

**République Algérienne Démocratique et Populaire**  
**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**  
**Université Mouloud Mammeri de Tizi Ouzou**  
**Faculté de Génie électrique et d'informatique**  
**Département informatique**



**Mémoire de fin d'études**  
En vue de l'obtention du diplôme de  
Master en Informatique  
Spécialité : Conduite de Projet Informatique

**Thème :**

**Utilisation d'un modèle Word Embedding pour  
extraire la similarité entre concepts d'enseignements**

Présenté par :

El Djama Soraya

Hamitouche Celia

Dirigé par:

Mme Lazib Lydia

Membres de jury :

Mme Bouarab Farida

Melle Khiali Lynda

## *Remerciements*

*Nous remercions Allah de nous avoir donné le courage, la santé et la motivation pour finir ce projet de fin d'études.*

*Nous tenons à remercier particulièrement notre promotrice Madame Lazib Lydia, nous tenons à lui exprimer notre profonde gratitude, pour nous avoir encadré avec un intérêt constant et une grande compétence, pour sa disponibilité, son soutien sans faille, ses conseils, et les encouragements qui nous ont permis de mener à bien ce travail.*

*Nous tenons à remercier les membres du jury d'avoir bien accepté de consacrer leur temps pour évaluer ce modeste travail.*

*Nous souhaitons vivement remercier nos familles pour leur soutien et leur compréhension tout au long de nos années d'études.*

*Enfin, on n'oublie pas de remercier nos amis d'études et tous ceux qui ont contribué de près ou de loin à notre formation et à l'aboutissement de ce projet.*

## Dédicaces

*A la mémoire de mon père*

*A ma chère maman.*

*A mes chers frères et belles-sœurs*

*A ma nièce Zira et mon neveu Ali*

*A mes chères amies*

*A mon ami Xavier*

*Soraya*

## Dédicaces

### ***Je dédie ce mémoire :***

*En premier lieu, A mes chers parents qui m'ont offert la vie et orienté mes pas, et qui continuent inlassablement à me guider vers le chemin de la réussite.*

*A ma petite sœur « Babi » qui m'a énormément aidé, je lui souhaite tout le succès et le bonheur du monde.*

*A mon fiancé, pour son soutien et ses encouragements.*

*A toute ma famille maternelle et paternelle petits et grands.*

*A ma belle-famille.*

*A mes meilleures amies « Tina, Sabrina, Aini ».*

*A mes aimables amis et collègues d'études.*

*Aux personnes qui m'ont toujours aidé et encouragé, à tous ceux qui étaient là à mes côtés et qui m'ont accompagné durant mon chemin d'étude*

*A tous ceux que j'aime.*

Celia

# Table des matières

Introduction générale.....	1
CHAPITRE 1 : Traitement automatique des langues naturelles .....	4
Introduction .....	5
I.1 Définition.....	6
I.2 Historique :.....	6
I.3 Les objectifs de TLAN.....	9
I.4 Les niveaux d'analyse d'un texte : .....	10
I.4.1 Analyse morphologique .....	11
I.4.2 Analyse syntaxique .....	12
I.4.3 L'analyse sémantique.....	13
I.4.4 L'analyse pragmatique .....	13
I.5 Les domaines d'application du TALN .....	14
I.6 Les difficultés de TALN : ambiguïté et implicite .....	18
I.6.1 L'ambiguïté.....	19
I.6.2 Implicite .....	20
I.7 Les outils du TALN .....	21
I.7.1 Stanford NLP Group Software .....	21
I.7.2 Le Natural Language Toolkit .....	21
I.7.3 Visualtext .....	22
I.7.4 NooJ .....	22
Conclusion.....	23
CHAPITRE 2 : L'apprentissage automatique .....	24
Introduction à l'apprentissage automatique .....	25
II.1 Définition de l'apprentissage automatique .....	25
II.2 Exemples de tâches d'apprentissage automatique .....	26
II.3 Les méthodes d'apprentissages automatiques .....	26
II.4 Les types de données utilisées dans l'apprentissage automatique .....	27
II.4.1 Données structurées .....	27
II.4.2 Données semi-structurées .....	27
II.4.3 Données non-structurées .....	28
II.5 Modèles de l'apprentissage automatique .....	28
II.5.1 Apprentissage supervisé.....	28
II.5.2 L'apprentissage non supervisé .....	29

II.5.3 Apprentissage semi supervisé .....	29
II.5.4 Apprentissage par renforcement.....	29
II.6 Objectifs des méthodes d'apprentissage automatique. ....	30
II.6.1 La classification .....	30
II.6.2 La régression.....	30
II.6.3 La segmentation.....	31
II.7 Introduction aux réseaux de neurones .....	31
II.8 Présentation d'un réseau de neurone .....	31
II.9 Topologie d'un réseau de neurone.....	33
II.10 Les couches du réseau de neurones .....	34
II.10.1 Comportement et structure d'un réseau de neurones artificiels .....	34
II.11 Architecture de réseaux de neurones .....	35
II.11.1 Réseaux statiques "feed-forward " .....	35
II.11.2 Réseaux multicouches (ou Perceptron Multi Couche PMC) .....	35
II.11.3 Réseau récurrents "Feed-back" .....	36
Conclusion.....	36
CHAPITRE 3: Le Word Embedding .....	37
Introduction .....	38
III.1 Présentation des Word Emebeddings .....	38
III.2 Un bref historique sur le Word Embedding .....	38
III.3 Les modèles Word Embeddings.....	39
III.3.1 Le model word2vec .....	40
III.4 Les couches du modèle Word2vec .....	43
III.5 Paramètres de Word2vec.....	44
III.6 Les domaines d'application du Word Embedding .....	44
III.6.2 Reconnaissances d'entités nommées .....	45
III.6.3 La traduction automatique .....	45
Conclusion.....	47
CHAPITRE 4 : LA CONCEPTION.....	48
Introduction .....	49
IV.1 Approche proposée .....	49
IV.1.1 Tâche 1 : Construction du corpus .....	51
C. Quelques statistiques du corpus .....	52
IV.1.2 Tâche 2 : Création du Modèle Word Embedding .....	53
IV.1.3 Tâche 3 : Utilisation du modèle pour extraire la similarité entre concepts .....	54
CHAPITRE 5 : Implémentation .....	56

