résultats, il atteint à un score de 87.9

6.5.4 Transformer models for NMT

L'architecture Transformer a réussi à couvrir un grand nombre de paires de langues avec une grande précision dans les tâches de MT, notamment dans des modèles tels que

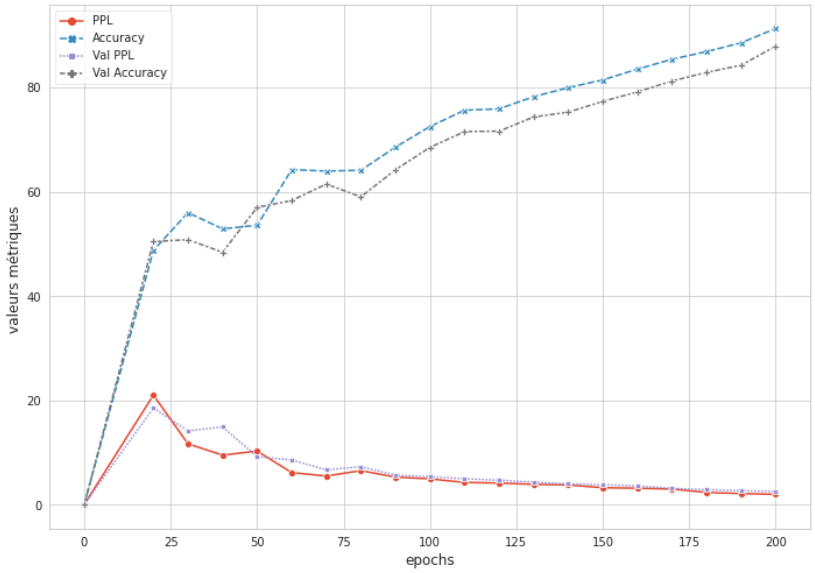

FIGURE 6.15 – Mesures PPL, Accuracy du modèle OPNNMT.

BART [95] et mBART [96]. mT5 [97] donne également de bons résultats avec un ensemble encore plus grand de langues, dont beaucoup sont considérées comme à faibles ressources. Ces modèles sont également très adaptables à d'autres tâches de NLP grâce à un réglage fin [96].

Nous avons construit un système de MT basé sur l'architecture Transformer. Étant donné que Transformer est l'architecture de pointe pour la MT et que notre système a été entraîné sur des quantités considérables de données du domaine, les traductions produites par notre système peuvent être considérées comme représentatives de la qualité qui peut être atteinte aujourd'hui par la MT pour la traduction de la langue amazigh à la langue anglaise.

epochs	**Loss**	**Accuracy**	**Val Loss**	**Val Accuracy**
10	1,25	65,25	1,56	64,25
20	0,95	73,56	0,97	67,56
30	0,85	75,8	0,94	69,3
40	0,77	77,23	0,89	70,65
50	0,68	81,25	0,78	72,6
60	0,63	83,56	0,73	75,56
70	0,59	85,21	0,68	78,24
80	0,54	86,24	0,61	79,84
90	0,51	87,14	0,58	80,44

epochs	Loss	Accuracy	Val Loss	Val Accuracy
100	0,47	87,86	0,52	82,13
110	0,43	89,32	0,49	83,26
120	0,41	90,15	0,46	83,94
130	0,39	90,98	0,45	84,5
140	0,36	91,2	0,41	85,21
150	0,31	91,8	0,39	86,3
160	0,29	92,3	0,37	87,2
170	0,27	92,75	0,35	88,5
180	0,26	93,5	0,34	89,4
190	0,25	93,9	0,33	90,21
200	0,24	94,56	0,31	91,3

