UNIVERSITÉ DU QUÉBEC À MONTRÉAL

VÉRIFICATION ET CORRECTION ORTHOGRAPHIQUE AUTOMATIQUE POUR LES LANGUES PARLÉES ET TRÈS PEU DOTÉES : ÉTUDE DE CAS SUR LE WOLOF

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

THIERNO IBRAHIMA CISSÉ

NOVEMBRE 2023

REMERCIEMENTS

En préambule à ce mémoire de fin d'études, je rends grâce à DIEU qui m'a permis de mener à bout ce travail.
Je souhaite ensuite adresser mes remerciements les plus chaleureux :

- À ma directrice de recherche, la Prof. Fatiha Sadat, pour le temps précieux qu'elle m'a accordé, ses conseils avisés et sa supervision attentive tout au long de la rédaction de ce mémoire ;
- À M. Billal Belainine, pour sa disponibilité, le partage de son expertise et ses précieux conseils ;
- À M. Ngoc Tan Le, pour ses recommandations et ses orientations judicieuses ;
- À M. Mathieu Mangeot, pour l'ensemble des ressources qu'il a généreusement partagées avec moi.

Je souhaite également exprimer ma gratitude à :

- L'ensemble des membres du jury, pour l'intérêt manifesté envers mon travail en acceptant d'examiner ce mémoire et d'y apporter leurs remarques et suggestions constructives ;
- L'intégralité du corps professoral et administratif de l'UQAM, pour la qualité de l'enseignement fourni et pour le soutien administratif tout au long de mon parcours académique.

En conclusion, mes remerciements s'adressent à toutes les personnes qui ont, de près ou de loin, contribué à la réalisation de ce travail.

Thierno Ibrahima CISSÉ

DÉDICACE

Je dédie ce mémoire à mes parents, en témoignage de mon profond amour pour eux.
À mon oncle **Mamadou Cissé** dit Modou Mbacké, en hommage à son influence inoubliable et sa mémoire impérissable.
À ma famille et à mes proches, pour leur soutien inconditionnel.
À ma chère épouse, avec qui j'ai partagé chaque moment d'émotion lors de la réalisation de ce document.
Puisse ce travail être l'accomplissement de vos voeux tant exprimés.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

ACRONYMES

ABR Arbre binaire de Recherche. viii, 34

ACW Accuracy Correct Word. viii, 86

Adam Adaptive Moment Estimation. viii, 17

AF Automate Fini. viii, 11

Ang Anglais. viii, xii

Ar Arabe. viii, 2

ASC Automatic Spelling Correction. viii, 1

BERT Bidirectional Encoder Representations from Transformers. viii, 76

BLEU BiLingual Evaluation Understudy. viii, 50

BP Brevity Penalty. viii, 51

BPE Byte-Pair-Encoding. viii, 81

CBOW Continuous Bag Of Words. viii, 18

CER Character Error Rate. viii, 52

ChrF Character n-gram F-score. viii, 53

ChrF++ Character unigrams and bigrams F-score. viii, 53

COA Correction Orthographique Automatique. viii, xii

Es Espagnol. viii, 2

FFN Feed Forward Network. viii, 14

Fr Français. viii, xii

GPT Generative Pre-trained Transformer. viii, 76

GPU Graphics Processing Unit. viii, 20

GRU Gated Recurrent Unit. viii, 19

IA Intelligence Artificielle. viii, 6

Loss Perte. viii, 85

LRL Low-Resource Language. viii, xii

LSTM Long-Short Term Memory. viii, 19

mBERT Multilingual Bidirectional Encoder Representations from Transformers. viii, 76

MRR Mean Reciprocal Rank. viii, 49

MT Machine Translation. viii, 1

MTurk Amazon Mechanical Turk. viii, 55

NLLB No Language Left Behind. viii, 59

NMT Neural Machine Translation. viii, 4

PPL Perplexité. viii, 85

RA Résumé Automatique. viii, 29

RAP Reconnaissance Automatique de la Parole. viii, 4

REN Reconnaissance des Entités Nommées. viii, 4

RI Recherche d'Informations. viii, 29

RLHF Reinforcement Learning from Human Feedback. viii, 98

RME Reconnaissance de l'écriture Manuscrite. viii, 52

RMSProp Root Mean Square Propagation. viii, 82

RNR Réseau de Neurones Récurrents. viii, 19

ROC Reconnaissance Optique de Caractères. viii, 41

seq2seq Séquence vers Séquence. viii, 20

SGD Stochastic Gradient Descent. viii, 17

T5 Text-to-Text Transfer Transformer. viii, 76

TA Traduction Automatique. viii, 1

TALN Traitement Automatique du Langage Naturel. viii, xii

TAN Traduction Automatique Neuronale. viii, 4

TAS Traduction Automatique Statistique. viii, 92

TPU Tensor Processing Unit. viii, 20

WER Word Error Rate. viii, 51

Wol Wolof. viii, xii

XLM-R XLM-RoBERTa. viii, 76

RÉSUMÉ

Au cours de la dernière décennie, le domaine du Traitement Automatique du Langage Naturel (TALN) a connu des avancées remarquables, principalement en raison des progrès technologiques. Une multitude d'outils linguistiques ont vu le jour, facilitant la recherche dans ce secteur. L'efficacité de ces outils découle de la disponibilité de vastes quantités de données, annotées ou non. Cependant, cette richesse concerne principalement les langues véhiculaires telles que l'Anglais (Ang) et le Français (Fr), et ne reflète pas la réalité de la majorité des langues du monde.

Les langues peu dotées, ou *Low-Resource Languages (LRLs)*, se distinguent par leur faible représentation dans le numérique. Elles souffrent d'un manque crucial de ressources, notamment les corpus textuels, essentiels au développement des systèmes de Correction Orthographique Automatique (COA). Ces contraintes posent des défis considérables au TALN pour les *LRLs*.

Cette recherche s'articule autour du développement de systèmes de vérification et de COA pour le Wolof (Wol), langue largement parlée en Afrique de l'Ouest mais considérée comme à très faibles ressources en linguistique computationnelle. Pour combler ce manque, nous avons développé deux systèmes de COA et créé deux corpus de test spécifiques au Wol.

Le premier système intègre la distance de Levenshtein pondérée, la programmation dynamique et la structure de données trie pour identifier et corriger les fautes orthographiques. Afin d'évaluer son efficacité, un corpus de mots Wol erronés, assortis de leurs corrections, a été conçu. La seconde approche utilise un modèle d'architecture encodeur-décodeur, entraîné sur un large corpus parallèle de phrases Wol, pour rectifier les erreurs orthographiques au niveau des phrases.

Les résultats issus de cette recherche attestent de la contribution notable de ces systèmes à l'amélioration des textes en Wol. Ils forment une solide base pour les futures recherches en COA pour le Wol et pour d'autres langues à ressources limitées. Grâce à ces efforts, nous aspirons à faire progresser la recherche en TALN pour le Wol et à contribuer à la préservation du patrimoine linguistique des nations africaines, assurant la pérennité de leurs expressions culturelles uniques pour les générations futures.

Mots clés : Wolof, Traitement Automatique du Langage Naturel, Correction Orthographique Automatique, langues à faibles ressources, distance de Levenshtein, programmation dynamique, structure de données trie, architecture encodeur-décodeur.

INTRODUCTION

0.1 Contexte et motivations

Avec l'avènement des échanges internationaux et de la technologie, le phénomène de mondialisation s'accélère de jour en jour. Dans notre ère numérique, l'écrit prend une place de plus en plus prépondérante. Ainsi, la COA est devenue une composante essentielle de la technologie moderne de communication, y compris les logiciels de traitement de texte, les moteurs de recherche en ligne et les applications de messagerie. Ces outils aident les utilisateurs à écrire correctement et à communiquer efficacement, sans avoir besoin de relecture manuelle.

Ces dernières années, plus particulièrement durant la dernière décennie, les avancées technologiques ont permis au domaine du TALN de connaître de remarquables progrès. Le TALN est une discipline qui s'intéresse à l'automatisation du traitement de certains aspects du langage humain. Bien que la Traduction Automatique (TA) également appelée *Machine Translation (MT)* soit actuellement le sous-domaine le plus en vogue du TALN, la COA ou encore *Automatic Spelling Correction (ASC)* n'en demeure pas moins importante.
La TA vise à automatiser la tâche de traduction d'une langue naturelle source vers une autre langue naturelle cible à l'aide de l'outil informatique (Henisz-Dostert *et al.*, 2011). La TA présente deux défis majeurs (Berment, 2004) : premièrement, comment programmer l'ordinateur pour qu'il comprenne un texte de la même manière qu'un être humain ; deuxièmement, comment créer de nouveaux textes dans des langues cibles de manière similaire à celle d'un être humain.
D'un autre côté, la COA se concentre sur l'utilisation de l'ordinateur pour détecter et corriger les erreurs orthographiques présentes dans un texte (Kukich, 1992). La difficulté de la COA réside dans la manière de programmer un outil informatique capable de reconnaître les mots mal orthographiés et de proposer une correction contextuellement pertinente.

Les langues peu dotées sont confrontées à d'importants défis, principalement en raison de la carence en données textuelles et audio. Cette absence d'informatisation engendre une pénurie criante de ressources linguistiques. Les corpus, indispensables pour le développement de systèmes de COA et de TA, sont quasiment inexistants pour ces langues. Face à ce contexte, le TALN

destiné aux LRLs est aujourd'hui au centre d'une problématique cruciale (Do, 2011).

En dépit du fait qu'environ 30% des langues parlées à travers le monde soient d'origine africaine (Eberhard *et al.*, 2019), leur accès à des données, tant textuelles qu'audio, demeure limité. Malgré leur forte présence et leur grande richesse morphosyntaxique, la plupart des langues africaines indigènes ne disposent que de ressources linguistiques très restreintes. La majorité des vérificateurs d'orthographe et des outils de correction ont été développés pour les langues à ressources élevées, telles que l'Ang, le Fr et l'Espagnol (Es), qui disposent de vastes corpus annotés et de technologies linguistiques avancées. En revanche, les langues à ressources limitées, manquent des ressources et de l'infrastructure nécessaires pour développer des outils de vérification et de correction efficaces. Le Wol, une langue particulièrement présente en Afrique de l'ouest, ne fait pas exception à cette règle. D'après notre revue de littérature en regard de cette dernière (voir le **Chapitre 1** pour plus de détails), il existe très peu de recherches ou publications concernant la COA de la langue Wol. C'est la raison pour laquelle nous avons choisi de nous concentrer sur cette thématique pour, dans un premier temps, contribuer au développement et à l'utilisation de la langue, et ensuite participer aux efforts de préservation de l'écriture authentique de la langue pour les générations futures.

0.2 Problématiques

Le Wolof est une langue transfrontalière principalement parlée dans trois pays d'Afrique occidentale : le Sénégal, la Gambie et la Mauritanie. Au Sénégal, où le Wolof est la langue véhiculaire dominante, plus de 80% de la population l'utilisait comme langue principale en 2017 (Diouf *et al.*, 2017).

Cependant, en dépit de sa prédominance en Afrique occidentale, le Wolof n'est la langue officielle dans aucun de ces pays. Cette situation est alarmante car, malgré sa large utilisation orale, le Wolof est peu représenté à l'écrit, ce qui le rend vulnérable à une possible extinction progressive. Au Sénégal, la langue officielle est le Français. Pour la Gambie et la Mauritanie, les langues officielles sont respectivement l'Anglais et l'Arabe (Ar). Cette configuration complique la collecte de ressources linguistiques pour le Wolof et diminue la probabilité de trouver des documents offi-

ciels dans cette langue. De plus, les rares ressources existantes et utilisables ne sont souvent pas accessibles aux chercheurs, comme le montre le corpus développé par Elhadji Mamadou Nguer *et al.* (2020).

La riche diversité linguistique du Sénégal, partagée par de nombreux autres pays africains, se manifeste par la coexistence de multiples langues nationales avec diverses langues étrangères. Cette cohabitation a entraîné un non-respect des conventions d'écriture du Wolof. La majorité des textes en Wolof disponibles sur le web ou dans les médias utilisent l'orthographe du Français, la langue officielle, plutôt que celle du Wolof. Le manque d'outils fiables de vérification et de correction orthographique est un frein majeur à la communication efficace et limite l'évolution du Wolof à l'ère numérique.

Récemment, grâce à l'intérêt croissant pour les LRLs, de nombreuses études ont été menées pour adapter les techniques de TALN aux langues africaines, comme le montrent les travaux de (Nekoto *et al.*, 2020) et (Reid *et al.*, 2021). D'autres, comme (Duh *et al.*, 2020) et (Abbott et Martinus, 2019), se concentrent davantage sur l'évaluation des performances de ces méthodes lorsqu'elles sont appliquées aux langues africaines. Dans (M'eric, 2014), une étude de cas est consacrée au bambara, langue minoritaire au Sénégal, dans le cadre de la COA. Toutefois, il est à noter que ces publications ne sont pas exclusivement consacrées au Wolof. À notre connaissance, l'unique recherche dédiée à la correction du Wol est celle de Lo *et al.* (2016), qui offre une revue complète de la situation et suggère des solutions pour élaborer un correcteur orthographique adapté au Wol.

0.3 Objectifs et contributions

Dans le cadre de notre travail de recherche, notre objectif consiste à apporter notre contribution au développement et à l'utilisation de la langue Wolof en fournissant de nouvelles ressources linguistiques exploitables par les communautés de chercheurs.
Nous avons concentré nos efforts sur la collecte de ressources linguistiques Wol et sur la COA des ressources collectées.
Ainsi, les contributions apportées par ce travail sont les suivantes :

— Un lexique Wolof exploitable non seulement pour la COA mais également pour diverses autres tâches de TALN tels que la Reconnaissance des Entités Nommées (REN), la Reconnaissance Automatique de la Parole (RAP), etc. ;

— Un outils de COA à base de règles pour aider à la tâche de normalisation de la langue Wol et participer à la préservation de la bonne syntaxe de la langue ;

— Un corpus annoté d'erreurs pour la langue Wol afin de servir à l'évaluation du système d'autocorrection developpé et de celle des futurs systèmes d'autocorrection se basant sur des techniques computationnelles ;

— Un modèle neuronal de COA pour le Wol qui tire profit des récentes techniques et avancées de la Traduction Automatique Neuronale (TAN), plus connu sous le nom de Neural Machine Translation (NMT) ;

— Un corpus parallèle annoté contenant des phrases avec des erreurs et leur version corrigée, pour aider à évaluer le modèle neuronal développé.

0.4 Organisation du mémoire

Le présent mémoire est structuré en quatre (04) chapitres :

— Le premier chapitre traite de la présentation de la langue Wol, en mettant l'accent sur les règles syntaxiques qui permettent de la comprendre, de lire et d'écrire dans cette langue. Ce chapitre évoque également quelques concepts relatifs à l'apprentissage automatique, qui sont essentiels à la compréhension des développements ultérieurs.

— Le deuxième chapitre est consacré à la revue de la littérature sur la thématique de recherche abordée dans ce mémoire. Il permet également de mettre en lumière les travaux antérieurs réalisés sur ce sujet.

— Dans le troisième chapitre, nous exposons notre méthodologie d'autocorrection fondée sur des techniques computationnelles traditionnelles, ainsi que l'évaluation du système proposé.

  Ce chapitre a abouti à la publication suivante :

Thierno Ibrahima Cissé et Fatiha Sadat. 2023. Automatic Spell Checker and Correction for Under-represented Spoken Languages : Case Study on Wolof. Dans *Proceedings of the Fourth workshop on Resources for African Indigenous Languages (RAIL 2023), pages 1–10, Dubrovnik, Croatia. Association for Computational Linguistics.*

— Le quatrième chapitre traite de notre méthodologie d'autocorrection basée sur les techniques d'apprentissage profond et présente une évaluation du modèle neuronal proposé. Ce chapitre a conduit à la rédaction d'un article qui a été soumis au *2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation.*
Actuellement, l'article est en cours d'évaluation.

Enfin, ce mémoire se conclut par une synthèse du travail accompli et une ébauche des perspectives futures pouvant être explorées dans ce domaine de recherche.

CHAPITRE 1
FONDEMENTS THÉORIQUES

1.1 Introduction

L'objectif principal de ce chapitre est d'établir les bases essentielles sur lesquelles reposera l'ensemble de cette recherche. Le but est d'offrir au lecteur une compréhension fondamentale de la langue Wolof, tout en lui permettant de se familiariser avec les concepts clés de l'apprentissage machine, en particulier ceux liés à l'apprentissage profond et au TAN.

La première section commence par aborder la signification socio-historique de la langue Wol, puis se penche sur les caractéristiques nécessaires pour comprendre, lire et écrire dans cette langue. Ensuite, quelques ressources et travaux dédiées à la langue Wol sont présentées, suivies d'une réflexion sur certaines particularités susceptibles de rendre son traitement automatique plus complexe pour les systèmes d'information.

La deuxième section du chapitre propose une introduction structurée aux principes clés de l'apprentissage automatique. Cette partie met l'accent sur des concepts essentiels tels que les réseaux de neurones, les plongements lexicaux et les modèles de type Transformer, qui jouent un rôle central dans de nombreux systèmes d'Intelligence Artificielle (IA) contemporains.

1.2 La langue Wolof (Wol)

Le Wol fait partie du groupe sénégambien, lui-même sous-ensemble de la branche nord de la grande famille linguistique atlantique, comme le stipule Wilson (1989). Cette famille atlantique est un élément de l'immense collection de langues nigéro-congolaises. Sapir (1971) a mis en évidence des liens linguistiques notables entre le Wol, le pulaar et le sérère.

La famille linguistique atlantique comprend environ une quarantaine de langues, dont le pulaar, qui est un dialecte du peul. Principalement, ces langues sont parlées dans des régions situées à proximité de la côte atlantique africaine, comme le précise Doneux (1978). Depuis la découverte

de cette famille linguistique, plusieurs classifications de ses langues ont été réalisées.

Sapir (1971) a initié une classification des langues de la famille atlantique en les répartissant en trois groupes : nord, sud et la langue bijago. Cette dernière est considérée comme divergente et provient des îles Bijagos situées au large de la Guinée-Bissau. Cette taxonomie est généralement considérée comme une classification traditionnelle. Cependant, à la lumière des recherches récentes, une nouvelle classification des langues atlantiques a été suggérée. Elle propose d'exclure toutes les langues appartenant au groupe sud, en les reclassant au sein de quatre branches distinctes de la famille nigéro-congolaise. Par ailleurs, les langues bak ont été dissociées des langues atlantiques du nord, et la langue bijago a été repositionnée parmi les langues bak. Une illustration de cette classification mise à jour, telle que présentée par Segerer et Pozdniakov (2017), est fournie à la Figure 1.1.

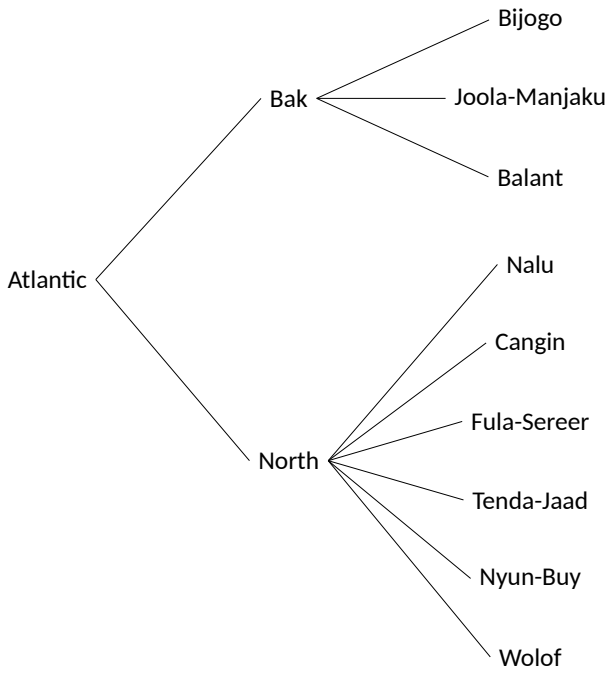

Figure 1.1 Classification révisée des langues atlantiques (Segerer et Pozdniakov, 2017)

1.2.1 Contexte socio-historique

Le Wol est majoritairement parlé au Sénégal, en Gambie et dans le sud de la Mauritanie. Des communautés Wol substantielles existent également au Mali, en Guinée, en Côte d'Ivoire, au Gabon,

en France et aux États-Unis (Eberhard *et al.*, 2019).

Reconnu comme langue dominante du Sénégal, le Wol est compris par plus de 90% de la population du pays et constitue le principal outil de communication des médias d'actualité (Eberhard *et al.*, 2019). Bien qu'elle ne soit la langue officielle d'aucun État, le Wol jouit d'un statut de langue nationale au Sénégal et en Mauritanie, statut consacré par les Constitutions de ces deux pays.

Historiquement, la société Wol se caractérise par une organisation hiérarchique basée sur le système de castes. Son économie était principalement agricole lors de sa formation. L'islamisation, débutée au onzième siècle dans certaines régions, s'est intensifiée durant la période coloniale, touchant aujourd'hui 94% de la population du Sénégal. Parallèlement, l'économie coloniale axée sur la culture de l'arachide a engendré un essor commercial significatif parmi les populations Wol. Ainsi, le Wol, outre son rôle de langue de communication, est devenu la langue des marchés, des écoles et des grandes villes.

1.2.2 Système d'écriture

Le Wol est une langue qui repose sur une tradition orale riche, mais il existe peu d'archives détaillées sur ses mécanismes linguistiques tels que son lexique et sa grammaire. En comparaison, le français dispose de ressources plus abondantes dans ce domaine.

Les études linguistiques descriptives du Wol remontent à l'époque coloniale (Boilat, 1858). Par la suite, plusieurs auteurs ont mené des recherches sur la morphologie et la syntaxe du Wol, tels que Diagne (1971), Mangold (1977), Church (1981), Dialo (1981), et Ka (1981). En ce qui concerne une analyse syntaxique plus approfondie du Wol, les travaux de Njie (1982) et Dunigan (1994) fournissent les principales études disponibles.

Ce qui distingue principalement le Wol, c'est son caractère aspectuel. Il accorde une importance particulière à l'aspect d'une action plutôt qu'à son temps. Cette particularité permet à son marqueur de l'imperfectif de se combiner avec différents marqueurs de temps. La langue possède un système verbal complexe qui comprend de nombreuses formes verbales de base et des pa-

radigmes. Des études systématiques sur ces paradigmes verbaux du Wol ont été réalisées par Mangold (1977) et Church (1981).

La graphie du Wol présente deux traditions d'écriture, qui méritent d'être soulignées. La première tradition, appelée wolofal, utilise des caractères arabes et remonte à une période plus ancienne. Quant à la deuxième tradition, elle est plus récente et se sert de caractères latins.

1.2.2.1 La graphie en caractères arabes

La transcription du Wol en caractères arabes semble remonter jusqu'au onzième siècle (Dunigan, 1994). Lors de l'islamisation, les populations Wol sont entrées en contact avec ces caractères et, à l'instar de nombreux autres peuples africains, elles ont essayé d'utiliser cette écriture pour noter leur langue. Il est important de souligner que le Wol, à la différence de l'arabe, est une langue où voyelles et consonnes ont un rôle équivalent. Les phonèmes non présents dans le système graphique arabe sont retranscrits par un signe proche ou en utilisant des signes diacritiques.

Malgré les défis posés par la systématisation du Wolofal, cette tradition scripturale présente un avantage indéniable : elle est aisément accessible à la majorité de la population, souvent alphabétisée grâce aux écoles coraniques.

1.2.2.2 La graphie en caractères latins

L'écriture du Wol en caractères latins fut réalisée par des explorateurs cherchant à recueillir des mots et des phrases de cette langue afin de faciliter les échanges entre les locuteurs wolofs et les navigateurs européens (Ka, 1981). Les missionnaires catholiques arrivèrent ultérieurement dans la région sénégambienne et initièrent les premières tentatives pour élaborer une graphie cohérente du Wol en caractères latins (Church, 1981). Ce n'est qu'en 1971 que l'écriture officielle du Wol en caractères latins fut normalisée (Robert, 2011). Sur le plan phonologique, cette orthographe ne consigne que les sons qui ont une fonction distinctive.

La langue Wol, telle qu'elle est employée actuellement compte 45 phonèmes consonantiques,

lesquels sont ensuite subdivisés en catégories distinctes (Cissé, 2004). Le tableau 1.1 illustre ces différents phonèmes consonantiques ainsi que leurs classifications respectives.

Phonèmes consonantiques		
Faibles	**Fortes**	
	Géminées	**Prénasalisées**
p, t, c, k, q, b, d, j, g, m, n, ñ, ŋ, f, r, s, x, w, l, y	pp, tt, cc, kk, bb, dd, jj, gg, ŋŋ, ww, ll, mm, nn, yy, ññ, qq	mp, nt, nc, nk, nq, mb, nd, nj, ng

Table 1.1 Classification des phonèmes consonantiques wolof (Cissé, 2004)

Par ailleurs, le Wol intègre un ensemble de 17 phonèmes vocaliques (Cissé, 2004), venant ainsi compléter les 45 phonèmes consonantiques déjà existants. Le tableau 1.2 offre un aperçu des phonèmes vocaliques ainsi que de leurs classifications respectives.

Phonèmes vocaliques	
Courtes	**Longues**
a, à, ã, i, o, ó, u, e, ë, é	ii, uu, éé, óó, ee, oo, aa

Table 1.2 Classification des phonèmes vocaliques wolof (Cissé, 2004)

1.2.3 Ressources existantes et applications technologiques

La langue Wol est diversifiée dans la littérature et d'autres ressources, se manifestant à travers des romans, des recueils de nouvelles et de la poésie. Toutefois, au Sénégal, l'accès à des documents

écrits en Wol demeure un défi. Récemment, plusieurs initiatives ont vu le jour pour améliorer la disponibilité des ressources informatiques pour les locuteurs du Wol.

D'abord, Enguehard et Mbodj (2004) a innové en proposant l'idée de développer un dispositif de COA pour les langues africaines. Par la suite, Diallo *et al.* (2012) a envisagé la création d'un dictionnaire électronique pour le Wol, une première dans ce secteur. Ce concept a été concrétisé par Nguer *et al.* (2015) qui a dirigé la mise en oeuvre du premier dictionnaire collaboratif en ligne pour le Wol. Ce travail est une composante du projet Dictionnaires Langues Africaines - Français (DiLAF) [1], qui a facilité l'établissement de dictionnaires pour sept langues africaines, dont le Wol. Cependant, à la date de rédaction de ce document, tous les dictionnaires de ce projet sont disponibles en ligne, sauf celui consacré au Wol. En parallèle, Dione (2012) a développé un outil d'analyse morphologique pour le Wol basé sur les Automate Fini (AF), bien qu'il ne soit pas largement accessible. De plus, Lo *et al.* (2016) a créé un corpus bilingue Wol-Fr pour la TA, bien que ce dernier ne soit pas ouvert au grand public. Gauthier *et al.* (2016) a compilé des fichiers audio pour quatre langues africaines, dont le Wol, permettant le développement du premier système de RAP pour cette langue. Plus récemment, Lo *et al.* (2020) et Cheikh M. Bamba Dione *et al.* (2022) ont travaillé sur le développement de systèmes de TA pour le Wol en utilisant des réseaux neuronaux, reflétant ainsi l'évolution constante de la technologie linguistique pour le Wol.

Le Wol, ayant un nombre limité de variations dialectales, est une langue remarquablement homogène (Robert, 2011). Néanmoins, le dialecte « lébou », dominant sur la presqu'île du Cap-Vert, se distingue.
Aujourd'hui, une dichotomie linguistique se profile entre le « Wolof urbain », enrichi d'emprunts au Français et simplifié dans sa grammaire, et le « Wolof rural », moins influencé par le Français.
Ces variations posent des défis pour le traitement automatique du Wol, accentués par le manque de ressources numériques et la complexité inhérente de la langue, rendant le développement d'outils linguistiques adaptés plus ardu.

1. `http://pagesperso.ls2n.fr/~enguehard-c/DiLAF/index.php`

1.3 L'apprentissage automatique

La période des années 90 a marqué un tournant pour le domaine du TALN, avec une adoption généralisée des techniques d'apprentissage automatique. L'intensification de l'utilisation de ces méthodes s'est accentuée avec l'émergence de l'apprentissage profond, un sous-domaine majeur de l'apprentissage automatique.

L'apprentissage automatique est souvent perçu comme un sous-ensemble ou un élément essentiel de l'IA, un terme dont la définition fait l'objet de nombreux débats. Toutefois, un consensus tend à se dégager autour de l'idée que l'IA englobe le concept de machines capables d'exécuter des tâches et de démontrer des comportements associés à l'intelligence (Xu *et al.*, 2021). Dans le cadre de l'apprentissage automatique, ce concept est adapté pour doter les machines de la capacité d'apprendre.

Marsland avance que l'apprentissage par expérience, qui caractérise l'intelligence humaine, est reproduit dans les machines en leur permettant d'apprendre à partir de données. Selon lui, l'objectif principal de l'apprentissage automatique est d'améliorer la précision des actions d'une machine, qu'il s'agisse de faire des prédictions ou de contrôler un robot. La précision est évaluée sur la base de la correspondance entre les actions effectuées et les actions attendues (Marsland, 2015).

Au cours de la dernière décennie, il est apparu que les algorithmes traditionnels d'apprentissage automatique éprouvent des difficultés à traiter le texte brut en langage naturel (LeCun *et al.*, 2015). Afin d'améliorer l'applicabilité des modèles d'apprentissage automatique, l'attention s'est progressivement portée sur les techniques d'apprentissage profond. Ces méthodes sont capables d'apprendre de façon autonome les représentations et les règles sémantiques à partir des données (LeCun *et al.*, 2015).