Introduction .....	57
V.1    Choix du langage de programmation.....	57
V.2    Environnement de développement.....	58
V.3    Bibliothèques Python utilisées .....	59
V.4    Matériel utilisé .....	60
V.5    Mise en Œuvre .....	61
V.5.1 Les étapes de création de notre corpus.....	61
V.6    Les étapes d’entrainement de model Word Emebeddings.....	66
V.6.1 Importer les bibliothèques .....	67
V.6.2 Importer les données.....	67
V.6.3 La création du modèle .....	67
V.6.4 Mise en application.....	68
V.7    Présentation de notre application.....	70
V.7.1 Démonstration sur notre applicaton .....	73
Conclusion générale .....	79
□Bibliographie.....	81
Webographies.....	85

## Listes des figures de chapitre 1

Figure I-1 : Evolution du traitement automatique du langage naturel .....	6
Figure I-2: la relation entre NLP, Machine Learning et Deep Learning .....	8
Figure I-3:Les niveaux d'analyse d'un texte.....	11
Figure I-4 : Représentation avec l'arbre syntaxique .....	13

## Listes des figures de chapitre 2

Figure 1: Méthodes d'apprentissage automatique. ....	27
Figure 2: Architecture d'un neurone.....	32
Figure 3: La différence entre un réseau de neurone biologique et un réseau de neurone artificiel .....	32
Figure 4: Structure d'un réseau de neurones artificiels. ....	34

## Listes des figures de chapitre 3 :

Figure 1: Les modèles contextuels et non-contextuels.....	40
Figure 2: Architecture du modèle CBOW.....	42
Figure 3: Architecture du modèle Skip gram .....	43
Figure 4: Exemple d'entraînement du modèle Word2vec à partir de couple de mot .....	46

## Liste des figures de chapitre 4:

Figure 1 : Architecture générale de notre approche proposée.....	50
Figure 2 : Étapes de construction et traitement du corpus.....	52

## Liste des figures de chapitre5 :

Figure 1: Les domaines qui constituent notre corpus .....	62
Figure 2: Le téléchargement des articles en mode multithread et leurs enregistrements en format zip .....	63
Figure 3: Similarité entre mots « ordinateur »et « informatique » .....	68
Figure 4: Similarité entre mots « biologie »et « microbiologie » .....	68
Figure 5: Similarité entre mots « biologie »et « informatique » .....	69
Figure 6: Interface de notre application .....	70
Figure 7:Top 5 des mots proches au mot informatique.....	74



Figure 8: Similarité entre deux mots .....	76
--	----

#### Liste des tableaux

Tableau 1: Passage d'un réseau biologique à un réseau de neurone artificiel .....	33
---	----

# Introduction générale

Depuis quelques années, l'Intelligence Artificielle fait parler d'elle à tous les instants. Aujourd'hui, elle sait tirer parti des données textuelles, grâce à une technique appelée « traitement automatique du langage naturel (TALN) »

Le traitement automatique du langage naturel (TALN) est un domaine à l'intersection du Machine Learning et de la linguistique. Il vise à modéliser et reproduire, à l'aide de machines, la capacité humaine à produire et à comprendre des énoncés linguistiques dans des buts de communication.

Avec le développement de l'Internet, le volume de l'information a explosé, ce qui rend l'accès aux documents souhaités de plus en plus difficile.

Pour accéder facilement et trouver rapidement les articles qu'on recherche, un système de recherche de mot est nécessaire, c'est pourquoi le domaine de recherche de mots similaires devient de plus en plus actif dans la communauté internationale. La technique qui nous permet de réaliser cette approche est bien les Word Emebeddings, qui sont un composant standard des architectures modernes de traitement automatique des langues naturelles (TALN), qui s'appuie sur l'exploitation d'un grand nombre de données.

Chaque fois qu'une avancée est obtenue dans l'apprentissage de Word Emebeddings, la grande majorité des tâches de traitement automatique des langues, telles que l'étiquetage morphosyntaxique, la reconnaissance d'entités nommées, la recherche de réponses à des questions, ou l'inférence textuelle, peuvent en bénéficier.

Dans ce travail nous allons nous baser sur les concepts d'enseignement dans le but de calculer la valeur de similarité entre deux mots, et d'afficher les mots similaires d'un mot qu'un utilisateur cherche sur un domaine spécifique et cela à partir d'une transformation en vecteur avec la méthode de Word Embeddings.

Pour ce faire, nous avons organisé notre mémoire en cinq chapitres. Le premier aborde les concepts de base du traitement automatique du langage naturel (TALN). Nous donnons les définitions sur les notions fondamentales du TALN, et mettons en évidence ses domaines d'application.

Le deuxième chapitre introduit le domaine du Machine Learning et ses différents types de modèles d'apprentissage. Puis, nous mettons le point sur les réseaux de neurones, en présentant les différentes architectures des réseaux de neurones et les couches qui les composent.

Le troisième chapitre aborde des notions sur le Word Embedding, des modèles Word2Vec, et des domaines d'application du Word Embedding.

Le quatrième chapitre est consacré à la présentation de notre approche. Nous développons la démarche suivie pour la construction de notre corpus, puis la génération de notre modèle de Word Embedding, et enfin l'utilisation de ce dernier pour extraire la similarité entre des termes et les mots similaires à un terme.

Le dernier chapitre présente les outils utilisés pour mettre en œuvre notre approche, et montre le travail réalisé, à savoir la similarité entre des termes donnés et les mots similaires à un terme.

# CHAPITRE 1 : Traitement automatique des langues naturelles

## Introduction

Aujourd'hui nous vivons dans un monde numérisé et connecté, c'est une évidence, et nos modes de vie, qu'il s'agisse d'étudier, de travailler, de se divertir, ou encore d'être citoyen, se sont en effet transformés radicalement. Cette société dite de l'information, caractérisée à ses débuts par une augmentation de la quantité de documents publiés, voit aujourd'hui une multiplication énorme de données. Dans ce contexte, les systèmes de traitement de données permettent non seulement de les stocker mais aussi de les manipuler. Parmi les objectifs poursuivis est d'extraire et de structurer des éléments d'informations, à partir de données brutes, afin d'élaborer des connaissances et d'être en capacité de les exploiter. Le traitement automatique de langage naturel dit (TALN) aide dans le traitement des données de nature langagières. [9]

Le traitement automatique de langage naturel (TALN) représente de manière générale les traitements informatiques capables de "traiter" de manière automatique (uniquement au moyen d'ordinateurs) le langage naturel (les langues parlées par les êtres humains comme la langue française par exemple).

Le but de TALN est de "comprendre" le sens des phrases d'un texte et les idées qui s'en dégagent et ce de manière à pouvoir les "traiter" de la manière la plus optimale et la plus naturelle d'un point de vue humain. [2]

Ce chapitre donne un aperçu sur le traitement automatique du langage naturel. Nous présentons dans un premier temps la définition de TALN puis un bref historique. Ensuite nous abordons l'objectif du TALN en précisant son analyse, sa génération tout en citant les différents niveaux d'analyse des textes ainsi qu'une brève explication des niveaux de traitement de TALN.