TABLE 6.3 : Les valeurs métriques du modèle Transformer

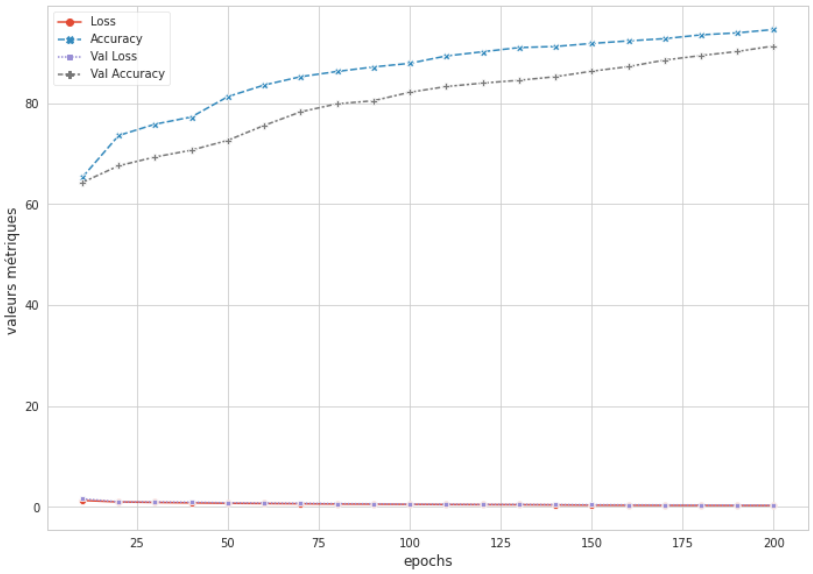

FIGURE 6.16 – Mesures Loss, et Accuracy du modèle de transformer.

La figure 6.16 et le tableau 6.3 présente l'évolution des métriques de performance après chaque 10 époques d'entraînement pendant l'entraînement des modèles. Il est évident que l'inclusion des informations syntaxiques apporte des avantages notables pour la plupart des métriques adoptées, on observe que l'accuarcy de l'entrainement augment du 65.25% jusqu'à 94.56%, et l'accuracy de validation augment du 64.25% jusqu'à 91.3%.

6.5.5 Comparison des systems NMT

Nous avons entraîné et évalué un système NMT bilingues basés sur les cellules GRU. Le tableau 6.4 résume les hyper-paramètres utilisés pour tous nos modèles. Les expériences GRU sont réalisées à l'aide de la boîte à outils NMT Nematus [98], tandis que les modèles LSTM et de transformateurs sont entraînés à l'aide de la boîte à outils open source OpenNMT-tf3. Les hyper-paramètres de formation et d'inférence pour les approches et les boîtes à outils sont fixés comme suit.

Encoder/ decoder	**Embiding size**	**Hidden units**	**Encoder depth**	**Decoder depth**	**Batch size**	**Accuracy%**
GRU	256	1024	2	2	64	85.2
LSTM	1024	1024	2	2	128	87.9
Transformer	512	512	6	6	64	91.3

TABLE 6.4 : Hyper-paramètres utilisés pour entraîner les modèles GRU, LSTM et de transformateur.

Les résultats montrent que le modèle de transformer est plus performants en terme d'accuracy, il atteint 91.3% dans la validation du modèle puis le modèle de LSTM atteint 87.9% finalement le modèle du GRU avec 85.2% dans la validation. D'après l'entrainement du modèle, on peut conclure que le modèle du transformer est plus rapide par 30% par rapport au modèle de GRU et 45% par rapport au modèle de LSTM, Cela est dû à l'utilisation de GPU dans les transformer.

6.6 Conclusion

La traduction neuronale de la traduction automatique est un nouveau paradigme dans la recherche sur la traduction automatique. Dans ce chapitre, un modèle d'encodage-décodage profond basé sur LSTM pour la traduction d'amazigh vers l'anglais est réalisé. Cette recherche a utilisé le mécanisme d'attention de Bahdanau. Un corpus parallèle amazigh-anglais de 137322 phrases a été utilisé. Pour évaluer l'efficacité du système proposé, plusieurs mesures d'évaluation automatiques comme BLEU, F-measure, Accuracy, Perplexité, etc. ont été utilisées. Après des simulations approfondies, des différents résultats sont obtenus ; 80,7% en utilisant la méthode statique, 85.2% par le modèle GRU, 87.9% par le modèle LSTM et 91.3% avec l'utilisation de transformer.

Conclusion générale et perspectives

La langue amazighe a été enseignée pour la première fois dans les écoles marocaines en septembre 2003, suite au discours de l'Ajdir Royal du 17 octobre 2001 et à la création de l'Institut Royal de la Culture Amazighe (IRCAM), qui est chargé de promouvoir la langue et la culture amazighes dans tous les aspects de la vie marocaine, y compris l'éducation et la formation. Verticalement, elle est enseignée de la 1ère à la 6ème année de l'école primaire, tandis qu'horizontalement, elle fait partie du programme de diverses écoles sans être enseignée de manière universelle. Son statut a été élevé de "langue nationale" à "langue officielle" en juillet 2011, et il a été reconnu comme la langue arabe dans la Constitution marocaine comme l'une des langues officielles du pays.