1.3.1 Les réseaux de neurones

L'objectif des réseaux de neurones artificiels est d'imiter la manière dont les êtres humains acquièrent de nouvelles connaissances. Leur fonctionnement est inspiré par le fonctionnement des

réseaux de neurones biologiques, d'où leur nom. Ces systèmes d'intelligence artificielle ont la capacité d'apprendre en analysant divers exemples sans qu'un programme spécifique ne soit écrit pour eux. Ils sont capables de tirer un sens des données non structurées, en découvrant des représentations cachées au sein de ces données. Ces représentations complexes sont généralement difficiles à identifier et à comprendre pour les humains.

L'idée des réseaux de neurones n'est pas nouvelle et remonte à 1943, lorsque McCulloch et Pitts ont développé un modèle de base pour imiter le neurone humain. Leur modèle était basé sur une fonction linéaire à seuil qui prenait un ensemble de valeurs d'entrée $x_1, x_2, ...x_n$ et les transformait en une sortie binaire y. Ainsi, le modèle de neurone était capable de prédire deux sorties différentes, selon que la valeur $y = x_1w_1 + x_2w_2 + ... + x_nw_n$ était positive ou négative (McCulloch et Pitts, 1943).

1.3.1.1 Le neurone

Le neurone représente l'unité fondamentale de calcul dans un réseau de neurones. Sa fonction principale consiste à traiter une représentation vectorielle qui intègre une série de variables désignées par $X = \{x_1, x_2, ...x_n\}$, pour ensuite générer une unique sortie, représentée par y.
L'algorithme Perceptron (Rosenblatt, 1958), a marqué l'histoire comme le premier modèle capable de déterminer les poids $\{w_1, w_2, ...w_n\}$ en se basant sur les données d'entrée de chaque catégorie. Cette réalisation représentait une percée significative dans le domaine à cette époque. Le Perceptron peut être perçu comme un classificateur binaire qui associe son entrée x à une unique valeur binaire y en suivant la formule 1.1.

$$y = \begin{cases} 1 & \text{si } w \cdot x + b > 0 \\ 0 & \text{sinon} \end{cases} \tag{1.1}$$

Où :

- $w \cdot x = \sum_{i=1}^{n} w_i x_i$ avec n = nbre variables entrée
- b : Biais

Une autre étape importante dans le développement des réseaux de neurones fut franchie en 1960

avec l'arrivée des neurones linéaires adaptatifs (Widrow et Lehr, 1990). Ces derniers ont contribué à améliorer le modèle du Perceptron en y intégrant une fonction non linéaire, ce qui a permis d'obtenir une prédiction en termes de nombres réels.
Le neurone linéaire adaptatif est conçu pour traiter un vecteur d'entrée X, de dimension n, en le mettant en corrélation avec un vecteur de poids W et un vecteur de biais B. La sortie générée par le neurone est calculée à partir de la formule 1.2 (LeCun *et al.*, 2015), qui incarne une caractéristique clé de son fonctionnement.

$$y = \sigma\left(\sum_{i=1}^{n} W_i X_i + B\right) \tag{1.2}$$

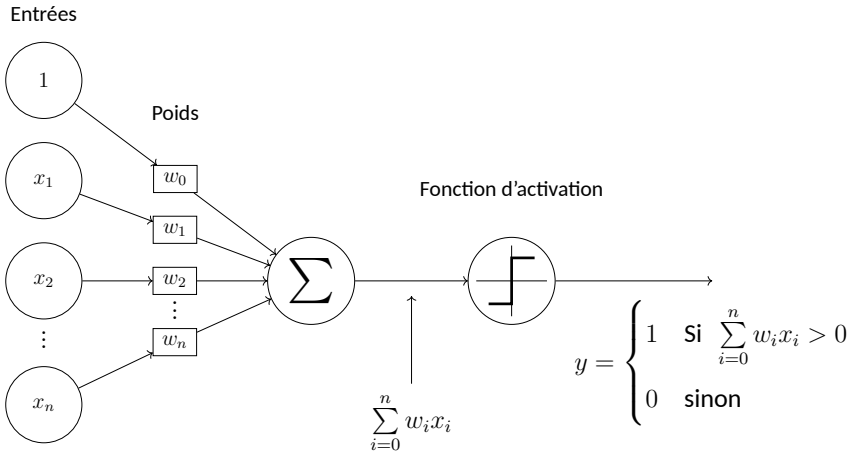

Figure 1.2 Représentation d'un neurone

1.3.1.2 Les réseaux de neurones à propagation avant

Le réseau de neurones à action directe, plus communément appelé Feed Forward Network (FFN), est un réseau multicouche dans lequel les sorties des neurones d'une couche sont acheminées vers les neurones de la couche subséquente. Ce réseau se structure autour de trois types de couches : une couche d'entrée, une ou plusieurs couches cachées et une couche de sortie. Chaque couche est totalement connectée aux couches précédente et suivante, signifiant ainsi que chaque neurone d'une couche reçoit en entrée l'ensemble des sorties de la couche précédente. De plus, on note l'absence de connexion entre les unités appartenant à une même couche.

Les unités de la couche d'entrée sont des valeurs scalaires, tandis que la couche cachée se compose d'unités neuronales qui calculent une somme pondérée de leurs entrées avant d'appliquer une fonction d'activation non linéaire. La sortie de la couche cachée, notée h, est représentée par l'équation 1.3 (LeCun *et al.*, 2015).

$$h = f(Wx + b) \tag{1.3}$$

Où :

- f : Une fonction d'activation
- W : La matrice de poids apprise
- x : Un vecteur d'entrée
- b : Un biais

La couche de sortie, quant à elle, génère une sortie finale basée sur le vecteur de représentation h. Cette sortie peut être soit un nombre réel, soit une distribution probabiliste à travers des mots de vocabulaire, en fonction de la tâche assignée au réseau. Tout comme la couche cachée, la couche de sortie possède une matrice de poids U et ne comporte généralement pas de vecteur de biais. Le réseau produit une sortie finale Z en multipliant la matrice de poids U par le vecteur caché h.

Lorsqu'on se trouve face à une situation qui nécessite une classification binaire, la fonction sigmoïde est couramment employée. Cette fonction, définie mathématiquement par l'équation 1.4, produit en sortie un vecteur à codage chaud, où l'élément le plus élevé a une valeur de 1 et les autres éléments ont une valeur de 0.

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{1.4}$$

Pour les problèmes de classification à plusieurs classes, une approche différente est recommandée afin de résoudre le problème de manière efficace. Dans de tels cas, il est courant d'utiliser la fonction $Softmax$ (Goodfellow *et al.*, 2016), qui est formellement exprimée par l'équation 1.5. Cette fonction est préférée pour les tâches de classification multi-classe en raison de sa capacité à normaliser les valeurs de sortie, les maintenant dans une plage de 0 à 1. Cela permet d'obtenir une distribution de probabilités pour chaque classe.

$$Softmax(Z_n) = \frac{\exp(Z_n)}{\sum_{j=1}^{N} \exp(Z_j)} \tag{1.5}$$

Où :

- $1 \leq n \leq N$
- N : Nombre d'unités de sortie

Un exemple de réseau de neurones à action directe est illustré dans la figure 1.3. Cette représentation graphique montre la structure du réseau et comment les différentes couches de neurones sont connectées entre elles.

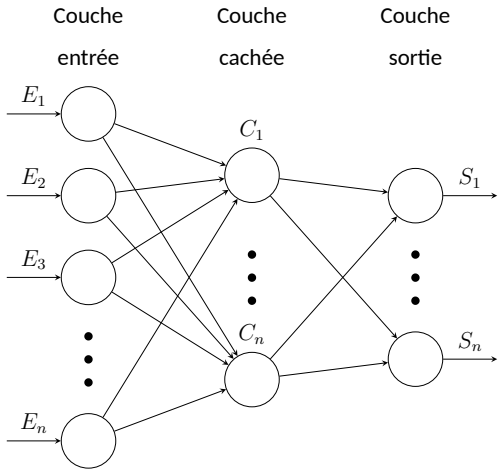

Figure 1.3 Architecture du FFN (LeCun *et al.*, 2015)

1.3.1.3 Entraînement d'un réseau de neurones

L'objectif principal de l'entraînement d'un réseau de neurones est d'identifier un ensemble optimal de poids qui réduira la distance entre la sortie prédite $\hat{y}$ générée par le modèle pour une entrée x donnée et la sortie y souhaitée pour la même entrée.

Initier l'entraînement d'un modèle nécessite d'abord la définition d'une fonction de perte, notée P. Le choix de cette fonction dépend intrinsèquement de la nature de la tâche à accomplir. Par exemple, dans le cas de tâches de classification, la fonction d'erreur couramment utilisée est la perte d'entropie croisée, qui est définie par l'équation 1.6.

$$P(f(x;\theta), y) = E(f(x;\theta), y) = -\sum_{i=0}^{n} y_i \log(f_i(x;\theta)) \tag{1.6}$$

Où :

— θ : Paramètres du modèle

— E : Entropie croisée

— $f_i(x;\theta)$: Prédiction pour le i^{eme} élément des n classes

— y_i : Valeur de référence

Suite à la sélection de la fonction d'erreur, l'actualisation des gradients du réseau devient une étape incontournable. Pour réaliser cette opération, l'usage courant est d'implémenter l'algorithme de rétro-propagation (Rumelhart *et al.*, 1986). Ce dernier consiste principalement à faire remonter l'erreur de la couche finale vers la couche initiale du réseau en ajustant les poids à l'aide du gradient. L'objectif est de réduire l'erreur et d'améliorer ainsi progressivement les prédictions du réseau au cours de l'entraînement.

Ce processus peut être décomposé en deux phases distinctes : durant la première, les gradients de l'erreur par rapport à l'ensemble des paramètres du réseau sont déterminés ; tandis que la seconde et dernière phase consiste à actualiser les poids en se basant sur une méthode d'optimisation choisie. Les méthodes d'optimisation varient depuis la plus élémentaire, le Stochastic Gradient Descent (SGD) (Lecun *et al.*, 1998), jusqu'à la plus sophistiquée, l'optimiseur Adaptive Moment Estimation (Adam) (Kingma et Ba, 2015).

1.3.2 Les plongements de mots

Les techniques de plongements de mots, ou plongements lexicaux, sont un ensemble de méthodes dans lesquelles chaque mot est représenté comme un vecteur à valeurs réelles dans un espace vectoriel prédéfini. Chaque mot est associé à un vecteur unique, et les valeurs de ces vecteurs sont déterminées à travers un processus d'apprentissage qui s'apparente à celui d'un réseau neuronal (Jiao et Zhang, 2021).

Les plongements lexicaux sont d'une importance cruciale pour révéler des relations sémantiques latentes entre les mots dans un corpus non annoté. Cette notion découle des travaux fondateurs de Bengio *et al.* (2003), les premiers à introduire la notion de représentation distribuée et continue. Cette représentation implique de représenter numériquement les mots sous forme de vecteurs multidimensionnels, rapprochant ainsi les mots sémantiquement similaires dans un espace vec-

toriel. En d'autres termes, elle encode les mots en utilisant des nombres, capturant ainsi leurs liens sémantiques de manière cohérente.

Cette représentation continue facilite la saisie de la similarité sémantique entre les mots. Les mots aux significations proches se retrouvent naturellement voisins les uns des autres dans cet espace vectoriel, simplifiant ainsi la détection des relations sémantiques entre les mots, même en l'absence d'annotations explicites.

L'idée sous-jacente aux plongements lexicaux repose sur le principe que des mots avec des sens similaires ont tendance à apparaître dans des contextes similaires dans le texte (Bengio *et al.*, 2003). Par conséquent, ils devraient avoir des représentations vectorielles similaires dans l'espace préétabli. Un réseau neuronal, initialement configuré de manière aléatoire, apprend à générer ces représentations en ajustant progressivement ses paramètres pour aligner les vecteurs des mots fréquemment utilisés ensemble (Jiao et Zhang, 2021).

Parmi les techniques existantes pour créer ces représentations vectorielles de mots, Word2Vec (Mikolov *et al.*, 2013) et GloVe (Pennington *et al.*, 2014) se distinguent. Word2Vec se base sur un réseau neuronal prédictif et apprend les vecteurs de mots en anticipant les mots environnants d'un mot cible. Il comprend deux variantes principales : le modèle Skip-Gram, qui prédit les mots contextuels à partir d'un mot central, et le modèle Continuous Bag Of Words (CBOW), qui prédit le mot central à partir des mots contextuels (Mikolov *et al.*, 2013).

Malgré sa capacité à déceler des relations sémantiques complexes, Word2Vec ne parvient pas à tirer parti de l'ensemble des statistiques de cooccurrence et de fréquence des mots (Pennington *et al.*, 2014). Pour surmonter cette limitation, GloVe a introduit un modèle des moindres carrés pondérés qui s'entraîne sur le nombre de cooccurrences entre deux mots, permettant ainsi d'exploiter plus efficacement les statistiques de fréquence du corpus (Pennington *et al.*, 2014).

1.3.3 Les réseaux de neurones récurrents

Les Réseaux de Neurones Récurrents (RNRs) constituent une catégorie spécifique de réseaux neuronaux qui suscitent un intérêt particulier dans le domaine du TALN en raison de leur caractéristique récurrente. Les RNRs sont capables de gérer des entrées de taille variable tout en parvenant à établir des liens entre les divers éléments de l'entrée. La différence majeure qui distingue les RNRs des FFNs mentionnés précédemment réside dans l'intégration de l'état caché précédent, en plus de l'entrée régulière, lors du calcul de chaque pas de temps.
Le calcul de la sortie des RNRs s'effectue de manière itérative, s'appuyant sur les équations 1.7 et 1.8.

$$h_t = f(W_x x_t + W_h h_{t-1} + b_h) \tag{1.7}$$

$$y_t = W_y h_t + b_y \tag{1.8}$$

Où :

- f : Une fonction d'activation
- x_t : Séquence d'entrée à l'instant t
- y_t : Séquence de sortie à l'instant t
- h : Séquence du vecteur caché du début jusqu'à t
- W : Poids des matrices
- b : Les biais

Les RNRs sont dotés d'un système de gestion de mémoire leur permettant de retenir le contexte des états précédemment calculés. Cependant, malgré leur capacité théorique à mémoriser le contexte de longues séquences, ils se heurtent en pratique à une limitation significative : ils ne peuvent conserver en mémoire que quelques étapes précédentes (Bengio *et al.*, 2003). Ce phénomène est dû à l'évolution combinatoire des valeurs du gradient pendant l'apprentissage, un problème connu sous le nom de disparition ou d'explosion du gradient (Goodfellow *et al.*, 2016).

Différentes architectures spécifiques ont été proposées pour résoudre ce problème, parmi lesquelles figurent le Long-Short Term Memory (LSTM) (Hochreiter et Schmidhuber, 1996) et le Gated Recurrent Unit (GRU) (Chung *et al.*, 2014). Ces architectures introduisent une méthode de calcul des états cachés innovante, qui utilise des mécanismes spécifiques appelés portes. Ces portes

jouent un rôle essentiel dans la régulation des informations provenant des états précédents, permettant de gérer à la fois l'oubli des informations non pertinentes et la mise à jour des entrées dans la cellule de mémoire à chaque itération.

1.3.4 Architecture Encodeur-Décodeur

L'usage des réseaux de neurones dans la TA a débuté il y a plusieurs décennies (Castaño *et al.*, 1997). Cependant, c'est l'accessibilité accrue des ressources informatiques, notamment les Graphics Processing Units (GPUs) et les Tensor Processing Units (TPUs), qui a facilité l'entraînement de modèles de grande taille et catalysé le véritable essor de cette pratique. L'approche prédominante dans le domaine de la TAN consiste à combiner les RNRs avec une architecture connue sous le nom de Séquence vers Séquence (seq2seq). Ce système, qui comprend un encodeur et un décodeur, a été largement adopté par les chercheurs (Kalchbrenner et Blunsom, 2013 ; Sutskever *et al.*, 2014).

Lors des premières phases du processus de TAN, l'encodeur reçoit les unités de base, également appelées "jetons" de la séquence source en entrée. Ces jetons, représentés généralement par des vecteurs à codage chaud, sont ensuite transformés en une représentation continue e, à l'aide de la matrice de plongement de mots, comme décrit dans l'équation 1.9.

$$e_i = W_e \cdot x_i \tag{1.9}$$

Où :

- W_e : Poids de la matrice de plongement
- x_i : Vecteur à codage chaud

L'encodeur traite ensuite ces représentations continues et les convertit en une représentation latente z. On peut concevoir cette représentation latente comme une condensation de l'information contenue dans la séquence d'entrée. Cette condensation est réalisée de telle sorte que la représentation latente conserve l'essentiel de l'information nécessaire pour la phase de décodage qui suit.

Compte tenu de la représentation latente z, le rôle du décodeur est de générer la phrase cible

y de manière séquentielle. Pour cela, il utilise une architecture de réseau neuronal qui opère de façon auto-régressive. Cela signifie que lors de la génération du jeton suivant, le modèle prend en compte la sortie de l'étape de temps actuelle. Toutefois, pendant l'entraînement, étant donné que nous disposons de la référence exacte de la séquence cible, le décodeur, à l'étape de temps t, utilise la sortie attendue de l'étape de temps t-1 plutôt que la sortie prédite par le modèle.

Le jeton prédit $\hat{y}$ est obtenu en deux étapes. Premièrement, la représentation cachée s_t dans le décodeur est transformée en un vecteur de la même taille que le vocabulaire en utilisant une certaine fonction f,comme le montre l'équation 1.10.

$$s_t = f(h_t) \tag{1.10}$$

Où :

— h_t : Représentation cachée du décodeur à l'étape de temps t.

Ensuite, le vecteur transformé s_t est converti en un vecteur de probabilités en appliquant la fonction softmax, comme indiqué dans l'équation 1.11.

$$P(\hat{y}|\hat{y}_{t-1}, x) = Softmax(s_t) \tag{1.11}$$

Où :

— $P(\hat{y}|\hat{y}_{t-1}, x)$: Probabilité conditionnelle de $\hat{y}$ sachant $\hat{y}_{t-1}$ et la séquence x

Finalement, le jeton prédit à l'étape de temps t est celui qui possède la probabilité la plus élevée dans la distribution de probabilité.

L'approche seq2seq présente certains inconvénients majeurs, tels que mentionné par Goodfellow *et al.*. Le premier inconvénient concerne la nécessité pour les états cachés de conserver des informations sur l'ensemble de la séquence source. Les états cachés de l'encodeur doivent être capables de représenter toutes les informations pertinentes de la phrase source, y compris les mots déjà traduits et ceux qui doivent encore l'être. Cette contrainte rend la tâche des états cachés difficile, car ils doivent représenter des informations complexes dans une taille prédéfinie.

Un deuxième inconvénient est lié à la taille des vecteurs cachés. Étant donné que les séquences peuvent varier en longueur, il peut arriver que la taille des vecteurs cachés soit insuffisante pour

encoder toutes les spécificités des phrases. En revanche, une taille trop grande peut entraîner un surajustement des données. Trouver la taille optimale des vecteurs cachés constitue souvent un défi nécessitant des ajustements empiriques.

Enfin, les modèles seq2seq peuvent avoir des difficultés à capturer les longues dépendances lorsque les phrases sont trop longues et que les mots à traduire sont éloignés les uns des autres. Par conséquent, des relations subtiles entre des mots éloignés peuvent être perdues lors de la traduction, ce qui peut entraîner une baisse de la qualité de traduction.

Pour surmonter ces problèmes, les chercheurs ont introduit un mécanisme d'alignement souple appelé attention (Bahdanau *et al.*, 2015 ; Luong *et al.*, 2015). Le mécanisme d'attention permet au décodeur de se concentrer partiellement sur les mots source à chaque étape de décodage. Au lieu de dépendre uniquement des états cachés de l'encodeur pour obtenir les informations nécessaires, le décodeur peut ajuster son attention en fonction de la pertinence des mots sources à chaque étape. Cela allège la charge des états cachés du côté du décodeur, tout en fournissant un meilleur contexte pour la génération de la phrase cible.

1.3.5 Le mécanisme d'attention

Le mécanisme d'attention, initialement introduit par Bahdanau *et al.* et ultérieurement affiné par Luong *et al.*, offre une méthode sophistiquée pour améliorer l'information disponible pour le décodeur à chaque étape en provenance de la source. Au coeur de ce processus se trouve le vecteur de contexte C_t, qui joue un rôle essentiel en tant qu'entrée pour les RNR.

Le vecteur de contexte est calculé en effectuant une somme pondérée des états cachés générés pendant la phase d'encodage, ce qui est formalisé dans l'équation 1.12.

$$C_t = \sum_{s=1}^{N} \alpha_{ts} \cdot \bar{h_s} \tag{1.12}$$

Où :

- N : Nombre de mots dans la séquence source
- $\bar{h_s}$: État caché à la position s dans la séquence source

α_{ts} représente le vecteur normalisé des scores d'attention. Ce dernier est une distribution de probabilités sur les mots sources. Ainsi, chaque valeur du vecteur peut être interprétée comme une mesure de l'importance du mot source correspondant pour la sortie à l'étape de décodage t. Chaque état caché de l'encodeur $\bar{h_s}$ est associé à une probabilité α_{ts} décrite par l'équation 1.13. Cette probabilité indique la pertinence de chaque état par rapport au contexte global, facilitant davantage le processus de décodage.

$$\alpha_{ts} = \frac{\exp(Z_{ts})}{\sum_{i=1}^{N} \exp(Z_{ti})} \tag{1.13}$$

Z_{ts} représente un modèle d'alignement capable de déterminer l'importance des mots sources qui sont proches de la position s pour l'étape de décodage t actuelle. Plusieurs définitions peuvent être attribuées à ce modèle d'alignement, ce qui entraîne des variations dans le calcul du score d'attention, comme illustré dans l'équation 1.14.

$$Z_{ts} = score(h_t, \bar{h_s}) = \begin{cases} v_a^T \tanh\left(W_1 h_t + W_2 \bar{h_s}\right) & \text{(Bahdanau } \textit{et al.}\text{, 2015)} \\ h_t^T W \bar{h_s} & \text{(Luong } \textit{et al.}\text{, 2015)} \end{cases} \tag{1.14}$$

1.3.6 Le modèle Transformer

Les RNRs sont réputés pour être complexes, principalement en raison de leurs cellules cachées. Cela entraîne des temps d'entraînement prolongés et nécessite un traitement séquentiel des données. Cependant, au cours des dernières années, avec les progrès des GPUs, il y a eu un intérêt croissant pour développer des modèles exploitant la parallélisation offerte par ces unités de traitement graphique.

Le Transformer, introduit en 2017 par des chercheurs de Google Brain, est un modèle d'apprentissage profond spécialement conçu pour traiter des séquences, telles que du texte en langage naturel, tout comme les RNRs. Néanmoins, il se distingue des RNRs en éliminant la nécessité de traiter les données de manière séquentielle (Vaswani *et al.*, 2017). Cette différence clé simplifie la parallélisation, ce qui permet au Transformer d'être entraîné plus rapidement que les RNRs.

La structure d'un Transformer peut varier. Il peut comporter une paire encodeur-décodeur (Figure 1.4), un seul bloc d'encodeur pour les modèles d'auto-encodage, ou un unique bloc de décodeur pour les modèles auto-régressifs.

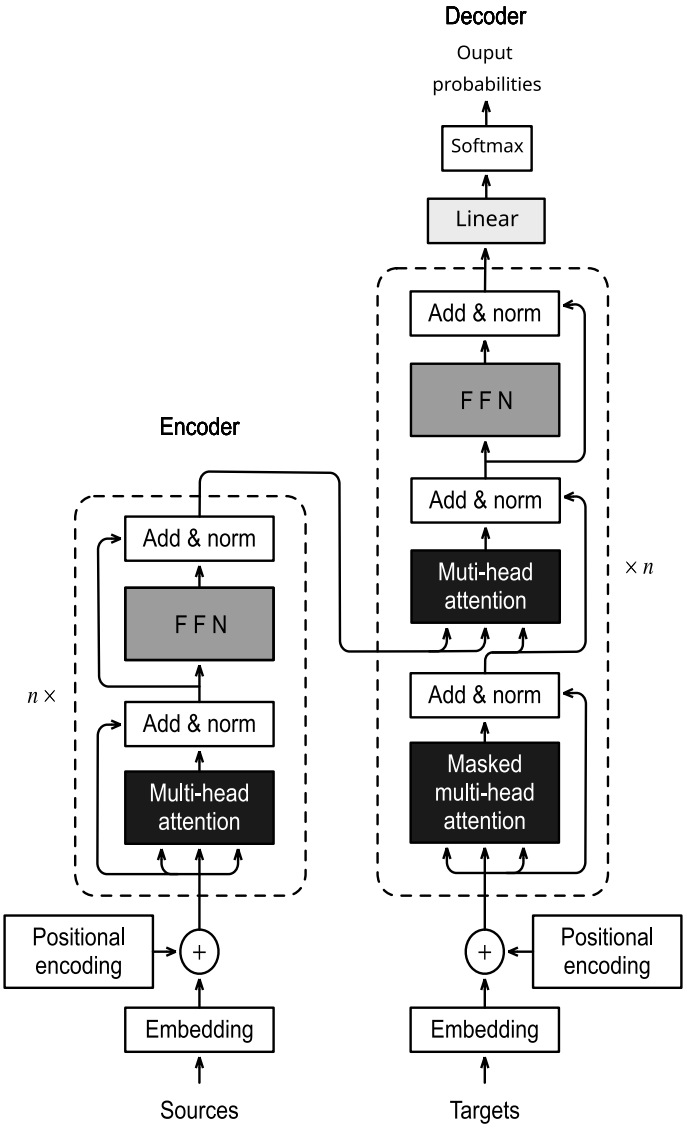

Figure 1.4 Architecture du modèle Transformer (Vaswani *et al.*, 2017)

1.3.6.1 Le mécanisme d'auto-attention

L'architecture du Transformer repose sur un mécanisme spécial d'attention appelé « auto-attention ». Contrairement aux RNRs qui utilisent une mémoire, l'auto-attention permet au Transformer de calculer les poids d'attention en utilisant un produit scalaire. Durant le traitement d'une phrase, les

poids d'attention entre les vecteurs de mots de cette phrase sont identifiés en parallèle. Cette approche permet de créer un contexte pour chaque mot dans la phrase, en rassemblant des informations provenant d'autres mots pertinents. De ce fait, le vecteur de contexte résultant comprend non seulement des données sur le mot proprement dit, mais aussi un ensemble d'informations issues d'autres mots significatifs. Ces données sont ajustées en se basant sur leurs poids d'attention respectifs.

Dans ce modèle, chaque composant d'auto-attention fait usage de trois matrices de poids distinctes : W_Q pour les requêtes, W_K pour les clés et W_V pour les valeurs. Quand un mot spécifique M_i est traité, son vecteur de plongement lexical x_i est multiplié par chacune de ces matrices de poids. Cela donne naissance à un vecteur de requête $q_i = x_i W_Q$, un vecteur clé $k_i = x_i W_K$, et un vecteur de valeur $v_i = x_i W_V$.

Les poids d'attention sont dérivés en utilisant le vecteur de requête et les vecteurs clés. Donc, pour chaque paire de mots M_i et M_j, le poids d'attention a_{ij} est calculé en effectuant le produit scalaire de q_i et k_j. Ces poids d'attention sont ensuite normalisés en les divisant par la racine carrée de la taille des vecteurs clés, soit $\sqrt{d_k}$. Cette étape de normalisation est cruciale car elle aide à stabiliser les gradients durant le processus d'apprentissage. Par la suite, les poids d'attention sont transmis à travers une fonction $Softmax$, qui les normalise de manière à ce que leur somme soit égale à 1.

La sortie de chaque composant d'attention pour un M_i est déterminée en sommant les vecteurs de valeur de tous les mots, en tenant compte de leurs poids d'attention a_{ij} respectifs. Cette approche permet de focaliser l'attention sur les mots qui sont les plus pertinents pour M_i, en fonction des poids d'attention calculés précédemment.
L'attention par produit scalaire normalisé peut être exprimée suivant l'équation 1.15 (Vaswani *et al.*, 2017).

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \times V \tag{1.15}$$

Dans le décodeur du modèle Transformer, en plus de l'auto-attention, on trouve une autre forme d'attention appelée « attention croisée » ou « attention entre les séquences ». Cette attention croi-

sée permet au décodeur d'incorporer les informations de la séquence d'entrée (représentées par les matrices K et V) lors de la génération de chaque mot de la séquence de sortie (représentée par la matrice Q). Cette mécanique aide le décodeur à saisir la relation entre les mots de la séquence d'entrée et ceux qu'il génère.

En sus des mécanismes d'attention, les couches d'encodeur et de décodeur du Transformer peuvent comporter d'autres modules supplémentaires. Par exemple, le FFN est utilisé comme une mémoire additionnelle dans chaque couche, afin de réaliser des transformations non linéaires sur les représentations des mots. Le module de normalisation par couche est responsable de la normalisation des activations des neurones à chaque couche, facilitant ainsi l'apprentissage. Les connexions résiduelles assurent la préservation du gradient lors de la rétro-propagation, en ajoutant les activations de l'entrée de la couche à la sortie de la couche avant l'application de la fonction d'activation. Ce module prévient la disparition du gradient, améliorant ainsi l'apprentissage dans les couches profondes.