Enfin, nous terminons notre chapitre par certains outils du TALN.

## I.1 Définition

Le traitement automatique de langage naturel (TALN) est un champ de savoir et de techniques élaborés autour de divers champs disciplinaires : l'intelligence artificielle « traditionnelle », l'informatique théorique, la logique, la linguistique, mais aussi les neurosciences, les statistiques, etc. [14]

L'objectif du traitement automatique de langage naturel est la conception de logiciels capables de traiter de façon automatique des données exprimées dans une langue dite « naturelle », par opposition aux langages formels de la logique mathématique. Ces données linguistiques peuvent, selon les cas, être de différents types (textes écrits, dialogues écrits ou oraux, etc.) et de taille variable (du texte entier au mot isolé, en passant par la phrase ou le syntagme). Qui dit « traitement » dit manipulation d'un objet d'entrée aboutissant à la modification de cet objet en un objet de sortie. C'est à dire permettre aux machines de lire, de déchiffrer, de comprendre et de donner sens au langage humain. [5]

## I.2 Historique :

Historiquement, les premiers travaux importants dans le domaine du TALN ont porté sur la traduction automatique. Les premières expériences remontent aux années 30.

Le TALN commence à se développer dans les années 1950, d'abord sous l'impulsion de chercheurs russes puis aux états unis. L'expérience Georgetown-IBM en 1954 était une démonstration influente de la traduction automatique, où IBM et l'Université de Georgetown s'associent pour créer le premier programme informatique capable de traduire une soixantaine de phrases du russe vers l'anglais. La machine était surnommée « le cerveau ». Bien que le vocabulaire ne comptât que 250 mots et la grammaire 6 règles, cette expérience a déclenché de nombreux travaux dans ce domaine [14]

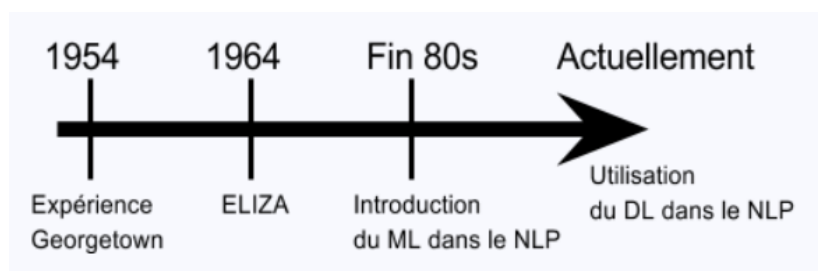


Figure 0-1 : Évolution du traitement automatique du langage naturel

Des années 60 et 70 :

Dans les années 60 cependant, les progrès n'ayant pas été assez rapides, les travaux sur le sujet commencent à être nettement moins nombreux, jusqu'au rapport ALPAC de 1966, très critique sur les efforts en cours et qui constate que les buts n'ont pas été atteints. Il suggère de favoriser la recherche fondamentale plutôt que les applications.

Malgré la désillusion et les subventions qui s'arrêtent aux Etats-Unis, les recherches continuent. Dans les années 60 et 70, parallèlement aux recherches en traduction automatique, on voit l'apparition des **premiers chatbot de l'histoire**, avec par exemple ELIZA en 1964, qui était capable de simuler une psychothérapie (reformulation des phrases du « patient » et questions contextuelles) avec seulement 5 pages de langage informatique.

- A partir des années 80 : une approche statistique du NLP

Jusque-là il s'agissait d'une approche symbolique de la discipline (développée sous l'impulsion de Noam Chomsky à la fin des années 60), c'est à dire que le savoir linguistique était codé sous forme de *grammaires* et de *bases de données lexicales* développées manuellement.

C'est à partir des années 80 que les premières approches statistiques voient le jour. Mais pour rendre la modélisation mathématique possible, il faut pouvoir transformer le texte en input numérique, alors les premières méthodes consistaient à représenter le texte sous forme de vecteurs, en comptant les occurrences des mots ou groupes de mots par exemple.

A la fin des années 80, l'augmentation des capacités de traitement informatique et l'introduction des algorithmes de machine Learning dans le traitement du langage donnent un nouveau souffle au TALN.

Avec cette approche qui vient compléter la précédente, la machine est en mesure de créer ses propres règles, déterminées par apprentissage à partir de textes.



## Années 90 : la révolution du deep Learning

C'est dans les années 90 qu'a lieu une avancée décisive avec les travaux de **Yann Le Cun** sur l'apprentissage profond au sein des laboratoires Bells, la mise au point du premier système basé sur les réseaux de neurones convolutifs permettant de lire les chèques bancaires. S'ensuivra pourtant une période « noire » pour le domaine, une dizaine d'années avec peu d'avancées significatives.

A partir de 2013, des réseaux de neurones comme Word2Vec, Glove puis Vanilla RNNs pour ne citer qu'eux, sont introduits dans la discipline, mais les chercheurs ne sont pas encore satisfaits et commencent à s'intéresser de plus près à ces réseaux de neurones convolutifs (initialement développés pour la reconnaissance d'images) aux résultats époustouflants.

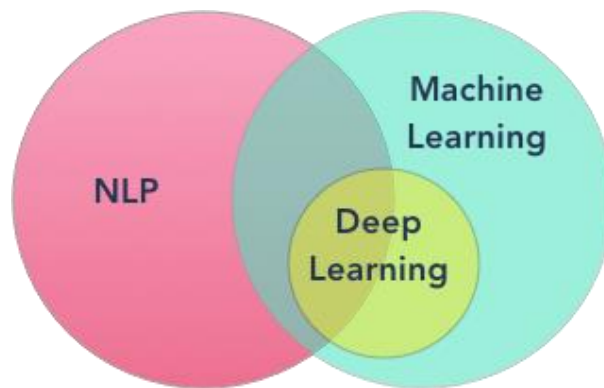


Figure 0-2: la relation entre NLP, Machine Learning et Deep Learning

Actuellement, grâce à la puissance de calcul exponentielle des ordinateurs, la disponibilité massive des données en open source de plus en plus importante et le perfectionnement continu des algorithmes de machine Learning, le TALN est donc de nouveau en plein essor. [5]

### **I.3 Les objectifs de TLAN**

L'objectif du TALN est la conception de logiciels, capables de traiter de façon automatique des données linguistiques, c'est-à-dire des données exprimées dans une langue "naturelle". Ces données linguistiques peuvent être des textes écrits, ou bien des dialogues écrits ou oraux, ou encore des unités linguistiques de taille inférieure à ce que l'on appelle habituellement des textes, telles que : des phrases, des énoncés, des groupes de mots ou simplement des mots isolés. [4]