La langue amazighe a nécessité la construction de son corpus pour son enseignement en raison de sa présence sur le territoire marocain sous forme de variétés (Tarifit, Tamazight, Tachelhit). Le choix de l'écriture avec laquelle cette langue sera enregistrée pour être enseignée, apprise et diffusée a été la première étape de sa normalisation, et l'écriture tifinagh a été choisie en février 2003.

Dans plusieurs langues et dialectes, l'étiquetage POS est le processus consistant à attacher des marqueurs de caractéristiques grammaticales à chaque mot d'un texte instructif. Une séquence de mots (tokenisés) et un ensemble d'étiquettes sont introduits dans un calcul d'étiquetage, et le résultat est un arrangement d'étiquettes, une pour chaque token. Pour appliquer des approches d'apprentissage automatique ou d'apprentissage profond sur la langue amazigh, on a créé un corpus annoté manuellement contient plus que 60 000 tokens écrits en tifinagh en se basant sur un corpus de 20 000 tokens écrit en latin.

Nous avons utilisé deux algorithmes d'apprentissage automatique pour prédire POS des mots amazigh arbre de décision et champs aléatoires conditionnels, ces deux algorithmes donnent des résultats satisfit en terme de précision 87.60% par l'algorithme d'arbre de décision et 93.40% par l'algorithme des champs aléatoires conditionnels. Ce qui concerne les techniques d'apprentissage profond, on a utilisé quatre algorithmes RNN, LSTM, GRU, et Bi-LSTM, En comparant les algorithmes d'apprentissage profond entre eux, nous avons constaté que le modèle RNN 95.20% est faible par rapport au modèle LSTM 97.88%, par rapport au modèle GRU 97.89%, et par rapport au modèle Bi-LSTM 97.92% en termes de précision, mais que le temps d'entraînement RNN est le meilleur.

Les résultats des modèles de classifications de POS a utilisé pour corriger la sortie d'un modèle OCR, avec un autre modèle de n-gramme, tout d'abord nous avons crée un système de reconnaissance des caractères de tifinagh,puis nous avons détecté les erreurs existe dans la sortie d'OCR, après nous avons appliqué une technique de n-gramme et un modèle de machine profond pour corriger les erreurs en se basant sur les candidats trouvés.

Dans le domaine de traitement de langue naturel, la reconnaissance d'entités nommées est une sous-tâche de la recherche d'information. Elle consiste à rechercher des éléments textuels (par exemple, un mot ou un ensemble de mots) qui peuvent être classés en plusieurs catégories, comme des noms de personnes, des noms d'organisations ou de sociétés, des noms de lieux, des nombres, des distances, des valeurs, des dates, etc. Pour enrichir la langue amazighe par des outils de traitement automatique nous avons créé des modèles de NER basés sur les techniques d'apprentissage profond, tel que modèle RNN, modèle LSTM, modèle GRU, modèle Bi-LSTM, et modèle Bi-GRU, ces différentes techniques appliquées sur le même corpus créé pour POS en ajoutant des étiquettes de NE. Nous avons obtenu 94,49% de précision utilisant l'algorithme basé sur RNN, 97,95% pour l'algorithme basé sur LSTM, 97,96% pour le GRU,97,99% pour le modèle Bi-LSTM et 98,01% pour la couche Bi-GRU par rapport à l'ensemble de test.

L'un des sujets les plus fréquents dans le traitement de la langue est la traduction automatique, ce sujet a reçu beaucoup d'attention des chercheurs, c'est pour cela nous avons contribué à la traduction de la langue amazigh en appliquant des technique de traduction statique et neuronal, nous avons créé tous d'abord un corpus parallèle de la langue amazighe-anglais de 137322 phrases, puis nous avons construit quatre modèles ; un statistique et trois basées sur les réseaux de neurones. Après de longues simulations, des résultats distincts sont obtenus pour chaque modèle : 80,7% avec l'approche statique, 85,2% avec le modèle GRU, 87,9% avec le modèle LSTM et 91,3% avec la transformation.

Perspectives de recherche

Au terme de ce travail, plusieurs pistes de recherches, en relation avec le traitement de la langue naturel pour la langue Amazigh :

- Amélioration des techniques d'étiquetage POS et NER en enrichissant notre corpus par autre phrase annoté ;
- Utilisation de Big Data pour améliorer le système de correction de la sortie d'un OCR ;
- Amélioration de système de correction de la sortie d'un OCR pour corriger les erreurs au niveau sémantique ;
- Amélioration des modèles de traduction en ajoutant des phrases à notre corpus et ajoutant une troisième langue pour réaliser un système de traduction multilangues ;
- Fusionnement des deux système (OCR et NMT) pour réaliser un système de traduction des images.

