الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université 8 Mai 1945 –Guelma-

Faculté des Mathématique, d'Informatique et des Sciences de la Matière

Département d'Informatique

Mémoire de Fin d'Etudes Master

Filière : Informatique

Option : Sciences et Technologie de l'Information et de la Communication

Thème

Correction des fautes d'orthographe par mesure de similarité sémantique entre les mots

Encadré par :

Dr. Farek Lazhar

Présenté par :

Haoues Nouhed

ANNÉE UNIVERSITAIRE: 2020/2021

Dédicace

Je dédie ce travail ….

A mes chers parents :

Aucune dédicace ne saurait assez éloquente pour exprimer ce que vous méritez pour tous les sacrifices, l'amour, la tendresse, le soutien et les prières tout au long de mes études.

A mes sœurs :

Les mots ne suffisent guère pour exprimer l'attachement, l'amour et l'affection que je porte pour vous, je vous dédie ce travail pour vos encouragements permanents, et votre soutien moral.

A tous les membres de ma famille :

Petit et grand, du loin ou de pré, veuillez trouver dans ce travail l'expression de notre affection la plus sincère.

A tous mes chers amis :

En souvenir de tous ce que nous avons vécus ensemble. J'espère que notre amitié durera éternellement.

Remerciement

Tout d’abord, je remercie le Dieu, notre créateur de m’avoir donné la force, la volonté et le courage afin d’accomplir ce modeste travail.

Ces remerciements vont d’abord au corps professoral et administratif du département d’informatique pour la richesse et la qualité de leur enseignement et qui déploient de grands efforts pour assurer à leurs étudiants une formation actualisée.

Mes sincères gratitudes à Dr. Farek Lazhar pour la qualité de son enseignement, ses conseils, s’est toujours montré à l’écoute et très disponible tout au long de la réalisation de ce mémoire.

Dans l’impossibilité de citer tous les noms, nos sincères remerciements vont à tous ceux et celles, qui de près ou de loin, ont permis par leurs conseils et leurs compétences la réalisation de ce mémoire.

Mes vifs remerciements vont également aux membres du jury pour l’intérêt qu’ils ont accepté d’examiner le travail et de l’enrichir par leurs propositions.

Résumé

Le but de ce travail est de concevoir et créer une approche de vérification et de correction automatique de textes anglais basée sur le Framework de plongement lexical Word2Vec, où les méthodes existantes basées sur dictionnaires traitement les mots comme des entités lexicales isolés sans prendre en considération leurs contextes, alors que notre approche prend en compte les mots du contexte pour corriger les fautes d'orthographe en utilisant des mesures de similarité sémantique. D'après les résultats obtenus, notre approche est plus performante que l'approche basée la distance de Levenshtein.

Mots-clés

Correction automatique ; erreur ; vérification; sémantique ; similarité ; distance.

Abstract

The aim of this work is to design and create an approach for automatic correction of English texts based on Word2Vec word-embedding Framework, where the existing methods based on dictionaries treat words as isolated lexical entities without taking into account their contexts, while our approach takes into account the words of the context to correct spelling errors using semantic similarity measures. Based on the results obtained, our approach performs better than Levenshtein distance-based approach.

Keywords

Automatic correction; error; verification; semantic; similarity; distance.

Table des matières

Liste des figures

Liste des Algorithmes

Introduction Générale

Introduction Générale

Dans un monde de plus en plus interconnecté, l'échange de documents, de courriels et de messages électroniques est devenu monnaie courante, et toute faute d'orthographe et de grammaire peut devenir un obstacle à une bonne communication.
Parmi les moyens d'aide opérationnels, la correction d'orthographe est l'une des plus utiles et plus facile pour l'utilisateur qui l'utilise pour créer des textes clairs et complets. Les fautes d'orthographe peuvent rendre le texte plus difficile à lire et encore plus difficile à traiter car une orthographe ou une numérisation incorrecte du texte diminue la valeur informationnelle.

Problématique

La mesure de similarité sémantique entre les mots joue un rôle important en text-mining, et s'utilise principalement en clustering des documents, la recherche d'information et l'extraction automatique de métadonnées. Pour que l'ordinateur puisse déterminer la similarité sémantique entre les mots, il est nécessaire de comprendre la sémantique d'un mot donné. L'ordinateur est une machine lexico-syntaxique et il ne comprend pas la sémantique. Par conséquent, il a toujours essayé de représenter les mots sémantiques comme des mots syntaxiques. De nos jours, de nombreuses méthodes ont été proposées pour trouver la similarité sémantique entre les mots.

Le monde évolue, pour cela nos méthodes de travail et de traitement des données doivent évoluer aussi, nous sommes dans l'obligation de se débarrasser des méthodes classiques qui se focalisent sur des dictionnaires, nous avons pensé qu'il serait agréable de réaliser une application pour détecter et corriger les fautes d'orthographe dans un document par mesure de similarité sémantique entre les mots, en prenant en considération le contexte des mots dans la phrase.

Organisation du Mémoire

Nous avons proposé d'organiser notre mémoire comme suit :

- Dans le premier chapitre, nous allons présenter un état de l'art sur la correction des fautes d'orthographe.

- Dans le deuxième chapitre, nous expliquons les techniques utilisées pour la représentation sémantique des mots.

- Dans le troisième chapitre, nous allons parler sur la conception de l'application où nous présenterons notre approche proposée.

- Dans le quatrième chapitre, nous allons présenter l'implémentation de notre approche, nous allons présenter sous forme de captures d'écran notre application, en expliquant son principe de fonctionnement, plus les outils et les technologies utilisées.

Chapitre 01 :

Correction des fautes d'orthographe (Etat de l'art)

1.1 Introduction

La correction automatique des fautes d'orthographe est l'un des axes les plus importants dans le domaine du traitement automatique du langage naturel (TALN) et a fait l'objet de plusieurs études depuis les années 1960 (Kukich, 1992).

La correction orthographique consiste à proposer des solutions (mots lexicaux) plus proches du mot incorrect. Cela implique la nécessité de développer des techniques plus efficaces capables de corriger les erreurs en temps optimal.

1.2 Types de correcteurs d'orthographe

Il y a deux grandes catégories de correcteurs, il est nécessaire de procéder à des distinctions plus fines :

1.2.1 Correcteurs intégrés [1]

- ✓ Dans des logiciels de traitement de texte
- ✓ Dans des logiciels de PAO (Publication Assistée par Ordinateur)
- ✓ Dans des logiciels de messagerie électronique
- ✓ Dans des outils de consultation de bases de données allant de l'encyclopédie au Web, le plus souvent très rudimentaires, souvent réduits à une fonction de phonétisation
- ✓ Dans des logiciels plus spécifiques ou en cours de développement. Nous donnerons l'exemple d'un projet d'un correcteur automatique d'épreuves d'examen, lequel intégrait dans ces données, entre autres "le nombre de fautes d'orthographe commises" :

  Un professeur de psychopédagogie de l'Université Duke dit qu'il a écrit un programme qui correspond à la capacité d'un être humain à évaluer la qualité des articles en anglais, sur la base de mesures pondérées de traits tels que la longueur et la structure des phrases, la précision de l'orthographe, le choix du vocabulaire, etc. (*Atlanta Journal-Constitution*, 14 Aug 1995).