Parmi les objectifs pour lesquels la linguistique informatique et le traitement automatique sont apparus sont :

#### **➤ Un meilleur contact humain**

Aider l'utilisateur à accéder aux informations qu'il souhaite facilement et rapidement grâce à un résumé automatique et une analyse automatique des textes en plus de la recherche intelligente d'informations sur Internet dans ses nouvelles générations qui connaissent Le web sémantique. [12]

#### **➤ Accès efficace à l'information**

La navigation, le traitement des informations sur le Web nécessitent le développement de logiciels qui peuvent accéder aux informations des documents et des pages Web. Ainsi, la technologie du langage humain pour la gestion de contenu est une condition nécessaire pour convertir la richesse des informations numériques en connaissances collectives. Le contenu multilingue sur le Web représente un défi supplémentaire pour la technologie du langage, car Le Web mondial ne peut être contrôlé qu'à l'aide d'outils multilingues pour l'indexation et la navigation sur le Web, les systèmes multilingues de gestion des connaissances et de l'information permettront de surmonter les barrières linguistiques au commerce électronique et à l'enseignement à distance. [12]

#### **➤ Meilleure communication avec l'ordinateur**

Les méthodes de recherche dans la langue régulière permettent à une personne de communiquer avec un ordinateur avec n'importe quelle langue en français, allemand, arabe ou toute autre langue. La communication avec l'ordinateur dans la langue parlée aura un impact majeur sur l'environnement de travail, de nouveaux et vaste domaines émergeront face aux technologies de l'information.

➤ **Meilleure gestion des informations et connaissances**

L'explosion des connaissances nécessite le développement de moyens automatisés pour réguler cette quantité croissante d'informations et améliorer l'efficacité de leur stockage, de leur récupération ainsi que de leur emploi.

➤ **Création de système automatisé experts.**

L'émergence de systèmes automatisés experts qui peuvent diagnostiquer les maladies et fournir des conseils techniques, juridiques et des systèmes automatisés pour l'auto-éducation, qui nécessitent la capacité de dialoguer avec l'utilisateur humain dans une langue facile comme son langage naturel [12]

## **I.4 Les niveaux d'analyse d'un texte :**

Pour parvenir à une compréhension complète d'un énoncé en langage naturel, nous passons par différents niveaux de traitement : [3]

- **Le niveau morphologique** ou lexical qui s'intéresse à la décomposition d'un texte en différents mots (ex : en langue française le découpage s'effectue selon les espaces).
- **Le niveau syntaxique** définit comment les mots sont agencés dans une phrase, et l'analyse de la forme d'un texte.
- **Le niveau sémantique** qui correspond à la signification des mots et des groupes de mots (analyse du sens d'un texte (ex : typage)).
- **Le niveau pragmatique** grâce auquel, le contexte est pris en compte.

Ces différents niveaux sont illustrés dans la figure I-3.

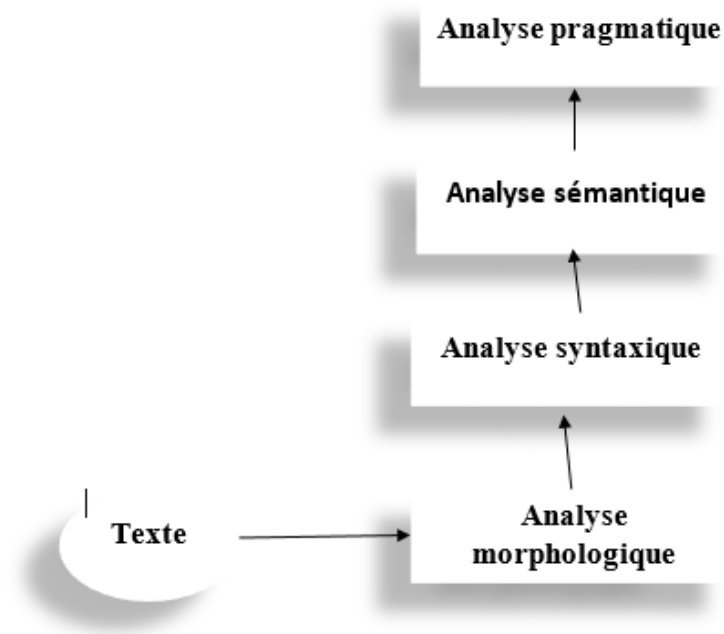


Figure 0-3:Les niveaux d'analyse d'un texte

#### **I.4.1 Analyse morphologique**

L'Analyse morphologique appelée aussi analyse morpho lexical, est liée à la syntaxe, qui s'occupe de la construction de la phrase. Dans cette phase, le texte est découpé et segmenté en un ensemble d'unités. Cette analyse permet d'interpréter comment les mots sont structurés et quels sont leurs rôles dans la phrase. [1]

##### **Exemple :**

Soit la chaîne de caractères : « Sofiane chante bien »

La segmentation se fera de la manière suivante :

$U_1$  = Sofiane

$U_2$  =Chante

$U_3$  = Bien

Maintenant, il est possible d'associer toutes sortes d'informations aux  $U_i(i = 1, 2, 3, \dots)$ . Par exemple :

U1= Sofiane

- Informations morphosyntaxiques : nom propre, masculin, singulier.
- Informations sémantiques : animé, humain, prénom

U2= chante

- Forme lemmatisée : chant
- Informations morphosyntaxiques (3ème personne, singulier) : verbe (chanter), passé (a chanté), indicatif (chante).

Idem pour U3.

### **I.4.2 Analyse syntaxique**

Définit l'agencement des mots pour obtenir des phrases bien formées ; la syntaxe définit également les relations entre les phrases et leurs constituants, donnant ainsi une description structurelle de chacun des types de phrases.

L'analyse syntaxique consiste à diviser une phrase en plusieurs groupes de mots appelés syntagmes.

Un syntagme est un ensemble de mots formant une seule unité catégorielle et fonctionnelle. Par exemple, on peut citer les types de syntagme suivants : syntagme nominal, syntagme verbal, syntagme adjectival, etc. [1]

Cette analyse est confrontée à divers règles, telles que les règles d'accord en genre, en nombre, ou positionnelles telles que celles qui contrôlent les positions relatives des mots dans la phrase. Le résultat de l'analyse syntaxique est représenté sous la forme d'un arbre, appelé arbre syntaxique.