Bibliographie

[1] A. Boukous, "Situation sociolinguistique de l Amazighe," *International Journal of the Sociology of Language*, vol. 123, Jan. 1997.

[2] M. Ennaji, "The sociology of Berber : change and continuity," *International Journal of the Sociology of Language*, vol. 123, Jan. 1997.

[3] L. Bianchini, "L'usage du français au Maghreb : Constellations francophones'," *Publifarum*, no. 7, 2007.

[4] C. A. Ferguson, "Diglossia," *WORD*, vol. 15, pp. 325–340, Jan. 1959.

[5] A. Youssi, *Grammaire et lexique de l'arabe marocain moderne*. Wallada, 1992.

[6] G. Stevanin, H. Azzedine, P. Denora, A. Boukhris, M. Tazir, A. Lossos, A. L. Rosa, I. Lerer, A. Hamri, and P. Alegria, "Mutations in SPG11 are frequent in autosomal recessive spastic paraplegia with thin corpus callosum, cognitive decline and lower motor neuron degeneration," *Brain*, vol. 131, no. 3, pp. 772–784, 2008. ISBN : 1460-2156 Publisher : Oxford University Press.

[7] Y. Es Saady, A. Rachidi, M. El Yassa, and D. Mammass, "AMHCD : A Database for Amazigh Handwritten Character Recognition Research," *IJCA*, vol. 27, pp. 44–48, Aug. 2011.

[8] L. NIHARMINE, B. OUTTAJ, and A. AZOUAOUI, "RECOGNITION OF HANDWRITTEN TIFINAGH CHARACTERS USING GRADIENT DIRECTION FEATURES.," *Journal of Theoretical & Applied Information Technology*, vol. 95, no. 13, 2017. ISBN : 1992-8645.

[9] E. W. Dadi, "Tifinagh-IRCAM Handwritten character recognition using Deep learning," *arXiv:1912.10338 [cs]*, Dec. 2019. arXiv : 1912.10338.

[10] M. Oujaoura, I. Rahil, W. Bouarifi, and A. Atlas, "Combination and fusion of some 2D invariant moments with generative and discriminative classifiers for recognition of isolated handwritten Tifinagh characters," *Annals of the University of Craiova - Mathematics and Computer Science Series*, vol. 46, pp. 457–473, Dec. 2019. Number : 2.

[11] B. Sabir, Y. Khazri, A. Jadir, B. Touri, and M. Moussetad, "Multiple Classifiers Combination Applied to OCR of Tifinagh Alphabets," *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 4, Nov. 2014.

[12] L. Niharmine, B. Outtaj, and A. Azouaoui, "RECOGNITION OF HANDWRITTEN TIFINAGH CHARACTERS USING GRADIENT DIRECTION FEATURES," 2017.

[13] M. Biniz and R. El Ayachi, "Recognition of Tifinagh Characters Using Optimized Convolutional Neural Network," *Sens Imaging*, vol. 22, p. 28, June 2021.

[14] S. Gounane, M. Fakir, and B. Bouikhalene, "Handwritten Tifinagh text recognition using fuzzy K-NN and Bi-gram language model," in *IJACSA Special Issue on Selected Papers from Third international symposium on Automatic Amazigh processing (SITACAM'13)*, pp. 29–32, Citeseer, 2013.

[15] M. Benaddy, O. El Meslouhi, Y. Es-saady, and M. Kardouchi, "Handwritten Tifinagh Characters Recognition Using Deep Convolutional Neural Networks," *Sens Imaging*, vol. 20, p. 9, Mar. 2019.

[16] O. Zealouk, H. Satori, M. Hamidi, and K. Satori, "Voice pathology assessment based on automatic speech recognition using Amazigh digits," in *Proceedings of the 2nd International Conference on Smart Digital Environment*, ICSDE'18, (New York, NY, USA), pp. 100–105, Association for Computing Machinery, Oct. 2018.

[17] M. Hamidi, H. Satori, O. Zealouk, and K. Satori, "Interactive Voice Application-Based Amazigh Speech Recognition," in *Embedded Systems and Artificial Intelligence* (V. Bhateja, S. C. Satapathy, and H. Satori, eds.), Advances in Intelligent Systems and Computing, (Singapore), pp. 271–279, Springer, 2020.

[18] H. Satori and F. ElHaoussi, "Investigation Amazigh speech recognition using CMU tools," *Int J Speech Technol*, vol. 17, pp. 235–243, Sept. 2014.