1.3.6.2 Le mécanisme d'attention multi-têtes

Au lieu de calculer une seule somme pondérée par l'attention des valeurs, l'attention multi-têtes effectue plusieurs calculs de sommes pondérées afin de capturer différents aspects de l'entrée. En concaténant plusieurs attentions indépendantes, nous sommes en mesure d'obtenir des informations sur différents sous-espaces.

Dans le but d'apprendre des représentations diversifiées, chaque tête de l'attention multi-têtes est une transformation linéaire unique de la représentation d'entrée utilisée en tant que requête, clé et valeur. Ensuite, l'attention par produit scalaire normalisé est calculée simultanément h fois. Les sorties de chaque tête sont ensuite concaténées suivant l'équation 1.16 (Vaswani *et al.*, 2017).

$$\text{Multi-tête}(Q, K, V) = Concat(tete_1, ..., tete_n)W^O \tag{1.16}$$

Où :

- $tete_h = Attention(QW_h^Q, KW_h^K, VW_h^V)$
- W^O : Matrice de poids

L'attention à plusieurs têtes est capable d'apprendre des informations pertinentes à partir de différents sous-espaces de représentation car chaque ensemble de requête, clé et valeur est initialisé de manière aléatoire.

1.3.6.3 L'encodage positionnel

Contrairement aux RNR, qui analysent les mots de manière séquentielle, le Transformer adopte une approche différente. Il ne repose pas sur un mécanisme de récurrence intrinsèque pour capturer l'ordre des mots. Au lieu de cela, il utilise l'encodage positionnel pour fournir cette information d'ordre aux mécanismes d'attention du Transformer.

L'encodage positionnel dans le Transformer consiste à attribuer un vecteur unique à chaque position dans la séquence d'entrée. Pour ce faire, nous utilisons une formule mathématique pour définir les vecteurs de position. Soit pos la position d'un mot dans une phrase d'entrée et $\vec{p_{pos}}$ son vecteur d'encodage de position correspondant. La taille des dimensions de l'encodage d est équivalente à celle des vecteurs de mots, où i représente une position dans le vecteur des mots. La formule 1.17 est utilisée pour calculer le vecteur d'encodage positionnel (Shiv et Quirk, 2019).

$$\vec{p_{pos}}^{(i)} = \begin{cases} \sin(\omega_i \cdot pos) & \text{si } i = 2k \\ \cos(\omega_i \cdot pos) & \text{si } i = 2k+1 \end{cases} \tag{1.17}$$

$$\omega_i = \frac{1}{10000^{2k/d}} \tag{1.18}$$

En ajoutant les vecteurs de position, le modèle est en mesure de différencier les mots qui apparaissent à différentes positions dans la séquence. Cela signifie que le modèle peut prendre en compte l'ordre des mots lorsqu'il génère la représentation contextuelle pour chaque mot. Ainsi, les mécanismes d'attention du Transformer peuvent capturer des relations à longue portée entre les mots, sans avoir besoin de traiter les mots de manière séquentielle.

La capacité du Transformer à prendre efficacement en compte l'ordre des mots a contribué à son succès dans de nombreuses tâches de TALN. Par exemple, dans la TA, le modèle peut capturer les

dépendances entre les mots source et cible, même lorsque les mots correspondants ne sont pas alignés de manière séquentielle. De même, dans la génération de texte, le modèle peut produire des phrases cohérentes en prenant en compte l'ordre des mots dans la séquence générée.

1.4 Conclusion

En conclusion, ce chapitre a établi les fondements essentiels qui guideront l'ensemble de cette recherche. Nous avons exploré deux volets majeurs : la langue Wolof et les concepts clés de l'apprentissage machine, notamment ceux relatifs à l'apprentissage profond et au TAN.

En ce qui concerne la langue Wol, nous avons discuté de sa signification socio-historique, des spécificités de son utilisation et de sa lecture, et avons identifié les défis uniques qu'elle présente pour le traitement automatique. Les ressources existantes et les travaux consacrés à cette langue ont également été passés en revue.

Parallèlement, nous avons introduit les principes essentiels de l'apprentissage machine, en mettant l'accent sur les réseaux de neurones, les plongements lexicaux et les modèles de type seq2seq, qui sont au coeur de nombreux systèmes d'IA contemporains.

Ce travail préliminaire a ainsi posé les bases pour le chapitre suivant, où nous nous pencherons sur les divers types d'erreurs d'orthographe pouvant survenir dans la langue Wol, et examinerons en détail les techniques de détection et de correction orthographique décrites dans la littérature. En outre, nous présenterons différentes méthodologies d'évaluation utilisées pour mesurer les performances des systèmes de COA actuels.

CHAPITRE 2
ÉTAT DE L'ART

2.1 Introduction

Les erreurs d'orthographe sont un problème fréquent dans l'utilisation des langues, souvent causé par un manque de compétences linguistiques ou simplement par négligence. Ces erreurs peuvent avoir un impact significatif sur la communication écrite, tels que les courriels ou les discours, et peuvent également influencer l'efficacité des recherches sur Internet. Compte tenu de cette importance, les chercheurs du domaine du TALN ont fourni des efforts considérables pour développer des solutions efficaces afin d'atténuer l'impact des fautes d'orthographe.

L'une de ces solutions est l'utilisation d'outils de COA, qui sont devenus une caractéristique essentielle dans diverses applications du TALN. Par exemple, ces outils jouent un rôle clé dans la tâche de Résumé Automatique (RA) (El-Kassas *et al.*, 2021), garantissant que l'information condensée est exempte de fautes d'orthographe, améliorant ainsi la qualité et la clarté du contenu résumé. Dans la tâche de TA (Yang *et al.*, 2020), ils contribuent à maintenir l'intégrité du texte traduit en corrigeant toutes les fautes d'orthographe, assurant ainsi des traductions précises entre différentes langues. De plus, les moteurs de Recherche d'Informations (RI) (Cai et De Rijke, 2016) bénéficient grandement de ces outils. Ils peuvent détecter et rectifier les fautes de frappe dans les requêtes de recherche, améliorant ainsi la précision des résultats et offrant une expérience utilisateur plus satisfaisante. Par conséquent, l'importance de ces correcteurs dans le monde de la communication numérique est évidente, fournissant un outil inestimable pour améliorer l'efficacité et la fiabilité du traitement du langage écrit.

Dans le domaine de la COA, une variété de techniques et de modèles ont été utilisés, allant des systèmes basés sur des règles aux modèles d'apprentissage automatique. Dans la suite de ce chapitre, nous aborderons les différents types d'erreurs d'orthographe que l'on peut rencontrer dans la section 2.2. Ensuite, dans la section 2.3, nous examinerons en détail les techniques de détection et de correction orthographique décrites dans la littérature. La section 2.4 présentera les différentes

méthodologies d'évaluation utilisées pour mesurer les performances des systèmes de COA. Enfin, nous conclurons ce chapitre par une synthèse de notre revue de littérature et une introduction préliminaire des solutions que nous proposons.

2.2 Classification des erreurs

Un logiciel de COA est un outil sophistiqué conçu pour détecter et rectifier les fautes orthographiques contenues dans un document écrit. L'exactitude dans la détection de ces erreurs est un élément fondamental pour toute approche visant à améliorer la qualité de la communication écrite. Baba et Suzuki (2012) soutiennent qu'une connaissance approfondie des raisons sous-jacentes aux erreurs orthographiques est indispensable pour concevoir un système de COA efficace.

Il existe un grand nombre de recherches consacrées aux erreurs orthographiques, notamment le travail remarquable de Mitton (1996) qui a procédé à une analyse minutieuse des différentes catégories d'erreurs orthographiques en anglais et a présenté des techniques pour construire un système de COA. Selon les recherches de Yannakoudakis et Fawthrop (1983), une proportion considérable, approximativement 80%, des fautes d'orthographes se caractérisent par des erreurs d'un seul caractère, que cela implique l'insertion, la suppression, la substitution ou la transposition de lettres. Ces erreurs sont généralement confinées à un écart d'une seule lettre par rapport à la longueur du mot correct. Par ailleurs, il convient de mentionner que les erreurs se produisent exceptionnellement au niveau du premier caractère d'un mot. Les auteurs ont démontré que cette tendance est prédominante dans la majorité des cas. Kukich (1992) ; Toutanova et Moore (2002) ; Pirinen et Lindén (2014) ont également fourni une synthèse des recherches faites sur les modèles d'erreurs orthographiques, classant les fautes orthographiques en deux groupes distincts :

— Les erreurs lexicales : elles proviennent des fautes commises sur des mots singuliers, sans tenir compte de leur contexte dans une phrase (Ten Hacken et Tschichold, 2001).

— Les erreurs grammaticales : elles comprennent les erreurs morphologiques et syntaxiques. Les erreurs morphologiques font référence aux insuffisances dans des constituants linguistiques tels que la dérivation, l'inflexion, les prépositions, les articles, les pronoms personnels, les verbes auxiliaires et les déterminants. Les erreurs syntaxiques proviennent de pro-

blèmes dans des composants linguistiques tels que la voix passive, le temps, les phrases nominales, les verbes auxiliaires, la concordance entre le sujet, le verbe et les déterminants (Gayo et Widodo, 2018).

D'après Peterson (1980), les sources d'erreurs orthographiques sont diverses et peuvent être d'origine cognitive ou typographique. Les erreurs cognitives se produisent lorsque l'individu ne maîtrise pas correctement l'orthographe d'un mot, tandis que les erreurs typographiques se produisent lorsqu'une mauvaise touche est pressée lors de la frappe. Les travaux sur la COA ont généralement traité ces types d'erreurs de manière indépendante, avec l'élaboration de méthodes spécifiques pour gérer chaque type (Kukich, 1992).

Dans le cadre de notre recherche, les erreurs identifiées dans les textes Wol sont réparties en trois principales catégories en fonction du niveau d'analyse auquel elles apparaissent : les erreurs au niveau phonétique, les erreurs au niveau syntaxique, et les erreurs au niveau sémantique.

2.2.1 Erreurs de niveau phonétique

Les erreurs de niveau phonétique sont couramment rencontrées lors du traitement de données audio, que ce soit lors de la conversion de la parole en texte (RAP) ou de la transcription de l'audio. Elles peuvent être dues à des distorsions du signal audio, à des accents ou des dialectes régionaux, ou à des variations dans la prononciation d'un même phonème.

2.2.2 Erreurs de niveau syntaxique

Les erreurs de niveau syntaxique se produisent lorsque la structure de la phrase ou de l'expression est incorrecte selon les règles de grammaire de la langue Wolof. Ces erreurs peuvent résulter d'un mauvais choix de mots, d'une mauvaise concordance de temps, ou de l'utilisation incorrecte des structures grammaticales.

2.2.3 Erreurs de niveau sémantique

Les erreurs de niveau sémantique sont liées au sens et à l'interprétation des mots et des expressions dans un contexte donné. Cela peut être dû à une mauvaise interprétation des nuances de sens, à l'utilisation incorrecte des termes spécifiques à un domaine, ou à la confusion entre les mots qui se ressemblent phonétiquement mais qui ont des significations différentes.

2.3 Techniques de vérification et de correction orthographique automatique

Historiquement, les méthodes de vérification et de correction orthographiques ont largement reposé sur l'exploration de dictionnaires pour identifier et corriger les erreurs. Au fil des années, l'emploi des approches probabilistes a pris de l'ampleur, apportant une nouvelle dimension à la correction orthographique. Le processus de vérification et de correction orthographiques est généralement biphasé : il débute par l'identification des erreurs, puis passe à la phase de correction. Cependant, certains systèmes intègrent ces deux phases en une seule étape.

Les techniques mises en oeuvre s'appuient principalement sur une comparaison entre les données entrantes, à savoir le texte contenant des erreurs, et une base de connaissances existante, telle qu'un dictionnaire, un lexique ou un corpus. La différenciation entre les diverses approches réside dans la structure des données qu'elles utilisent pour organiser ces connaissances. De plus, la manière dont l'information entrante est représentée peut également varier selon la technique employée.

2.3.1 Techniques de vérification orthographique

La vérification orthographique consiste à vérifier l'exactitude d'un mot saisi en le comparant avec un dictionnaire. Si le mot correspond parfaitement à une entrée du dictionnaire, on considère que ces deux mots sont interchangeables sans aucun changement de sens, comme le soulignent Hall et Dowling (1980). Cette correspondance est déterminée par l'utilisation de méthodes de comparaison de chaînes de caractères.

Selon Kukich (1992), deux principales méthodes sont utilisées pour détecter les fautes d'ortho-

graphe dans un texte : la vérification par dictionnaire et l'analyse des n-grammes. Un mot est considéré comme invalide s'il ne figure pas dans un dictionnaire donné ou s'il ne correspond pas à une forme orthographique reconnue.

La vérification par dictionnaire implique l'utilisation d'algorithmes de fouille efficaces et/ou d'algorithmes de correspondance de patrons. Elle peut également utiliser diverses méthodes pour diviser le dictionnaire et différentes techniques de traitement morphologique pour examiner à la fois le mot entré et le dictionnaire stocké en mémoire.

L'analyse de n-grammes, quant à elle, se base sur les fréquences ou les probabilités d'apparition des n-grammes dans un texte et un dictionnaire, un lexique ou, dans la plupart des cas, un corpus. Un n-gramme est une séquence de n éléments consécutifs dans un texte, où ces éléments peuvent être des lettres, des mots ou d'autres unités textuelles, en fonction du contexte. Par exemple, un 2-gramme (ou bigramme) serait une séquence de deux mots consécutifs.

2.3.1.1 Vérification dans un dictionnaire

Les techniques de fouille de dictionnaires sont couramment utilisées pour comparer et identifier des chaînes de caractères dans différentes ressources telles que les lexiques, les corpus ou une combinaison de ces derniers. L'objectif principal de ces méthodes standard est de réduire le temps nécessaire pour effectuer des recherches dans le dictionnaire. Dans le domaine de la détection des erreurs d'orthographe, des techniques de correspondance exacte de chaînes de caractères sont utilisées. Si une chaîne n'est pas présente dans le lexique ou le corpus sélectionné, elle est considérée comme mal orthographiée ou invalide.

L'étude réalisée par Kukich (1992) se concentre sur les techniques de fouille de dictionnaires qui visent à réduire le temps de recherche en utilisant des algorithmes ou structures de données spécifiquement conçus pour cette tâche. Ces méthodes comprennent entre autres, des techniques de correspondance de patrons, des stratégies de segmentations de dictionnaires et des approches de traitement morphologique. Parmi les techniques les plus pertinentes pour la recherche dans des dictionnaires, on note l'usage du hachage, des arbres binaires de recherche, des arbres trie et des

automates finis.

2.3.1.1.1 Le hachage

Le hachage, une méthode de fouille efficace et largement reconnue, fait l'objet de discussions approfondies dans des travaux tels que Bloom (1970) ; Knuth (1998) ; Thareja (2014). Le principe central du hachage repose sur un calcul habile effectué sur une séquence d'entrée pour identifier l'emplacement d'une éventuelle correspondance. En d'autres termes, le hachage permet d'explorer une chaîne d'entrée dans une table de hachage préétablie, en utilisant une clé ou une adresse de hachage liée à un mot, puis de retrouver le mot à cette adresse spécifique. Si des collisions se sont produites lors de la création de la table, il peut être nécessaire de traverser un ou plusieurs liens, ou d'effectuer une recherche linéaire. Par conséquent, si une fonction de hachage tend à générer des adresses de hachage similaires ou identiques, le processus de recherche pourrait en être ralenti.

Dans le cadre de la COA, une correspondance est trouvée si le mot stocké à l'adresse de hachage est le même que la séquence d'entrée. Toutefois, si le mot d'entrée ne correspond pas au mot récupéré, ou si l'adresse de hachage est vide, le mot d'entrée est considéré comme mal orthographié. L'accès aléatoire qu'offrent les tables de hachage élimine la nécessité d'effectuer un grand nombre de comparaisons pour effectuer des recherches. Par conséquent, c'est une technique de recherche plus rapide que la recherche séquentielle ou la recherche basée sur les arbres, en particulier lorsqu'il s'agit de parcourir une grande quantité de données. Cependant, l'inconvénient d'une table de hachage est qu'une fonction de hachage optimale est nécessaire, sinon une table de hachage de grande taille serait requise pour prévenir les collisions. En moyenne, les tables de hachage offrent un temps de recherche constant $O(1)$. Dans le pire des cas, le hachage a une complexité temporelle de $O(n)$, où n est le nombre de mots dans un dictionnaire ou un corpus.

2.3.1.1.2 Arbre binaire de Recherche (ABR)

Un Arbre binaire de Recherche (ABR) (Windley, 1960 ; Frost et Peterson, 1982 ; Culberson, 1989) est une structure de données spécifique, de type arborescent, qui a pour particularité de conserver

une hiérarchie bien définie entre ses éléments, que l'on appelle des noeuds. Dans cette hiérarchie, chaque noeud possède une clé, ou une valeur unique, qui permet de le distinguer des autres. Les noeuds dont la clé est plus grande qu'une clé donnée se trouvent dans la branche droite de l'arbre, tandis que ceux dont la clé est égale ou inférieure se trouvent dans la branche gauche. Cette disposition spécifique est ce qui permet une recherche binaire efficace. En effet, chaque comparaison de clés élimine environ la moitié de l'arbre restant à examiner, ce qui signifie que la performance de la recherche suit le logarithme en base deux du nombre total de noeuds.

Cependant, la performance d'un ABR peut être fortement impactée par l'ordre dans lequel les noeuds sont insérés. Une séquence d'insertions arbitraire peut en effet conduire à un arbre déséquilibré et donc moins efficace, augmentant considérablement le temps nécessaire pour des opérations comme la recherche, l'insertion, et la suppression de noeuds. Dans le pire des cas, la complexité temporelle de ces opérations peut passer de $O(\log(n))$ à $O(n)$, où 'n' est le nombre total de noeuds.
Pour pallier à ce problème, des types de ABR spécifiques appelés arbres binaires de recherche équilibrés (Sleator et Tarjan, 1985), comme les arbres AVL (Adel'son-Vel'skii et Landis, 1962) ou les arbres-B (Bayer et McCreight, 1970), ont été développés. Ces arbres ont la particularité de s'ajuster automatiquement pour garder une hauteur en rapport logarithmique avec le nombre total de noeuds, indépendamment de l'ordre d'insertion et de suppression des noeuds. Cette propriété assure une complexité de recherche qui suit le logarithme en base deux, même dans le pire des cas, ce qui optimise l'efficacité globale de l'arbre.

Les qualités des ABR en font un choix populaire pour des tâches comme la fouille de dictionnaires ou la vérification orthographique (Gowri *et al.*, 2022). Dans ces applications, un ABR est utilisé pour stocker des termes de vocabulaire, chaque noeud représentant un test binaire qui dirige la recherche vers l'un des deux sous-arbres dépendant de ce noeud. Ainsi, une recherche efficace peut être réalisée avec un nombre de comparaisons qui est $O(\log(m))$, où m est le nombre total de termes dans le dictionnaire.

2.3.1.1.3 Les arbres trie

Les arbres trie (Fredkin, 1960 ; Connelly et Morris, 1995 ; Amir *et al.*, 2008), aussi désignés par arbres de préfixes, sont une forme spécialisée de structure de données arborescente, conçue spécifiquement pour le traitement et la recherche de séquences de caractères. Alors que les arbres de recherche binaires sont utilisés pour stocker des éléments comparables tels que les nombres, les arbres trie se focalisent sur l'enregistrement de séquences de valeurs.

Chaque noeud de l'arbre représente un caractère spécifique d'une séquence, avec le noeud racine représentant généralement une séquence ou un caractère vide. Ces noeuds peuvent se ramifier en plusieurs directions, chaque branche représentant un caractère potentiel des clés, et la fin d'une clé est signalée par un noeud feuille. Dans un arbre de préfixes, chaque niveau de récursion correspond à la valeur du i^{eme} élément de la liste entrante, contrairement à un arbre binaire qui compare la valeur recherchée à chaque noeud. Cette différence permet à un arbre de préfixes d'intégrer une séquence dans sa structure et de stocker uniquement une seule fois les préfixes communs des séquences, ce qui permet d'économiser de la mémoire.

Dans les arbres trie, la durée d'une opération de recherche est directement proportionnelle à la longueur de la séquence recherchée, indépendamment du volume de données stockées. De plus, l'exploration erronée des séquences est limitée à quelques caractères seulement, spécifiquement ceux constituant le préfixe commun le plus long entre la séquence recherchée et le contenu existant de l'arbre. La rapidité de ces arbres trie est également observable lors des opérations d'insertion et de suppression. En raison de la taille constante de chaque comparaison, la complexité temporelle de ces opérations est de l'ordre de $O(1)$, faisant preuve d'une efficacité remarquable.

Comparativement, un arbre trie peut localiser un préfixe d'une séquence en $O(m)$ étapes, où m est la longueur de la séquence, tandis qu'un arbre de recherche binaire nécessiterait $O(m \times \log(n))$ étapes, n étant le nombre de noeuds. Cette différence de performance devient plus notable à mesure que la taille de l'ensemble de données s'accroît.

2.3.1.1.4 Automate Fini (AF)

Un AF (Arbib, 1969 ; Hopcroft et Ullman, 1979 ; Linz, 2006) est un outil de calcul numérique qui fonctionne avec un ensemble limité d'états. Il utilise un ensemble spécifique de symboles d'entrée, ainsi qu'une fonction de transition qui établit un lien entre les combinaisons d'états et d'entrées pour conduire à de nouveaux états. Un AF dispose d'un état de départ, ainsi que d'un ou plusieurs états finaux ou acceptants.

Un AF démarre à son état initial et commence à lire une séquence de symboles provenant de son ensemble de symboles. À chaque lecture d'un symbole, elle se dirige vers un nouvel état, défini par la fonction de transition. Si après avoir lu la totalité de la séquence de symboles, l'AF se termine dans un état acceptant, elle valide cette séquence. Dans le cas contraire, elle la refuse.

Les AF peuvent être soit déterministes, soit non-déterministes. Dans le cas des AF non-déterministes, plusieurs transitions sont possibles pour une même combinaison d'état et de symbole. Une propriété majeure des AF est leur manque de mémoire : l'état suivant est uniquement déterminé par l'état actuel et le symbole entrant, sans tenir compte des états ou symboles précédents. Cette caractéristique simplifie leur utilisation et améliore leur efficacité pour certains types de tâches, mais elle limite également leur aptitude à gérer des tâches nécessitant le souvenir des entrées précédentes.

Les AF sont particulièrement utiles pour la recherche dans un dictionnaire et la vérification orthographique. Ils peuvent être utilisés pour représenter un dictionnaire où chaque mot correspond à une séquence validée par la machine. Dans le contexte de la vérification orthographique, une AF peut déterminer si un mot est bien orthographié (s'il est présent dans le dictionnaire) ou non.

La construction d'une AF commence par la création d'une table de transition, qui est remplie en fonction des séquences que la AF doit reconnaître. Le processus de construction a une complexité de $O(m \times k)$, où m est la longueur de la séquence à reconnaître et k est le nombre de symboles différents. La recherche de séquence principale fonctionne avec une complexité temporelle de $O(n)$.

2.3.1.2 Analyse de n-grammes

L'analyse de n-grammes (Peterson, 1980 ; Zamora *et al.*, 1981) est une méthode permettant de vérifier l'exactitude des n-grammes présents dans un texte en les comparant à des statistiques préétablies sur les n-grammes tirées de grands corpus de textes. Cette analyse s'articule autour de deux stratégies principales : la méthode binaire et la méthode probabiliste.

La méthode binaire consiste à générer une matrice à deux dimensions de bigrammes, intégrant des valeurs binaires pour chaque association de lettres ou de mots. Ces valeurs binaires, 0 et 1, symbolisent respectivement l'absence ou la présence de n-grammes spécifiques dans le lexique choisi. Si un n-gramme d'un mot ne réussit pas à obtenir une valeur de 1, alors le segment de texte englobant est identifié comme étant invalide.

En contraste, la méthode probabiliste s'appuie sur les valeurs dérivées de la fréquence d'occurrence des n-grammes dans de larges corpus. Chaque n-gramme d'une séquence d'entrée est comparé à la table statistique des n-grammes, aboutissant à un indice de bizarrerie pour chaque mot en se basant sur ses n-grammes inclus. Bien que cette technique offre une façon rapide et préliminaire de détecter les erreurs non lexicales, son caractère probabiliste implique qu'elle peut valider par erreur des mots mal orthographiés, et inversement.

Pour accroître l'exactitude dans la détection des erreurs d'orthographe, l'analyse des n-grammes est régulièrement combinée à d'autres techniques, incluant la fouille de dictionnaire, la transcription phonétique, et l'analyse morphologique.

2.3.2 Techniques de correction orthographique

La fonction principale des outils de correction orthographique est de rectifier les fautes d'orthographe identifiées. Pour cela, ils se basent sur un dictionnaire ou un ensemble de textes, qui servent de source pour les corrections possibles. On peut distinguer deux formes majeures de correction orthographique : l'approche interactive et l'approche automatique.

L'approche interactive, la plus simple des deux, propose à l'utilisateur une sélection de corrections

possibles parmi lesquelles il peut choisir. Cette méthode met le contrôle entre les mains de l'utilisateur, lui permettant de décider de la correction à appliquer. En revanche, l'approche automatique est plus complexe et fait appel à l'intelligence artificielle. Elle permet de corriger les erreurs sans nécessiter d'intervention de l'utilisateur, comme on le constate dans certaines applications de RAP (Zhang *et al.*, 2019).

Il est possible de catégoriser les techniques de correction orthographique en deux sous-groupes : la correction d'erreurs de mots individuels et la correction d'erreurs qui dépendent du contexte. La correction d'erreurs de mots individuels s'occupe de rectifier les mots mal écrits indépendamment du contexte linguistique ou textuel, tandis que la correction contextuelle tient compte du contexte linguistique environnant.

La nature de la technique de correction mise en oeuvre dépend en grande partie des types d'erreurs orthographiques qu'elle est conçue pour corriger. Le processus de correction utilise des techniques de correspondance approximative de chaînes (Hall et Dowling, 1980) pour trouver des mots bien orthographiés qui ressemblent à celui qui est mal orthographié. Ce processus est divisé en deux phases : la génération de corrections orthographiques possibles et le classement de ces corrections suggérées.

Dans la phase de classement, le système peut se servir d'une mesure de similarité lexicale entre les mots mal orthographiés et les corrections proposées, ou d'une évaluation probabiliste de la chance de chaque mot correct. Ceci permet de classer les suggestions par ordre de pertinence. Cependant, cette étape pourrait ne pas être nécessaire dans le cadre des correcteurs orthographiques interactifs, où c'est l'utilisateur qui a le dernier mot dans le choix des mots.

Plusieurs techniques, souvent mentionnées dans la littérature, sont employées pour la correction des erreurs. Parmi celles-ci, on retrouve la distance d'édition minimale, la clé de similarité, les techniques basées sur des règles, les techniques probabilistes et les réseaux de neurones.

2.3.2.1 La distance d'édition minimale

La distance d'édition minimale est une technique largement exploitée en correction orthographique, qui vise à évaluer le plus petit nombre d'opérations d'édition nécessaires pour transformer un ensemble de caractères en un autre. Les distances d'édition les plus utilisées sont : la distance de Levenshtein et la distance de Damerau.

La distance de Levenshtein (1965) prend en compte trois types d'opérations : les insertions, les suppressions et les substitutions. Cependant, il ne considère pas la transposition comme une opération valide, se basant sur l'idée que la transposition pourrait être traitée par une combinaison de suppressions et d'insertions ou de deux substitutions.

La distance de Damerau (1964), quant à elle, ajoute la transposition de deux caractères adjacents aux opérations valides, en plus de celles prises en compte par Levenshtein. Damerau soutient que ces quatre types d'opérations correspondent à plus de 80% des erreurs orthographiques commises par les humains.

Pour détailler, les insertions se réfèrent à l'ajout d'une lettre à un mot incorrectement orthographié pour le corriger, tandis que les suppressions font référence à la suppression d'une lettre pour atteindre le même objectif. Les substitutions impliquent le remplacement d'une lettre incorrecte par la lettre correcte dans un mot mal orthographié, et les transpositions englobent l'échange des positions de deux lettres adjacentes pour corriger l'orthographe. Selon une étude préliminaire réalisée par Gates (1937) et citée par Damerau (1964), la majorité des erreurs d'orthographe peuvent être corrigées par une seule opération : insertion, suppression, substitution, ou transposition de deux caractères.

Dans leur application concrète, les algorithmes d'éditions minimales effectuent des comparaisons entre le mot mal orthographié et chaque entrée du dictionnaire sélectionné. Pour optimiser le processus de recherche, certaines techniques sont mises en place. Par exemple, la méthode inverse de Damerau, utilisée par Kernighan *et al.* (1990), crée toutes les permutations d'erreur uniques possibles du mot incorrectement orthographié et les compare avec le dictionnaire. Si une permu-

tation donne un mot valide, elle est ajoutée à un ensemble de corrections potentielles. L'emploi de structures de données comme les arbres trie (Muth et Tharp, 1977) peut aussi contribuer à minimiser le temps de recherche.

2.3.2.2 La clé de similarité

Les méthodes de clé de similarité, fondamentalement, attribuent à chaque mot une clé unique, rendant ainsi plus aisée la détection de mots ayant une orthographe semblable. La clé associée à un mot spécifique est élaborée de façon à correspondre ou ressembler aux clés de mots orthographiés de la même manière. Quand on les met en pratique sur un mot mal orthographié, ces méthodes peuvent efficacement proposer des rectifications possibles en se référant à un dictionnaire et en identifiant des mots ayant des clés analogues, éliminant ainsi la nécessité d'une recherche exhaustive dans le dictionnaire.