Par exemple :

La représentation morphologique de la phrase « le chat mange la souris » est :

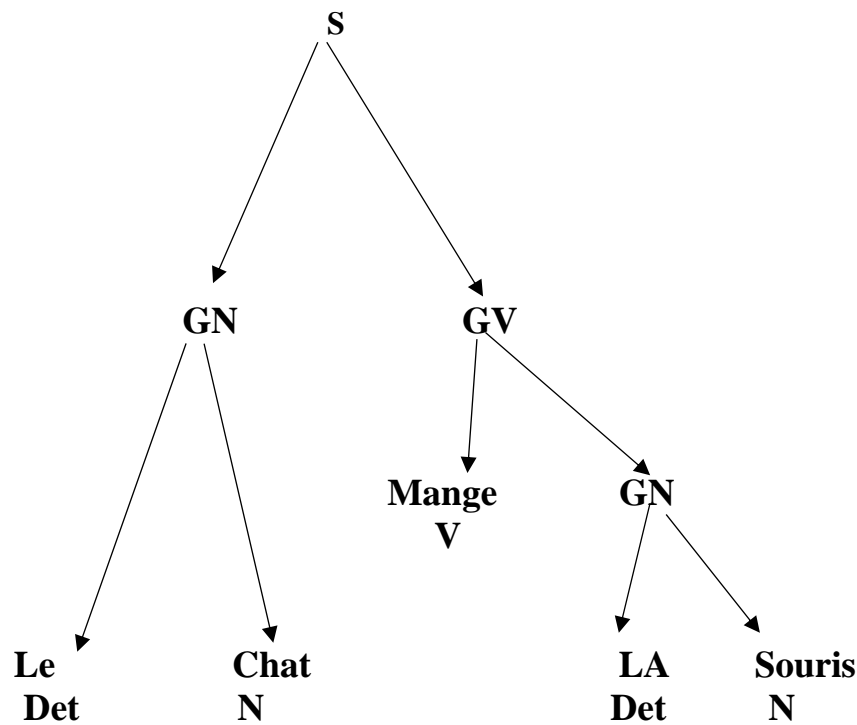
U1= le, U2= chat, U3= mange, U4= la, U5=la souris

Le résultat de l'analyse syntaxique est représenté l'arbre illustré dans la figure I-4 :

Où :

GN : Groupe nominal

GV : Groupe verbal



**Figure 0-4 : Représentation avec l'arbre syntaxique**

Dans l'arbre d'analyse syntaxique, les nœuds sont étiquetés par des symboles grammaticaux de telle manière que les feuilles sont étiquetées par des terminaux et les nœuds intérieurs par des non-terminaux, la racine est étiquetée par le symbole de départ de la grammaire.

### **I.4.3 L'analyse sémantique**

L'analyse sémantique se préoccupe du sens des énoncés, les connaissances sémantiques sont celles qui définissent si une phrase a un sens ou pas, pour cela on utilise un analyseur syntaxique qui va permettre d'attribuer un sens à la phrase structurée. Ces phrases ou (connaissances sémantiques) se composent d'un certain nombre de mots identifiés par l'analyse morphologique, et regroupés en structures par l'analyse syntaxique. Ces derniers constituent autant d'indices pour le calcul du sens d'une phrase. Dans l'analyse sémantique on vérifie par exemple que le programme est correctement typé ; qu'il n'y a pas de double déclaration. [1]

### **I.4.4 L'analyse pragmatique**

Le niveau pragmatique est parfaitement dissociable du niveau sémantique. Alors que la sémantique se préoccupe du sens des énoncés, la pragmatique porte sur les attitudes (vérité, désirabilité, probabilité) que les locuteurs adoptent vis à vis des énoncés.

Au niveau sémantique, on s'attend à ce que tout énoncé ait un sens au niveau pragmatique on attend que l'énoncé soit pertinent.[13]

## **I.5 Les domaines d'application du TALN**

La demande de TALN vient pour répondre à deux tendances : d'une part la nécessité de concevoir des interfaces de plus en plus ergonomiques, d'autre part la nécessité de pouvoir traiter (produire, lire, rechercher, classer, analyser, traduire) de manière plus « intelligente » les informations disponibles sous forme textuelle, de manière à résister à leur prolifération exponentielle. Les applications des techniques de TALN sont donc nombreuses et variées. Le traitement automatique du langage naturel à travers son parcours historique et ses divers succès et développements au cours desquels le traitement automatisé a réussi à creuser pour lui-même des domaines pratiques dans divers chemins, dont certains ont commencé à voir ses effets pratiques tels que les outils de traduction automatique actuellement disponibles gratuitement sur les moteurs de recherche célèbres tels que Google.

Nous traitons ci-dessous, certaines applications de traitement automatique de la langue qui incluent la traduction automatique, le résumé automatique, la génération automatique de la langue, l'extraction d'informations, la récupération d'informations, système de question réponse l'exploration de texte, la conversion de texte en mots parlés et la compréhension du son.

### **➤ L'indexation automatique**

L'indexation automatique est le processus informatisé de numérisation de grands volumes de documents par rapport à un vocabulaire, un thésaurus ou une ontologie contrôlés. L'utilisation de ces termes contrôlés pour indexer rapidement et efficacement de grands dépôts de documents électroniques... Il y a des parties qui s'ajoutent à cela, telles que la syntaxe, d'utilisation et d'autres algorithmes basés sur ce système et qui nécessitent l'indexation. Ceci est pris en compte à l'aide d'instructions booléennes pour rassembler et capturer les informations d'indexation hors du texte.

Comme le nombre de documents augmente de façon exponentielle avec la prolifération d'Internet, l'indexation automatique devient essentielle pour maintenir la capacité de trouver des informations pertinentes dans une mer d'informations non pertinentes. Les systèmes de langage naturel sont utilisés pour former un système basé sur sept méthodes différentes pour aider avec

cette mer d'informations non pertinentes. Ces méthodes sont morphologiques, lexicales, syntaxiques, numériques, sémantiques et pragmatiques ...