1.2.2 Correcteurs autonomes [1]

Outils très complets, liés à une application (en général à un logiciel de traitement de texte), ils associent la correction orthographique, la correction syntaxique, la correction stylistique, mais encore fort peu la correction sémantique.

1.3 Utilité de correction automatique

Le rédacteur n'a peut-être pas assez de temps ou de capacité pour corriger les fautes d'orthographe. Le système de vérification orthographique automatique (Automatic spelling correction) aide à trouver la forme de mot requise. Il identifie les mots incorrects et proposent un ensemble de mots candidats alternatifs. Les candidats sont généralement classés en fonction de leurs similarités lexicales, et les mots environnants. La meilleure correction peut être sélectionnée de manière interactive, diviser l'ensemble du processus en trois étapes :

- Détection d'une erreur.
- Génération de candidats à la correction.
- Classement des corrections des candidats.

L'importance de la relecture est possible dans un certain nombre de domaines, notamment :

- Assurer l'exactitude du contenu
- Assurer l'intégrité et la signification du contenu
- Assurer le bon formatage du contenu
- Assurer la clarté de l'œuvre et sa signification pour le lecteur

Ainsi, nous voyons que la correction automatique est d'une grande importance dans la recherche scientifique. A travers la relecture, le rédacteur garantit une recherche scientifique idéale sans erreur et claire, et ainsi le lecteur peut lire la recherche sans aucune confusion et dans un temps record et à moindre coût.

1.4 Différentes techniques et algorithmes proposés pour corriger les fautes d'orthographe [2]

Il a longtemps été difficile de trouver une solution au problème de vérification orthographique. Plusieurs chercheurs se sont penchés sur ce problème et, grâce à leurs efforts, diverses technologies et de nombreux algorithmes ont émergé. La détection d'erreur est la découverte de mots mal orthographiés dans le texte. L'application chargée de vérifier l'orthographe marque alors les mots considérés comme incorrects. Si le mot est en effet faux - parce que ce n'est pas toujours le cas - nous disons qu'une erreur a été détectée. Des recherches dans ce domaine ont été menées par de nombreux auteurs, par exemple (Enguehard et al., 2011). Les principales techniques utilisées pour identifier les mauvais mots dans le texte sont basées sur l'analyse des

n-grammes ou la recherche dans le dictionnaire (Kukich, 1992). (Peterson, 1980) a donné un algorithme de détection basé sur un dictionnaire.

La table de hachage est l'une des structures de données la plus utilisée, ce qui peut réduire le temps de réponse lors de la recherche dans le dictionnaire (Kukich, 1992).
Bien que la technique de calcul des n-grammes à partir d'un dictionnaire soit bonne, sa précision est faible par rapport aux techniques qui utilisent toutes les informations du dictionnaire. Mais cette dernière s'est avérée longue, lorsque la structure de données du dictionnaire a été mal choisie.

À ce jour, *la distance minimale d'édition* ou simplement *la distance d'édition* est la technique la plus courante pour corriger les fautes d'orthographe. Il a été appliqué à presque toutes les fonctions de vérification orthographique, y compris les éditeurs de texte et les interfaces de langage de commande. (Damerau, 1964) a proposé le premier algorithme de correction orthographique basé sur cette technologie. Presque au même moment, Levenshtein a également développé un algorithme similaire, qui semble être l'algorithme le plus couramment utilisé. Plusieurs autres algorithmes sur la distance d'édition ont été développés plus tard. Wagner définit la distance d'édition comme le nombre minimum d'opérations d'édition requises pour convertir un mot en un autre (Kukich, 1992). Ces opérations sont l'insertion, la suppression, la substitution et la transposition.

Dans la plupart des cas, la correction des fautes d'orthographe nécessite l'insertion, la suppression ou la substitution d'une seule lettre ou la transposition de deux lettres. Lorsqu'un mot incorrect est converti en mot du dictionnaire en inversant l'une de ces opérations, le mot du dictionnaire est considéré comme une correction raisonnable. Afin de réduire le temps de recherche, nous utilisons la technologie de distance d'édition inversée. Une autre façon de réduire le nombre de comparaisons est de trier ou de diviser le dictionnaire selon certaines conditions (ordre des lettres, longueur des mots, occurrence des mots). De nombreuses autres techniques sont également utilisées pour la correction d'erreurs, telles que : des clés similaires, des systèmes de règles, des techniques basées sur les n-grammes, des techniques probabilistes et des réseaux de neurones.

1.5 Conclusion

Concernant la connaissance du fonctionnement du "correcteur", même s'il s'agit de connaissances générales, les utilisateurs doivent considérer cet outil comme un outil de vérification et une aide à la correction en fonction de leurs besoins.

Dans la littérature sur ce sujet, de plus en plus les gens prévoient l'utilisation d'une nouvelle génération de correcteurs liés à la technologie de l'intelligence artificielle, qui permettraient de prendre en compte la dimension sémantique sans laquelle la vérification et la correction d’un texte ne peuvent être réellement performantes.

Chapitre 02:

Plongement Lexical (Word Embedding)

2.1 Introduction

De manière intuitive, en tant qu'être humain, il n'a pas beaucoup de sens d'utiliser des nombres pour représenter des mots ou tout autre objet dans l'univers, parce que les nombres sont utilisés pour la quantification. Pourquoi aurait-on besoin de quantifier les mots ?

De même, en dehors de la science, nous utilisons des nombres pour quantifier la qualité. Lorsque nous citons le prix d'un objet, nous essayons de quantifier sa valeur, la taille d'un vêtement, et nous essayons de quantifier la proportion du corps qui lui convient le mieux. En utilisant des nombres, nous facilitons l'analyse et la comparaison en fonction de ces qualités. Maintenant que nous savons que la représentation numérique des objets peut faciliter l'analyse en quantifiant une certaine qualité, la question est, quelle qualité des mots voulons-nous quantifier ?

La réponse est que nous devons quantifier la sémantique. Nous voulons représenter les mots d'une manière qui capture leur signification comme les humains. Pas la signification exacte du mot, mais la signification du contexte.

2.2 Méthodes de pondération

Les méthodes de pondération (eng. Weighting Scheme) visent à représenter le mot par une valeur numérique discrète (poids), Il existe plusieurs méthodes de pondération.

2.2.1 Bag-Of-Words (BOW) [3]

Un modèle de sac de mots (BOW) est un moyen d'extraire des caractéristiques du texte à utiliser dans la modélisation, comme avec les algorithmes d'apprentissage automatique, l'approche est très simple et flexible et peut être utilisée de multiples façons pour extraire des mots à partir de documents.
BOW est une représentation de texte qui décrit l'occurrence de mots dans un document. Cela implique deux choses :

- Un vocabulaire de mots connus.
- Une mesure de la présence de mots connus.

C'est ce qu'on appelle un « sac » de mots, car toute information sur l'ordre ou la structure des mots dans le document est supprimée. Le modèle se préoccupe uniquement de savoir si les mots connus apparaissent dans le document, et non à l'endroit du document.