Ces méthodes utilisent des clés qui englobent différents types de relations entre les caractères d'un mot, tels que la similarité positionnelle, matérielle et ordinale (Faulk, 1964).

La similarité positionnelle concerne la conformité des positions des caractères entre deux séquences. Cette forme de similarité est souvent rencontrée dans les textes issus de la Reconnaissance Optique de Caractères (ROC) et les comparaisons de textes. Cependant, elle a été jugée trop restrictive pour être utilisée seule dans la correction orthographique (Peterson, 1980 ; Angell *et al.*, 1983).

La similarité matérielle, de son côté, évalue dans quelle mesure deux séquences possèdent exactement les mêmes caractères, sans prendre en compte leur ordre. Cette forme de similarité a été quantifiée en utilisant des coefficients de corrélation entre le mot mal orthographié et d'autres mots contenant les mêmes caractères, mais dans un ordre différent (Sidorov, 1979). Cependant, cette mesure est considérée comme imprécise pour la correction orthographique car tous les anagrammes sont classés dans la catégorie de similarité matérielle.

Finalement, la similarité ordinale, qui ressemble à la similarité positionnelle, estime le degré au-

quel les caractères correspondants de deux séquences suivent le même ordre. Cette forme de similarité, bien qu'étroitement liée à la similarité positionnelle, met davantage l'accent sur la séquence globale des caractères, offrant une dimension supplémentaire de comparaison pour les tâches de correction orthographique.

2.3.2.3 Approches basées sur les règles

Les méthodes qui s'appuient sur des règles sont des algorithmes qui contiennent la connaissance des patrons d'erreur d'orthographe qui reviennent fréquemment. Ces règles encadrent cette connaissance, qui couvre divers domaines, y compris des informations morphologiques de base et la longueur des mots incorrectement orthographiés. Le processus de correction orthographique repose sur l'application de toutes les règles pertinentes à un mot mal orthographié, et chaque mot valide que le dictionnaire propose est ensuite retenu. Les suggestions pour les mots de remplacement sont triées en fonction d'une évaluation préétablie de la probabilité de l'erreur que la règle en question est conçue pour traiter. Il est important de souligner que cette approche fonctionne de manière autonome, sans s'appuyer sur un cadre grammatical ou une formule de décomposition, et peut simplement nécessiter une procédure de recherche dans le dictionnaire (Yannakoudakis et Fawthrop, 1983 ; Armstrong *et al.*, 1995).

2.3.2.4 Approches probabilistes

Les techniques basées sur la probabilité utilisent souvent des modèles de langue construits à partir de n-grammes (Stolcke, 2000). Nous distinguons principalement deux sortes de probabilités : les probabilités de Markov et les probabilités de confusion.

Les probabilités de Markov, aussi appelées probabilités de transition, calculent la probabilité qu'une lettre spécifique suive une autre dans une langue précise. Ces probabilités sont évaluées grâce aux données de fréquence des n-grammes extraites d'un vaste corpus linguistique. L'idée centrale est que la langue fonctionne comme une source de Markov, ce qui signifie que la probabilité d'une transition est indépendante des transitions qui l'ont précédée.

Les probabilités de confusion, également appelées probabilités d'erreur, estiment la probabilité qu'une lettre donnée remplace une autre dans le cas où une erreur est survenue. Ces probabilités sont déduites soit en insérant un texte de test dans un appareil de ROC et en compilant les erreurs statistiques, soit en laissant l'appareil ROC générer un vecteur de n éléments, chacun représentant une probabilité pour chaque lettre de l'alphabet lors de la reconnaissance de caractères (Bledsoe et Browning, 1959).

La différence majeure entre les probabilités de Markov et les probabilités de confusion réside dans leur dépendance. Les probabilités de Markov dépendent de la langue, en se basant sur l'idée que la langue est une source de Markov et que le corpus étudié est souvent spécifique à un certain domaine. À l'inverse, les probabilités de confusion dépendent de la source, en variant avec les techniques et les caractéristiques des différents appareils ROC, tels que les types de polices. Chaque appareil produit une distribution de probabilité de confusion unique. Les probabilités de confusion peuvent également prendre en compte les erreurs humaines en récupérant les statistiques d'erreurs à partir d'échantillons de textes saisis.

Avec les approches probabilistes, les corrections possibles sont généralement triées grâce à un algorithme bayésien ou un canal bruyant (Kernighan *et al.*, 1990). Plusieurs essais ont été menés pour rectifier l'orthographe en se basant sur ces deux types de probabilités, en utilisant souvent l'algorithme de Viterbi (1967). Cet algorithme emploie un graphe pour capturer la structure du dictionnaire et les caractéristiques du canal de l'appareil ROC, en calculant les probabilités de transition du dictionnaire et les probabilités de confusion à partir des caractéristiques de l'appareil.

2.3.2.5 Approches basées sur les réseaux de neurones

Les réseaux neuronaux se distinguent par leur potentiel exceptionnel pour corriger les erreurs orthographiques, grâce à leur aptitude à établir des connexions mémorielles même face à des entrées partielles ou inexactes. Ils peuvent être ajustés pour répondre à des schémas d'erreur spécifiques à un domaine particulier en les formant sur des erreurs orthographiques réelles, offrant ainsi la possibilité d'accroître l'exactitude de la correction dans ledit domaine.

L'exploration de méthodes basées sur les données, notamment l'apprentissage non supervisé (Soricut et Och, 2015) et la technique de bootstrapping (Yarowsky *et al.*, 2001), a pris de l'ampleur ces dernières années. La performance de ces méthodes a été renforcée par l'apprentissage par transfert (Täckström *et al.*, 2012) entre différentes langues et l'utilisation de corpus comparables (Madnani *et al.*, 2012) en vue d'optimiser la correction orthographique dans les langues peu dotées. Toutefois, l'efficacité de ces stratégies peut être contrainte par la disponibilité et la qualité des corpus parallèles.

Des modèles séquence à séquence (Sutskever *et al.*, 2014) ont également été employés dans le domaine de la COA, avec des résultats encourageants. Ces modèles s'appuient sur une architecture d'encodeur-décodeur pour associer des séquences mal orthographiées à leurs correctes correspondances (Hládek *et al.*, 2019). De plus, des mécanismes d'attention (Bahdanau *et al.*, 2015) ont été intégrés pour améliorer l'alignement entre les séquences d'entrée et de sortie (Garg *et al.*, 2019).

L'avènement des modèles Transformer a renforcé les capacités des réseaux neuronaux dans le domaine de la COA. Ces modèles, qui exploitent les mécanismes d'auto-attention, sont particulièrement efficaces pour détecter des dépendances à longue distance et fournir des représentations sensibles au contexte. Des recherches récentes ont mis en oeuvre des modèles Transformer pour détecter et corriger les erreurs orthographiques (Sorokin *et al.*, 2016), et certains ont adapté ces modèles aux langues disposant de ressources limitées (Al-Ghamdi *et al.*, 2023).

2.4 Méthodologies d'évaluation

L'évaluation des systèmes de vérification orthographique et de correction joue un rôle essentiel en fournissant des informations sur leur efficacité et leur applicabilité à un large éventail de situations. Cette évaluation permet de mesurer la précision des systèmes, leur capacité à détecter les erreurs et leur efficacité à proposer des corrections appropriées. Mener ce processus d'évaluation est également crucial, car il a une incidence directe sur l'expérience de l'utilisateur et la confiance accordée à la fiabilité de ces systèmes.

Cependant, il est important de noter qu'il n'existe pas de norme universellement acceptée ni de méthode prédéfinie pour évaluer ces systèmes de vérification orthographique et de correction. La communauté scientifique et technologique n'a pas encore trouvé de consensus sur les paramètres à utiliser lors de ces évaluations. Cette absence de standardisation est principalement due à la diversité des types d'erreurs d'orthographe et à la complexité inhérente à leur correction précise.

Malgré cette absence de norme globale, plusieurs méthodologies ont émergé comme des pratiques courantes dans le domaine. Parmi celles-ci, trois sont basées sur des métriques d'évaluation automatique : l'utilisation de métriques de classification, de métriques de Traduction Automatique (TA), et de métriques de Recherche d'Informations (RI). Parallèlement à ces métriques automatisées, l'évaluation manuelle a également été reconnue pour sa valeur intrinsèque.

Chacune de ces méthodologies présente des avantages spécifiques et des défis potentiels lorsqu'il s'agit d'évaluer ces systèmes. Néanmoins, en les utilisant conjointement, elles offrent une vision globale des performances d'un système. Elles constituent un cadre solide et inclusif qui nous permet d'approfondir notre compréhension des systèmes de vérification orthographique et de correction, en nous fournissant des informations sur leur fonctionnement, leurs possibilités d'amélioration et leur efficacité globale.

2.4.1 Métriques de Classification

Les métriques de classification fournissent une évaluation détaillée de la capacité d'un système à classer avec précision les mots comme étant correctement orthographiés ou mal orthographiés. Pour calculer ces métriques d'évaluation, il est primordial de comprendre les concepts suivants :

- Vrai positif (TP) : Mot correctement orthographié (valide) reconnu comme tel par le correcteur orthographique.
- Faux positif (FP) : Mot incorrectement orthographié (invalide) reconnu à tort comme correct par le correcteur orthographique.
- Faux négatif (FN) : Mot correctement orthographié reconnu à tort comme incorrect par le correcteur orthographique.
- Vrai négatif (TN) : Mot incorrectement orthographié reconnu comme tel par le correcteur

orthographique.

Une fois ces concepts établis, ils permettront de calculer les métriques de classification proposées par Van Rijsbergen (1979) ; Paggio et Music (1998) ; Paggio et Underwood (1998) ; King (1999) ; Voorhees et Garofolo (2000) ; Starlander et Popescu-Belis (2002).

2.4.1.1 Rappel lexical ou R_c

Le calcul du rappel lexical repose sur la détermination du rapport entre le nombre de mots valides correctement identifiés par le correcteur orthographique et le nombre total de mots valides contenus dans le texte étudié. Cette procédure est démontrée dans la Formule 2.1, laquelle est présentée en détail par Paggio et Music (1998) ; Paggio et Underwood (1998).

$$R_c = \frac{T_p}{T_p + F_n} \tag{2.1}$$

2.4.1.2 Rappel d'erreur ou R_i

Le rappel d'erreur est défini comme la proportion de mots incorrects détectés par le correcteur orthographique par rapport au nombre total de mots incorrects dans le texte. Les travaux de Paggio et Music (1998) ; Paggio et Underwood (1998) ont fourni une formulation précise de cette mesure, qui est présentée en détail dans la Formule 2.2.

$$R_i = \frac{T_n}{T_n + F_p} \tag{2.2}$$

2.4.1.3 Précision lexicale ou P_c

La précision lexicale est évaluée en calculant le ratio entre le nombre de mots valides correctement détectés par le correcteur orthographique et la somme des mots valides détectés par le correcteur, ainsi que des mots invalides qui n'ont pas été identifiés comme incorrects par le correcteur orthographique. Cette mesure est présentée en détail dans la Formule 2.3, selon les explications

fournies par Paggio et Music (1998) ; Paggio et Underwood (1998).

$$P_c = \frac{T_p}{T_p + F_p} \tag{2.3}$$

2.4.1.4 Précision d'erreur ou P_i

La précision d'erreur est évaluée en effectuant la division entre le nombre total de prédictions correctes effectuées par le correcteur orthographique et le nombre total de prédictions émises par le système. Cette évaluation est démontrée en détail dans la Formule 2.4, telle qu'exposée par Paggio et Music (1998) ; Paggio et Underwood (1998).

$$P_i = \frac{T_n}{T_n + F_n} \tag{2.4}$$

2.4.1.5 F-mesure ou Fm

En règle générale, la moyenne harmonique, également connue sous le nom de H_{mean}, est couramment utilisée pour calculer une moyenne de ratios. Sa définition est la suivante :

$$H_{mean} = \frac{n}{\sum_{i=1}^{n} \frac{1}{m_i}} \tag{2.5}$$

Si nous prenons l'exemple où n = 2 avec m_1 = P et m_2 = R, la formule se transforme de la manière suivante :

$$H_{mean} = \frac{2}{\frac{1}{P} + \frac{1}{R}}$$

$$H_{mean} = \frac{2 \times P \times R}{P + R} \tag{2.6}$$

La mesure F1, également connue sous les noms de F-mesure ou F-score (Van Rijsbergen, 1979),

est soumise à certaines contraintes. Par conséquent, pour évaluer cette mesure, nous pouvons appliquer la formule suivante :

$$Fm = \frac{2 \times P \times R}{P + R} \tag{2.7}$$

Étant donné que nous avons défini plusieurs catégories de précision et de rappel, il est possible de dériver les moyennes harmoniques suivantes à partir de la mesure F1 principale :

1. F-score lexical ou Fm_c : Le F-score lexical est utilisé pour calculer la moyenne harmonique entre le rappel lexical et la précision lexicale. Cette mesure est détaillée dans la Formule 2.8.

$$Fm_c = \frac{2 \times P_c \times R_c}{P_c + R_c} \tag{2.8}$$

2. F-score d'erreur ou Fm_i : Le F-score d'erreur est calculée en utilisant la moyenne harmonique du rappel d'erreur et de la précision d'erreur. Cette évaluation est détaillée dans la Formule 2.9.

$$Fm_i = \frac{2 \times P_i \times R_i}{P_i + R_i} \tag{2.9}$$

2.4.1.6 Précision globale ou PA

La précision globale permet de quantifier la probabilité qu'un mot, qu'il soit correct ou incorrect, soit correctement traité par le correcteur orthographique. Cette probabilité est calculée à l'aide de la Formule 2.10, telle qu'énoncée par Starlander et Popescu-Belis (2002).

$$PA = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{2.10}$$

2.4.1.7 Précision de suggestion ou SA

La précision de suggestion évalue la capacité de notre correcteur orthographique à proposer les alternatives appropriées pour les mots mal orthographiés. Nous notons S une suggestion appropriée pour un mot incorrect et N représente le nombre total de mots mal orthographiés. La précision des suggestions fournies par notre système est calculée à l'aide de la Formule 2.11, telle qu'exposée par King (1999).

$$SA = \frac{1}{N} \sum_{i=1}^{n} S_i \tag{2.11}$$

2.4.2 Métriques de Recherche d'Informations

La Correction Orthographique Automatique peut être considérée comme une variante de la Recherche d'Informations, où le mot invalide fonctionne comme une requête et la liste de suggestions fournies par le système de correction correspond à la réponse à cette requête.
Notre correcteur orthographique adopte une approche systématique en sélectionnant toujours le premier mot de la liste de suggestions, l'identifiant comme la correction la plus probable pour l'erreur orthographique. Cependant, nous reconnaissons que la première suggestion n'est pas nécessairement la plus appropriée. Par conséquent, nous employons une mesure pour évaluer l'efficacité de la méthode de classement des suggestions du système.
La mesure que nous utilisons est le Mean Reciprocal Rank (MRR), un indice statistique couramment utilisé pour évaluer les performances des systèmes de recherche d'information qui produisent une liste classée de réponses à une requête.
Le calcul du MRR prend en considération le nombre total de mots invalides (N) ainsi que la position de la suggestion correcte ($Rank_{i,c}$) dans la liste de suggestions correspondant au i^{eme} mot mal orthographié parmi les N. La formule 2.12, proposée par Voorhees et Garofolo (2000), est utilisée pour cet objectif.

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{Rank_{i,c}} \tag{2.12}$$

2.4.3 Métriques de Traduction Automatique

Diverses méthodes quantitatives ont été développées pour évaluer de manière automatique l'efficacité des systèmes de traduction. Ces méthodes impliquent généralement une comparaison entre les phrases générées par le modèle de traduction et les phrases de référence. Chaque métrique fait appel à des méthodes distinctes pour évaluer la qualité des traductions, en mettant l'accent sur des aspects tels que la correspondance des n-grammes, l'agencement des mots, la synonymie, et la distance d'édition.

La tâche de la Correction Orthographique Automatique peut être envisagée comme un défi de Traduction Automatique, où l'objectif est de traduire un mot mal orthographié en un mot correctement orthographié. Étant donné les nombreuses similitudes entre les deux processus, il est courant de nos jours d'employer les métriques de TA pour évaluer la performance des systèmes de COA.

2.4.3.1 La métrique BiLingual Evaluation Understudy (BLEU)

Le score BLEU est la métrique d'évaluation la plus populaire dans la tâche de TA. Elle a été introduite par Papineni *et al.* (2002) et a montré une corrélation significative avec l'évaluation humaine.

Pour détailler davantage, nous définissons les variables P_1, P_2, P_3,... P_n comme suit : P_1 représente la précision des unigrammes, c'est-à-dire la proportion de mots correctement traduits parmi tous les mots candidats. De même, P_2 mesure la précision des bigrammes, c'est-à-dire la proportion de bigrammes corrects parmi tous les bigrammes candidats, et ainsi de suite pour P_3 (trigrammes) et les suivants.

Par la suite, pour combiner toutes ces précisions, nous calculons la moyenne géométrique des

ratios, que nous appelons G_{mean}, comme suit :

$$
\begin{aligned}
G_{mean} &= \left(\prod_{i=1}^{n} P_i \right)^{\frac{1}{n}} \\
&= \exp\left(\log_e \left(\prod_{i=1}^{n} P_i \right)^{\frac{1}{n}} \right)
\end{aligned}
$$

$$
G_{mean} = \exp\left(\frac{1}{n} \sum_{i=1}^{n} \log_e(P_i) \right) \tag{2.13}
$$

Ensuite, nous introduisons une Brevity Penalty (BP) ou pénalité de brièveté, qui évalue la corrélation entre la longueur de la référence (r) et la longueur de l'hypothèse (c), selon la formule suivante :

$$
BP = \begin{cases} 1 & \text{si } c > r \\ \exp\left(1 - \frac{r}{c}\right) & \text{sinon} \end{cases} \tag{2.14}
$$

Finalement, la métrique d'évaluation BLEU est obtenue grâce à la formule suivante :

$$
BLEU = BP \times G_{mean} \tag{2.15}
$$

2.4.3.2 La métrique Word Error Rate (WER)

Le taux d'erreur de mots, également connu sous le nom de WER, est calculé en utilisant la distance de Levenshtein (1965) appliquée aux mots plutôt qu'aux caractères. Il représente le pourcentage de mots incorrectement reconnus par rapport à un texte de référence. La formule 2.16, illustrée par Popović et Ney (2007), décrit en détail cette mesure.

$$
WER = \frac{\text{\# Insertions + \# Substitutions + \# Suppressions}}{\text{\# Mots dans la référence}} \tag{2.16}
$$

Où :

— # Insertions : Fait référence au nombre de mots alignés ajoutés à l'hypothèse.
— # Substitutions : Indique combien de mots de la référence ont été substitués par des mots alignés dans l'hypothèse.
— # Suppressions : Correspond au nombre de mots présents dans l'hypothèse mais absents de la référence

Le taux d'erreurs de mots est généralement calculé en déterminant le nombre minimal d'opérations d'édition nécessaires pour faire correspondre les phrases générées par un modèle aux phrases de référence. Notons que le WER est un indice variant de 0 à 1. Un WER de 0 signifie que le modèle a atteint une précision parfaite, c'est-à-dire qu'il n'y a aucune différence entre la sortie du modèle et la référence. À l'opposé, un WER de 1 signifie que la sortie du modèle ne correspond pas du tout à la référence, indiquant ainsi l'erreur maximale possible.

2.4.3.3 La métrique Character Error Rate (CER)

Le taux d'erreurs de caractères, ou CER, est une métrique standard employée dans un éventail de domaines, incluant la ROC, la Reconnaissance de l'écriture Manuscrite (RME), et la RAP. Sa fonction principale est d'évaluer la capacité d'un système à identifier correctement les caractères individuels au sein d'un ensemble de données déterminé.

Le CER est défini comme le rapport du nombre minimal d'opérations effectuées au niveau des caractères (soit des insertions, des suppressions ou des substitutions) nécessaire pour transformer la sortie du système en une référence, sur le nombre total de caractères présents dans cette dernière. Cette définition est basée sur la distance de Levenshtein (1965) et peut être formulée mathématiquement de la manière suivante :

$$CER = \frac{\text{\# Insertions + \# Substitutions + \# Suppressions}}{\text{\# Caractères dans la référence}} \tag{2.17}$$

Il est important de noter que le taux d'erreur de caractères et le taux d'erreur de mots sont tous les deux des indicateurs standard de précision pour la reconnaissance. Cependant, ces deux mesures présentent des différences significatives. Le WER est calculé au niveau des mots, tandis que le CER est calculé au niveau des caractères.

En outre, le CER peut être considéré comme une mesure plus précise que le WER. Par exemple, une seule erreur de caractère dans un mot long peut entraîner une modification importante du WER, tandis que son impact sur le CER serait relativement faible.

2.4.3.4 Les métriques Character n-gram F-score (ChrF) et Character unigrams and bigrams F-score (ChrF++)

Le Score ChrF repose sur l'évaluation de la précision et du rappel des n-grammes de caractères. Il a vu le jour grâce à Popović (2015) et s'est révélé utile dans divers contextes, notamment dans l'évaluation de la qualité des traductions automatiques.

Le calcul du ChrF implique plusieurs étapes. Tout d'abord, on compte le nombre de n-grammes de caractères dans la traduction qui correspondent à ceux présents dans la référence. Ensuite, ce nombre est divisé par le nombre total de n-grammes dans la référence pour obtenir le rappel, et par le nombre total dans la traduction pour déterminer la précision. En utilisant la moyenne harmonique de ces deux valeurs, on obtient le score ChrF.

Il est important de souligner que ce calcul est effectué pour chaque ordre de n-gramme de caractères, jusqu'à un ordre maximum spécifique (généralement fixé à 6). Ensuite, une moyenne des scores obtenus pour chaque ordre est réalisée pour produire le score ChrF final.

La formule du score ChrF est la suivante :

$$ChrF_\beta = \left(1 + \beta^2\right) \frac{P \cdot R}{\beta^2 \cdot P + R} \tag{2.18}$$

Où :

— P est la précision des n-grammes de caractères : P = n-grammes corrects / total des n-grammes dans la sortie du système.
— R est le rappel des n-grammes de caractères : R = n-grammes corrects / total des n-grammes dans la référence.
— β est un paramètre permettant de donner plus de poids au rappel ou à la précision.

$$\begin{cases} \beta > 1 \Rightarrow Poids_{rappel} > Poids_{precision} \\ \beta < 1 \Rightarrow Poids_{precision} > Poids_{rappel} \end{cases}$$

La métrique ChrF a évolué pour inclure les n-grammes de mots, engendrant ainsi une variante améliorée connue sous le nom de ChrF++ (Popović, 2017). Contrairement à son prédécesseur, le ChrF++ ne se contente pas de la précision et du rappel des n-grammes de caractères, il intègre également les n-grammes de mots. Par défaut, l'ordre des n-grammes de caractères utilisés est fixé à 6, tandis que l'ordre des n-grammes de mots est établi à 2.

Cette nouvelle métrique reprend la formule du ChrF original, avec une modification majeure : l'ajout d'une composante liée aux n-grammes de mots. Cet ajout confère au ChrF++ un avantage considérable sur le ChrF traditionnel. En effet, le ChrF++ ne considère pas uniquement les n-grammes de caractères, mais aussi les n-grammes de mots. Cela lui permet de capturer plus efficacement le sens des phrases, en tenant compte non seulement de l'ordre des caractères, mais aussi de celui des mots.

Enfin, le ChrF++ a été conçu pour offrir une meilleure corrélation entre l'évaluation humaine et l'évaluation automatique, renforçant ainsi sa pertinence et son utilité dans le domaine de la traduction automatique.

2.4.4 Évaluation manuelle

Selon l'étude menée par Grundkiewicz *et al.* (2015), l'utilisation d'évaluations humaines est cruciale pour apprécier la qualité des outils de COA. Bien que ces évaluations soient plus coûteuses et qu'elles aient un caractère non reproductible, elles présentent des avantages majeurs face aux

méthodes automatisées.

Plutôt que de dépendre uniquement de critères objectifs prédéfinis, l'évaluation manuelle capitalise sur l'expérience et l'expertise des linguistes. Ces professionnels, grâce à leur connaissance approfondie de la langue, peuvent détecter des erreurs qui resteraient invisibles aux outils automatisés. Ils évaluent également la pertinence des corrections proposées avec une grande précision, tenant compte de la variabilité et de la complexité intrinsèque de la langue.

Bien que l'évaluation manuelle offre ces atouts, elle peut être judicieusement complétée par des mesures automatiques. Par ailleurs, l'avènement de plateformes collaboratives telles que Amazon Mechanical Turk (MTurk) [1] facilite la mise en place de ces évaluations humaines à grande échelle.

2.5 Conclusion

En conclusion, les erreurs d'orthographe constituent un problème majeur dans le domaine de la communication écrite, notamment dans les courriels, les discours et les recherches sur Internet. Les chercheurs en TALN ont réalisé des efforts significatifs pour développer des solutions efficaces pour minimiser l'impact de ces erreurs. Parmi ces solutions, les outils de COA ont démontré leur valeur et leur pertinence dans diverses applications de TALN, notamment dans les tâches de RA, TA et RI.

Dans ce chapitre, nous avons exploré les différents types d'erreurs d'orthographe, les techniques de détection et de correction orthographique ainsi que les différentes méthodologies d'évaluation utilisées pour mesurer les performances des systèmes de COA. Ces systèmes, qui ont évolué de solutions basées sur des règles à des modèles d'apprentissage automatique, ont fait preuve d'une grande diversité de techniques et de modèles. Cependant, malgré les avancées significatives dans ce domaine, la plupart des techniques abordées dans ce chapitre se sont concentrées sur les langues véhiculaires. On observe un manque notable d'efforts pour développer des outils de COA pour les langues à faibles ressources. Bien que certaines tentatives aient été faites pour adapter des techniques standards de COA à quelques langues autochtones africaines (Boago Ok-

1. https://www.mturk.com/

getheng *et al.*, 2022 ; M'eric, 2014 ; Salifou et Naroua, 2014), aucun effort n'a été entrepris pour le Wolof. À notre connaissance, la seule recherche dédiée exclusivement à la COA du wolof est celle de (Lo *et al.*, 2016), qui donne un aperçu de l'état actuel du domaine et propose des solutions potentielles pour développer un système de correction adapté à la langue Wolof.

Dans le prochain chapitre, nous présenterons notre première méthodologie basée sur des règles pour la création d'un outil de vérification et de correction orthographique pour la langue Wol.

CHAPITRE 3
MÉTHODOLOGIE DE CORRECTION BASÉE SUR LES RÈGLES

3.1 Introduction

Ce chapitre expose la méthodologie initiale utilisée pour élaborer un premier outil de Correction Orthographique Automatique pour le Wol. Cette approche est fondée sur une combinaison de plusieurs techniques, notamment la distance de Levenshtein pondérée, la programmation dynamique, la structure de données trie, un lexique Wol, ainsi que les règles d'écriture spécifiques à cette langue. Cette méthode tire parti de la nature agglutinante du Wol, qui permet de décomposer les mots en unités plus petites et de construire une structure de données trie représentant tous les mots valides possibles, incluant leurs préfixes et suffixes.

Le lexique Wol élaboré à l'aide d'une méthodologie d'annotation semi-automatique, sert de référence pour les mots Wol valides. Cela nous permet de comparer chaque mot dans un texte donné avec les mots du lexique pour déterminer s'il est valide ou non. Cette approche renforce la précision et l'étendue de la correction orthographique en ne prenant en compte que des mots valides.

En outre, la méthodologie intègre les règles d'écriture spécifiques au Wol pour vérifier la validité des mots. Ces règles couvrent les contraintes phonologiques, morphologiques et syntaxiques du Wol, tels que la structure consonne-voyelle (CV), l'accord des formes verbales avec leurs sujets, l'utilisation de prépositions et d'articles, etc. En appliquant ces règles, il est possible de détecter et corriger les erreurs d'orthographe pour les mots qui ne respectent pas les conventions d'écriture du Wol.

Enfin, la méthodologie s'appuie sur la distance de Levenshtein, qui permet de mesurer le nombre minimal d'opérations (insertion, suppression, substitution) nécessaires pour transformer une chaîne de caractères en une autre. En pondérant les opérations en fonction de leur coût et de leur fréquence relative en Wol, nous sommes en mesure d'attribuer un score à chaque correction candidate et de sélectionner la correction la plus probable à l'aide de la programmation dynamique.

Le reste de ce chapitre est structuré de la manière suivante : la section 3.2 détaille l'approche proposée pour développer notre système d'autocorrection. Les résultats des expérimentations et de l'évaluation sont présentés dans les sections 3.3 et 3.4, respectivement. Ces résultats sont accompagnés d'une analyse des erreurs générées par notre outil, ainsi que des limitations de notre système. Enfin, la section 3.5 conclut le chapitre en résumant les principales contributions de notre étude.

3.2 Approche proposée

L'objectif est de développer un vérificateur automatique pour la langue Wol, qui se concentre sur la détection et la correction des mots erronés. Pour atteindre cet objectif, nous avons élaboré et implémenté l'architecture présentée dans la Figure 3.1.