### ➤ **Le résumé automatique**

Une application de traitement automatique de la langue qui crée un texte bref à partir d'un fichier ou d'un document à l'aide d'un programme informatique, à condition que le texte abrégé contienne les idées les plus importantes dans le texte d'origine. Les logiciels développés pour fournir des résumés cohérents devraient prendre en compte plusieurs variables telles que la longueur, le style d'écriture et la construction afin de créer un résumé utile. On peut distinguer deux types de programmes de synthèse automatisés : les programmes d'extraction et d'abstraction. Le logiciel d'extraction repose sur la copie d'informations importantes pour le résumé (telles que des phrases de base et des paragraphes importants). Quant à l'abstraction ou au focus, il nécessite une reformulation. En général, l'abstraction et la reformulation sont plus puissantes et concentrent les informations plus que l'extraction, mais les programmes qui le font sont très difficiles à extraire. Programmation et développement car elle nécessite une technologie de génération de langage naturel, qui est toujours un domaine en croissance [12]

### ➤ **La Génération automatique de la langue**

Le langage automatisé signifie la création d'un texte dans un langage naturel à partir d'un système de représentation automatisé tel qu'une base de connaissances ou une forme logique. Certaines personnes considèrent la génération automatique de langage comme une contrepartie à la compréhension du langage naturel. Dans les langages de génération automatisée, le système doit prendre des décisions sur la façon de formuler un concept. Les applications les plus réussies pour la génération automatique de langue sont les systèmes de transformation de données en texte qui préparent des résumés de texte pour des données numériques et non linguistiques car ils mélangent l'analyse de données et la génération automatique de langue. [12]

### ➤ **L'Extraction d'informations**

L'extraction d'informations dans le traitement du langage naturel signifie la récupération d'informations, l'objectif étant d'extraire automatiquement des informations structurées et classifiées ainsi que le contexte et les connaissances de documents électroniques non organisés.



L'un des objectifs généraux de l'extraction d'informations est de déduire des inférences à partir du contenu logique des données saisies, cet objectif vient à la lumière de la croissance des informations sous des formes non organisées sur Internet, qui peuvent être facilement accessibles en les codant avec des codes XML, Sur Internet, facilement accessible en balisant les codes XML. Une application idéale pour extraire des informations est de numériser un ensemble de documents écrits en langage naturel et de remplir la base de données avec les informations consultées. Parmi les sous-tâches pour extraire des informations, distinguer les phrases nominales qui se réfèrent à la même chose, extraire les termes, c'est-à-dire trouver les termes ainsi que les relations pour un thésaurus de langue spécifique [12]

### ➤ **Les récupérations des informations**

Les applications de la recherche d'informations sont nées de la science de la recherche de documents et d'informations dans les ordinateurs, les bases de données et les référentiels de données, le World Wide Web et bien d'autres, les systèmes automatisés de recherche d'informations sont utilisés pour réduire le débordement de connaissances. De nombreuses universités et bibliothèques publiques utilisent des systèmes de recherche d'informations pour faciliter la recherche de livres périodiques et autres documents et les moteurs de recherche sont l'incarnation des applications de recherche d'informations. La relation entre la recherche d'informations et le traitement automatique du langage naturel consiste à développer les capacités du moteur de recherche ou de la base de données pour comprendre des phrases et des questions dans le langage courant, pour relier les systèmes de recherche d'informations aux systèmes de conversion de la parole en d'autres termes pour convertir du texte en discours parlé [12]

### ➤ **Les systèmes de questions / réponses**

Cette application est considérée comme l'une des applications de récupération d'informations car l'ordinateur est censé répondre à toutes les questions, en recherchant dans une énorme collection de documents et tels que le World Wide Web, pour répondre aux questions, il faut des méthodes de traitement automatique avancées pour les langues telles que la récupération de documents, beaucoup le considèrent comme l'étape après les moteurs de recherche. [6,12]

### ➤ **Fouille de texte (excavations / exploitation)**

Il se réfère au processus d'extraction d'informations de qualité à partir de textes, des informations de qualité sont dérivées de la division des modèles et des tendances par des moyens tels que l'apprentissage statistique des modèles. Le processus de prospection dans les textes comprend la structuration des textes entrés via la syntaxe avec l'ajout de fonctionnalités linguistiques dérivées, la suppression d'autres avantages, la prochaine entrée dans la base de données, la dérivation de modèles dans les données structurées et finalement, l'évaluation et l'interprétation des résultats. Les tâches d'exploration des textes comprennent la classification des textes, l'extraction de concepts et d'identités, la production de classifications graduées, etc. Les techniques d'exploration de texte sont utilisées dans les applications de protection, les soins médicaux, les logiciels et les applications, les résultats de recherche améliorés, ainsi que les applications académiques. [12]

#### ➤ **La conversion du texte en discours**

L'une des applications importantes du traitement automatique du langage naturel. Cette application consiste à convertir un texte écrit en parole audible et en parole parlée, le système informatique utilisé à cet effet est appelé synthétiseur vocal, le système de synthèse vocale convertit le texte de la langue normale en parole, soit. La parole synthétisée peut être créée en combinant des parties séquentielles de la parole enregistrée stockées dans une base de données de thésaurus parlé. Les systèmes diffèrent par la taille des unités vocales stockées. Elle permet de stocker des mots entiers ou des phrases complètes pour produire une parole de qualité. Une autre manière consiste à inclure un modèle de l'appareil vocal et d'autres caractéristiques de la voix humaine pour produire un son entièrement accordé. La qualité du synthétiseur vocal est jugée en fonction de sa similitude avec la voix humaine ou de l'étendue de sa compréhension, le programme de conceptualisation de la synthèse vocale permet aux aveugles et aux malvoyants d'écouter des œuvres écrites via un ordinateur. De nombreux systèmes d'exploitation informatiques incluent des synthétiseurs vocaux depuis le début des années 1980. [12]

#### ➤ **Dictée vocale**

Cette application diffère de la reconnaissance vocale de la précédente en ce qu'elle ne reconnaît pas un texte écrit et le convertit en discours parlé, mais écoute plutôt une voix audible

et la reconnaît et identifie son propriétaire, en convertissant le son en symboles que la machine comprend et reconnaît, il diffère également du terme reconnaissance vocale. La reconnaissance vocale signifie reconnaître la voix de l'orateur lui-même, pas les mots qu'il dit. Parmi les applications de la compréhension vocale figurent la communication vocale, le renvoi d'appels, le contrôle des appareils électroménagers, la recherche vocale de contenu, la saisie de données simple, la préparation de documents structurés, la conversion de la parole en texte écrit [12]

#### ➤ **L'analyse des sentiments**

L'analyse des sentiments à partir de sources textuelles dématérialisées sur de grandes quantités de données le big data.

Ce procédé apparaît au début des années 2000 et connaît un succès grandissant dû à l'abondance de données provenant de réseaux sociaux, notamment celles fournies par Twitter.