2.2.2 La pondération TFs [5]

TFs (Term Frequency) qui consiste à calculer la fréquence d'un mot par rapport à un document. Plusieurs variations de TF existent le voir dans la Figure 2.1

Schéma de pondération	formule du TF
binaire	$0, 1$
fréquence brute	$f_{t,d}$
normalisation logarithmique	$1 + \log(f_{t,d})$
normalisation « 0.5 » par le max	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
normalisation par le max	$K + (1 - K)\frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Figure 2.1: Variantes de TF

2.2.3 La pondération TF-IDF [5]

Le TF-IDF est une métrique largement utilisée qui permet d'estimer globalement l'importance des termes contenus dans un document par rapport à une collection ou un corpus, elle est obtenue en multipliant TF par IDF.

$$\text{TF-IDF (W)} = \text{IDF (W)} * \text{TF (W)}$$

2.2.4 La pondération TFminmax [5]

La pondération TFminmax utilise une autre mesure statistique, qui est le logarithme du nombre de fois qu'un mot apparaît dans le corpus divisé par le nombre total de mots dans le corpus :

$$TF\log(W) = -\log(\frac{Cpt}{N})$$

Où Cpt et le nombre de fois qu'apparait W dans le corpus, N = nombre de mots total dans le corpus. Une fois les TFlogs de tous les mots calculés, une normalisation est appliquée à ces derniers afin d'obtenir les TFminmax en divisons le tf-log du mot par le max des tf-logs obtenus:

$$TFminmax(W) = \frac{TF\log(\mathrm{W})}{Max(TF\mathrm{logs})}$$

2.2.5 La pondération TF-IDFminmax [5]

Différent du TF-IDF de base que nous avons vu précédemment, le but de TF-IDFminmax est de faire une évaluation globale de l'importance des termes contenus dans le document (par rapport à la collection ou au corpus), tandis que le TF-IDFminmax génère une estimation globale de l'importance du mot par rapport à la langue entière. Cette estimation est obtenue par la formule :

TF-IDFminmax(W)= IDF(W) * TFminmax (W)

2.3 Méthodes de plongement lexical (Word Embedding)[5]

Le Word Embedding est l'une des représentations les plus populaires du vocabulaire documentaire. Il désigne un ensemble de techniques de machine learning qui visent à représenter les mots ou les phrases d'un texte par des vecteurs de nombres réels. Cette nouvelle représentation a ceci de particulier que les mots apparaissant dans des contextes similaires possèdent des vecteurs correspondants qui sont relativement proches.

Les Word embeddings caractérisent chaque mot par un ou plusieurs vecteurs denses, de faible dimension ayant des éléments réels, capturant les spécificités latente (de contexte) du mot et les propriétés syntaxiques et sémantiques utiles. Il existe plusieurs méthodes de plongement.

2.3.1 Word Vector [4]

Un vecteur de mot n'est qu'un vecteur numérique qui représente la signification d'un mot. Cela signifie que les mots sont des nombres à virgule flottante continus multidimensionnels, dans lesquels des mots ayant une sémantique similaire sont mappés à des points adjacents dans l'espace géométrique. En termes plus simples, un vecteur de mot est une rangée de nombres réels (par opposition aux nombres virtuels), où chaque point capture la dimension de la signification du mot, et les mots avec une sémantique similaire ont des vecteurs similaires. En d'autres termes, les mots utilisés dans des contextes similaires seront mappés sur des espaces vectoriels proches (nous verrons comment créer ces vecteurs de mots ci-dessous). L'avantage de représenter les mots sous forme vectorielle est qu'ils conviennent aux opérateurs mathématiques. Par exemple, nous pouvons ajouter et soustraire des vecteurs - l'exemple typique montre ici qu'en utilisant des vecteurs de mots, nous pouvons déterminer :

Roi - Homme + Femme = Reine

En d'autres termes, nous pouvons soustraire un sens du vecteur de mot du roi (c'est-à-dire, masculinité), ajouter un autre sens (féminin) et montrer que ce nouveau vecteur de mot (Roi-Homme + Femme) est le mieux lié au vecteur du mot 'Reine'.

Les nombres dans le vecteur de mot représentent le poids distribué du mot à travers les dimensions. Dans un sens simplifié, chaque dimension représente une signification et le poids numérique du mot sur cette dimension capture la proximité de son association avec et avec cette signification. Ainsi, la sémantique du mot est intégrée à travers les dimensions du vecteur.

2.3.2 Word2Vec (Mikolov et al. 2013) [5]

Word2Vec est un modèle de représentation distribué des mots peu profond (par rapport à un réseau de neurone traditionnel) où le principe est de générer des vecteurs dimensionnels à partir d'un apprentissage sur des données d'entrée non étiquetées. En fait, il prédit les mots en fonction de leur contexte en utilisant l'un des deux modèles neuronaux distincts : CBOW et Skip-Gram.

2.3.2.1 Idée:

- Nous avons un grand corpus de texte.
- Chaque mot d'un vocabulaire fixe est représenté par un vecteur.
- Parcourez chaque position t dans le texte, qui a un mot central c et mots de contexte o (« extérieurs »).
- Utilisez la similitude des vecteurs de mots pour c et o pour calculer la probabilité de o étant donné c (ou vice versa).
- Continuez à ajuster les vecteurs de mots pour maximiser cette probabilité.
- Pour la couche de sortie, la fonction de transfert est softmax.

2.3.2.2 Word2vec: fonction de prédiction [6]

La fonction *Softmax* mappe des valeurs arbitraires x_i à une probabilité distribuée pi

$$Softmax(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp\left(x_j\right)} = pi$$

• «max » car amplifie la probabilité du plus grand x_i.

• « soft » car attribue toujours une certaine probabilité à un x_i plus petit.

• Fréquemment utilisée dans le Deep Learning.

2.3.2.3 Continuous Bag-of-Words « CBOW » [5]

Le modèle "CBOW" prédit un mot commun en fonction de son contexte. Le contexte correspond à un certain nombre de mots adjacents à gauche et à droite du mot. Dans le processus CBOW, trois couches sont utilisées, comme le montre la Figure 2.1.

La couche d'entrée est le contexte. La couche cachée correspond à la projection de chaque mot dans la couche d'entrée dans la matrice de poids, qui est projetée dans la troisième couche en tant que couche de sortie. La dernière étape du modèle consiste à comparer sa sortie avec le mot lui-même afin de corriger sa représentation en fonction de la rétro-propagation du gradient d'erreur.

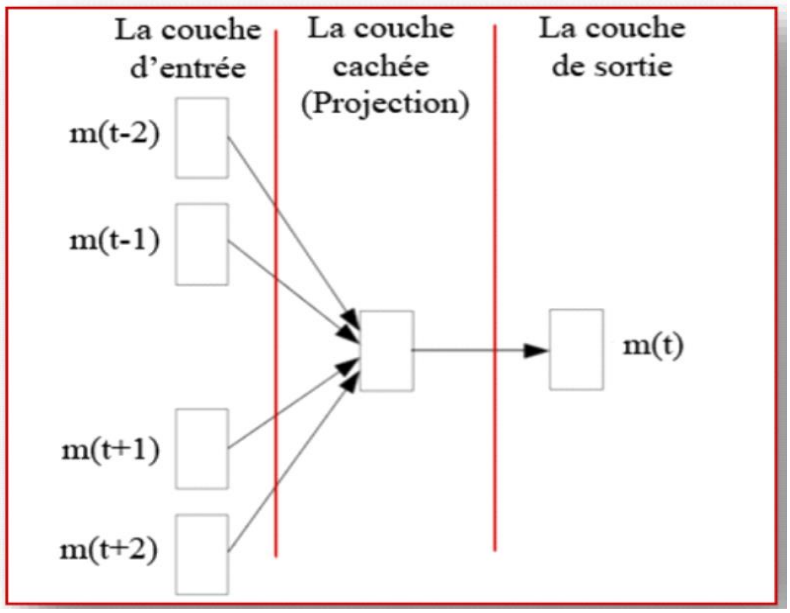


Figure 2.2: Schéma du modèle CBOW

2.3.2.4 Skip-Gram [5]

Contrairement au modèle CBOW. En fait, comme le montre la Figure 2.2, la couche d'entrée correspond au mot cible, et la couche de sortie correspond au contexte. Par conséquent, Skip-Gram cherche à prédire le contexte d'un mot donné, plutôt que de prédire le mot connu. Le contexte est CBOW. La dernière étape de Skip-Gram est de comparer sa sortie avec chaque mot du contexte afin de corriger sa représentation en fonction de la rétro-propagation du gradient d'erreur.