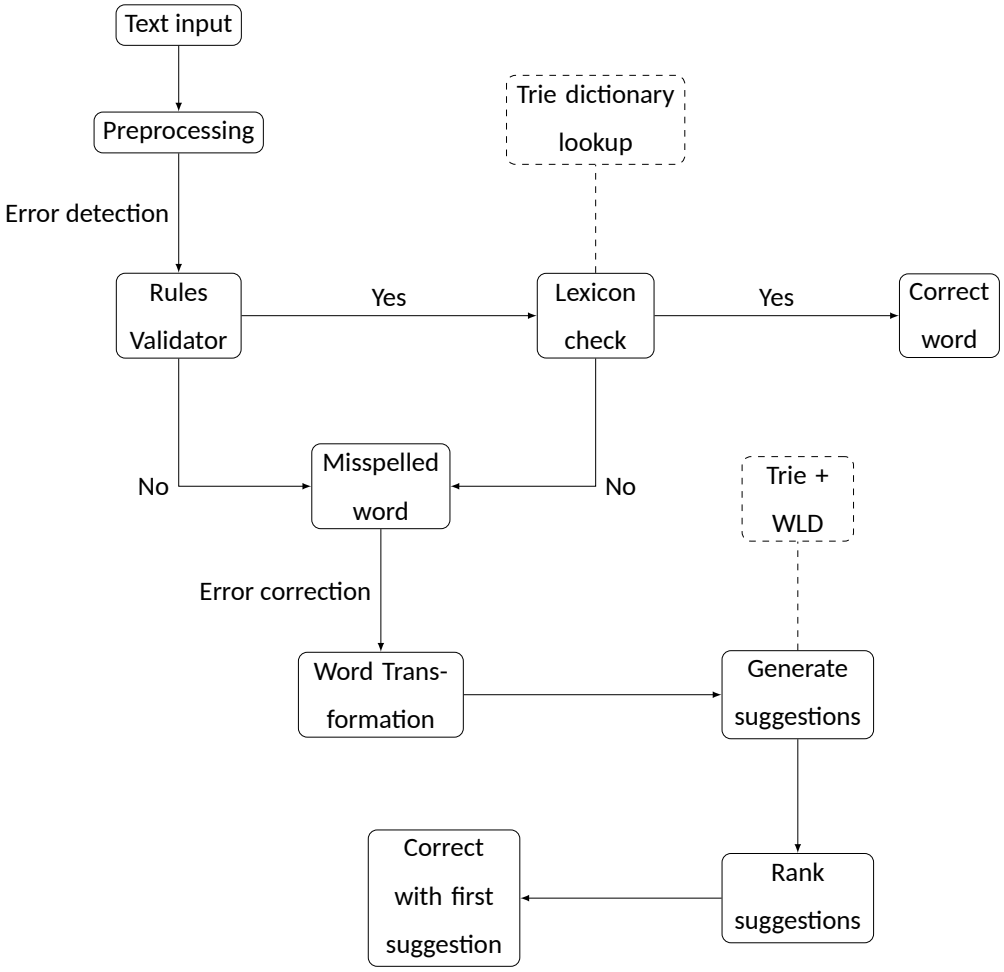

Figure 3.1 Architecture du système d'autocorrection

Le schéma exposé dans la Figure 3.1 offre une représentation graphique des divers éléments et procédés impliqués dans le dispositif de vérification orthographique décrit dans ce chapitre. Le système se compose de dispositifs d'entrée et de sortie, de techniques de détection et de correction d'erreurs, ainsi que de structures de données et de modèles favorisant sa fonctionnalité.

Notre approche consiste ici en quatre étapes principales : (1) Génération d'un lexique Wol, (2) prétraitement, (3) détection des mots invalides et (4) corrections des mots invalides.

3.2.1 Génération d'un lexique wolof

Le développement d'un lexique Wol précis et fiable est un processus crucial qui nécessite une grande attention et une grande précision. Tels que mentionné dans les chapitres précédents, la rareté des ressources disponibles en Wol a présenté un défi important dans la génération du lexique Wol. Par conséquent, notre approche a impliqué la combinaison de méthodes d'annotation manuelle et d'extraction automatique pour surmonter cette contrainte.

Le processus d'annotation manuelle a nécessité un examen minutieux du corpus de texte FLORES-200 (Team *et al.*, 2022 ; Goyal *et al.*, 2022) pour identifier les mots Wol uniques et les extraire dans une liste. FLORES-200 est un corpus de TA qui englobe plusieurs langues considérées comme étant à faibles ressources. Ce corpus a été publié par Facebook dans le cadre du projet No Language Left Behind (NLLB) [1], qui vise à promouvoir la diversité linguistique et l'inclusion des langues à ressources limitées. Le corpus est composé de 842 articles Web distincts, représentant un total de 3001 phrases. Ces phrases ont été réparties en trois ensembles distincts, à savoir dev, devtest et test (caché). En moyenne, chaque phrase a une longueur d'environ 21 mots. Le Tableau 3.1 présente en détail la composition des ensembles relatifs à la langue Wol au sein du corpus FLORES-200.

Cette méthode d'extraction manuelle a permis une étude complète de la langue Wol et a assuré un contrôle manuel de l'inclusion des mots dans le lexique. Cette approche a également permis d'assurer la précision du lexique en s'assurant de l'ajout de mots spécifiques à la langue Wol qui n'auraient peut-être pas été capturés par une technique d'extraction automatique.

1. https://ai.facebook.com/research/no-language-left-behind

Ensembles	# phrases
dev	997
devtest	1012
test	inconnu

Table 3.1 Statistiques du corpus FLORES-200

D'autre part, la méthode d'extraction automatique a impliqué l'utilisation de techniques de Reconnaissance Optique de Caractères (ROC) et d'autres scripts en langage Python, appliqués à plusieurs dictionnaires wolof-français. Les deux principaux dictionnaires qui ont été exploité dans le processus de génération de notre lexique sont (Fal *et al.*, 1990) et (Diouf et Kenkyūjo, 2001).
Ce processus d'extraction automatique a étendu la couverture du lexique initialement obtenu et a permis l'identification de mots supplémentaires qui n'auraient pas été capturés par l'annotation manuelle. Toutefois, il est important de noter que cette approche a également introduit du bruit dans le lexique, car le processus d'extraction automatique n'est pas parfait et peut inclure des mots mal orthographiés ou erronés.

Pour garantir la validité du lexique, l'ensemble des données résultant a été relu et corrigé manuellement. Cette étape a permis de confirmer l'exactitude des mots et de leur orthographe, permettant ainsi toutes les révisions nécessaires avant la génération du lexique final. Cette étape était également essentielle pour s'assurer que le lexique ne contenait que des mots Wol valides, réduisant ainsi le nombre de faux positifs pouvant survenir lors du processus de vérification orthographique.

Il est important de souligner que le lexique final ne contient que 4280 mots différents en raison de la rareté des ressources Wol. Malgré cette contrainte, la combinaison de méthodes d'annotation manuelle et d'extraction automatique a permis la génération d'un lexique Wol fiable. Ce lexique constitue une composante critique pour notre première méthodologie et sera exploité par les modules de détection et de génération des suggestions. Sans un lexique fiable, notre méthodologie ne pourrait pas fonctionner correctement, compromettant ainsi l'exactitude du processus de vérification et de correction orthographique.

3.2.2 Prétraitement

Notre système de vérification orthographique met en oeuvre une étape préliminaire, qui implique la suppression des entrées contenant des caractères numériques, des signes de ponctuation ou des mots empruntés à des langues étrangères. Le résultat de cette phase de prétraitement sert de base au module chargé de la détection des mots invalides présents dans le texte.

La phase de prétraitement inclut donc trois opérations fondamentales à savoir : (1) suppression des signes de ponctuation, (2) normalisation de l'entrée et (3) segmentation du texte (Tokénisation) en mots individuels.

3.2.2.1 Suppression de la ponctuation

L'objectif de la première étape de prétraitement consiste en l'élimination des signes de ponctuation non essentiels du texte d'entrée.
En effet, ces signes peuvent compromettre l'efficacité de la vérification orthographique et perturber l'analyse sémantique ultérieure (Rahimi et Homayounpour, 2022). Par conséquent, leur suppression garantit que les étapes ultérieures du système puissent traiter le texte de manière optimale et rapide.

L'algorithme implémenté à cette étape procède à l'examen minutieux du texte d'entrée et élimine systématiquement les occurrences de signes de ponctuation, notamment les virgules, les points, les points d'exclamation et les points d'interrogation. Ce faisant, le résultat obtenu est un texte épuré de tout élément superflu, prêt à subir une analyse approfondie dans les étapes suivantes du processus.

Il est également à noter qu'une fois le processus de correction effectué, les ponctuations seront réintégrés au texte final généré en sortie par le correcteur.

3.2.2.2 Normalisation de l'entrée

Dans la phase de normalisation de notre étape de prétraitement, le texte d'entrée est transformé en une forme normalisée. Cela s'accomplit en convertissant toutes les lettres alphabétiques en minuscules et en supprimant les mots qui n'appartiennent pas à la langue Wol.

La conversion en minuscules revêt une importance particulière, car de nombreuses techniques en TALN considèrent les mots sous des casse différentes comme des entités distinctes. En convertissant l'intégralité du texte en minuscules, nous éliminons l'impact de la sensibilité à la casse sur l'analyse, en accord avec les recommandations de HaCohen-Kerner *et al.* (2020).

Pour ce qui est de la suppression des mots provenant de langues étrangères, nous avons utilisé le modèle d'identification de langue (Hughes *et al.*, 2006) intégré à la bibliothèque Polyglot (version 16.7.4). Ce modèle est polyvalent et capable de détecter plusieurs langues. Cependant, étant donné que les principales langues étrangères que nous rencontrons dans les textes en Wol sont le Fr et l'Ang, nous nous sommes principalement concentrés sur la détection de ces deux langues.

En conséquence, ce modèle nous a permis de procéder à une élimination sélective des mots écrits en Fr ou en Ang, garantissant ainsi que le texte analysé reste principalement en langue Wol. Cette approche a grandement amélioré la précision de notre analyse en réduisant le risque d'introduire des mots dépourvus de signification sémantique dans le contexte de la langue Wol.

3.2.2.3 Segmentation du texte

La tokenisation est une étape critique dans le processus de vérification orthographique automatique, car elle segmente le texte d'entrée en unités plus petites appelées jetons.

Il existe une multitudes de techniques utilisées pour la segmentation qui varient en fonction de la langue et de la tâche à accomplir. Celles-ci peuvent inclure la séparation sur les espaces et la ponctuation, l'utilisation d'expressions régulières ou l'utilisation de dictionnaires et de règles morphologiques (Dalrymple *et al.*, 2006).

Le processus de tokenisation aboutit à des unités qui peuvent varier des caractères ou des mots individuels aux phrases ou même aux phrases entières. Cependant, la segmentation de mots est la forme de segmentation la plus couramment utilisée dans les systèmes de vérification orthographique, car elle sépare le texte en mots individuels et fournit une base solide pour l'identification des erreurs d'orthographe. Cette approche a été notamment employée dans des travaux récents tels que ceux de Mosavi Miangah (2013) ; Rahman *et al.* (2021) et Abdulrahman et Hassani (2022).

La tokenisation de mots améliore non seulement la précision et l'efficacité du processus de vérification orthographique, mais permet également une analyse du contexte dans lequel chaque mot apparaît. Cela permet au système de vérification orthographique de faire des suggestions plus informées pour des corrections d'orthographe appropriées, améliorant ainsi la précision des résultats.

Dans la présente étude, nous mettons en oeuvre le processus de tokenisation en mettant l'accent sur la segmentation au niveau des mots.

3.2.3 Détection des mots invalides

La phase de détection d'erreurs dans notre système de vérification orthographique nous permet d'identifier l'ensemble des mots invalides présents dans notre texte. Ceci est réalisé en s'appuyant sur un processus en deux étapes rigoureuses.

Tout d'abord, le texte est validé par rapport aux règles d'écriture du wolof, qui sont essentielles pour garantir la cohérence et l'exactitude de l'orthographe. Ensuite, le texte est comparé avec le lexique Wol précédemment construit.

3.2.3.1 Validateur de règles

L'orthographe des mots en langue Wol obéit à des conventions strictement établies. Parmi celles-ci, on note la forme CVC pour les mots monosyllabiques et CVCV(C) pour les mots dissyllabiques, tel que décrit par Merrill (2021). Ces conventions définissent notamment des restrictions sur l'utilisa-

tion des voyelles et des consonnes longues. De plus, certaines consonnes, comme les consonnes pré-nasalisées, doivent apparaître à des positions spécifiques au sein d'un mot. Sur la base des conventions d'écriture de la langue Wol (Njie, 1982 ; Dunigan, 1994 ; Merrill, 2021), nous avons intégré à notre système 53 règles d'écriture propres à cette langue. Le Tableau 3.2 présente quelques-unes de ces règles, utilisées pour vérifier la validité orthographique d'un mot en Wol.

#	Règles
1	Un mot ne finit jamais par une consonne et une voyelle longues
2	Une voyelle longue n'est jamais suivie d'une consonne forte
3	Parmi les consonnes fortes, seules les prénasalisées peuvent débuter un mot.

Table 3.2 Quelques conventions d'écriture (Njie, 1982)

Au sein de notre système de vérification orthographique, une étape de validation rigoureuse a été mise en place. Elle consiste à comparer chaque mot du texte d'entrée avec les conventions d'écriture mentionnées précédemment. Les mots conformes à ces règles grammaticales sont validés pour l'étape suivante, tandis que ceux qui ne le sont pas sont identifiés comme étant incorrects et nécessitent une correction.

3.2.3.2 Vérification lexicale

Dans la phase de vérification lexicale, le système de vérification orthographique analyse chaque mot du texte d'entrée en le comparant avec le lexique wolof pour établir sa validité. Cependant, étant donné que le lexique est un répertoire considérable de mots, la recherche d'un mot précis peut représenter un défi pour un processus de recherche rapide et efficace. Par conséquent, plusieurs techniques ont été développées pour faciliter les recherches dans les grands lexiques (voir la **Section 2.3** pour plus de détails).

Dans le vérificateur orthographique actuel, le système utilise une structure de données trie, qui organise le lexique en noeuds représentant les caractères individuels et un nœud racine représentant la chaîne vide. Cette structure permet une recherche efficace des mots dans le lexique en suivant le chemin correspondant aux caractères du mot cible (Feng *et al.*, 2012). Si le chemin mène

à un noeud terminal, le mot est considéré comme faisant partie du lexique et est jugé valide. En revanche, si le chemin ne mène pas à un noeud terminal, le mot est considéré comme incorrect et des suggestions de correction seront proposées.

Dans la Figure 3.2, nous présentons un exemple d'utilisation des structures de tries avec quelques mots tirés du lexique wolof. Les noeuds noirs indiquent que le chemin jusqu'à ce point dans la structure de trie représente un mot valide.

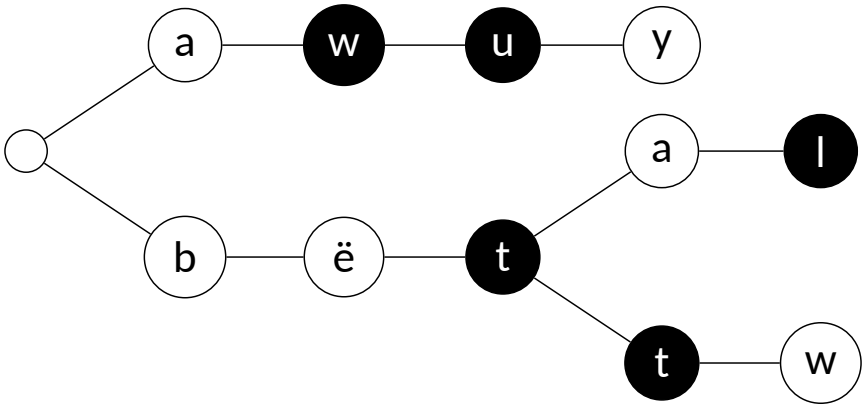

Figure 3.2 Tries avec mots wolof valides et invalides

3.2.4 Correction des mots invalides

La dernière étape de la procédure de correction orthographique consiste à générer des alternatives pour les mots mal orthographiés. Nos techniques de correction, décrites ci-dessous, se concentrent exclusivement sur le mot et ne tiennent pas compte du contexte dans lequel il apparaît. Le processus de correction comprend trois phases distinctes : (1) transformation des sons composés du français en wolof, (2) génération de suggestions candidats, (3) classement de ces suggestions en fonction de leur pertinence suivi de la correction effective du mot invalide.

3.2.4.1 Transformations des sons français composés

Avant de mettre en place le module chargé de translittérer les sons composés du français vers le Wolof, nous avons au préalable procédé à une collecte d'une quantité limitée de données en Wolof

auprès de diverses sources. Les sources exploitées inclues : des sites d'actualités [2], des plateformes de médias sociaux [3] [4] et des sites web religieux [5].

Une analyse minutieuse des données collectées a été menée pour identifier les erreurs les plus courantes commises par les locuteurs wolofs lorsqu'ils rédigent dans leur langue maternelle. Les résultats ont révélé que ces erreurs étaient principalement dues à l'utilisation de l'alphabet français, qui engendrait l'apparition de sons composés français ou de lettres qui ne sont pas originaires du Wolof.
En outre, il a été également observé que les accents, qui jouent un rôle crucial pour garantir la bonne prononciation ainsi que le sens approprié des mots, étaient souvent négligés. Pour étayer ces résultats, le Tableau 3.3 expose certaines des fautes d'orthographe les plus fréquentes et leur équivalent correct en Wolof.

Mots invalides	**Mots corrigés**
dadialé	dajale (Rassembler)
guinaw	ginnaaw (Derrière)
mousiba	musiba (danger)
deuk	dëkk (village)
thiossane	cosaan (tradition)
gnopati	ñoppati (pincer)
niaar	ñaar (deux)
sakhar	saxaar (train)
tank	tànk (pied)

Table 3.3 Exemples de mots wolof invalides et corrections

En prenant en considération les erreurs les plus récurrentes relevées lors de l'analyse des données, notre module a été conçu pour examiner systématiquement chaque mot en quête de sons

2. https://www.wolof-online.com
3. https://twitter.com/SaabalN
4. https://www.facebook.com/wolofakxamle
5. http://biblewolof.com

composés d'origine française ou de lettres n'appartenant pas à l'alphabet Wolof. Lorsqu'un tel son est détecté, notre module se charge de le traduire en son équivalent Wolof. Les lettres qui ne font pas partie de l'alphabet Wolof sont quant à elles automatiquement supprimées.
Une fois ces transformations effectuées, la sortie du module sera dirigée vers la phase suivante du processus de correction, à savoir le module de suggestions de mots candidats.

3.2.4.2 Génération des suggestions

Dans notre système actuel, pour générer des alternatives potentielles aux mots mal orthographiés, nous avons implémenté une méthode de comparaison de distance lexicographique. Ce processus implique de déterminer le nombre minimum d'opérations d'édition nécessaires pour changer un mot en un autre (Vienney, 2004) (voir la **Section 2.3.2.1** pour plus de détails). Parmi les différentes mesures de distance lexicographique, nous avons fait le choix d'utiliser la distance de Levenshtein (Levenshtein, 1965), qui est la plus couramment utilisée pour les systèmes de COA.

Soit $Lev_{\alpha,\beta}$ la distance de Levenshtein entre la sous-séquence formée avec les α premiers caractères d'un mot W_1 et la sous-séquence formée avec les β premiers caractères d'un mot W_2. La distance de Levenshtein entre les deux sous-séquences W_1 et W_2 (de longueur respectivement $|W_1|$ et $|W_2|$) peut être calculée récursivement en utilisant la Formule 3.1 (Levenshtein, 1965).

$$Lev_{\alpha,\beta} = \begin{cases} \max(\alpha,\beta) & \textbf{si } \min(\alpha,\beta) = 0 \\ \min \begin{cases} Lev_{\alpha-1,\beta} + 1 \\ Lev_{\alpha,\beta-1} + 1 & \textbf{sinon} \\ Lev_{\alpha-1,\beta-1} + 1_{(W_{1\alpha} \neq W_{2\beta})} \end{cases} \end{cases} \tag{3.1}$$

La mesure de la distance de Levenshtein, obtenue par l'application de son équation récursive, peut être onéreuse en termes de ressources de calcul (Gusfield, 1997). Cette complexité est particulièrement significative pour des distances importantes, en raison de la nature exponentielle de la complexité temporelle de l'algorithme, qui est de l'ordre de $O(3^{min(|W_1|,|W_2|)})$. Afin de résoudre

ce problème, notre approche combine deux techniques : la programmation dynamique (Bellman, 1957) et l'utilisation d'une structure de données trie.

La programmation dynamique est une technique de résolution de problèmes qui consiste à les décomposer en sous-problèmes plus gérables et à stocker les solutions de ces sous-problèmes (Almudevar, 2001). Elle permet de réduire le nombre de calculs redondants en fournissant un stockage plus efficace des résultats intermédiaires. Lorsqu'elle est appliquée à la distance de Levenshtein, elle permet de stocker les résultats intermédiaires des calculs partiels dans une matrice, ce qui conduit à un calcul plus efficace du résultat final.

En conjuguant la technique de programmation dynamique avec la structure de données trie (Wagner, 1974), notre approche permet une élagage et une exploration efficace de l'espace de recherche tout en évitant les calculs redondants. Cette approche combinée se révèle particulièrement performante pour le calcul de la distance de Levenshtein, même pour des ensembles de données volumineuses, offrant ainsi une solution rapide et efficiente.

Dans la distance de Levenshtein standard, toutes les opérations d'édition sont considérées comme équivalentes et se voient attribuer un coût uniforme de 1. Cependant, en prenant en compte les résultats discutés précédemment, une matrice de coûts a été introduite pour permettre l'attribution de coûts variables à différentes opérations d'édition. Cette approche offre une représentation plus nuancée de l'importance de chaque opération. Plus précisément, les coûts d'insertion et de suppression demeurent à 1 pour tous les caractères. En revanche, pour les opérations de substitution entre les caractères source et cible, un coût de 1 est attribué si le couple de caractères correspond à ceux répertoriés dans le Tableau 3.4, sinon un coût de 2 est assigné.

Notre module de suggestion génère des mots candidats potentiels pour un mot invalide donné en calculant la distance de Levenshtein pondérée entre le mot mal orthographié et chaque mot valide du lexique Wolof.
Pour chaque mot invalide, le coût de transformation de ce dernier vers le mot valide candidat est également fourni.

Couples	Coûts de substitution
(’a’, ’à’)	1
(’a’, ’ã’)	1
(’o’, ’ó’)	1
(’e’, ’é’)	1
(’e’, ’ë’)	1
(’é’, ’ë’)	1
(’x’, ’q’)	1

Table 3.4 Coûts de substitution pour des couples spécifiques

3.2.4.3 Trie et correction effective

Au stade final de notre méthodologie, une évaluation rigoureuse des mots candidats générés lors de l’étape précédente est réalisée. Le classement des mots candidats est basé sur leur degré de similitude avec le mot invalide initial. La priorité est accordée au mot candidat qui implique le coût de modification le plus faible, ce qui en fait la substitution la plus probable pour le mot mal orthographié.

3.3 Expérimentations

Pour parvenir aux résultats présentés et discutés dans la section 3.4, nous avons élaboré un corpus d’évaluation constitué de mots Wolof mal orthographiés ainsi que de mots Wolof valides. Pour construire ce corpus, nous avons opté pour une approche similaire à celle utilisée pour la génération du lexique wolof.

Nous avons utilisé une méthode d’annotation manuelle et automatique, suivie d’une relecture. L’approche consistait à sélectionner les mots wolof couramment mal orthographiés découverts à travers les réseaux sociaux, les sites web religieux et les sites d’actualités. Pour chaque erreur d’orthographe, nous avons ajouté manuellement sa correction. Ce processus a abouti à la formation d’un corpus comprenant 3070 mots, dont 1075 mots valides et 1995 mots invalides. La distance d’édition entre les mots mal orthographiés et leurs formes corrigées est présentée dans le Tableau

3.5.

Distance d'édition	Décompte	Pourcentage
1	400	20.05%
2	412	20.65%
3	445	22.31%
4	281	14.09%
5	204	10.23%
6	114	5.71%
7	67	3.36%
8	36	1.80%
9	23	1.15%
10	9	0.45%
11	2	0.1%
12	1	0.05%
13	1	0.05%
Total	1995	100%

Table 3.5 Distance d'édition des mots invalides comparés aux mots corrigés

Une fois que notre corpus d'évaluation a été finalisé, nous l'avons soumis à notre système d'auto-correction afin d'en évaluer les performances.

3.4 Évaluations

Il convient de prendre en considération divers facteurs lors de l'évaluation des correcteurs orthographiques. Les métriques conventionnelles, telles que le rappel et la précision, mentionnées dans le Chapitre 2, ont été largement utilisées depuis longtemps pour évaluer la compétence linguistique des outils de correction orthographique. Cependant, du point de vue centré sur l'utilisation, ces paramètres d'évaluation ont des limites en raison de l'absence de certaines variables inhérentes à l'évaluation des correcteurs orthographiques dans ces métriques.

Nous avons utilisé les métriques présentées dans le **Chapitre 2** (voir les **Section 2.4.1** et **Section 2.4.2** pour plus de détails) pour évaluer notre système. Il est à noter que malgré leur ancienneté, ces métriques demeurent largement employées dans l'état actuel de l'art pour évaluer les correcteurs orthographiques, en particulier ceux conçus pour les langues à faibles ressources, tels que cités par Abdulrahman et Hassani (2022) et Boago Okgetheng *et al.* (2022).

3.4.1 Résultats

Les résultats de notre correcteur orthographique automatique, tels que présentés dans le Tableau 3.6, montrent un niveau remarquable de performance dans divers aspects de la correction orthographique.

Métriques	**Ratio**	**Pourcentage**
R_c	1023/1075	95.16%
R_i	1995/1995	100%
P_c	1023/1023	100%
P_i	1995/2047	97.46%
Fm_c	0.9752	97.52%
Fm_i	0.9871	98.71%
PA	3018/3070	98.31%
SA	1862/1995	93.33%
MRR	0.9604	96.04%

Table 3.6 Performances du système de correction automatique

Les scores de rappel de 95,16% (R_c) et de 100% (R_i) témoignent de l'exhaustivité du lexique utilisé par le correcteur orthographique, ainsi que de son état relativement épuré (sans mots invalides). Le correcteur orthographique fait preuve d'un niveau remarquable de précision, avec des scores de 100% (P_c) et 97,46% (P_i), indiquant sa fiabilité pour identifier avec précision les erreurs d'orthographe ainsi que les mots valides.
Les scores F_1 de 97,52% (Fm_c) et 98,71% (Fm_i) démontrent que notre système de correction orthographique automatique ne fait pas usage de stratégies simplistes, assurant ainsi son efficacité.

Considérant l'exactitude des suggestions (SA) du correcteur orthographique, le score de 93,33% atteste de la pertinence des alternatives les plus probables aux mots invalides présentée à l'utilisateur final par le système.
Le score de rang réciproque moyen (MRR) de 96,04% met en évidence la qualité du classement des suggestions présentées par le correcteur orthographique.
Enfin, la performance linguistique globale du correcteur orthographique, telle qu'indiquée par son score de précision globale (PA) de 98,31%, est hautement satisfaisante.

3.4.2 Analyse des erreurs

Afin de parvenir à une compréhension détaillée et à l'identification précise des limites linguistiques inhérentes à notre correcteur orthographique, nous avons entrepris une enquête détaillée sur les distances d'édition des mots mal orthographiés, pour lesquels notre système n'a pas fourni de suggestion adéquate. Les résultats de cette enquête ont été synthétisés et sont présentés de manière exhaustive dans le tableau 3.7.

Distance d'édition	**Décompte**	**Pourcentage**
1	4	3.01%
2	17	12.78%
3	32	24.06%
4	20	15.04%
5	22	16.54%
6	13	9.77%
7	10	7.52%
8	11	8.27%
9	3	2.26%
10	1	0.75%
Total	133	100%

Table 3.7 Distances d'édition des mots invalides avec une suggestion erronée

Après avoir minutieusement examiné les résultats obtenus, nous avons été surpris de constater

l'absence de corrélation linéaire significative entre la distance d'édition des mots mal orthographiés et la probabilité que le correcteur orthographique fournisse des suggestions erronées. Ces résultats sont en accord avec ceux présentés dans le Tableau 3.5. La plupart des mots de notre corpus de mots mal orthographiés avaient une distance d'édition de 3, ce qui a accru la probabilité que le correcteur orthographique produise des suggestions inexactes pour les mots mal orthographiés présentant une distance d'édition de 3.

De plus, étant donné que les mots mal orthographiés ayant des distances d'édition de 11, 12 et 13 étaient sous-représentés dans notre corpus, les suggestions du correcteur orthographique pour ces mots étaient toutes correctes. Ces résultats viennent confirmer notre conclusion selon laquelle plus la fréquence des mots mal orthographiés présentant une distance d'édition spécifique est élevée, plus les chances que le correcteur orthographique génère des suggestions erronées pour les mots mal orthographiés ayant la même distance d'édition sont importantes.

En outre, nous présentons ci-dessous le Tableau 3.8 qui illustre plusieurs exemples de phrases renfermant des termes invalides. Pour corriger ces phrases, nous avons utilisé notre système de correction orthographique automatique. Nous avons également inclus les phrases de référence qui ne contiennent aucune erreur pour permettre une comparaison. Les phrases corrigées par notre système mettent en évidence l'efficacité et l'importance de ce dernier pour améliorer la qualité de la langue écrite. Toutefois, ces exemples exposent également les limites et les faiblesses de notre système.