L'objectif de l'opinion mining est d'analyser une grande quantité de données afin d'en déduire les différents sentiments qui y sont exprimés. Les sentiments extraits peuvent ensuite faire l'objet de statistiques sur le ressenti général d'une communauté. Le but de l'analyse de données est de déterminer si le sentiment dégagé par une phrase est positif ou négatif. La principale difficulté de l'analyse réside au cœur même de l'utilisation de la langue. Le sentiment dégagé par une phrase dépend directement du contexte dans laquelle elle est utilisée et du type de langage [12]

#### ➤ **Reconnaissance optique des lettres**

OCR signifie la conversion mécanique ou électronique d'images d'écriture manuscrite, dactylographie ou de texte imprimé, qui est généralement capturé par le scanner en texte modifiable et lisible sur l'ordinateur. La technologie de reconnaissance optique a beaucoup progressé il existe des applications avancées pour la reconnaissance optique des lettres, mais elles sont très coûteuses et moins répandues. [12]

## **I.6 Les difficultés de TALN : ambiguïté et implicite**

Les difficultés que l'on rencontre en TALN sont principalement de deux ordres, soit de l'ambiguïté du langage, soit de la quantité d'implicite contenue dans les communications naturelles.

### **I.6.1 L'ambiguïté**

Le langage naturel est ambigu, cette ambiguïté se manifeste par la multitude d'interprétations possibles pour chacune des entités linguistiques pertinentes pour un niveau de traitement, comme en témoignent les exemples suivants :

- Ambiguïté des graphèmes (lettres) dans le processus d'encodage orthographique : comparez la prononciation du « i dans : lit, poire, maison.
- Ambiguïté des terminaisons dans les processus de conjugaison et d'infection : ainsi un /s /final marque à la fois le pluriel des noms (Les hommes), des adjectifs (belles), et la deuxième (parfois également la première) personne du singulier des formes verbales ; par exemple : je comprends, tu comprends
- Ambiguïté dans les propriétés grammaticales et sémantiques (i.e. associées à son sens) d'une forme graphique donnée : ainsi manges est ambigu à la fois morpho-syntaxiquement, puisqu'il correspond aux formes indicative et subjonctive du verbe manger), mais aussi sémantiquement. En effet, cette forme peut aussi bien référer (dans un style familier) à un ensemble d'actions conventionnelles (comme de s'asseoir à une table, mettre une serviette, utiliser divers ustensiles, ceci éventuellement en maintenant une interaction avec un autre humain) avec pour visée finale d'ingérer de la nourriture (auquel cas il ne requière pas de complément d'objet direct) ; et à l'action consistant à effectivement ingérer un type particulier de nourriture (auquel cas il requiert un complément d'objet direct), etc. Comparez en effet :

(a) Demain, Paul mange avec ma sœur.

(b) Paul mange son pain au chocolat.

À partir de ces deux énoncés les déductions que l'on peut faire :

De (a), on peut raisonnablement conclure que Paul sera assis à une table, disposera de couverts, tout ceci n'est pas nécessairement vrai dans le cas de l'énoncé (b). [5]

- Ambiguïté de la fonction grammaticale des groupes de mots, illustrés par la phrase :  
(3) il poursuit le voleur à vélo .

Dans cet exemple à vélo est soit un complément de manière de poursuivre (et s'est-il qui pédale), soit un complément de nom de voleur (et c'est lui qui mouline) ;

- Ambiguïté de la portée des quantificateurs, des conjonctions, des prépositions. Ainsi, dans :
    - (4) Tous mes amis ont pris un verre, on peut supposer que chacun avait un verre différent, mais dans
    - (5) Tous les témoins ont entendu un cri, est probable que c'était le même cri pour tous les témoins. De même, lorsque l'on évoque les chiens et les chats de Paul, l'interprétation la plus naturelle consiste à comprendre de Paul comme le complément de nom du groupe les chats et les chiens ; cette lecture est beaucoup moins naturelle dans les chiens de race et les chats de Paul ;
  - Ambiguïté sur l'interprétation à donner en contexte à un énoncé. Comparez ainsi la « signification » De non, dans les deux échanges suivants :
  - (6)
    - (a) Si je vais en cours demain ? Non (négation)
    - (a) Tu vas en cours demain ! Non ! (j'y crois pas)
- Conformément au parti pris de ce cours d'introduction, nous avons surtout insisté sur les ambiguïtés de reconnaissance (compréhension), mais les problèmes se posent naturellement de manière symétrique pour ce qui est de la génération : comment choisir les phrases produites de manière à limiter les ambiguïtés pour le récepteur ? Comment sélectionner parmi un ensemble de synonymes ? Parmi un ensemble de paraphrases ?

## **I.6.2 Implicite**

L'activité langagière s'inscrit toujours dans un contexte d'interaction entre deux humains, sensément dotés d'une connaissance du monde et de son fonctionnement telle que l'immense majorité des éléments de contexte nécessaires à la désambiguïsation mais aussi à la compréhension d'un énoncé naturel peuvent rester implicites. [16] La situation change du tout au tout dès qu'une machine tente de s'insérer dans un processus de communication naturel avec un humain : la machine ne dispose pas de cette connaissance d'arrière-plan, ce qui rend la compréhension complète de la majorité des énoncés difficile, voire impossible, si l'on ne dispose pas de bases de connaissance additionnelles, donnant accès à la fois à un savoir sur le monde (ou le domaine) en général (connaissance statique) et sur le contexte de l'énonciation

(connaissance dynamique). Un exemple typique est la désambiguïsation du référent du pronom personnel il dans les trois énoncés suivants : Le professeur envoya l'élève chez le censeur parce qu' ...:(7)

- a. ...il lançait des boulettes (il réfère probablement à l'élève)
- b. ...il voulait avoir la paix (il réfère probablement au professeur)
- c. ...il voulait le voir (il réfère probablement au censeur) [5]

## **I.7 Les outils du TALN**

Le traitement automatique du langage naturel (TALN) est une discipline de l'intelligence artificielle qui exploite la linguistique et l'informatique pour rendre le langage humain intelligible aux machines. Tout en permettant aux ordinateurs d'analyser automatiquement des ensembles massifs de données, les outils du TALN naturel peuvent nous aider à découvrir des informations précieuses dans du texte non structuré et à résoudre divers problèmes d'analyse de texte, tels que l'analyse des sentiments, la classification des sujets...