[19] S. El Ouahabi, M. Atounti, and M. Bellouki, "Toward an automatic speech recognition system for amazigh-tarifit language," *Int J Speech Technol*, vol. 22, pp. 421–432, June 2019.

[20] M. Outahajala, L. Zekouar, P. Rosso, and M. A. Martí, "Tagging amazigh with anconrapipe," in *Proceeding of the Workshop on Language Resources and Human Language Technology for Semitic Languages*, pp. 52–56, Citeseer, 2010.

[21] M. Outahajala, Y. Benajiba, P. Rosso, and L. Zenkouar, "POS Tagging in Amazighe Using Support Vector Machines and Conditional Random Fields," in *Natural Language Processing and Information Systems* (R. Muñoz, A. Montoyo, and E. Métais, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 238–241, Springer, 2011.

[22] S. Amri, L. Zenkouar, and M. Outahajala, "Amazigh Part-of-Speech Tagging Using Markov Models and Decision Trees," *IJCSIT*, vol. 8, pp. 61–71, Oct. 2016.

[23] M. Outahajala, Y. Benajiba, P. Rosso, and L. Zenkouar, "Using confidence and informativeness criteria to improve POS-tagging in amazigh," *Journal of Intelligent & Fuzzy Systems*, vol. 28, pp. 1319–1330, Jan. 2015. Publisher : IOS Press.

[24] A. Samir, Z. Lahbib, and O. Mohamed, "Amazigh PoS Tagging Using Machine Learning Techniques," in *Innovations in Smart Cities and Applications* (M. Ben Ahmed and A. A. Boudhir, eds.), Lecture Notes in Networks and Systems, (Cham), pp. 551–562, Springer International Publishing, 2018.

[25] M. Outahajala, L. Zenkouar, Y. Benajiba, and P. Rosso, "The development of a fine grained class set for Amazigh POS tagging," in *2013 ACS International Conference on Computer Systems and Applications (AICCSA)*, pp. 1–8, May 2013. ISSN : 2161-5330.

[26] S. Amri and L. Zenkouar, "Amazigh POS Tagging Using TreeTagger : A Language Independant Model," in *Advanced Intelligent Systems for Sustainable Development (AI2SD'2018)* (M. Ezziyyani, ed.), Advances in Intelligent Systems and Computing, (Cham), pp. 622–632, Springer International Publishing, 2019.

[27] M. F. Porter, "An algorithm for suffix stripping," *Program*, 1980. ISBN : 0033-0337 Publisher : MCB UP Ltd.

[28] L. Weissweiler and A. Fraser, "Developing a Stemmer for German Based on a Comparative Analysis of Publicly Available Stemmers," in *Language Technologies for the Challenges of the Digital Age* (G. Rehm and T. Declerck, eds.), Lecture Notes in Computer Science, (Cham), pp. 81–94, Springer International Publishing, 2018.

[29] A. Berko, Y. Matseliukh, Y. Ivaniv, L. Chyrun, and V. Schuchmann, "The Text Classification Based on Big Data Analysis for Keyword Definition Using Stemming," in *2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT)*, (LVIV, Ukraine), pp. 184–188, IEEE, Sept. 2021.

[30] S. Wehnert, V. Sudhi, S. Dureja, L. Kutty, S. Shahania, and E. W. De Luca, "Legal norm retrieval with variations of the bert model combined with TF-IDF vectorization," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, (São Paulo Brazil), pp. 285–294, ACM, June 2021.

[31] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *arXiv:1301.3781 [cs]*, Sept. 2013. arXiv : 1301.3781.

[32] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data," *Departmental Papers (CIS)*, June 2001.

[33] R. M. Hanifa, K. Isa, S. Mohamad, S. M. Shah, S. S. Nathan, R. Ramle, and M. Berahim, "Voiced and unvoiced separation in malay speech using zero crossing rate and energy," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 16, pp. 775–780, Nov. 2019. Number : 2.

[34] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.

[35] M. Zulqarnain, R. Ghazali, Y. M. Mohmad Hassim, and M. Rehan, "A comparative review on deep learning models for text classification," *IJEECS*, vol. 19, no. 1, 2020.

[36] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *arXiv:1412.3555 [cs]*, Dec. 2014. arXiv : 1412.3555.

[37] C. Xu, J. Shen, X. Du, and F. Zhang, "An Intrusion Detection System Using a Deep Neural Network With Gated Recurrent Units," *IEEE Access*, vol. 6, pp. 48697–48707, 2018.

[38] J.-Y. Antoine, "Pour une ingénierie des langues plus linguistique," *HDR informatique*, 2003.