Après avoir analysé quelques phrases (1, 3 et 6) générées par notre système de correction orthographique automatique, il est apparu que les alternatives proposées pour les termes erronés présents dans les phrases invalides étaient les suggestions adéquates. Cependant, pour les phrases 2, 4 et 5, bien que les alternatives proposées par notre système soient syntaxiquement correctes, elles ne sont pas les choix les plus appropriés en termes de contexte et de sens des phrases. Cette lacune révèle donc un manque de prise en compte du contexte dans notre système.

#	Phrases invalides	Phrases générées	Phrases références
1	Maa ngui nieuw légui (J'arrive tout de suite)	Maa ngi ñëw léegi (J'arrive tout de suite)	Maa ngi ñëw léegi (J'arrive tout de suite)
2	Dinaa pouss radio bi (Je vais pousser la radio)	Dinaa puso rajo bi (Je vais aiguille la radio)	Dinaa puus rajo bi (Je vais pousser la radio)
3	Beug naa ame aye tiyaba (J'aimerai avoir des hasanates)	Bëgg naa am ay tuyaaba (J'aimerai avoir des hasanates)	Bëgg naa am ay tuyaaba (J'aimerai avoir des hasanates)
4	Sama diamalé bi nieuwna demb (Ma coépouse est venue hier)	Sama jaale bi ñëwna demb (Ma condoléance est venue hier)	Sama jamaale bi ñëwna demb (Ma coépouse est venue hier)
5	Modou khamp na mbourou mi (Modou a mordu dans le pain)	Modu xam na mburu mi (Modou a connu dans le pain)	Modu xàmp na mburu mi (Modou a mordu dans le pain)
6	Lane ngueine sokhla wone ? (De quoi aviez-vous besoin ?)	Lan ngeen soxla woon ? (De quoi aviez-vous besoin ?)	Lan ngeen soxla woon ? (De quoi aviez-vous besoin ?)

Table 3.8 Phrases invalides, phrases générées et phrases références

3.4.3 Limites du système proposé

Pour expliquer les erreurs précédemment produites par notre système, plusieurs facteurs doivent être pris en considération. Tout d'abord, notre outil de COA est limité aux mots présents dans le lexique Wol que nous avons créé. Par conséquent, il ne peut pas identifier les mots qui ne sont pas inclus dans ce lexique.

Deuxièmement, l'utilisation de l'algorithme de distance de Levenshtein pondéré peut parfois ne pas refléter avec précision la probabilité de différents types d'erreurs, ce qui peut entraîner des suggestions incorrectes.

Troisièmement, les choix de la programmation dynamique et des structures de données basées sur les tries peuvent également entraîner des suggestions fausses positives. Ces approches ne permettent pas de prendre en compte le sens sémantique des mots, ce qui peut conduire à des suggestions qui ne sont pas pertinentes sur le plan sémantique.

En outre, il est important de noter également que notre approche peut entraîner un coût com-

putationnel significatif, en particulier lorsque nous traitons des lexiques volumineux ou des mots avec de nombreuses corrections possibles.

3.5 Conclusion

Dans le présent chapitre, nous avons introduit un système innovant de COA innovant pour la langue Wolof. Grâce à l'efficacité de la combinaison des structures de données trie, la programmation dynamique et l'algorithme de Levenshtein pondérée, ce correcteur a démontré tout son potentiel. L'approche adoptée consistant en une annotation manuelle et automatique a permis de constituer un lexique exhaustif ainsi qu'un corpus robuste de mots mal orthographiés. Ce corpus de mots invalides a constitué une base solide pour évaluer les performances du correcteur orthographique, malgré la disponibilité limitée de données pour cette langue spécifique. Les résultats obtenus dans ce chapitre fournissent des preuves convaincantes quant à la viabilité du correcteur orthographique automatique pour la langue Wol et ouvrent la voie à des améliorations et des explorations futures.

Dans le prochain chapitre, nous allons développer un modèle d'autocorrection plus avancé en utilisant des réseaux neuronaux afin de capturer le contexte et de proposer des corrections pertinentes du point de vue du contexte. Cette approche de modélisation du langage basée sur les réseaux de neurones est particulièrement adaptée aux langues agglutinantes telles que le Wol, où une analyse contextuelle approfondie est nécessaire pour identifier et corriger les erreurs d'orthographe.

CHAPITRE 4
MÉTHODOLOGIE DE CORRECTION BASÉE SUR L'APPRENTISSAGE PROFOND

4.1 Introduction

Le développement des réseaux de neurones, en particulier des techniques d'apprentissage profond, a considérablement transformé le domaine du TALN au cours des dernières années. Ces modèles ont fait preuve d'une performance remarquable dans un large éventail d'applications, allant du RA de texte (See *et al.*, 2017) à l'analyse des sentiments (Socher *et al.*, 2013), en passant par la Correction Orthographique Automatique (COA). L'un des modèles les plus notables dans le domaine est le modèle Transformer, qui constitue la base de diverses approches avancées, telles que le Generative Pre-trained Transformer (GPT) (Radford et Narasimhan, 2018), le Bidirectional Encoder Representations from Transformers (BERT) (Devlin *et al.*, 2019) et plus récemment le Text-to-Text Transfer Transformer (T5) (Raffel *et al.*, 2020).
Par ailleurs, le développement continu de modèles pré-entraînés spécifiquement conçus pour les LRLs, tels que Multilingual Bidirectional Encoder Representations from Transformers (mBERT) (Pires *et al.*, 2019) et XLM-RoBERTa (XLM-R) (Conneau *et al.*, 2020), a permis de renforcer davantage les capacités de ces systèmes.

Dans ce chapitre, nous présentons la deuxième méthodologie mise en oeuvre dans notre recherche : un modèle basé sur l'architecture Transformer pour aborder la tâche de COA en langue Wolof. Nous avons choisi cette approche en raison de sa capacité prometteuse à appréhender des structures linguistiques complexes et à gérer efficacement des longueurs d'entrée variables. De plus, le mécanisme d'attention inhérent aux modèles Transformer permet une parallélisation efficace, rendant le modèle capable de s'adapter à des ensembles de données volumineux.

Le chapitre est structuré de la manière suivante : la section 4.2 expose l'architecture modifiée de notre modèle Transformer, mettant en évidence les ajustements spécifiques effectués ainsi que les hyperparamètres associés. Dans la section 4.3, sont abordées les étapes de prétraitement et la création du corpus parallèle. Ce dernier est constitué de phrases Wolof présentant des erreurs et

de phrases correctement rédigées. La configuration et les hyperparamètres utilisés lors de l'entraînement du modèle sont discutés dans la section 4.4, en mettant en avant les algorithmes d'optimisation et les méthodes pour éviter le surapprentissage. Les résultats obtenus par notre modèle, incluant une analyse des erreurs et des limitations, sont présentés dans la section 4.5. Une étude comparative entre nos deux systèmes de COA est menée dans la section 4.6. Cette étude vise à évaluer les améliorations apportées par le système basé sur l'apprentissage neuronal à travers des analyses statistiques. Enfin, la section 4.7 conclut le chapitre en résumant les contributions principales.

4.2 Modèle proposé

Notre modèle est constitué de deux composants principaux : un encodeur et un décodeur, comme indiqué dans la Figure 1.4 (Vaswani *et al.*, 2017). L'objectif principal de l'encodeur dans notre adaptation du modèle est de traiter les séquences d'entrée qui contiennent des erreurs d'orthographe, tandis que le décodeur se concentre sur la génération de séquences de sortie sans fautes d'orthographe.

Au cours de la phase d'encodage, chaque mot d'entrée est converti en une représentation vectorielle à l'aide d'une couche de plongement lexical. Afin d'insérer des informations supplémentaires, telles que la position des mots, à ces représentations, un encodage positionnel est appliqué. Chacun des composants de notre modèle, l'encodeur et le décodeur, comprend 5 couches identiques (Biljon *et al.*, 2020). Chaque couche est pourvue d'un mécanisme d'auto-attention multi-têtes, avec 2 têtes d'attention. Ce mécanisme est suivi d'un réseau de neurones à action directe positionnel ou position-wise FFN, ayant une taille cachée de 256 et une dimension de propagation avant de 1024.

Le décodeur est constitué de deux blocs d'attention à plusieurs têtes au sein d'une couche, l'un pour les séquences cibles et l'autre pour les sorties de l'encodeur. La première attention à plusieurs têtes est masquée pour éviter que les scores d'attention soient calculés pour les mots suivants. La seconde couche d'attention à plusieurs têtes utilise les sorties de l'encodeur comme requêtes et clés, tandis que les sorties de la première couche d'attention à plusieurs têtes sont utilisées comme

valeurs.
Ce mécanisme permet au décodeur de déterminer les entrées les plus pertinentes de l'encodeur pour le processus de génération de séquences de sortie sans fautes d'orthographe. La sortie de la dernière couche des réseaux de neurones à action directe positionnels est ensuite transmise à une couche linéaire qui agit en tant que classificateur, suivie d'une couche Softmax pour générer le texte corrigé.

Pour l'initialisation du modèle, nous adoptons la méthode Xavier avec un gain de 1.0 pour tous les poids entraînables, tandis que les termes de biais sont initialisés à zéro (Glorot et Bengio, 2010). Les plongements de mots sont également initialisés selon la méthode Xavier avec un gain distinct de 1.0.
Une pratique courante pour réduire le nombre de paramètres entraînables est de lier les plongements lexicaux source et cible ainsi que la couche Softmax (Press et Wolf, 2017). Grâce à notre méthode de segmentation des données, qui est détaillée dans la **Section 4.3.2**, nous avons un vocabulaire relativement restreint, ce qui nous permet de fortement diminuer le nombre de paramètres entraînables.
Les plongements de mots dans l'encodeur et le décodeur ont été fixés à une dimension de 256 pour assurer leur compatibilité avec la taille cachée de nos réseaux FFN. De plus, les plongements lexicaux ont été mis à l'échelle par la racine carrée de leur taille.
Afin de prévenir le surapprentissage, nous utilisons la technique d'abandon sur différents composants du Transformer. Initialement, nous appliquons un taux d'abandon de 10^{-1} aux plongements lexicaux dans l'encodeur et le décodeur, ce qui permet d'ignorer certains mots de la matrice de plongement de mots (Gal et Ghahramani, 2016). De plus, un taux d'abandon de 3×10^{-1} est appliqué uniquement sur les couches du décodeur (Srivastava *et al.*, 2014).

4.3 Préparation des données

Notre processus d'acquisition des données et d'annotation du corpus comprend deux phases principales. Tout d'abord, nous avons identifié des sources appropriées pour les données du corpus, qui étaient disponibles sous différents formats (PDF, texte, HTML, etc.), puis nous avons procédé à l'extraction du contenu. Ensuite, nous avons utilisé une approche hybride, combinant des tech-

niques d'annotation manuelles et automatiques, et effectué une relecture pour pouvoir générer un corpus parallèle de phrases correctes et incorrectes.

4.3.1 Sélection des données

Le processus de collecte de données pour notre étude a été mené de manière exhaustive, en recueillant des données provenant de sources variées. Ces sources incluaient des sites d'actualité reconnus [1], des plateformes de médias sociaux populaires [2] [3], des sites web à vocation religieuse [4], des fichiers PDF de nature religieuse (Diagne, 1997), ainsi que des dictionnaires bilingues Wol-Fr (Fal *et al.*, 1990 ; Diouf et Kenkyūjo, 2001) et des corpus bilingues Wol-Fr publiés (Adelani *et al.*, 2022 ; Team *et al.*, 2022 ; Goyal *et al.*, 2022).

Dans l'ensemble, nous avons collecté 78.384 phrases pour notre corpus. Tout au long du processus de collecte, une attention particulière a été accordée à la diversité du contenu, afin d'assurer que notre corpus englobe des phrases issues de divers domaines et genres. Cette approche garantit la représentativité et la pertinence de notre étude, en tenant compte des différentes utilisations de la langue Wol dans différents contextes.

4.3.2 Annotation des données

Dans un premier temps, nous avons implémenté des scripts en langage Python afin d'extraire des données de sites d'information, de plateformes de médias sociaux et de sites web religieux. Ce processus a abouti à l'obtention de 25.860 phrases en provenance des sites web religieux, de 21.341 phrases issues des plateformes de médias sociaux, ainsi que de 13.245 phrases recueillies auprès des sites d'actualité.

Par la suite, nous avons procédé à l'extraction de 10.087 phrases à partir de fichiers PDF à vocation

1. https://www.wolof-online.com
2. https://twitter.com/SaabalN
3. https://www.facebook.com/wolofakxamle
4. http://biblewolof.com

religieuse, ainsi que de dictionnaires bilingues Wol-Fr. Parallèlement, nous avons tiré parti des données en langue Wol présentes au sein des corpus bilingues Wol-Fr publiés par Facebook (voir la **Section 3.2.1** pour plus de détails) et par Masakhane (Adelani *et al.*, 2022). Masakhane [5] est une organisation qui a pour mission principale d'améliorer et de promouvoir le développement de méthodologies et de ressources en TALN spécifiquement dédiées aux langues africaines. Le Tableau 4.1 fournit une présentation détaillée de la composition des ensembles relatifs à la langue Wol au sein du corpus Masakhane.

Ensembles	# phrases
Entrainement	3360
Développement	1506
Test	1500

Table 4.1 Statistiques du corpus Masakhane

L'ensemble des phrases collectées a été enregistré dans des fichiers au format TXT conformément à la norme d'encodage UTF-8. Une analyse approfondie des phrases collectées a révélé la présence fréquente d'erreurs lexicales et grammaticales. Afin de créer un corpus parallèle comprenant à la fois des phrases mal orthographiées et leurs équivalents sans erreurs, nous avons utilisé notre outil de correction orthographique (Cissé et Sadat, 2023) développé dans la première partie de ce travail. Cet outil a permis de générer un fichier contenant les formes corrigées des phrases. Par la suite, une relecture manuelle approfondie du fichier obtenu a été réalisée afin de corriger toute erreur grammaticale ou lexicale subsistante. Cette démarche nous a assuré d'obtenir un corpus de qualité, prêt à être utilisé par notre modèle d'apprentissage.
Pour les phrases qui étaient initialement sans erreurs, nous avons introduit diverses erreurs typographiques. La majorité des erreurs introduites impliquent la duplication, l'omission, la transposition ou la substitution de caractères. Le Tableau 4.2 offre un aperçu des fautes de frappe introduites.

À la fin de la construction de notre corpus synthétique parallèle, nous sommes confrontés à une décision cruciale avant d'entamer la phase de prétraitement des données et d'entraînement du

5. https://www.masakhane.io

Mot initial	Catégorie d'erreurs	Mot incorrect
Waxtu	Duplication	Waxxtu
Bunt	Omission	Bnt
Juddu	Transposition	udJdu
Ñëw	Substitution	Gneuw
Xaar	Substitution + Omission	Khare
Jàppale	Substitution + Omission	Diapalé
Caabi	Substitution + Omission	Thiabi
Sàkk	Substitution	Spkk

Table 4.2 Types d'erreurs introduits

modèle : nous devons déterminer l'unité linguistique atomique sur laquelle le modèle va opérer. Un nombre important de modèles de TA ont traditionnellement utilisé les tokens comme leur plus petite unité. Cependant, une tendance émergente vers l'utilisation des unités de sous-mots (BPE) (Sennrich *et al.*, 2016b) comme éléments de base a été observée.

Initialement, l'idée d'utiliser les mots comme entrées pour notre modèle semblait être une bonne stratégie par défaut, reflétant l'approche observée dans de nombreux modèles de TALN. Cependant, lorsqu'elle est appliquée à la tâche de COA, l'approche par token peut devenir rapidement complexe en raison des inexactitudes potentielles liées à l'utilisation de la ponctuation. De plus, la nécessité pour les modèles de TALN de fonctionner avec un vocabulaire fixe implique que le vocabulaire de notre outil de COA devrait être suffisamment complet pour inclure toutes les erreurs d'orthographe possibles de chaque mot rencontré pendant le processus d'entraînement. Les implications de cette exigence aboutiraient à un modèle coûteux, tant en termes d'entraînement que de maintenance.

En tenant compte de ces facteurs, nous avons décidé d'utiliser le caractère comme élément de base pour notre modèle. Cette approche s'est révélée très efficace dans les tâches de TA selon (Lee *et al.*, 2017). L'adoption de la segmentation au niveau des caractères nous a également permis de préserver une taille de vocabulaire gérable.

À des fins expérimentales, nous avons procédé à la division de l'intégralité du jeu de données en

trois sous-ensembles distincts : un ensemble d'entraînement, un ensemble de validation et un ensemble de test. Nous avons sélectionné aléatoirement 5% du corpus généré pour constituer les ensembles de validation et de test. Cette démarche s'est avérée essentielle pour garantir une représentativité fidèle de l'ensemble du jeu de données. Les 90% restants des données ont ensuite été utilisés pour former notre ensemble d'entraînement.

4.4 Expérimentations

En ce qui concerne la procédure d'entraînement de notre modèle, nous avons procédé à une sélection minutieuse des hyperparamètres en nous appuyant sur notre étude approfondie de l'état de l'art. Nous avons également entrepris de tester un large éventail de valeurs et de combinaisons de différentes configurations afin de parvenir aux meilleures choix possibles. Cette approche a été primordiale pour s'assurer de la pertinence et la performance de notre modèle.

Pour garantir la reproductibilité des résultats, nous avons adopté une approche d'entraînement déterministe en utilisant une graine aléatoire fixée à une valeur de 42.
La graine aléatoire est un nombre utilisé pour initialiser le générateur de nombres aléatoires. En fixant la graine aléatoire, nous nous assurons que les poids initiaux du modèle sont toujours les mêmes à chaque exécution de l'entraînement. Cela signifie que l'initialisation aléatoire sera la même à chaque fois, ce qui permet de reproduire les mêmes conditions d'entraînement et de comparer les résultats de manière cohérente.

Afin d'optimiser les paramètres internes du réseau en vue de minimiser l'erreur globale, nous avons fait le choix d'utiliser l'optimiseur Adam (Kingma et Ba, 2015).
L'optimiseur Adam est un choix répandu dans la communauté scientifique. Il combine les avantages de deux autres optimiseurs, à savoir l'optimiseur SGD avec moment et l'optimiseur Root Mean Square Propagation (RMSProp). Cette combinaison permet à l'optimiseur Adam de maintenir un taux d'apprentissage adaptatif pour chaque paramètre du modèle, ce qui lui permet d'ajuster l'apprentissage en fonction des caractéristiques spécifiques de chaque paramètre. De plus, l'optimiseur Adam utilise des moments de premier et de second ordre pour estimer l'adaptation des paramètres lors de la mise à jour des poids. Pour notre modèle, nous avons fixé les paramètres

du premier et du second ordre respectivement aux valeurs de $\beta_1 = 9 \times 10^{-1}$ et $\beta_2 = 999 \times 10^{-3}$.

Le taux d'apprentissage est l'élément le plus important lors de l'entraînement des modèles d'apprentissage automatique, notamment pour les modèles utilisant l'optimisation basée sur le gradient. Il joue un rôle déterminant dans l'ajustement des poids du modèle à chaque étape de l'entraînement, ce qui a un impact direct sur la performance globale du modèle. Un taux d'apprentissage optimal permet d'atteindre un équilibre entre la vitesse de convergence et la précision, favorisant ainsi un entraînement efficace. Il évite de passer outre le minimum de la fonction de perte ou de rester piégé dans des minima locaux sous-optimaux. De plus, un taux d'apprentissage approprié favorise une meilleure généralisation du modèle en trouvant le juste milieu entre le sous-apprentissage et le surapprentissage, ce qui améliore la performance du modèle sur des données inconnues.
Après plusieurs cycles d'expérimentation et d'ajustement, nous avons déterminé qu'un taux d'apprentissage de 10^{-4} offrait le meilleur équilibre pour notre modèle et notre jeu de données spécifique. Ce taux d'apprentissage a permis une convergence stable et efficace pendant l'entraînement, et a conduit à un modèle qui se généralise bien à de nouvelles données. Un seuil minimal de 10^{-8} a été établi pour mettre fin à l'entraînement dès lors que la convergence ou la quasi-convergence est atteinte.

Pour optimiser le processus d'apprentissage, nous avons mis en place une stratégie de planification basée sur le plateau (Smith, 2017).
La stratégie de planification plateau consiste à ajuster dynamiquement le taux d'apprentissage afin de permettre au modèle de sortir d'un plateau de performance local et de converger vers une solution optimale. En réduisant le taux d'apprentissage, nous effectuons des mises à jour plus petites des poids du modèle, ce qui peut favoriser une convergence vers un minimum global de la fonction de perte et améliorer la performance générale du modèle.
Dans notre approche, avec une patience fixée à 5, le taux d'apprentissage a été réduit d'un facteur de 7×10^{-1} si le score de validation ne montrait aucune amélioration lors de 5 époques de validation consécutives. Cette adaptation dynamique du taux d'apprentissage, basée sur la rétroaction de performance, a abouti à une convergence et une optimisation améliorées du modèle.

En vue de prévenir efficacement le surapprentissage de notre modèle, nous avons également adopté une technique d'arrêt précoce (Prechelt, 1996).
Le concept de l'arrêt précoce consiste à interrompre l'entraînement du modèle lorsque sa performance sur l'ensemble de validation commence à se détériorer, plutôt que de poursuivre jusqu'à ce que sa performance sur l'ensemble d'entraînement soit maximisée. En minimisant notre fonction de perte d'entropie croisée sur l'ensemble de validation, nous avons pu sélectionner le modèle présentant la meilleure capacité de généralisation, plutôt que celui qui obtient les meilleurs résultats uniquement sur les données d'entraînement spécifiques.

Nous avons opté pour une taille de lot de 4096 jetons (Ott *et al.*, 2018) lors de l'entraînement de notre modèle. Le terme « taille de lot » fait référence au nombre d'exemples d'entraînement inclus dans chaque époque lors de la mise à jour des poids du modèle.
En choisissant une taille de lot plus élevée, nous avons pu optimiser l'utilisation des ressources de calcul disponibles. En effet, les opérations matricielles nécessaires pour la mise à jour des poids peuvent être effectuées de manière parallèle, ce qui a accéléré le processus d'entraînement et a permis d'exploiter efficacement les capacités de calcul de notre système.
De plus, en utilisant une taille de lot plus importante, nous avons bénéficié d'une estimation plus stable du gradient. Une taille de lot élevée permet d'obtenir une meilleure approximation du gradient global, car elle repose sur un plus grand nombre d'exemples d'entraînement. Cette approche a favorisé une convergence plus régulière de notre modèle en évitant les oscillations et les variations brusques lors de la mise à jour des poids.

Afin de favoriser des prédictions diversifiées, nous avons intégré une technique de régularisation connue sous le nom de lissage d'étiquettes (Szegedy *et al.*, 2016).
Cette approche de régularisation vise à redistribuer les poids de probabilité des jetons de référence vers d'autres jetons du vocabulaire en introduisant une certaine incertitude dans les étiquettes d'entraînement. En utilisant un coefficient de lissage de 10^{-1}, nous avons équilibré l'impact de l'information contenue dans les étiquettes de référence avec une certaine incertitude contrôlée, permettant ainsi au modèle d'explorer différentes possibilités lors de la génération de prédictions.

Tout au long de la procédure d'entraînement, nous avons systématiquement évalué la capacité de généralisation et les performances de notre modèle. À cet effet, nous avons effectué des validations à des intervalles de 2000 étapes, couvrant une période totale de 50 époques. Cette fréquence d'évaluation nous a permis d'obtenir un aperçu régulier de la progression de l'entraînement et d'évaluer de manière approfondie les performances du modèle.
Par ailleurs, nous avons établi une fréquence de journalisation de 200 étapes, ce qui nous a permis de suivre de près l'évolution de l'entraînement et d'obtenir des informations détaillées sur son déroulement. Cette journalisation régulière nous a fourni une vue d'ensemble complète de l'avancement de l'entraînement et nous a permis de prendre des décisions éclairées pour améliorer notre approche.

Pendant la phase d'entraînement, notre principal indicateur d'évaluation était le score BLEU, une métrique largement adoptée et reconnue. En plus du score BLEU, nous avons également utilisé la Perplexité (PPL) (Sennrich, 2012) pour évaluer le degré d'incertitude de notre modèle vis-à-vis d'une séquence de caractères donnée. La PPL est une mesure de l'incertitude associée à la prédiction d'une séquence de mots. Plus la PPL est faible, plus le modèle est sûr de ses prédictions, tandis qu'une PPL élevée indique une plus grande incertitude.
En outre, nous avons utilisé notre fonction Perte (Loss) pour quantifier l'écart entre les prédictions générées par notre modèle et les valeurs attendues.
Afin de garantir une évaluation fiable, nous avons fait usage d'une stratégie de regroupement par jetons lors des phases de validation, avec une taille de lot constante de 1024 jetons. Le regroupement par jetons permet également de faciliter le traitement parallèle des calculs, contribuant ainsi à accélérer le processus d'entraînement et d'évaluation sans compromettre la précision des résultats obtenus.
Par ailleurs, dans notre algorithme de recherche en faisceau (Beam Search) (Meister *et al.*, 2020), nous avons fixé la taille à 5, ce qui signifie que nous avons considéré les 5 meilleures hypothèses lors de la génération des séquences de traduction. La recherche en faisceau est une technique d'exploration de l'espace des hypothèses de séquences de mots, permettant de trouver la séquence la plus probable selon le modèle. La taille du faisceau, détermine le nombre de séquences candidates considérées simultanément, ce qui influence la qualité des traductions générées.

Pour gérer efficacement la longueur des séquences générées lors du processus de décodage, nous avons établi une limite supérieure fixe de 175 jetons (Araabi et Monz, 2020) pour la longueur maximale des sorties. Cette stratégie nous a permis de contrôler la longueur des phrases générées et de prévenir les problèmes de débordement ou de surcharge d'informations.
De plus, nous avons veillé à maintenir une surveillance constante du progrès et de l'intégrité de la validation en imprimant systématiquement 3 phrases de validation à chaque exécution de la procédure de validation. Cette approche nous a permis d'évaluer de manière régulière et cohérente la qualité des prédictions du modèle, en nous assurant que les phrases générées pendant le décodage respectent les critères de cohérence, de compréhensibilité et de fidélité aux références fournies.

4.5 Évaluations

Nous avons réalisé une série d'évaluations pour mesurer à la fois le temps d'apprentissage et les performances de traduction de notre système. Le temps d'apprentissage a été mesuré en nombre d'heures, en nombre d'époques et en nombre d'étapes effectuées lors de chaque époque.
Étant donné que notre correcteur orthographique automatique fonctionne en traduisant un texte source contenant des erreurs vers sa forme correcte la plus probable, les mesures détaillées dans le **Chapitre 2** (voir la **Section 2.4.3** pour plus de détails) semblent être les plus appropriées pour évaluer les performances de notre système. Parmi ces mesures, la métrique BLEU a été largement utilisée dans différentes études pour évaluer les performances des outils de COA, comme l'illustrent les travaux de recherche réalisés par Gerdjikov *et al.* (2013), Mitankin *et al.* (2014) et Sariev *et al.* (2014). De plus, la métrique WER a été employée dans une étude menée par Evershed et Fitch (2014).
En complément de ces métriques d'évaluation, nous avons également évalué notre modèle en termes de précision, ou Accuracy Correct Word (ACW), tel que défini par Streiner et Norman (2006).

4.5.1 Résultats

L'entraînement de notre modèle neuronal a été effectué sur une carte graphique NVIDIA GTX 2080 Ti, ce qui s'est avéré être un choix judicieux en raison de sa puissance considérable. Le processus d'entraînement s'est déroulé en 50 époques, nécessitant un temps total de 96 heures. Au cours de cette période, notre modèle a réalisé 900×10^3 étapes d'apprentissage. Bien que cette durée puisse sembler relativement longue, elle s'est révélée essentielle pour permettre à notre modèle de converger vers un état optimal. Il est important de noter qu'à la fin du processus d'entraînement, notre modèle a pu obtenir une PPL finale remarquablement basse de 1.17.

Après avoir terminé la phase d'apprentissage de notre modèle, nous avons procédé à une évaluation détaillée de ses performances. Les résultats de cette évaluation sont résumés dans le tableau 4.3 ci-dessous. Il est important de souligner que cette évaluation a été effectuée en utilisant l'ensemble de données de test, garantissant ainsi une évaluation solide et objective de la performance de notre modèle.

Mètriques	**Ratio**	**Pourcentage**
BLEU	0.83	83%
WER	0.08	8%
CER	0.03	3%
ChrF++	0.94	94%
ACW	0.88	88%

Table 4.3 Performances du modèle neuronal d'autocorrection

La valeur de 83% obtenue pour le score BLEU, une mesure évaluant la similitude entre le texte corrigé et le texte de référence basée sur la superposition des n-grammes, témoigne de l'habileté de notre modèle à corriger et produire un texte qui s'aligne de manière significative avec le texte source, autant au niveau du vocabulaire que de la construction grammaticale.

Le taux d'erreurs de mots WER s'élevant à 0.08 signifie que, en général, seulement 8% des mots dans les énoncés corrigés s'écartent des énoncés de référence. De même, le taux d'erreurs de caractères CER de 0.03 montre que les énoncés corrigés présentent en moyenne seulement 3%

de différence au niveau des caractères par rapport aux énoncés de référence. Ces chiffres illustrent clairement la capacité du correcteur orthographique à repérer et rectifier efficacement les erreurs à l'échelle des mots et des caractères.

De plus, le score ChrF++ de 94% révèle un haut degré de ressemblance entre les énoncés corrigés et les énoncés de référence, en prenant en compte de multiples critères tels que la précision, le rappel, et le F-score au niveau des caractères.