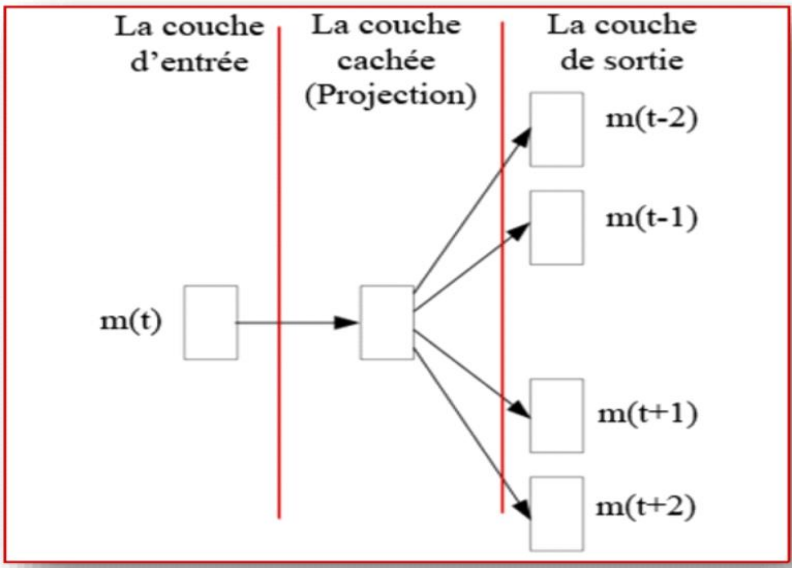


Figure 2.3: Schéma du modèle Skip-Gram

2.3.3 GloVe (Pennington, et al.) [7]

Alors que la technique Word2Vec s'appuie sur des statistiques locales (contexte local entourant le mot) pour dériver la sémantique locale d'un mot, la technique Glove va encore plus loin en combinant des statistiques locales avec des statistiques globales telles que l'analyse sémantique latente (Word Co-occurrence Matrix) pour capturer les relations sémantiques globales d'un mot.

2.3.4 ELMO [7]

ELMO utilise le LSTM bidirectionnel, qui est pré-entraîné sur un grand corpus de texte pour produire des vecteurs de mots. Il fonctionne par entraînement pour prédire le mot suivant à partir d'une séquence de mots. Cette tâche est également appelée modélisation du langage.
Les représentations ELMO sont :

- **Contexte :** Les vecteurs ELMO produits pour un mot dépendent du contexte de la phrase dans laquelle le mot est utilisé.
- **Basé sur les caractères :** les représentations ELMO sont purement basées sur des caractères, ce qui permet au réseau d'utiliser des indices morphologiques pour former

des représentations robustes pour des jetons hors vocabulaire invisibles lors de la formation.

2.3.5 BERT (Google en 2018) [8]

BERT est l'acronyme de Bidirectional Encoder Representations from Transformers. Contrairement aux modèles de représentation linguistiques récents, BERT est conçu pour pré-entraîner des représentations bidirectionnelles profondes à partir de texte non étiqueté en conditionnant conjointement le contexte gauche et droit dans toutes les couches. En conséquence, le modèle BERT préformé peut être affiné avec une seule couche de sortie supplémentaire pour créer des modèles de pointe pour un large éventail de tâches, telles que la réponse aux questions et l'inférence linguistique, sans tâche substantielle. Modifications spécifiques de l'architecture.
BERT est conceptuellement simple et empiriquement puissant. Il obtient de nouveaux résultats de pointe sur onze tâches de traitement du langage naturel, amélioration.

Ce n'est pas une liste exhaustive, il existe de nombreux autres Word Embeddings tels que LSA et GPT qui ont également fait des percées avancées dans les applications NLP.

2.4 Discussion

Les méthodes de pondération sont confrontées à un ou plusieurs problèmes en termes d'évolutivité, de parcimonie et de pertinence du contexte.
Par conséquent, nous préférons les méthodes de plongement car elles peuvent résoudre tous les problèmes ci-dessus. Le plongement mappe chaque mot sur un espace à N dimensions, Par conséquent, nous avons résolu le problème de l'évolutivité. Par rapport à chaque vecteur d'incorporation, chaque vecteur de l'incorporation est densément rempli, nous résolvons donc également le problème de parcimonie. Par conséquent, le modèle peut maintenant être mieux appris et peut être bien généralisé. Enfin, apprenez ces vecteurs d'une manière qui capture le contexte partagé et les dépendances entre les mots.

2.5 Conclusion

Nous avons présenté quelques méthodes de représentation de mots chevauchant notre projet ainsi que les définitions nécessaires pour la compréhension de la suite, mais également les techniques du plongement lexical que nous allons adapter au cours de ce projet. Dans le prochain chapitre, nous allons détailler d'avantage le fonctionnement de la correction d'orthographe à l'aide du Framework Word2vec qui est l'une des méthodes du plongement lexical et s'est avérée très efficace dans la réalisation qui pourrait être utilisée pour mesurer les similitudes syntaxiques et sémantiques entre les mots.

Chapitre 03 :

Correction des fautes d'orthographe à l'aide du Framework Word2Vec

3.1 Introduction

Comme nous avons mentionné précédemment dans le chapitre 2, il existe deux méthodes pour représenter les mots ou les phrases d'un texte, la première méthode pour représenter les mots par une valeur numérique discrète (méthode de pondération) et la deuxième représenter par des vecteurs (méthode de plongement lexical).

Dans notre projet nous travaillerons avec Word2vec qui est l'une des méthodes du plongement lexical et s'est avérée très efficace dans la réalisation qui pourrait être utilisée pour mesurer les similitudes syntaxiques et sémantiques entre les mots, pour cela nous allons proposer une approche qui résoudra notre problème avec deux techniques, dont cette technologie.

3.2 Approche proposée

Notre approche proposée pour corriger les fautes d'orthographe est une approche hybride, c'est-à-dire une combinaison de deux techniques, qui dans notre cas consiste à écrire du texte mal orthographié pour que notre approche corrige ces erreurs en mélangeant le Framework word2vec et la distance de Levenshtein.