[39] J. K. Baker, "Stochastic modeling for automatic speech understanding," in *Readings in speech recognition*, pp. 297–307, 1990.

[40] A. Ratnaparkhi, "A maximum entropy model for part-of-speech tagging," in *Conference on empirical methods in natural language processing*, 1996.

[41] K. W. Church, "A stochastic parts program and noun phrase parser for unrestricted text," in *International Conference on Acoustics, Speech, and Signal Processing,*, pp. 695–698, IEEE, 1989.

[42] P. F. Brown, J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. Lafferty, R. L. Mercer, and P. S. Roossin, "A statistical approach to machine translation," *Computational linguistics*, vol. 16, no. 2, pp. 79–85, 1990.

[43] E. Levin and R. Pieraccini, "Concept-based spontaneous speech understanding system," in *Fourth European Conference on Speech Communication and Technology*, 1995.

[44] M. Outahajala, P. Rosso, and L. Zenkouar, "Building an annotated corpus for Amazigh," in *Proceedings of 4th International Conference on Amazigh and ICT*, (Rabat Morocco), 2011.

[45] O. Maarouf and R. El Ayachi, "Part-of-Speech Tagging Using Long Short Term Memory (LSTM) : Amazigh Text Written in Tifinaghe Characters," in *Business Intelligence* (M. Fakir, M. Baslam, and R. El Ayachi, eds.), vol. 416, pp. 3–17, Cham : Springer International Publishing, 2021. Series Title : Lecture Notes in Business Information Processing.

[46] L. A. Ramshaw and M. P. Marcus, "Text Chunking using Transformation-Based Learning," *arXiv:cmp-lg/9505040*, May 1995. arXiv : cmp-lg/9505040.

[47] M. Boukabous and M. Azizi, "A comparative study of deep learning based language representation learning models," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, pp. 1032–1040, May 2021. Number : 2.

[48] D. Jurafsky and J. H. Martin, *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*, vol. 26 of *Prentice Hall series in artificial intelligence*. Upper Saddle River, N.J : Prentice Hall, 2000.

[49] T. Tan, B. Ranaivo-Malançon, L. Besacier, Y. Yeong, K. H. Gan, and E. K. Tang, "Evaluating LSTM Networks, HMM and WFST in Malay Part-of-Speech Tagging," 2017.

[50] H. Zen and H. Sak, "Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4470–4474, Apr. 2015. ISSN : 2379-190X.

[51] A. Berard, O. Pietquin, C. Servan, and L. Besacier, "Listen and Translate : A Proof of Concept for End-to-End Speech-to-Text Translation," *arXiv:1612.01744 [cs]*, Dec. 2016. arXiv : 1612.01744.

[52] D. Das, A. K. Kolya, A. Ekbal, and S. Bandyopadhyay, "Temporal Analysis of Sentiment Events – A Visual Realization and Tracking," in *Computational Linguistics and Intelligent Text Processing* (A. F. Gelbukh, ed.), vol. 6608, pp. 417–428, Berlin, Heidelberg : Springer Berlin Heidelberg, 2011.

[53] O. Maarouf, R. El Ayachi, and M. Biniz, "Amazigh part-of-speech tagging with machine learning and deep learning," *IJEECS*, vol. 24, p. 1814, Dec. 2021.

[54] V. Claveau and A. Ncibi, "Découverte de connaissances dans les séquences par CRF non-supervisés," in *20ème conférence sur le Traitement Automatique des Langues Naturelles, TALN*, vol. 1, (Sables d'Olonne, France), p. volume 1, June 2013.

[55] M. Silfverberg, T. Ruokolainen, K. Lindén, and M. Kurimo, "Part-of-Speech Tagging using Conditional Random Fields : Exploiting Sub-Label Dependencies for Improved Accuracy," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, (Baltimore, Maryland), pp. 259–264, Association for Computational Linguistics, 2014.

[56] M. Otman, E. A. Rachid, and B. Mohamed, "Amazigh Part Of Speech Tagging using Gated recurrent units (GRU)," in *2021 7th International Conference on Optimization and Applications (ICOA)*, (Wolfenbüttel, Germany), pp. 1–6, IEEE, May 2021.

[57] S. Manoharan, "A SMART IMAGE PROCESSING ALGORITHM FOR TEXT RECOGNITION, INFORMATION EXTRACTION AND VOCALIZATION FOR THE VISUALLY CHALLENGED," *JIIP*, vol. 1, pp. 31–38, Oct. 2019.