En dernier lieu, la mesure de Précision ACW est de 0.88, soit 88%, ce qui démontre que le modèle prédit avec une grande exactitude l'orthographe des mots. La précision ACW mesure spécifiquement le pourcentage de mots qui ont été correctement orthographiés dans la sortie du modèle, donnant ainsi une mesure de l'exactitude au niveau des mots qui est indépendante des mesures au niveau des phrases ou des caractères. Avec 88% de mots orthographiés correctement, le modèle témoigne d'une forte habileté à détecter et corriger précisément des mots, contribuant ainsi de manière significative à l'exactitude et à l'efficacité générales du correcteur orthographique.

Ces résultats mettent en évidence les performances considérables réalisés par notre modèle neuronal de COA. Les hauts scores BLEU et ChrF++, les valeurs faibles de WER et CER, de même que le score prometteur de ACW, témoignent conjointement de la compétence de notre modèle à rectifier avec précision les mots incorrectement orthographiés et à produire des phrases Wol cohérentes et dépourvues d'erreurs.

4.5.2 Analyse des erreurs

Bien que les métriques d'évaluation mettent en évidence l'efficacité globale du modèle neuronal proposé pour la vérification orthographique et la correction des textes en Wol, il est essentiel de se concentrer également sur les erreurs spécifiques que notre système génère. Cette démarche nous a permis d'obtenir des informations sur les aspects où le modèle excelle et ceux qui nécessitent des améliorations supplémentaires. L'analyse des erreurs suivante, effectuée sur un sous-ensemble des données de test, a identifié trois catégories principales d'erreurs récurrentes produites par notre modèle : les erreurs liées aux (1) voyelles longues, aux (2) entités nommées et aux (3) accents.

1. Voyelles longues : La première catégorie d'erreurs concerne la correction des voyelles longues dans les mots. En Wolof, la distinction entre les voyelles longues et courtes peut considérablement modifier la signification des mots. Notre modèle, cependant, éprouve des difficultés constantes à discerner avec précision quand remplacer une voyelle courte par une voyelle longue, conduisant à des corrections incorrectes.
   Par exemple :
   (hypothèse) Man, maa mel ni garabu oliw → (référence) Man, ma mel ni garabu oliw
   (hyp) Ngir ya ma def ántalpareet → (réf) Ngir yaa ma def ántalpareet
   (hyp) Ay bés a ngi dév, di ñëw → (réf) Ay bés a ngii dév, di ñëw

2. Entités nommées : La deuxième catégorie d'erreurs est associée aux entités nommées. Les entités nommées, qui ne respectent pas souvent les règles d'écriture standard du Wolof, semblent entraîner une confusion significative pour le modèle. Dans certains cas, le modèle fait à tort l'hypothèse que ces entités nommées sont erronées et essaie de les corriger. Dans d'autres cas, lorsque certaines entités nommées sont effectivement mal orthographiées et ne font pas partie du vocabulaire, le modèle propose des corrections incorrectes.
   Par exemple :
   (hypothèse) Allemañe dëkk bou mag la → (référence) Almaañ dëkk bou mag la
   (hyp) Móoritani biir Afrig la nekk → (réf) Gànnaar biir Afrig la nekk
   (hyp) Njiitu réw mi Maki Sal la → (réf) Njiitu réw mi Maki Sàll la

3. Accents : La troisième catégorie d'erreurs est liée à la gestion des accents. Les accents en Wolof jouent un rôle crucial dans la différenciation et la prononciation des mots. Nous avons constaté que notre modèle rencontre des difficultés persistantes lorsqu'il s'agit de localiser avec précision et de restaurer les accents manquants dans les mots.
   Par exemple :
   (hypothèse) Ba loolu amee mu jël nebbon bi sang yérey biir yi → (référence) Ba loolu amee mu jël nebbon bi sàng yérey biir yi
   (hyp) Woorlu ak may askan wi ñuy jot ci téerey bokk-moomeel yi → (réf) Wóorlu ak may askan wi ñuy jot ci téerey bokk-moomeel yi
   (hyp) Ci depàrtemaa bu Binjoona la jëkk a ñakke → (réf) Ci depàrtemãa bu Binjoona la jëkk

a ñàkke

Une observation intéressante découlant de notre évaluation est que notre modèle parvient à corriger efficacement des phrases mal orthographiées, même lorsque celles-ci ont des phrases références comportant des erreurs. Ce phénomène, en plus d'avoir un impact négatif sur les scores d'évaluation obtenus par notre modèle, suggère la présence d'artefacts dans les données d'entraînement. Cette présence d'artefacts pourrait expliquer la génération de certaines erreurs par notre modèle.
Par exemple :
(hyp) Yéene roy imaam boo dee ku ñuy jiite → (ref) Yéenee roy imaam boo dee ku ñuy jiite
(hyp) Toppleen li ñu wàcce ci yéen mu joge ca seen Boroom → (ref) Toppleen li ñu wàcce ci yéen mu jóge ca seen Boroom.

La présence de ces artefacts souligne une fois de plus l'importance considérable de la qualité des données lors de l'entraînement de modèles efficaces.

4.5.3 Limites du modèle proposé

Notre modèle de COA a démontré de bonnes performances, comme l'indiquent ses scores élevés de BLEU, ChrF++ et ACW, ainsi que les scores relativement faibles de WER et CER. Cependant, il existe encore des limites qui nécessitent une investigation et des améliorations supplémentaires.

Tout d'abord, il convient de noter que les modèles basés sur les caractères en tant qu'unité linguistique, tels que celui utilisé dans notre étude, sont intrinsèquement complexes et peuvent nécessiter beaucoup de temps pour être entraînés. Cela est dû à la grande quantité de données qu'ils doivent assimiler, comparativement aux modèles s'appuyant sur les mots en tant qu'unité linguistique. Le coût computationnel de l'entraînement de tels modèles peut être particulièrement élevé lorsqu'on travaille avec de vastes ensembles de données ou des langues comportant un large éventail de caractères.

En outre, un autre défi auquel notre modèle est confronté réside dans sa capacité à saisir les dé-

pendances à longue distance présentes dans le texte. Les relations entre les mots au sein d'une phrase, qui s'étendent souvent sur plusieurs caractères, peuvent être difficiles à appréhender pour les modèles fonctionnant au niveau des caractères. Par conséquent, cela pourrait éventuellement impacter les performances du modèle dans des situations requérant une compréhension approfondie de la sémantique à l'échelle de la phrase.

En troisième lieu, notre modèle ne bénéficie pas de l'avantage d'utiliser des plongements de mots pré-entraînés, qui capturent les relations sémantiques et syntaxiques entre les mots. Par conséquent, la compréhension sémantique du modèle peut être moins nuancée par rapport aux modèles qui opèrent au niveau du mot.

En quatrième lieu, il convient de souligner que les modèles fonctionnant au niveau des caractères peuvent être davantage sensibles au bruit présent dans les données d'entrée. Les erreurs orthographiques, l'usage incohérent de la ponctuation et d'autres formes de perturbations peuvent avoir un impact plus marqué sur ces modèles, ce qui peut potentiellement entraîner des performances plus faibles dans certaines situations.

En outre, bien que notre modèle ait été développé pour prendre en charge toutes les langues qui utilisent un alphabet similaire à celui du wolof, il peut rencontrer des difficultés avec les langues qui dépendent fortement de l'ordre des mots. Cette limitation découle de l'absence de compréhension au niveau des mots dans notre modèle, une caractéristique qui pourrait s'avérer utile dans ces situations.

Enfin, il convient de noter que notre modèle peut éprouver des difficultés lors de la désambiguïsation. Par exemple, des mots ayant une orthographe identique mais des significations différentes peuvent représenter un défi pour les modèles basés sur les caractères, étant donné que ces modèles n'ont pas accès aux informations sémantiques spécifiques aux mots.

Compte tenu de ces considérations, il existe plusieurs axes qui pourraient être ciblés pour l'amélioration. Premièrement, le modèle pourrait être davantage entraîné sur une variété plus large de données textuelles afin d'améliorer sa capacité à gérer des erreurs moins courantes ou plus com-

plexes. Étant donné notre attention actuelle portée sur une langue avec des ressources limitées, l'utilisation de la technique de back-translation se présente comme une stratégie prometteuse. Cette approche a constamment démontré son efficacité dans divers domaines, tels que la Traduction Automatique Statistique (TAS) (Bojar et Tamchyna, 2011), la TAN supervisée (Sennrich *et al.*, 2016a), et la TA non supervisée (Lample *et al.*, 2017). Dans le contexte de la COA, l'adoption de cette approche consisterait à entraîner un modèle pour introduire intentionnellement un nombre substantiel d'erreurs d'orthographe réalistes au sein d'un texte bien orthographié. Par la suite, le corpus résultant de textes modifiés peut être utilisé pour affiner notre modèle de COA.

De plus, nous suggérons d'explorer davantage les modèles hybrides qui combinent les avantages du traitement à la fois au niveau des caractères et des mots. De tels modèles pourraient potentiellement tirer profit de la granularité des modèles au niveau des caractères tout en conservant une compréhension de niveau supérieur de la sémantique des mots et des phrases. De plus, l'inclusion d'une évaluation humaine supplémentaire pourrait fournir une évaluation plus complète des sorties du modèle.

Enfin, étant donné les coûts computationnels associés aux modèles basés sur les caractères, il serait avantageux de se pencher sur des méthodes d'entraînement plus efficaces. En procédant ainsi, nous pourrions réduire la charge computationnelle et améliorer l'efficacité globale du processus d'entraînement.

4.6 Étude comparative des systèmes proposés

L'essence de la recherche et du développement réside dans la comparaison et l'évaluation de systèmes ou de modèles concurrents, en particulier lorsque l'objectif est de distinguer le plus performant. Dans notre quête pour faire progresser le domaine de la COA pour la langue Wol, nous avons élaboré deux systèmes distincts. Bien que chacun ait démontré ses atouts de manière individuelle, il est essentiel de les mettre en parallèle de manière structurée pour véritablement discerner lequel prédomine. Par conséquent, cette section vise à offrir une comparaison méticuleuse des deux systèmes de COA précédemment présentés.

4.6.1 Évaluation humaine

Pour conduire une évaluation manuelle, nous avons adopté une démarche structurée en plusieurs étapes essentielles.
Dans un premier temps, nous avons débuté par une sélection aléatoire de 100 phrases Wol issues de nos corpus formés. Ces phrases, comportant une variété d'erreurs orthographiques, ont servi de base de test pour nos systèmes. Cette sélection variée nous a assuré une évaluation complète, couvrant différentes facettes des capacités des systèmes en question. Suite à cette première étape, chacune des phrases choisies a été introduite dans les deux systèmes de COA afin d'observer et d'analyser les corrections proposées.

La sélection de l'évaluateur approprié s'est avérée un défi majeur. En l'absence d'un centre culturel dédié au Wol et face à la rareté des experts en linguistique Wol, la décision de collaborer avec un locuteur natif du Wol s'est imposée. Cette collaboration s'est avérée bénéfique, car elle nous a permis d'obtenir un retour d'expérience authentique sur la qualité des corrections. Les propositions de correction des deux systèmes ont été méticuleusement examinées par l'évaluateur, sans révélation préalable concernant l'origine de chaque correction, pour assurer impartialité et objectivité.

Le système de notation adopté était le suivant : une note de « 3 » était attribuée à une phrase parfaitement corrigée et conforme à la phrase de référence. Une note de « 2 » était réservée aux corrections qui, bien qu'elles contiennent des erreurs mineures, conservaient le sens initial de la phrase. Enfin, une note de « 1 » était attribuée aux corrections totalement incorrectes ou inadéquates.

Pour résumer les évaluations menées sur l'ensemble des phrases, nous avons élaboré le Tableau 4.4. Celui-ci présente la répartition des notes attribuées à chaque système.

Suite à l'analyse des notes fournies par l'évaluateur, il est manifeste que le système neuronal semble surpasser le système de COA traditionnel en termes d'efficacité. Toutefois, il convient d'approcher cette observation avec prudence.

Note attribuée	Système traditionnel	Système neuronal
1	36/100 (36%)	0/100 (0%)
2	51/100 (51%)	54/100 (54%)
3	13/100 (13%)	46/100 (46%)

Table 4.4 Récapitulatif des notes obtenues par chaque système

Afin de consolider cette hypothèse et d'arriver à une conclusion définitive, il est impératif de mener des tests complémentaires. Ces évaluations supplémentaires permettront de confirmer, ou de nuancer, la supériorité apparente du système neuronal et d'assurer une robustesse à notre analyse.

4.6.2 Test de significativité statistique

L'évaluation de notre système basé sur des règles et de notre modèle neuronal a été méticuleusement réalisée en utilisant des métriques distinctes. Le système basé sur des règles a été examiné à l'aide des métriques traditionnelles : le rappel, la précision, et la mesure F1. Le modèle neuronal, quant à lui, a été évalué en utilisant des métriques plus spécifiques, telles que le score BLEU, le WER et le CER. En raison des différences entre ces métriques d'évaluation, une comparaison directe entre les deux approches s'avère complexe. Afin d'obtenir une évaluation comparative objective de l'efficacité des deux systèmes, une évaluation basée sur le jugement humain a été privilégiée.

Dans la section précédente, nous avons effectué une évaluation qualitative en faisant intervenir des évaluations humaines. Ces évaluations nous ont permis de formuler l'hypothèse que le système neuronal présente des capacités de correction supérieures à celles du système basé sur des règles, spécifiquement dans le contexte de la langue Wol. Cependant, afin de confirmer empiriquement cette hypothèse, nous avons entrepris un test de significativité statistique.

L'essence de notre démarche reposait sur une série de phrases mal orthographiées, dont chaque instance avait des corrections proposées par nos deux systèmes de COA. En tenant compte de la nature ordinale des évaluations, le test des rangs signés de Wilcoxon est apparu comme le choix

le plus pertinent pour déterminer s'il existe une différence statistiquement significative entre nos deux systèmes.

Nous avons défini les hypothèses suivantes pour ce test :

$$\begin{cases} H_0 & : \text{Il n'y a pas de différence notable entre les deux systèmes.} \\ H_a & : \text{Le modèle neuronal est nettement meilleur.} \end{cases}$$

Le test de Wilcoxon, initialement présenté par Wilcoxon (1945), représente une approche non-paramétrique largement employée dans le but de comparer deux échantillons appariés. Cette méthode trouve son utilité dans les situations où les hypothèses relatives à la distribution des données ne sont pas satisfaites ou lorsque les données sont ordinales.

Dans le contexte de notre étude, considérons des observations appariées (x_i, y_i) pour $i = 1, ..., n$, où x_i et y_i représentent respectivement les scores du système basé sur les règles et du système neuronal. Les différences inter-paires sont définies par $d_i = x_i - y_i$. Les observations pour lesquelles $d_i = 0$ sont éliminées car elles n'apportent aucune information sur le signe de la différence. Les différences absolues $|d_i|$ sont ensuite classées par ordre croissant, attribuant un rang de 1 à la plus petite différence, un rang de 2 à la suivante, et ainsi de suite. En cas d'égalité, des rangs moyens sont attribués.
Avec ces rangs obtenus, nous déterminons les statistiques de test signés suivantes :

$$\begin{cases} W^+ & = \sum_{d_i>0} \text{rang}(|d_i|) \\ W^- & = \sum_{d_i<0} \text{rang}(|d_i|) \end{cases} \tag{4.1}$$

Enfin, la statistique de test globale W est dérivée en prenant le minimum entre les deux sommes calculées précédemment :

$$W = \min(W^+, W^-) \tag{4.2}$$

Un niveau de significativité standard ($\alpha = 0,05$) est retenu pour ce test. Un résultat sera considéré comme statistiquement significatif si la valeur-p est inférieure à α.
À la lumière de cette méthodologie, les résultats obtenus pour la statistique-W ainsi que la valeur-p sont consignés dans le tableau 4.5.

Métriques	**Scores**
Statistique-W	0.0
Valeur-p	4.92×10^{-17}

Table 4.5 Récapitulatif de la statistique-W et de la Valeur-p

La statistique-W représente la somme des rangs des différences entre les observations appariées en faveur du système neuronal. Une statistique-W de 0.0 indique que dans la majorité des cas comparés, le système neuronal a été plus performant que le système basé sur des règles.

La valeur-p traduit la probabilité d'obtenir, par hasard, une différence aussi marquée entre les deux systèmes si l'hypothèse nulle était vraie. Dans le cadre de notre test de rang signé de Wilcoxon, l'hypothèse nulle postule qu'aucune différence notable n'existe entre les performances des deux systèmes. Une valeur-p extrêmement faible, comme celle calculée (4.92×10^{-17}), présente une preuve solide à l'encontre de cette hypothèse nulle (H_0), renforçant ainsi la validité de notre hypothèse alternative (H_a).

4.7 Conclusion

La présente étude constitue une avancée significative dans le domaine de la Correction Orthographique Automatique (COA), notamment pour la langue Wolof (Wol), qui présente de grandes contraintes en termes de ressources disponibles. Notre modèle, qui utilise une architecture basée sur les transformateurs, a produit des résultats encourageants sur plusieurs métriques d'évaluation, y compris BLEU, WER, CER, ChrF++, et ACW. Ces résultats mettent en évidence le potentiel des techniques avancées d'apprentissage profond pour relever quelques défis spécifiques posés par les langues à ressources limitées.

En outre, l'utilisation de l'architecture Transformer et surtout du mécanisme de self-attention in-

hérent à cette architecture a permis de résoudre le problème posé par notre précédent outil d'autocorrection basé sur les approches traditionnelles. Grâce à cette dernière, notre modèle a pu capturer quelques informations contextuelles lors du processus de correction.

Par ailleurs, il est essentiel de souligner la comparaison faite entre notre système neuronal et un système de COA basé sur des règles. L'évaluation manuelle minutieuse nous a démontré la supériorité de notre approche neuronale. Cette observation a été renforcée par le test de rang signé de Wilcoxon, rejetant l'hypothèse nulle avec une confiance notable. Cela indique que la différence de performance entre le système neuronal et celui basé sur des règles est significative.

Malgré ces résultats prometteurs, notre travail a également révélé certains axes qui pourraient bénéficier d'améliorations pour accroître davantage les performances du système proposé.Notre modèle présente certaines limites comme la complexité de calcul, la difficulté à capter les dépendances à longue distance, et la sensibilité au bruit dans les données d'entrée. De plus, l'absence de compréhension au niveau des mots pourrait conduire à des difficultés potentielles avec les langues qui dépendent fortement de l'ordre des mots ou font face à des défis de désambiguïsation.

CONCLUSION

En conclusion, ce projet de recherche a fait des avancées significatives dans le domaine du Traitement Automatique du Langage Naturel (TALN) pour les langues à ressources limitées, plus particulièrement le Wolof (Wol). Les travaux décrits dans ce mémoire ont impliqué l'utilisation à la fois de techniques traditionnelles et des techniques les plus récentes du domaine de l'IA, dont celles issues de l'apprentissage profond. Ces approches ont permis de concevoir et de développer des architectures innovatrices et efficaces, en liaison avec les systèmes de vérification et de Correction Orthographique Automatique (COA).

Malgré les défis inhérents à l'absence de données et à la complexité morphosyntaxique du Wol, nous avons démontré la possibilité de développer des systèmes robustes de vérification et de COA pour cette langue. Le principal obstacle rencontré, l'indisponibilité des données pour la langue Wol, a été contourné par la création de corpus inédits spécifiquement pour ce travail de recherche. La complexité morphosyntaxique de la langue a ajouté un défi supplémentaire mais non moins intéressant, nous incitant à explorer de nouvelles frontières de ce qui est actuellement possible en TALN.

Pour les recherches à venir, nous envisageons d'intégrer un module d'analyse morphologique dans notre premier système de correction. Cela nous permettrait de mieux prendre en compte les traits morphologiques et syntaxiques distinctifs du Wolof dans la correction orthographique.
Par ailleurs, nous prévoyons d'explorer le Reinforcement Learning from Human Feedback (RLHF) pour améliorer notre modèle neuronal d'autocorrection. Le RLHF, qui utilise l'évaluation humaine pour guider l'entraînement des modèles neuronaux, pourrait offrir un moyen efficace d'affiner notre système en le rendant plus sensible aux nuances et aux exceptions du Wol.
De surcroît, nous comptons fusionner notre système de COA basé sur les règles avec notre modèle neuronal. Un tel système hybride pourrait tirer parti des forces de chaque méthode, apportant ainsi plusieurs bénéfices. Entre autres, une précision accrue, une rapidité améliorée et une détection plus large d'erreurs seraient envisageables. De plus, cela renforcerait l'aptitude du système à s'ajuster aux nuances linguistiques tout en conservant une explicabilité pertinente pour les

utilisateurs.

Par cette recherche, nous espérons non seulement améliorer les systèmes de TALN pour le Wol, mais aussi contribuer à une meilleure préservation et valorisation de ce riche patrimoine linguistique africain. En développant des outils capables de gérer les langues à ressources limitées, nous favorisons l'intégration de ces langues dans l'ère numérique, garantissant ainsi leur pérennité et leur évolution continue pour les générations futures.

ANNEXE A

PUBLICATIONS

Liste des publications scientifiques qui rentrent dans le cadre des travaux de recherche de ce mémoire :

Thierno Ibrahima Cissé et Fatiha Sadat. 2023. Automatic Spell Checker and Correction for Underrepresented Spoken Languages : Case Study on Wolof. Dans *Proceedings of the Fourth workshop on Resources for African Indigenous Languages (RAIL 2023)*, *pages 1–10*, *Dubrovnik*, *Croatia*. *Association for Computational Linguistics*.

Thierno Ibrahima Cissé et Fatiha Sadat. 2023. Advancing Language Diversity and Inclusion : Towards a Neural Network-based Spell Checker and Correction for Wolof. Dans *2024 Joint International Conference on Computational Linguistics*, *Language Resources and Evaluation*. **(soumis)**

RÉFÉRENCES

Abbott, J. et Martinus, L. (2019). Benchmarking neural machine translation for Southern African languages. Dans *Proceedings of the 2019 Workshop on Widening NLP*, 98–101., Florence, Italy. Association for Computational Linguistics.

Abdulrahman, R. et Hassani, H. (2022). A language model for spell checking of educational texts in Kurdish (Sorani). Dans *Proceedings of the 1st Annual Meeting of the ELRA/ISCA Special Interest Group on under-Resourced Languages*, 189–198., Marseille, France. European Language Resources Association.

Adelani, D., Alabi, J., Fan, A., Kreutzer, J., Shen, X., Reid, M., Ruiter, D., Klakow, D., Nabende, P., Chang, E., Gwadabe, T., Sackey, F., Dossou, B. F. P., Emezue, C., Leong, C., Beukman, M., Muhammad, S., Jarso, G., Yousuf, O., Niyongabo Rubungo, A., Hacheme, G., Wairagala, E. P., Nasir, M. U., Ajibade, B., Ajayi, T., Gitau, Y., Abbott, J., Ahmed, M., Ochieng, M., Aremu, A., Ogayo, P., Mukiibi, J., Ouoba Kabore, F., Kalipe, G., Mbaye, D., Tapo, A. A., Memdjokam Koagne, V., Munkoh-Buabeng, E., Wagner, V., Abdulmumin, I., Awokoya, A., Buzaaba, H., Sibanda, B., Bukula, A. et Manthalu, S. (2022). A few thousand translations go a long way ! Leveraging pre-trained models for African news translation. Dans *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, 3053–3070., Seattle, United States. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/2022.naacl-main.223

Adel'son-Vel'skii, G. M. et Landis, E. M. (1962). An algorithm for organization of information. *Doklady Akademii Nauk SSSR*, *146*(2), 263–266.

Al-Ghamdi, S., Al-Khalifa, H. et Al-Salman, A. (2023). Fine-tuning BERT-Based pre-trained models for arabic dependency parsing. *Applied Sciences*, *13*(7). http://dx.doi.org/10.3390/app13074225

Almudevar, A. (2001). A dynamic programming algorithm for the optimal control of piecewise deterministic markov processes. *SIAM Journal on Control and Optimization*, *40*(2), 525–539. http://dx.doi.org/10.1137/S0363012999364474

Amir, A., Turpin, A. et Moffat, A. (dir.) (2008). *String Processing and Information Retrieval : 15th International Symposium, SPIRE 2008, Melbourne, Australia, November 10-12, 2008 : Proceedings*. Numéro 5280 de Lecture Notes in Computer Science. Berlin ; New York : Springer.

Angell, R. C., Freund, G. E. et Willett, P. (1983). Automatic spelling correction using a trigram similarity measure. *Information Processing & Management*, *19*(4), 255–261. http://dx.doi.org/10.1016/0306-4573(83)90022-5

Araabi, A. et Monz, C. (2020). Optimizing Transformer for Low-Resource Neural Machine Translation. Dans *Proceedings of the 28th International Conference on Computational*

Linguistics, 3429–3435., Barcelona, Spain (Online). International Committee on Computational Linguistics.
http://dx.doi.org/10.18653/v1/2020.coling-main.304

Arbib, M. A. (1969). *Theories of Abstract Automata*. Prentice-Hall Series in Automatic Computation. Englewood Cliffs, N.J : Prentice-Hall.

Armstrong, S., Russell, G., Petitpierre, D. et Robert, G. (1995). An open architecture for multilingual text processing. Dans *From Texts to Tags : Issues in Multilingual Language Analysis. Proceedings of the ACL Sigdat Workshop.*, 30–34., Dublin.

Baba, Y. et Suzuki, H. (2012). How are spelling errors generated and corrected ? A study of corrected and uncorrected spelling errors using keystroke logs. Dans *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2 : Short Papers)*, 373–377., Jeju Island, Korea. Association for Computational Linguistics.

Bahdanau, D., Cho, K. et Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. Dans Y. Bengio et Y. LeCun (dir.). *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Bayer, R. et McCreight, E. (1970). Organization and maintenance of large ordered indices. Dans *Proceedings of the 1970 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control - SIGFIDET '70*, p. 107., Houston, Texas. ACM Press.
http://dx.doi.org/10.1145/1734663.1734671

Bellman, R. (1957). *Dynamic Programming* (1 éd.). Princeton, NJ, USA : Princeton University Press.

Bengio, Y., Ducharme, R., Vincent, P. et Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, *3*(null), 1137–1155.

Berment, V. (2004). *Méthodes Pour Informatiser Les Langues et Les Groupes de Langues " Peu Dotées "*. (Theses). Université Joseph-Fourier - Grenoble I.

Biljon, E. V., Pretorius, A. et Kreutzer, J. (2020). On optimal transformer depth for low-resource language translation. *CoRR*, *abs/2004.04418*.

Bledsoe, W. W. et Browning, I. (1959). Pattern recognition and reading by machine. Dans *Papers Presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference on - IRE-AIEE-ACM '59 (Eastern)*, 225–232., Boston, Massachusetts. ACM Press.
http://dx.doi.org/10.1145/1460299.1460326

Bloom, B. H. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, *13*(7), 422–426. http://dx.doi.org/10.1145/362686.362692

Boago Okgetheng, G. Malema, Ariq Ahmer, Boemo Lenyibi et Ontiretse Ishmael (2022). Bantu Spell Checker and Corrector using Modified Edit Distance Algorithm (MEDA). *Data Science and Machine Learning*. http://dx.doi.org/10.5121/csit.2022.121524

Boilat, D. (1858). *Grammaire de La Langue Woloffe*. Paris : Imprimerie impériale.

Bojar, O. et Tamchyna, A. (2011). Improving translation model by monolingual data. Dans *Proceedings of the Sixth Workshop on Statistical Machine Translation*, 330–336., Edinburgh, Scotland. Association for Computational Linguistics.

Cai, F. et De Rijke, M. (2016). A Survey of Query Auto Completion in Information Retrieval. *Foundations and Trends® in Information Retrieval*, *10*(4), 273–363. http://dx.doi.org/10.1561/1500000055

Castaño, A., Casacuberta, F. et Vidal, E. (07 23-25 1997). Machine translation using neural networks and finite-state models. Dans *Proceedings of the 7th Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, St John's College, Santa Fe. Not mentionned on TOC.

Cheikh M. Bamba Dione, Alla Lo, Elhadji Mamadou Nguer et Silèye O. Ba (2022). Low-resource Neural Machine Translation : Benchmarking State-of-the-art Transformer for Wolof<->French. *International Conference on Language Resources and Evaluation*.

Chung, J., Gulcehre, C., Cho, K. et Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. http://dx.doi.org/10.48550/ARXIV.1412.3555

Church, E. (1981). *Le Système Verbal Du Wolof*. Rapport technique, Université de Dakar, Dakar.

Cissé, M. (2004). *Dictionnaire Francais-Wolof* (2.éd. révisée et augmentée éd.). Dictionnaires des Langues O. Paris : Langues et MOndes, L'Asiatheque.