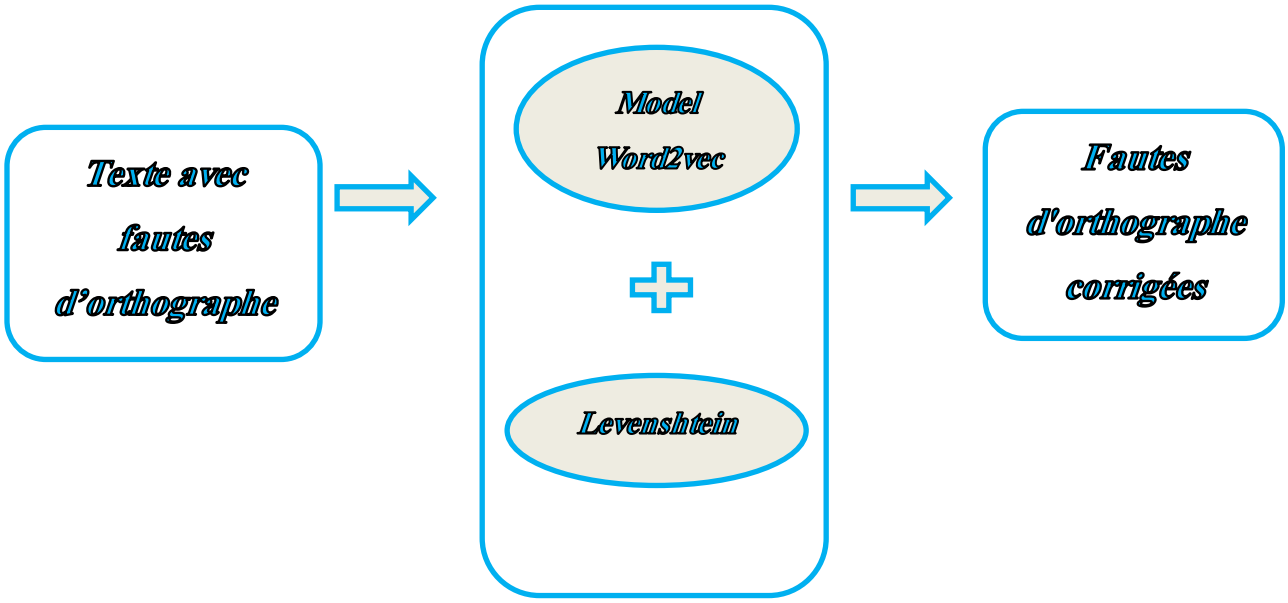


Figure 3.1 : Architecture générale de notre approche.

3.2.1 La distance de Levenshtein [9]

La distance de Levenshtein mesure la similarité entre deux chaînes de caractères. Elle est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre.

Son nom provient de Vladimir Levenshtein qui l'a définie en 1965. Elle est aussi connue sous le nom de «distance d'édition» ou encore de «déformation dynamique temporelle», surtout en reconnaissance vocale.

Cette distance est d'autant plus grande que le nombre de différences entre les deux chaînes est grand. On nomme distance de Levenshtein entre deux mots M et P le coût minimal pour aller de M à P en effectuant les opérations élémentaires suivantes :

- substitution d'un caractère de M en un caractère de P.
- ajout dans M d'un caractère de P.
- suppression d'un caractère de M.

On associe ainsi à chacune de ces opérations un coût. A titre d'exemple, dans les exemples suivants, le coût est toujours égal à 1, sauf dans le cas d'une substitution de caractères semblables.

3.2.2 Word2vec (CBOW)

Comme nous l'avons déjà mentionné dans le chapitre précédent, Word2vec a deux modèles, où chacun de ces modèles à son propre avantage.
À titre d'exemple, Skip-Gram est plus efficace avec de petites données d'entraînement, de plus il représente bien même des mots ou des phrases rares.

D'un autre côté, CBOW est plus rapide à entraîner que le skip-gram, la précision légèrement meilleure pour les mots fréquents.

Dans notre problème, étant donné un « mot de contexte », nous voulons former un modèle tel que le modèle puisse prédire un « mot cible », Pour cela nous allons choisir le modèle "CBOW" car il prédit le mot cible qui est le mot incorrect et nous corrigeons l'erreur d'après le contexte

et cela nous aidera à résoudre notre problème actuel, Le principe de fonctionnement de cette technique est décrit en détail dans 2.3.2.3.

Maintenant, En se basant sur CBOW, la question qui se pose : est-t-il possible de corriger un mot lexicalement incorrect en se basant sur les mots du contexte ?

3.2.3 Prétraitement du corpus textuel

Il existe de nombreux types de données à partir desquels des informations pertinentes pour une tâche donnée peuvent être extraites. Cependant, quel que soit le type de données, les données doivent être prétraitées.

Dans cette étape, nous nous intéressons aux données textuelles, qui nécessitent le plus de prétraitement en raison de leur ambiguïté. Et pour cela nous allons passer par plusieurs étapes de suppression et de changement :

- Supprimer les signes de ponctuation (point, virgule, point-virgule, etc.), caractères non-ascii, et Stop-Words.
- Convertir les majuscules en minuscules.

Pour cela nous avons proposé cet algorithme pour résoudre ces étapes-là :

Algorithme 3.1 : Prétraitement du corpus textuel

```
ListeMots[] = text.Liste();
stopwords[];//Liste de stopwords
Pour i=0 à length(ListeMots) faire
      Pour j=0 à length(stopwords) faire
          Si TabMots[i] = stopwords[j] alors
              ListeMots[i].Supprimer()
          Fin Si
    Fin Pour
ListeMot[i]← Supprimer non-ascii ;
ListeMots[i]← Supprimer la ponctuation ;
```

ListeMots[i]← Convertir les Majuscules en Minuscules ;

Fin Pour

Return ListeMots []

Fin.

3.2.4 Traitement

Etant donné un mot lexicalement incorrect, les étapes à suivre pour prédire le mot correct sont les suivantes :

- Extraire une liste de mots candidats avec le modèle entrainé CBOW.
- Extraire une liste de mots candidats en utilisant la distance de Levenshtein.
- Combiner les deux listes, de la première et la deuxième étape. Parmi les choix proposées (CBOW et Levenshtein), choisir le mot correct sémantiquement.

Et c'est ce que nous allons détailler les sections suivantes :

3.2.4.1 Traitement par CBOW

On extrait les mots voisins (du contexte) qui entouré le mot mal orthographié de la phrase (à gauche (resp. À droite) et/ou droite (resp. À gauche)) et comme CBOW déjà construit utilise une fonction softmax qui donne une distribution probabiliste si on lui fournit les mots du contexte, on peut donc obtenir une liste des mots ordonnés par poids, dont le mot le plus probable se trouve au début de la liste.

3.2.4.2 Traitement par Levenshtein

En utilisant la distance de Levenshtein, nous construisons la liste des mots candidats similaires au mot mal orthographié, puis nous utilisons la liste des mots candidats pour choisir le mot qui vérifie la distance minimale.