[58] M. Biniz and R. El Ayachi, "Recognition of Tifinagh Characters Using Optimized Convolutional Neural Network," *Sens Imaging*, vol. 22, p. 28, Dec. 2021.

[59] J. S. Shah and V. N. Gokani, "A Simple and Effective Optical Character Recognition System for Digits Recognition using the Pixel-Contour Features and Mathematical Parameters," 2014.

[60] K. Verspoor and K. B. Cohen, "Natural Language Processing," in *Encyclopedia of Systems Biology* (W. Dubitzky, O. Wolkenhauer, K.-H. Cho, and H. Yokota, eds.), pp. 1495–1498, New York, NY : Springer New York, 2013.

[61] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural Language Processing : State of The Art, Current Trends and Challenges," *arXiv:1708.05148 [cs]*, Aug. 2017. arXiv : 1708.05148.

[62] R. Beaufort and S. Roekhaut, "Le TAL au service de l'ALAO/ELAO L'exemple des exercices de dictée automatisés," *18 me Conférence sur le Traitement Automatique des Langues Naturelles 2011*, p. 6, 2011.

[63] S. Boulaknadel, *Traitement Automatique des Langues et Recherche d'Information en langue arabe dans un domaine de spécialité : Apport des connaissances morphologiques et syntaxiques pour l'indexation*. PhD thesis, Université de Nantes, Nantes., 2008.

[64] F. Agnaou, K. Ansar, F. Ataa Allah, A. Bouhjar, and S. Boulaknadel, "L'amazighe dans les sciences du numérique : expérience de l'IRCAM," *Actes de l'atelier «Diversité Linguistique et TAL»*, p. 11, 2017.

[65] k. younis and a. khateeb, "Arabic Hand-Written Character Recognition Based on Deep Convolutional Neural Networks," *JJCIT*, vol. 3, no. 3, p. 186, 2017.

[66] L. G. M. Pinto, F. Mora-Camino, P. L. de Brito, A. C. B. Ramos, and H. F. Castro Filho, "A SSD – OCR Approach for Real-Time Active Car Tracking on Quadrotors," in *16th International Conference on Information Technology-New Generations (ITNG 2019)* (S. Latifi, ed.), vol. 800, pp. 471–476, Cham : Springer International Publishing, 2019. Series Title : Advances in Intelligent Systems and Computing.

[67] H. Modi and M. C., "A Review on Optical Character Recognition Techniques," *IJCA*, vol. 160, pp. 20–24, Feb. 2017.

[68] A. F. Mollah, S. Basu, N. Das, R. Sarkar, M. Nasipuri, and M. Kundu, "Binarizing Business Card Images for Mobile Devices," *arXiv:1003.0645 [cs]*, Mar. 2010. arXiv : 1003.0645.

[69] J. Lázaro, J. L. Martín, J. Arias, A. Astarloa, and C. Cuadrado, "Neuro semantic thresholding using OCR software for high precision OCR applications," *Image and Vision Computing*, vol. 28, no. 4, pp. 571–578, 2010. ISBN : 0262-8856 Publisher : Elsevier.

[70] N. Islam, Z. Islam, and N. Noor, "A Survey on Optical Character Recognition System," *arXiv:1710.05703 [cs]*, Oct. 2017. arXiv : 1710.05703.

[71] W. B. Lund, D. J. Kennard, and E. K. Ringger, "Combining multiple thresholding binarization values to improve OCR output," in *Document Recognition and Retrieval XX*, vol. 8658, p. 86580R, International Society for Optics and Photonics, 2013.

[72] K. Yindumathi, S. S. Chaudhari, and R. Aparna, "Analysis of Image Classification for Text Extraction from Bills and Invoices," in *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, (Kharagpur, India), pp. 1–6, IEEE, July 2020.

[73] R. Deepa and K. N. Lalwani, "Image Classification and Text Extraction using Machine Learning," in *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, (Coimbatore, India), pp. 680–684, IEEE, June 2019.

[74] D. Saravanan and D. Joseph, "Image Data Extraction Using Image Similarities," in *Microelectronics, Electromagnetics and Telecommunications* (G. Panda, S. C. Satapathy, B. Biswal, and R. Bansal, eds.), Lecture Notes in Electrical Engineering, (Singapore), pp. 409–420, Springer, 2019.

[75] A. F. Mollah, N. Majumder, S. Basu, and M. Nasipuri, "Design of an Optical Character Recognition System for Camera- based Handheld Devices," vol. 8, no. 4, p. 7, 2011.