Cissé, T. I. et Sadat, F. (2023). Automatic spell checker and correction for under-represented spoken languages : Case study on Wolof. Dans *Proceedings of the Fourth Workshop on Resources for African Indigenous Languages (RAIL 2023)*, 1–10., Dubrovnik, Croatia. Association for Computational Linguistics.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L. et Stoyanov, V. (2020). Unsupervised cross-lingual representation learning at scale. Dans *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 8440–8451., Online. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/2020.acl-main.747

Connelly, R. H. et Morris, F. L. (1995). A generalization of the trie data structure. *Mathematical Structures in Computer Science*, *5*(3), 381–418. http://dx.doi.org/10.1017/S0960129500000803

Culberson, J. (1989). Explaining the Behaviour of Binary Search Trees Under Prolonged Updates : A Model and Simulations. *The Computer Journal*, *32*(1), 68–75. http://dx.doi.org/10.1093/comjnl/32.1.68

Dalrymple, M., Liakata, M. et Mackie, L. (2006). Tokenization and morphological analysis for Malagasy. Dans *International Journal of Computational Linguistics & Chinese Language Processing, Volume 11, Number 4, December 2006*, 315–332.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, *7*(3), 171–176. http://dx.doi.org/10.1145/363958.363994

Devlin, J., Chang, M.-W., Lee, K. et Toutanova, K. (2019). BERT : Pre-training of deep bidirectional transformers for language understanding. Dans *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies, Volume 1 (Long and Short Papers)*, 4171–4186., Minneapolis, Minnesota. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/N19-1423

Diagne, P. (1971). *Grammaire de Wolof Moderne*. Paris : Presence Africaine.

Diagne, P. (1997). *Al Xuraan ci Wolof*. Paris : [Dakar] : Harmattan ; Sankoré.

Diallo, H., Corenthin, A. et Lishou, C. (2012). Formalization of the Wolof with NooJ : Implementation on the Wolof dictionary. *International Journal of Computer Engineering Science*, *2*(4).

Dialo, A. (1981). *Structures Verbales Du Wolof Contemporain*. Dakar : Centre de Linguistique Appliquée de Dakar.

Dione, C. M. B. (2012). A Morphological Analyzer For Wolof Using Finite-State Techniques. Dans *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, 894–901., Istanbul, Turkey. European Language Resources Association (ELRA).

Diouf, I., Tidiane Ndiaye, C. et Binta Dieme, N. (2017). Dynamique et transmission linguistique au Sénégal au cours des 25 dernières années. *Cahiers québécois de démographie*, *46*(2), 197–217. http://dx.doi.org/10.7202/1054052ar

Diouf, J. L. et Kenkyūjo, T. G. D. A. A. G. B. (2001). *Dictionnaire wolof : wolof-français, français-wolof*. Tokyo : Institute for the Study of Languages and Cultures of Asia and Africa (ILCAA), Tokyo University of Foreign Studies.

Do, T. N. D. (2011). *Extraction de Corpus Parallèle Pour La Traduction Automatique Depuis et Vers Une Langue Peu Dotée*. (Theses). Université de Grenoble ; Université de Hanoi - Vietnam.

Doneux, J. L. (1978). Les liens historiques entre les langues du Sénégal. *Réalités africaines et langues française : Bulletin du Centre de la Linguistique Appliquée de Dakar*, *7*, 6–55.

Duh, K., McNamee, P., Post, M. et Thompson, B. (2020). Benchmarking neural and statistical machine translation on low-resource African languages. Dans *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 2667–2675., Marseille, France. European Language Resources Association.

Dunigan, M. B. (1994). *On the Clausal Architecture of Wolof*. (Thèse de doctorat). University of North Carolina, Chapel Hill.

Eberhard, D., Simons, G. et Fennig, C. (2019). *Ethnologue : Languages of the World* (22nd edition éd.). Dallas, Texas : SIL International.

El-Kassas, W. S., Salama, C. R., Rafea, A. A. et Mohamed, H. K. (2021). Automatic text summarization : A comprehensive survey. *Expert Systems with Applications*, *165*, 113679. http://dx.doi.org/10.1016/j.eswa.2020.113679

Elhadji Mamadou Nguer, Nguer, E. M., Nguer, E. M., Lo, A., Dione, M. B., Ba, S. O., Ba, S., Sileye O. Ba et Lo, M. (2020). SENCORPUS : A French-Wolof Parallel Corpus. *International Conference on Language Resources and Evaluation*, 2803–2811.

Enguehard, C. et Mbodj, C. (2004). Des correcteurs orthographiques pour les langues africaines. *Bulletin de linguistique appliquée et générale*, 51–68.

Evershed, J. et Fitch, K. (2014). Correcting Noisy OCR : Context Beats Confusion. Dans *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, DATeCH '14, 45–51., New York, NY, USA. Association for Computing Machinery. http://dx.doi.org/10.1145/2595188.2595200

Fal, A., Santos, R. et Doneux, J. (1990). *Dictionnaire Wolof-Français : Suivi d'un Index Français-Wolof*. 22-24, boulevard Arago, 75013 Paris : Karthala.

Faulk, R. D. (1964). An inductive approach to language translation. *Communications of the ACM*, *7*(11), 647–653. http://dx.doi.org/10.1145/364984.365067

Feng, J., Wang, J. et Li, G. (2012). Trie-join : A trie-based method for efficient string similarity joins. *The VLDB Journal*, *21*(4), 437–461. http://dx.doi.org/10.1007/s00778-011-0252-8

Fredkin, E. (1960). Trie memory. *Communications of the ACM*, *3*(9), 490–499. http://dx.doi.org/10.1145/367390.367400

Frost, R. A. et Peterson, M. M. (1982). A Short Note on Binary Search Trees. *The Computer Journal*, *25*(1), 158–158. http://dx.doi.org/10.1093/comjnl/25.1.158

Gal, Y. et Ghahramani, Z. (2016). A theoretically grounded application of dropout in recurrent neural networks. Dans *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, 1027–1035., Red Hook, NY, USA. Curran Associates Inc.

Garg, S., Peitz, S., Nallasamy, U. et Paulik, M. (2019). Jointly learning to align and translate with transformer models. Dans *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 4453–4462., Hong Kong, China. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/D19-1453

Gates, A. I. (1937). *Spelling Difficulties in 3867 Words*. New York : Bureau of Publications, Teachers College, Columbia University.

Gauthier, E., Besacier, L., Voisin, S., Melese, M. et Elingui, U. P. (2016). Collecting Resources in Sub-Saharan African Languages for Automatic Speech Recognition : A Case Study of Wolof. *International Conference on Language Resources and Evaluation*, 3863–3867.

Gayo, H. et Widodo, P. (2018). An Analysis of Morphological and Syntactical Errors on the English Writing of Junior High School Indonesian Students. *International Journal of Learning, Teaching and Educational Research*, *17*(4), 58–70. http://dx.doi.org/10.26803/ijlter.17.4.4

Gerdjikov, S., Mitankin, P. et Nenchev, V. (2013). Realization of common statistical methods in computational linguistics with functional automata. Dans *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, 294–301., Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

Glorot, X. et Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. Dans Y. W. Teh et D. M. Titterington (dir.). *AISTATS*, volume 9 de *JMLR Proceedings*, 249–256. JMLR.org.

Goodfellow, I., Bengio, Y. et Courville, A. (2016). *Deep Learning*. Adaptive Computation and Machine Learning. Cambridge, Massachusetts : The MIT Press.

Gowri, S., Sathish Kumar, P., Geetha Rani, K., Surendran, R. et Jabez, J. (2022). Usage of a binary integrated spell check algorithm for an upgraded search engine optimization. *Measurement : Sensors*, *24*, 100451. http://dx.doi.org/10.1016/j.measen.2022.100451

Goyal, N., Gao, C., Chaudhary, V., Chen, P.-J., Wenzek, G., Ju, D., Krishnan, S., Ranzato, M., Guzmán, F. et Fan, A. (2022). The Flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, *10*, 522–538. http://dx.doi.org/10.1162/tacl_a_00474

Grundkiewicz, R., Junczys-Dowmunt, M. et Gillian, E. (2015). Human Evaluation of Grammatical Error Correction Systems. Dans *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 461–470., Lisbon, Portugal. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/D15-1052

Gusfield, D. (1997). Algorithms on strings, trees, and sequences : Computer science and computational biology. *SIGACT News*, *28*(4), 41–60. http://dx.doi.org/10.1145/270563.571472

HaCohen-Kerner, Y., Miller, D. et Yigal, Y. (2020). The influence of preprocessing on text classification using a bag-of-words representation. *PloS one*, *15*(5), e0232525. http://dx.doi.org/10.1371/journal.pone.0232525

Hall, P. A. V. et Dowling, G. R. (1980). Approximate String Matching. *ACM Computing Surveys*, *12*(4), 381–402. http://dx.doi.org/10.1145/356827.356830

Henisz-Dostert, B., Macdonald, R. R. et Zarechnak, M. (2011). *Machine Translation*. Berlin, New York : De Gruyter Mouton. http://dx.doi.org/doi:10.1515/9783110816679

Hládek, D., Pleva, M., Staš, J. et Liao, Y.-F. (2019). Sequence to sequence convolutional neural network for automatic spelling correction. Dans *Proceedings of the 31st Conference on Computational Linguistics and Speech Processing (ROCLING 2019)*, 102–111., New Taipei City, Taiwan. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

Hochreiter, S. et Schmidhuber, J. (1996). LSTM can solve hard long time lag problems. Dans M. Mozer, M. Jordan, et T. Petsche (dir.). *Advances in Neural Information Processing Systems*, volume 9. MIT Press.

Hopcroft, J. E. et Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Series in Computer Science. Reading, Mass : Addison-Wesley.

Hughes, B., Baldwin, T., Bird, S., Nicholson, J. et MacKinlay, A. (2006). Reconsidering language identification for written language resources. Dans *5th International Conference on Language Resources and Evaluation, LREC 2006*, 485–488., Genoa, Italy. European Language Resources Association (ELRA).

Jiao, Q. et Zhang, S. (2021). A Brief Survey of Word Embedding and Its Recent Development. Dans *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 1697–1701., Chongqing, China. IEEE. http://dx.doi.org/10.1109/IAEAC50856.2021.9390956

Ka, O. (1981). *La Derivation et La Composition En Wolof*, volume 77 de *Les Langues Nationales Au Senegal*. Dakar : Centre de Linguistique Appliquée de Dakar.

Kalchbrenner, N. et Blunsom, P. (2013). Recurrent continuous translation models. Dans *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1700–1709., Seattle, Washington, USA. Association for Computational Linguistics.

Kernighan, M. D., Church, K. W. et Gale, W. A. (1990). A spelling correction program based on a noisy channel model. Dans *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING '90, 205–210., USA. Association for Computational Linguistics. http://dx.doi.org/10.3115/997939.997975

King, M. (1999). Evaluation design : The EAGLES framework. Dans *Proceedings of the Konvens '98 : Evaluation of the Linguistic Performance of Machine Translation Systems*, Bonn, Germany. Rita Nübel, Uta Seewald-Heeg, Gardezi Verlag, St. Augustin.

Kingma, D. P. et Ba, J. (2015). Adam : A method for stochastic optimization. Dans Y. Bengio et Y. LeCun (dir.). *3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA.

Knuth, D. (1998). *Art of computer programming, the : Volume 3 : Sorting and searching* (2nd edition. éd.). Addison-Wesley Professional.

Kukich, K. (1992). Techniques for automatically correcting words in text. *Acm Computing Surveys*, *24*(4), 377–439. http://dx.doi.org/10.1145/146370.146380

Lample, G., Denoyer, L. et Ranzato, M. (2017). Unsupervised machine translation using monolingual corpora only. *CoRR*, *abs/1711.00043*.

LeCun, Y., Bengio, Y. et Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. http://dx.doi.org/10.1038/nature14539

Lecun, Y., Bottou, L., Bengio, Y. et Haffner, P. (Nov./1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. http://dx.doi.org/10.1109/5.726791

Lee, J., Cho, K. et Hofmann, T. (2017). Fully Character-Level Neural Machine Translation without Explicit Segmentation. *Transactions of the Association for Computational Linguistics*, *5*, 365–378. http://dx.doi.org/10.1162/tacl_a_00067

Levenshtein, V. I. (1965). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, *10*, 707–710.

Linz, P. (2006). *An Introduction to Formal Languages and Automata* (4th ed éd.). Sudbury, Mass : Jones and Bartlett Publishers.

Lo, A., Dione, C. B., Mangeot, M., Khoule, M., Bao-Diop, S., Cissé, M.-T. *et al.* (2016). Correction orthographique pour la langue wolof : état de l'art et perspectives. Dans *JEP-TALN-RECITAL 2016 : Traitement Automatique Des Langues Africaines TALAF 2016*.

Lo, A., Dione, C. M. B., Nguer, E. M., Ba, S. O. et Lo, M. (2020). Using LSTM to translate french to senegalese local languages : Wolof as a case study. *CoRR*, *abs/2004.13840*.

Luong, T., Pham, H. et Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. Dans *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1412–1421., Lisbon, Portugal. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/D15-1166

Madnani, N., Heilman, M., Tetreault, J. et Chodorow, M. (2012). Identifying high-level organizational elements in argumentative discourse. Dans *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, 20–28., Montréal, Canada. Association for Computational Linguistics.

Mangold, M. (1977). *Wolof Pronoun Verb Patterns and Paradigms*. Numéro Bd. 3 de Forschungen Zur Anthropologie Und Religionsgeschichte. Saarbrücken : Homo et Religio.

Marsland, S. (2015). *Machine Learning : An Algorithmic Perspective* (second edition éd.). Chapman & Hall/CRC Machine Learning & Pattern Recognition Series. Boca Raton : CRC Press.

McCulloch, W. S. et Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133. http://dx.doi.org/10.1007/BF02478259

Meister, C., Vieira, T. et Cotterell, R. (2020). Best-First Beam Search. *Transactions of the Association for Computational Linguistics*, *8*, 795–809. http://dx.doi.org/10.1162/tacl_a_00346

M'eric, J.-J. (2014). Un vérificateur orthographique pour la langue bambara. *TALAf@TALN*, 141–146.

Merrill, J. T. M. (2021). The evolution of consonant mutation and noun class marking in Wolof. *Diachronica*, *38*(1), 64–110. http://dx.doi.org/10.1075/dia.18046.mer

Mikolov, T., Chen, K., Corrado, G. et Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. http://dx.doi.org/10.48550/ARXIV.1301.3781

Mitankin, P., Gerdjikov, S. et Mihov, S. (2014). An Approach to Unsupervised Historical Text Normalisation. Dans *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, DATeCH '14, 29–34., New York, NY, USA. Association for Computing Machinery. http://dx.doi.org/10.1145/2595188.2595191

Mitton, R. (1996). *English Spelling and the Computer*. Studies in Language and Linguistics. London ; New York : Longman.

Mosavi Miangah, T. (2013). FarsiSpell : A spell-checking system for Persian using a large monolingual corpus. *Literary and Linguistic Computing*, *29*(1), 56–73. http://dx.doi.org/10.1093/llc/fqt008

Muth, F. E. et Tharp, A. L. (1977). Correcting human error in alphanumeric terminal input. *Information Processing & Management*, *13*(6), 329–337. http://dx.doi.org/10.1016/0306-4573(77)90053-X

Nekoto, W., Marivate, V., Matsila, T., Fasubaa, T., Fagbohungbe, T., Akinola, S. O., Muhammad, S., Kabongo Kabenamualu, S., Osei, S., Sackey, F., Niyongabo, R. A., Macharm, R., Ogayo, P., Ahia, O., Berhe, M. M., Adeyemi, M., Mokgesi-Selinga, M., Okegbemi, L., Martinus, L., Tajudeen, K., Degila, K., Ogueji, K., Siminyu, K., Kreutzer, J., Webster, J., Ali, J. T., Abbott, J., Orife, I., Ezeani, I., Dangana, I. A., Kamper, H., Elsahar, H., Duru, G., Kioko, G., Espoir, M., van Biljon, E., Whitenack, D., Onyefuluchi, C., Emezue, C. C., Dossou, B. F. P., Sibanda, B., Bassey, B., Olabiyi, A., Ramkilowan, A., Öktem, A., Akinfaderin, A. et Bashir, A. (2020). Participatory research for low-resourced machine translation : A case study in African languages. Dans *Findings of the Association for Computational Linguistics : EMNLP 2020*, 2144–2160., Online. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/2020.findings-emnlp.195

Nguer, E. H. M., Khoule, M., Thiam, M. N., Thiam, M. B., Thiare, O., Cissé, M.-T. et Mangeot, M. (2015). Dictionnaires wolof en ligne : état de l'art et perspectives. Dans *Colloque National Sur La Recherche En Informatique et Ses Applications*, Thiès, Senegal.

Njie, C. M. (1982). *Description Syntaxique Du Wolof de Gambie*. Dakar : Les Nouvelles Editions Africaines.

Ott, M., Edunov, S., Grangier, D. et Auli, M. (2018). Scaling neural machine translation. Dans *Proceedings of the Third Conference on Machine Translation : Research Papers*, 1–9., Brussels, Belgium. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/W18-6301

Paggio, P. et Music, B. (1998). Evaluation in the SCARRIE project. *International Conference on Language Resources and Evaluation*, *Granada*, *Spain*, 277–282.

Paggio, P. et Underwood, N. L. (1998). Validating the TEMAA LE evaluation methodology : A case study on Danish spelling checkers. *Natural Language Engineering*, *4*(3), 211–228. http://dx.doi.org/10.1017/s1351324998001995

Papineni, K., Roukos, S., Ward, T. et Zhu, W.-J. (2002). Bleu : A method for automatic evaluation of machine translation. Dans *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318., Philadelphia, Pennsylvania, USA. Association for Computational Linguistics. http://dx.doi.org/10.3115/1073083.1073135

Pennington, J., Socher, R. et Manning, C. (2014). Glove : Global Vectors for Word Representation. Dans *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543., Doha, Qatar. Association for Computational Linguistics. http://dx.doi.org/10.3115/v1/D14-1162

Peterson, J. L. (1980). Computer programs for detecting and correcting spelling errors. *Communications of The Acm*, *23*(12), 676–687. http://dx.doi.org/10.1145/359038.359041

Pires, T., Schlinger, E. et Garrette, D. (2019). How multilingual is multilingual BERT ? Dans *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4996–5001., Florence, Italy. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/P19-1493

Pirinen, F. A. et Lindén, K. (2014). State-of-the-art in weighted finite-state spell-checking. Dans *Conference on Intelligent Text Processing and Computational Linguistics*.

Popović, M. (2015). chrF : Character n-gram F-score for automatic MT evaluation. Dans *Proceedings of the Tenth Workshop on Statistical Machine Translation*, 392–395., Lisbon, Portugal. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/W15-3049

Popović, M. (2017). chrF++ : Words helping character n-grams. Dans *Proceedings of the Second Conference on Machine Translation*, 612–618., Copenhagen, Denmark. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/W17-4770

Popović, M. et Ney, H. (2007). Word error rates : Decomposition over POS classes and applications for error analysis. Dans *Proceedings of the Second Workshop on Statistical*

Machine Translation, 48–55., Prague, Czech Republic. Association for Computational Linguistics.

Prechelt, L. (1996). Early Stopping-But When ? In G. B. Orr et K.-R. Müller (dir.), *Neural Networks : Tricks of the Trade*, volume 1524 de *Lecture Notes in Computer Science* 55–69. Springer.

Press, O. et Wolf, L. (2017). Using the output embedding to improve language models. Dans *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics : Volume 2, Short Papers*, 157–163., Valencia, Spain. Association for Computational Linguistics.

Radford, A. et Narasimhan, K. (2018). Improving language understanding by generative pre-training.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. et Liu, P. J. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, *21*(140), 1–67.

Rahimi, Z. et Homayounpour, M. M. (2022). The impact of preprocessing on word embedding quality : A comparative study. *Language Resources and Evaluation*. http://dx.doi.org/10.1007/s10579-022-09620-5

Rahman, M. M., Mahmud, H., Rupa, R. S. et Rinty, M. R. (2021). A robust bangla spell checker for search engine. Dans *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, 1329–1334. http://dx.doi.org/10.1109/ICCES51350.2021.9489146

Reid, M., Hu, J., Neubig, G. et Matsuo, Y. (2021). AfroMT : Pretraining strategies and reproducible benchmarks for translation of 8 African languages. Dans *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 1306–1320., Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/2021.emnlp-main.99

Robert, S. (2011). Le wolof. *Bulletin de la Société de Linguistique de Paris*, *81*. http://dx.doi.org/10.2143/BSL.81.1.2013699

Rosenblatt, F. (1958). The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. http://dx.doi.org/10.1037/h0042519

Rumelhart, D. E., Hinton, G. E. et Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Vol. 1 : Foundations* 318–362. Cambridge, MA, USA : MIT Press.

Salifou, L. et Naroua, H. (2014). Design and Implementation of a Spell Checker for Hausa Language (’Etude et conception d’un correcteur orthographique pour la langue haoussa) [in French]. *TALAf@TALN*, 147–158.

Sapir, J. D. (1971). West Atlantic : An inventory of the languages, their noun class systems and consonant alternation. In T. A. Sebeok (dir.), *Current Trends in Linguistics, 7 : Linguistics in Sub-Saharan Africa*, numéro 7 de Current Trends in Linguistics 7 (Ed. by T. Sebeok) 45–112. The Hague & Paris : Mouton & Co.

Sariev, A., Nenchev, V., Gerdjikov, S., Mitankin, P., Ganchev, H., Mihov, S. et Tinchev, T. (2014). Flexible Noisy Text Correction. Dans *2014 11th IAPR International Workshop on Document Analysis Systems*, 31–35., Tours, France. IEEE. http://dx.doi.org/10.1109/DAS.2014.12

See, A., Liu, P. J. et Manning, C. D. (2017). Get to the point : Summarization with pointer-generator networks. *CoRR*, *abs/1704.04368*.

Segerer, G. et Pozdniakov, K. (2017). A genealogical classification of atlantic languages. In F. Luepke (dir.), *The Oxford Guide to the Atlantic Languages of West Africa*, The Oxford Guide to the Atlantic Languages of West Africa. Oxford University Press.

Sennrich, R. (2012). Perplexity Minimization for Translation Model Domain Adaptation in Statistical Machine Translation. Dans *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, 539–549., Avignon, France. Association for Computational Linguistics.

Sennrich, R., Haddow, B. et Birch, A. (2016a). Improving Neural Machine Translation Models with Monolingual Data. Dans *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, 86–96., Berlin, Germany. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/P16-1009

Sennrich, R., Haddow, B. et Birch, A. (2016b). Neural machine translation of rare words with subword units. Dans *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 : Long Papers)*, 1715–1725., Berlin, Germany. Association for Computational Linguistics. http://dx.doi.org/10.18653/v1/P16-1162

Shiv, V. et Quirk, C. (2019). Novel positional encodings to enable tree-based transformers. Dans H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, et R. Garnett (dir.). *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Sidorov, A. A. (1979). Analysis of word similarity on spelling correction systems. *Programming and Computer Software*, *5*(4), 274–277.

Sleator, D. D. et Tarjan, R. E. (1985). Self-adjusting binary search trees. *Journal of the ACM*, *32*(3), 652–686. http://dx.doi.org/10.1145/3828.3835

Smith, L. N. (2017). Cyclical learning rates for training neural networks. Dans *2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017, Santa Rosa, CA, USA, March 24-31, 2017*, 464–472. IEEE Computer Society. http://dx.doi.org/10.1109/WACV.2017.58

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. et Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. Dans *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1631–1642., Seattle, Washington, USA. Association for Computational Linguistics.

Soricut, R. et Och, F. (2015). Unsupervised morphology induction using word embeddings. Dans *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, 1627–1637., Denver, Colorado. Association for Computational Linguistics. http://dx.doi.org/10.3115/v1/N15-1186

Sorokin, A., Baytin, A., Galinskaya, I., Rykunova, E. et Shavrina, T. (2016). SpellRuEval : The first competition on automatic spelling correction for Russian. Dans *Proceedings of the Annual International Conference "Dialogue"*, volume 15, Moscow, Russia.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. et Salakhutdinov, R. (2014). Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Starlander, M. et Popescu-Belis, A. (2002). Corpus-based evaluation of a French spelling and grammar checker. Dans *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, Las Palmas, Canary Islands - Spain. European Language Resources Association (ELRA).

Stolcke, A. (2000). Entropy-based pruning of backoff language models. *CoRR*, *cs.CL/0006025*.

Streiner, D. L. et Norman, G. R. (2006). "Precision" and "Accuracy" : Two Terms That Are Neither. *Journal of Clinical Epidemiology*, *59*(4), 327–330. http://dx.doi.org/10.1016/j.jclinepi.2005.09.005

Sutskever, I., Vinyals, O. et Le, Q. V. (2014). Sequence to sequence learning with neural networks. Dans *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, 3104–3112., Cambridge, MA, USA. MIT Press.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. et Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. Dans *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818–2826., Las Vegas, NV, USA. IEEE. http://dx.doi.org/10.1109/CVPR.2016.308

Täckström, O., McDonald, R. et Uszkoreit, J. (2012). Cross-lingual word clusters for direct transfer of linguistic structure. Dans *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics : Human Language Technologies*, 477–487., Montréal, Canada. Association for Computational Linguistics.

Team, N., Costa-jussà, M. R., Cross, J., Çelebi, O., Elbayad, M., Heafield, K., Heffernan, K., Kalbassi, E., Lam, J., Licht, D., Maillard, J., Sun, A., Wang, S., Wenzek, G., Youngblood, A., Akula, B., Barrault, L., Gonzalez, G. M., Hansanti, P., Hoffman, J., Jarrett, S., Sadagopan, K. R., Rowe, D., Spruit, S., Tran, C., Andrews, P., Ayan, N. F., Bhosale, S., Edunov, S., Fan, A., Gao, C., Goswami, V., Guzmán, F., Koehn, P., Mourachko, A., Ropers, C., Saleem, S., Schwenk, H. et Wang, J. (2022). No Language Left Behind : Scaling Human-Centered Machine Translation.

Ten Hacken, P. et Tschichold, C. (2001). Word Manager and CALL : Structured access to the lexicon as a tool for enriching learners' vocabulary. *ReCALL*, *13*(1), 121–131. http://dx.doi.org/10.1017/S0958344001001112

Thareja, R. (2014). *Data Structures Using C* (second edition éd.). New Delhi : Oxford University Press.

Toutanova, K. et Moore, R. (2002). Pronunciation modeling for improved spelling correction. Dans *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 144–151., Philadelphia, Pennsylvania, USA. Association for Computational Linguistics. http://dx.doi.org/10.3115/1073083.1073109

Van Rijsbergen, C. J. (1979). *Information Retrieval* (2d ed éd.). London ; Boston : Butterworths.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. et Polosukhin, I. (2017). Attention is all you need. Dans I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, et R. Garnett (dir.). *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Vienney, S. (2004). *Correction Automatique : Bilan et Perspectives*. Presses universitaires de Franche-Comté.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, *13*(2), 260–269. http://dx.doi.org/10.1109/TIT.1967.1054010

Voorhees, E. et Garofolo, J. (2000). The TREC Spoken Document Retrieval Track. *Bulletin of the American Society for Information Science and Technology*, *26*(5), 18–19. http://dx.doi.org/10.1002/bult.170

Wagner, R. A. (1974). Order-*n* correction for regular languages. *Communications of the ACM*, *17*(5), 265–268. http://dx.doi.org/10.1145/360980.360995

Widrow, B. et Lehr, M. (Sept./1990). 30 years of adaptive neural networks : Perceptron, Madaline, and backpropagation. *Proceedings of the IEEE*, *78*(9), 1415–1442. http://dx.doi.org/10.1109/5.58323

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, *1*(6), 80. http://dx.doi.org/10.2307/3001968

Wilson, W. A. A. (1989). Atlantic. In J. Bendor-Samuel (dir.), *The Niger-Congo Languages : A Classification and Description of Africa's Largest Language Family* 81–104. Lanham, MD : University Press of America.

Windley, P. F. (1960). Trees, Forests and Rearranging. *The Computer Journal*, *3*(2), 84–88. http://dx.doi.org/10.1093/comjnl/3.2.84

Xu, Y., Liu, X., Cao, X., Huang, C., Liu, E., Qian, S., Liu, X., Wu, Y., Dong, F., Qiu, C.-W., Qiu, J., Hua, K., Su, W., Wu, J., Xu, H., Han, Y., Fu, C., Yin, Z., Liu, M., Roepman, R., Dietmann, S., Virta, M., Kengara, F., Zhang, Z., Zhang, L., Zhao, T., Dai, J., Yang, J., Lan, L., Luo, M., Liu, Z., An, T., Zhang, B., He, X., Cong, S., Liu, X., Zhang, W., Lewis, J. P., Tiedje, J. M., Wang, Q., An, Z., Wang, F., Zhang, L., Huang, T., Lu, C., Cai, Z., Wang, F. et Zhang, J. (2021). Artificial intelligence : A powerful paradigm for scientific research. *The Innovation*, *2*(4), 100179. http://dx.doi.org/10.1016/j.xinn.2021.100179

Yang, S., Wang, Y. et Chu, X. (2020). A Survey of Deep Learning Techniques for Neural Machine Translation.

Yannakoudakis, E. et Fawthrop, D. (1983). The rules of spelling errors. *Information Processing & Management*, *19*(2), 87–99. http://dx.doi.org/10.1016/0306-4573(83)90045-6

Yarowsky, D., Ngai, G. et Wicentowski, R. (2001). Inducing multilingual text analysis tools via robust projection across aligned corpora. Dans *Proceedings of the First International Conference on Human Language Technology Research*, HLT '01, 1–8., USA. Association for Computational Linguistics. http://dx.doi.org/10.3115/1072133.1072187

Zamora, E., Pollock, J. et Zamora, A. (1981). The use of trigram analysis for spelling error detection. *Information Processing & Management*, *17*(6), 305–316. http://dx.doi.org/10.1016/0306-4573(81)90044-3

Zhang, S., Lei, M. et Yan, Z. (2019). Automatic Spelling Correction with Transformer for CTC-based End-to-End Speech Recognition. http://dx.doi.org/10.48550/ARXIV.1904.10045