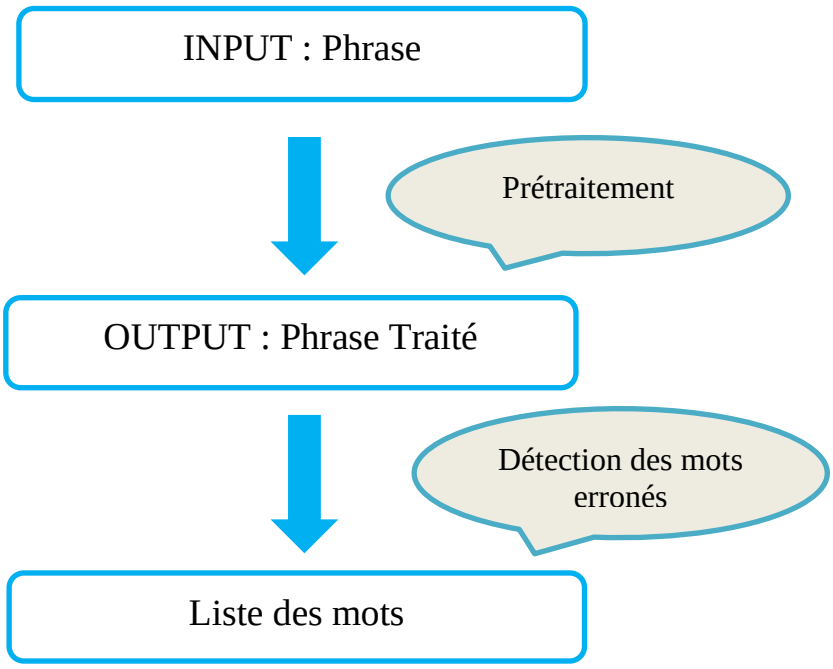
3.2.4.3 La combinaison de CBOW et Levenshtein

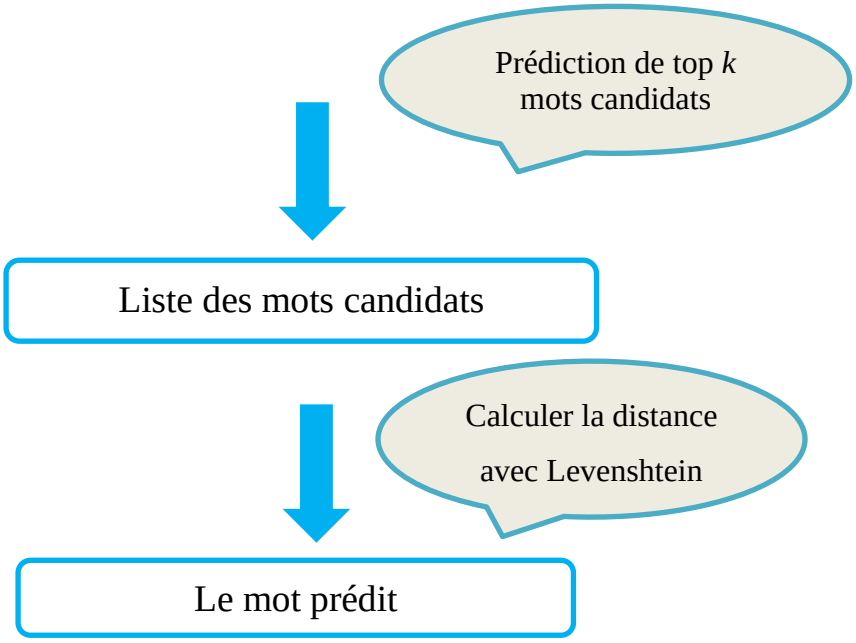
Nous combinons CBOW avec la distance de Levenshtein. CBOW pour que nous puisons trouver une liste de mots candidats (probables), puis avec la distance de Levenshtein, nous choisissons le bon mot qui vérifie la distance minimale, donc nous avons proposé cet algorithme qui fait une combinaison des deux listes pour prévoir le bon mot candidat:

Algorithme 3.2: predict(text)

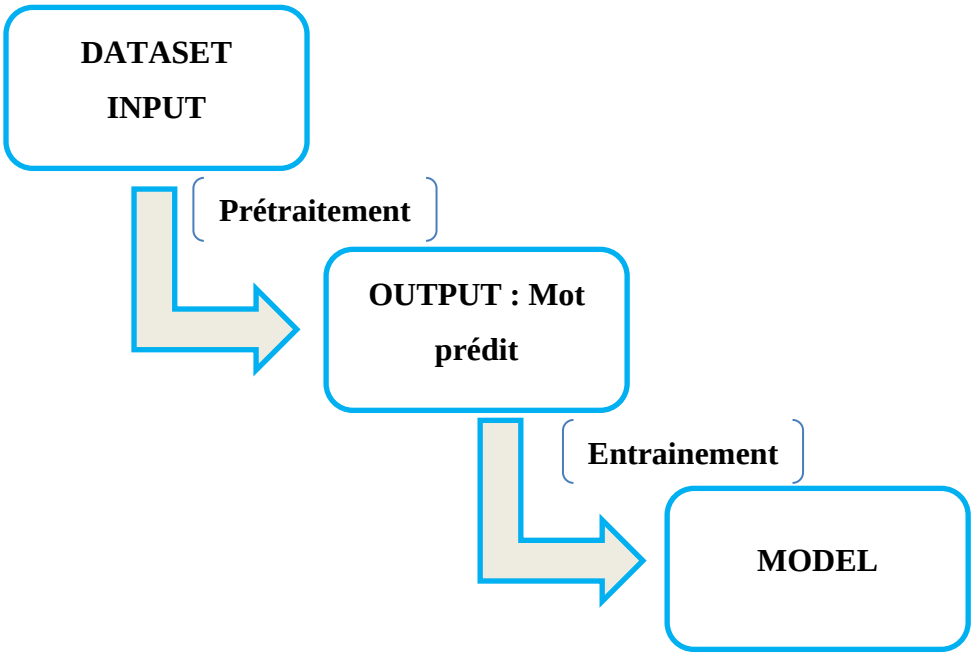
ListeMots = traitement(text);
motIncorrect = spellchecker. checkErreur (ListeMots);
motPredictLenven =spellchecker(motIncorrect);
ListeCbow = model.cbow(motIncorrect, top=*k*);
motCorrect= intersection (ListeCbow, motPredictLenven) ;
text. replace (motIncorrect, motCorrect)
Return text
Fin.

3.2.5 Schéma d'entrainement





3.2.6 Schéma de test



3.3 Conclusion

Après avoir réalisé la conception, qui se traduit en fait par la présentation de notre idée proposée pour corriger les fautes d'orthographe sémantiquement à l'aide de Framework Word2vec et la distance de Levenshtein, il serait judicieux de penser à implémenter notre projet. En effet, cela nous mène au quatrième chapitre, ce dernier est dédié à l'implémentation de notre application.

Chapitre 04 :

Expérimentation et évaluation des résultats

4.1 Introduction

Après avoir terminé la conception de notre approche, nous avons commencé à programmer notre application et notre côté technique. Où nous expliquerons dans ce chapitre une méthode générale de performance (l'exécution de notre application), ainsi que les différents outils de développement et les logiciels utilisées pour y parvenir.

4.2 Implémentation du correcteur

4.2.1 Langages et outils de développement

Pour la réalisation de notre projet, nous avons utilisé les outils de développement suivants :

4.2.1.1 Python [10]

Python est le langage de programmation open source le plus employé par les informaticiens. Ce langage s'est propulsé en tête de la gestion d'infrastructure, d'analyse de données ou dans le domaine du développement de logiciels. En effet, parmi ses qualités, Python permet notamment aux développeurs de se concentrer sur ce qu'ils font plutôt que sur la manière dont ils le font. Il a libéré les développeurs des contraintes de formes qui occupaient leur temps avec les langages plus anciens. Ainsi, développer du code avec Python est plus rapide qu'avec d'autres langages.