[76] Ming-Hsuan Yang and N. Ahuja, "Recognizing hand gesture using motion trajectories," in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, (Fort Collins, CO, USA), pp. 466–472, IEEE Comput. Soc, 1999.

[77] S. Sahoo, N. Dash, and P. Sahoo, "Word Extraction from Speech Recognition using Correlation Coefficients," *IJCA*, vol. 51, pp. 21–25, Aug. 2012.

[78] M. D. Kemighan, K. W. Church, and W. A. Gale, "A Spelling Correction Program Based on a Noisy Channel Model," in *COLING 1990 Volume 2 : Papers presented to the 13th International Conference on Computational Linguistics*, 1990.

[79] E. Mays, F. J. Damerau, and R. L. Mercer, "Context based spelling correction," *Information Processing & Management*, vol. 27, pp. 517–522, Jan. 1991.

[80] D. Jurafsky and J. H. Martin, *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall series in artificial intelligence, Upper Saddle River, N.J : Pearson Prentice Hall, 2nd ed ed., 2009. OCLC : 213375806.

[81] H. Lane, C. Howard, and H. M. Hapke, *Natural language processing in action : understanding, analyzing, and generating text with Python*. Shelter Island, NY : Manning Publications Co, 2019. OCLC : on1099603035.

[82] S. Amri and L. Zenkouar, "Neural Networks Architecture for Amazigh POS Tagging," in *Advanced Intelligent Systems for Sustainable Development (AI2SD'2018)* (M. Ezziyyani, ed.), Advances in Intelligent Systems and Computing, (Cham), pp. 955–965, Springer International Publishing, 2019.

[83] D. P. Kingma and J. Ba, "Adam : A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017. arXiv : 1412.6980.

[84] M. Singh, R. Kumar, and I. Chana, "Machine translation systems for Indian languages : review of modelling techniques, challenges, open issues and future research directions," *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 2165–2193, 2021. ISBN : 1886-1784 Publisher : Springer.

[85] S. Jindal, V. Goyal, and J. S. Bhullar, "English to Punjabi statistical machine translation using moses (Corpus Based)," *Journal of Statistics and Management Systems*, vol. 21, no. 4, pp. 553–560, 2018. ISBN : 0972-0510 Publisher : Taylor & Francis.

[86] A. Waibel, A. Jain, A. McNair, H. Saito, A. Hauptmann, and J. Tebelskis, "JANUS : a speech-to-speech translation system using connectionist and symbolic processing strategies," in *[Proceedings] ICASSP 91 : 1991 International Conference on Acoustics, Speech, and Signal Processing*, (Toronto, Ont., Canada), pp. 793–796 vol.2, IEEE, 1991.

[87] M. L. Forcada and R. P. Ñeco, "Recursive hetero-associative memories for translation," in *International Work-Conference on Artificial Neural Networks*, pp. 453–462, Springer, 1997.

[88] D. S. Rawat, "Survey on machine translation approaches used in India," *International Journal of Engineering and Technical Research (IJETR)*, vol. 3, no. 6, pp. 36–39, 2015.

[89] W. J. Hutchins, "Machine translation : A brief history," in *Concise history of the language sciences*, pp. 431–445, Elsevier, 1995.

[90] O. Craciunescu, C. Gerding-Salas, and S. Stringer-O'Keeffe, "Machine translation and computer-assisted translation," *Machine Translation and Computer-Assisted Translation*, 2004.

[91] P. Koehn, *Statistical machine translation*. Cambridge University Press, 2009.

[92] W. Weaver, "Translation," in *Proceedings of the Conference on Mechanical Translation*, 1952.

[93] W. A. Gale and K. W. Church, "A program for aligning sentences in bilingual corpora," *Computational linguistics*, vol. 19, no. 1, pp. 75–102, 1994.

[94] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba, "Addressing the Rare Word Problem in Neural Machine Translation," *arXiv:1410.8206 [cs]*, May 2015. arXiv : 1410.8206.

[95] 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.

[96] Y. Liu, J. Gu, N. Goyal, X. Li, S. Edunov, M. Ghazvininejad, M. Lewis, and L. Zettlemoyer, "Multilingual denoising pre-training for neural machine translation," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 726–742, 2020. Publisher : MIT Press.

[97] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mT5 : A massively multilingual pre-trained text-to-text transformer," *arXiv preprint arXiv:2010.11934*, 2020.

[98] R. Bawden, R. Sennrich, A. Birch, and B. Haddow, "Evaluating discourse phenomena in neural machine translation," *arXiv preprint arXiv:1711.00513*, 2017.