Les principales utilisations de Python par les développeurs sont :

- la programmation d'applications.
- la création de services web.
- la génération de code.
- la méta-programmation.

Figure 4.1 : Logo de python.

4.2.1.2 Jupyter Notebook [11]

Python Notebook est maintenant connu sous le nom de Jupyter Notebook. Il s'agit d'une application Web open source qui vous permet de créer et de partager des documents contenant du code en direct, des équations, des visualisations et du texte narratif. Les utilisations incluent :

le nettoyage et la transformation des données, la simulation numérique, la modélisation statistique, la visualisation des données, l'apprentissage automatique et bien plus encore. L'application Jupyter Notebook peut être exécutée sur un bureau local ne nécessitant aucun accès Internet ou peut être installée sur un serveur distant et accessible via l'Internet.

Figure 4.2 : Logo de jupyter.

4.2.2 Les Bibliothèques utilisée

4.2.2.1 Pyspellecheker [12]

Vérification orthographique Pure Python basée sur l'article de blog de Peter Norvig sur la configuration d'un algorithme de vérification orthographique simple.
Il utilise un algorithme de distance de Levenshtein pour trouver les modifications possibles sur le mot d'origine. Il compare ensuite toutes les permutations (insertions, suppressions, remplacements et transpositions) aux mots connus dans une liste de mots (dictionnaire). Les mots que l'on trouve le plus souvent dans la liste sont plus probablement les bons résultats. Pyspellchecker prend en charge plusieurs langues, dont l'anglais, l'espagnol, l'allemand, le français et le portugais.

4.2.2.2 Gensim[13]

Gensim est une bibliothèque Python open source gratuite permettant de représenter des documents sous forme de vecteurs sémantiques. Gensim est conçu pour traiter des textes non structurés (« texte brut ») à l'aide d'algorithmes d'apprentissage automatique non supervisés.
Les algorithmes de Gensim, tels que Word2Vec, FastText, Latent Semantic Indexing (LSI, LSA, LsiModel), Latent Dirichlet Allocation (LDA, LdaModel), etc., découvrent automatiquement la structure sémantique des documents en examinant les modèles statistiques de cooccurrence au sein d'un corpus de documents d'apprentissage. Ces algorithmes ne sont

pas supervisés, ce qui signifie qu'aucune intervention humaine n'est nécessaire - vous n'avez besoin que d'un corpus de documents en texte brut.
Une fois ces modèles statistiques trouvés, tous les documents en texte brut (phrase, mot) peuvent être exprimés succinctement dans la nouvelle représentation sémantique et interrogés pour une similarité thématique avec d'autres documents (mots, phrases).

4.2.2.3 Pandas [14]

Pandas est une bibliothèque Python utilisée pour travailler avec des ensembles de données.
Il a des fonctions d'analyse, de nettoyage, d'exploration et de manipulation des données.
Le nom "Pandas" fait référence à la fois à "Panel Data" et à "Python Data Analysis" et a été créé par Wes McKinney en 2008.
Pandas nous permet d'analyser de grandes données et de tirer des conclusions basées sur des théories statistiques.
Pandas peuvent nettoyer des ensembles de données désordonnés et les rendre lisibles et pertinents.

4.2.2.4 NLTK (Natural Language Toolkit) [15]

NLTK est une bibliothèque de traitement automatique des langues, écrite en langage Python, développée par Steven Bird et Edward Loper du département d'informatique de l'université de Pennsylvanie. En plus de la bibliothèque, NLTK fournit des démonstrations graphiques, des données-échantillon, des tutoriels, ainsi que la documentation de l'interface de programmation (API).

4.2.2.5 NumPy [16]

NumPy est une bibliothèque écrite en langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.

Plus précisément, cette bibliothèque libre et open source fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynôme.

4.2.3 Présentation de Dataset

Notre ensemble de données est une concaténation de deux datasets.
La première (bbc-text) est constituée d'ensembles de données d'articles d'actualités, provenant de BBC News, fournis à titre de référence pour la recherche en apprentissage automatique. Les données d'origine sont traitées pour former un fichier csv unique pour une utilisation facile, le titre de l'actualité et le nom du fichier texte associé sont conservés avec le contenu de l'actualité et sa catégorie. Il se compose de 2225 documents du site Web d'actualités de BBC correspondant à des articles dans 5 domaines thématiques (affaires, divertissement, politique, sport, technologie).

Alors que la deuxième dataset (imdb) est contient des critiques et des notes pour différents films, et se compose de 100.000 documents.

4.3 Présentation de l'application

Dans cette partie, nous présentons le fonctionnement de l'application à travers les différentes captures d'écrans.

Après le téléchargement et le prétraitement de l'ensemble de données (Dataset), nous avons divisé notre programme en 3 catégories: Levenshtein, W2V, notre approche hybride (Levenshtein + Word2Vec).

- Nous commençons donc par la première méthode où nous faisons une liste "incorrecte" contient 10 phrases avec des mots mal orthographiés.

```
a great movii to watch
I watched a good fiml
great movies to watche
the principal actrr played his role so well
I get bored easily when watching movis
i need helpp
this film is absolutely appaling and awful
as this movie is just so pore
i wanted too like it
it would be nisse
```

Figure 4.3 : Phrases avec des mots mal orthographiés.

- Puis nous présentons les résultats finals par la distance de Levenshtein sous la forme suivante : mot mal orthographié, mot proposé, mot correct, les candidats.

	Misspelled	Proposed Word	Correct Word	Candidates
0	movii	movie	movie	{movei, movi, movie, movin}
1	fiml	film	film	{fime, fim, himl, fill, fiel, film, fil}
2	watche	watch	watch	{watched, watches, watcher, watch, watcha}
3	actrr	actor	actor	{actor, actr}
4	movis	movie	movies	{movi, moves, movie, mavis, mois, movin, movies}
5	helpp	help	help	{help, helpa, helps, helpl}
6	appaling	appealing	appalling	{appealing, appalling}
7	nisse	disse	nice	{niste, nissen, disse, misse}

Figure 4.4 : Les résultats finals par la distance de Levenshtein.

- Ensuite, nous passons à la deuxième méthode, qui dépend de Word2Vec et la présentons sous la même forme que la première méthode.

```
Sentence with mistakes:  a great movii to watch
        Word to correct: movii
        Proposed word: movies
        Correct word: movie
Sentence with mistakes:  I watched a good fiml
        Word to correct: fiml
        Proposed word: movies
        Correct word: film
Sentence with mistakes:  great movies to watche
        Word to correct: watche
        Proposed word: watch
        Correct word: watch
Sentence with mistakes:  the principal actrr played his role so well
        Word to correct: actrr
        Proposed word: actor
        Correct word: actor
Sentence with mistakes:  I get bored easily when watching movis
        Word to correct: movis
        Proposed word: movies
        Correct word: movies
Sentence with mistakes:  i need helpp
        Word to correct: helpp
        Proposed word: desperately
        Correct word: help
Sentence with mistakes:  this film is absolutely appaling and awful
        Word to correct: appaling
        Proposed word: appalling
        Correct word: appalling
Sentence with mistakes:  as this movie is just so pore
        Word to correct: pore
        Proposed word: watching
        Correct word: poor
Sentence with mistakes:  i wanted too like it
        Word to correct: too
        Proposed word: desperately
        Correct word: to
Sentence with mistakes:  it would be nisse
        Word to correct: nisse
        Proposed word: benefited
        Correct word: nice
```

Figure 4.5 : Les phrases avec des mots mal orthographiés et ses résultats finals par Word2Vec.

	Misspelled	Proposed Word	Correct Word	Candidates
0	movii	movies	movie	[movies, joy, pleasure, movie, film, want, new]
1	fiml	movies	film	[movies, film, yesterday, recently, reason, ne...
2	watche	watch	watch	[watch, watching, deal, b, horror, geetanjali,...
3	actrr	actor	actor	[actor, dual, thankless, pivotal, plays, actress]
4	movis	movies	movies	[movies, to, started, tired, tears, offended]
5	helpp	desperately	help	[desperately, urgent, apply, help, desperate, ...
6	appaling	appalling	appalling	[appalling, terrible, horrible, dreadful, to, ...
7	pore	watching	poor	[watching, poor, great, watch, easily, b, watc...
8	too	desperately	to	[desperately, movies, felt, malibu, looked, se...
9	nisse	benefited	nice	[benefited, hoped, preferred, gladly, expect, ...

Figure 4.6 : Les résultats finals par Word2Vec.

- Et enfin, nous connectons les deux méthodes ensemble (Levenshtein et W2V) par une intersection pour obtenir notre approche hybride.

	Misspelled	Proposed Word	Correct Word	Candidates
0	movii	movie	movie	[movies, joy, pleasure, movie, film, want, new]
1	fiml	film	film	[movies, film, yesterday, recently, reason, ne...
2	watche	watch	watch	[watch, watching, deal, b, horror, geetanjali,...
3	actrr	actor	actor	[actor, dual, thankless, pivotal, plays, actress]
4	movis	movies	movies	[movies, to, started, tired, tears, offended]
5	helpp	help	help	[desperately, urgent, apply, help, desperate, ...
6	appaling	appalling	appalling	[appalling, terrible, horrible, dreadful, to, ...
7	pore	appalling	poor	[watching, poor, great, watch, easily, b, watc...
8	too	appalling	to	[desperately, movies, felt, malibu, looked, se...
9	nisse	benefited	nice	[benefited, hoped, preferred, gladly, expect, ...

Figure 4.7 : Les résultats finals par notre approche.

- **NB :** S'il n'y a pas une intersection entre les 2 techniques, il va donner le résultat avec CBOW.

4.4 Evaluation des résultats

Dans cette section on va comparer notre approche hybride (Levenshtein + CBOW) par rapport à Levenshtein qui est la méthode classique de correction des erreurs en calculant le pourcentage de bonnes réponses pour faire une évaluation.

$$pourcentage\ de\ réussite = \frac{mots\ corrects * 100}{total\ des\ mots}$$

```
percentOfSucces = percentOfSucces * 100 /i
percentOfSuccesLeven = percentOfSuccesLeven * 100 /i
print('percent of succes of our methode ',percentOfSucces,'%')
print('percent of succes of levenshtein ',percentOfSuccesLeven,'%')
resulta_intersection
```

```
10


percent of succes of our methode  60.0 %
percent of succes of levenshtein  50.0 %
```

Figure 4.8 : Evaluation des résultats.

Nous sommes arrivés à la conclusion que l'hybridation permet d'atteindre des résultats nettement meilleurs.

4.5 Conclusion

Dans ce chapitre, nous présentons nos résultats de test avec des captures d'écran tout en décrivant notre architecture. Nous avons également présenté l'évaluation des résultats qui montre la fiabilité de notre approche.

Conclusion et Perspectives

Le travail réalisé dans ce mémoire nous a permis d'atteindre notre objectif qui est la résolution du problème de correction d'orthographe par la sémantique, en proposant une approche hybride, basée sur le modèle CBOW du Framework Word2Vec et la distance de Levenshtein.

Après avoir parlé de notre projet, nous pensons que l'étape la plus importante réside dans: Comment dérouler un travail de recherche, en profitant des travaux existants dans le domaine, et les nouvelles technologies boostées par l'intelligence artificielle, qui nous ont permet de venir à bout pour finaliser le projet.

En perspective, nous proposons d'envisager cette nouvelle technique qui va beaucoup nous aider et de l'ajouter aux logiciels spécialisés dans ce domaine en appliquant le concept humain dans la correction des erreurs et en abandonnant ainsi les anciennes méthodes qui obligent le rédacteur à surveiller le texte.

Référence

[1] Jacquet-Pfau Christine, « Correcteurs orthographiques et grammaticaux. Quel(s) outil(s) pour quel rédacteur ? », *Revue française de linguistique appliquée*, 2001/2 (Vol. VI), p. 81-94. DOI : 10.3917/rfla.062.0081.

[2] Lawaly. S, Harouna N. (2014). Etude et conception d'un correcteur orthographique pour la langue haoussa. *21ème Traitement Automatique des Langues Naturelles, Marseille, 2014*, 147-158.

[3] Brownlee, J. (2019, août 7). *A Gentle Introduction to the Bag-of-Words Model*. Machine Learning Mastery. https://machinelearningmastery.com/gentle-introduction-bag-words-model/

[4] Ahire, J. B. (2019, 25 mai). *Introduction to Word Vectors*. Dzone.Com. https://dzone.com/articles/introduction-to-word-vectors

[5] Bentekkouka.A et Cheik.A,(2020), « MÉMOIRE DE MASTER INFORMATIQUE TAL Généralisation d'approches basées Word Embedding pour un système d'évaluation des réponses courtes adapté à la langue anglaise», http://di.univ-blida.dz:8080/jspui/bitstream/123456789/4943/1/BENTEKKOUKA%20%20%20Abderrahmane%28G%C3%A9n%C3%A9ralisation%20d%E2%80%99approches%20bas%C3%A9es%20Word%20Embedding%20pour%20un%20syst%C3%A8me%20d%E2%80%99%C3%A9valuation.pdf

[6] https://fr.wikipedia.org/wiki/Fonction_softmax

[7] Dua, A. (2020, août 5). *Introduction to Word Embeddings and its Applications*. Medium. https://medium.com/compassred-data-blog/introduction-to-word-embeddings-and-its-applications-8749fd1eb232

[8] *Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT.*

[9] http://www.bonne-mesure.com/distance_de_levenshtein.php

[10] RÉ ; Daction, L. (2020, 31 mars). *Python : définition et utilisation de ce langage informatique*. https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445304-python-definition-et-utilisation-de-ce-langage-informatique/

[11] Project *Jupyter*. (2021). Home. https://jupyter.org/

[12] *pyspellchecker*. (2021, 27 mars). PyPI. https://pypi.org/project/pyspellchecker/

[13] *Gensim : topic modelling for humans*. (2021). https://Radimrehurek.Com/Gensim/Intro.html

[14] *Pandas Introduction*. (2021). https://Www.W3schools.Com/Python/Pandas/Pandas_intro.Asp

[15] Wikipedia contributors. (2020, 28 mars). *Natural Language Toolkit*. https://fr.wikipedia.org/wiki/Natural_Language_Toolkit

[16] Wikipedia contributors. (2021, août 5). *NumPy*. https://fr.wikipedia.org/wiki/NumPy. https://fr.wikipedia.org/wiki/NumPy