Nous allons citer ci-dessous les outils open-source les plus utilisés et les plus connus :

### **I.7.1 Stanford NLP Group Software**

L'un des groupes de recherche les plus importants dans le domaine du traitement automatique du langage naturel. Ils proposent un ensemble d'outils pour traiter et comprendre les langues humaines. Ils permettent de définir la forme de base des mots (segmentation en unité), la fonction des mots (étiquetage morphosyntaxique qui consiste à associer aux mots d'un texte les informations grammaticales correspondantes) et la structure des phrases (arbre syntaxique qui permet de représenter la structure syntaxique d'une phrase). De plus, il existe des outils pour les processus compliqués comme le deep Learning pour lequel le contexte de la phrase est pris en compte. L'ensemble des programmes du Stanford NLP sont écrits en langage Java et sont disponibles en plusieurs langues telles que le français, anglais, allemand, espagnol et chinois. [8,10]

### **I.7.2 Le Natural Language Toolkit**

C'est un ensemble d'outils TALN en langage Python. L'outil propose un accès à plus de 100 corpus de textes, parmi lesquels des textes en anglais, portugais, polonais...etc

De plus, le kit peut effectuer le traitement de différents textes, comme l'étiquetage morphosyntaxique, l'arbre syntaxique, la segmentation (*tokenisation*, la première étape du TALN) et la synthèse de texte. Le kit d'outils TALN comporte également une introduction à la programmation et une documentation détaillée. [8]

### **I.7.3 Visualtext**

C'est un ensemble d'outils écrits dans un langage de programmation propre au TALN : le langage NLP++.

Il est considéré comme étant un outil idéal pour développer rapidement des systèmes d'extraction d'informations précis et rapides, depuis une grande quantité de textes. Il permet par exemple de résumer des textes longs mais aussi de regrouper des événements sur un thème précis à partir de plusieurs sites Web [13] il est utilisé dans divers types d'applications telles que Des systèmes comme l'Indexation qui prendre en charge des capacités de recherche de qualité sur le World Wide Web, mais également dans le filtrage pour déterminer si un document est pertinent ou pas [7]

### **I.7.4 NooJ**

Nooj est considéré comme un moteur linguistique, il intègre les outils de traitement automatique du langage conçu spécialement pour l'enseignement des langues et de la linguistique.il permet aux utilisateurs de traiter de grands ensembles de textes Les utilisateurs peuvent construire, et gérer des grammaires morphologiques et syntaxiques organisées dans des bibliothèques réutilisables. Il traite les textes qui sont annoté ; les annotations sont stockées dans chaque texte .Il offre également possibilités de gérer des procédures de recherche, de test, et d'entraînement pour l'étudiant. [12 ; 11]

## Conclusion

Nous avons explicité à travers ce chapitre les différentes connaissances liées au traitement automatique du langage naturel. On a vu qu'il représente un défi colossal dans le domaine de l'informatique, en effet le langage peut être à double sens et pour bien le comprendre il est nécessaire de connaître le contexte dans lequel il s'insère. De nombreux progrès restent à accomplir pour mieux comprendre cette faculté et pour bâtir des systèmes capables de soutenir la comparaison avec l'être humain, mais l'état des connaissances nous permet aujourd'hui de proposer de nombreuses solutions efficaces à des problèmes et des demandes réels.



## CHAPITRE 2 : L'apprentissage automatique

## **Introduction à l'apprentissage automatique**

La question de l'apprentissage fascine les spécialistes de l'informatique et des mathématiques tout autant que neurologues, pédagogues. Une définition qui s'applique à un programme informatique comme à un robot, un animal de compagnie ou un être humain est celle proposée par Fabien Benureau (2015) : « L'apprentissage est une modification d'un comportement sur la base d'une expérience »

En effet l'apprentissage automatique se trouve au carrefour de nombreux autres domaines : intelligence artificielle, statistiques, théorie des probabilités, il est considéré comme un champ d'études en mutation constante qui a su s'imposer, il a été utilisé depuis des décennies

Il fait référence à des algorithmes informatiques qui permettent à une machine d'évoluer grâce à un processus d'apprentissage. Il repose sur deux piliers fondamentaux : d'une part, les données, qui sont les exemples à partir duquel l'algorithme va apprendre ; et d'autre part, l'algorithme d'apprentissage, qui est la procédure que l'on fait tourner sur ces données pour produire un modèle. On appelle entraînement le fait de faire tourner un algorithme d'apprentissage sur un jeu de données.

Le chapitre qui suit a pour but de présenter et de faire une introduction au domaine de l'apprentissage automatique. En expliquant les concepts clés qui le composent.

### **II.1 Définition de l'apprentissage automatique**

L'apprentissage automatique Machine Learning (ML) est une branche de l'intelligence artificielle (IA) qui consiste à programmer des algorithmes permettant d'apprendre automatiquement à partir des données et d'expériences passées ou par des interactions avec l'environnement.

L'apprentissage automatique se fonde sur des approches mathématiques et statistiques pour donner aux ordinateurs la capacité d'apprendre à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des tâches sans être « explicitement programmés » pour chacune. Plus largement, il concerne la conception, l'analyse, l'optimisation, développement et l'implémentation de telles méthodes. [36]

L'ensemble du processus d'apprentissage nécessite un ensemble de données comme suit :

- ✓ Ensemble de données pour l'entraînement : c'est la base de connaissance utilisée pour entraîner notre algorithme d'apprentissage. Pendant cette phase, les paramètres du modèle peuvent être réglés (ajustés) en fonction des performances obtenues.
- ✓ Ensemble de données pour le test : cela est utilisé juste pour évaluer les performances du modèle sur les données non vues.

## **II.2 Exemples de tâches d'apprentissage automatique**

Les algorithmes de machine learning sont capables d'effectuer des tâches diverses et variées, selon le domaine d'application. Voici un exemple de tâches que peut effectuer un algorithme d'apprentissages automatiques :

Exemple 1 : Supposons que l'on dispose d'une collection d'articles de journaux. L'algorithme d'apprentissage automatique permet dans ce cas d'identifier des groupes d'articles portant sur un même sujet (faire une classification d'articles).

Exemple 2 : Supposons que l'on dispose d'une base de données regroupant les caractéristiques de logements dans une ville : superficie, quartier, étage, prix, année de construction, nombre d'occupants, montant des frais de chauffage. L'algorithme d'apprentissage automatique permet dans ce cas de prédire la facture de chauffage à partir des autres caractéristiques pour un logement qui n'appartiendrait pas à cette base.

Exemple 3 : Supposons que l'on dispose d'un certain nombre d'images représentant des chiens, et d'autres représentants des chats. Comment classer automatiquement une nouvelle image dans une des catégories « chien » ou « chat » ? Avec les algorithmes d'apprentissages automatiques on pourra faire une classification ces animaux.

## **II.3 Les méthodes d'apprentissages automatiques**

Il existe plusieurs façons d'apprendre automatiquement à partir des données, dépendamment des problèmes à résoudre et des données disponibles. Les algorithmes d'apprentissage peuvent se catégoriser selon le mode d'apprentissage qu'ils emploient : on peut les classer en trois catégories distinctes : apprentissage supervisé, apprentissage non supervisé, apprentissage par renforcement.

La figure ci-dessous illustre les méthodes de l'apprentissage automatique :

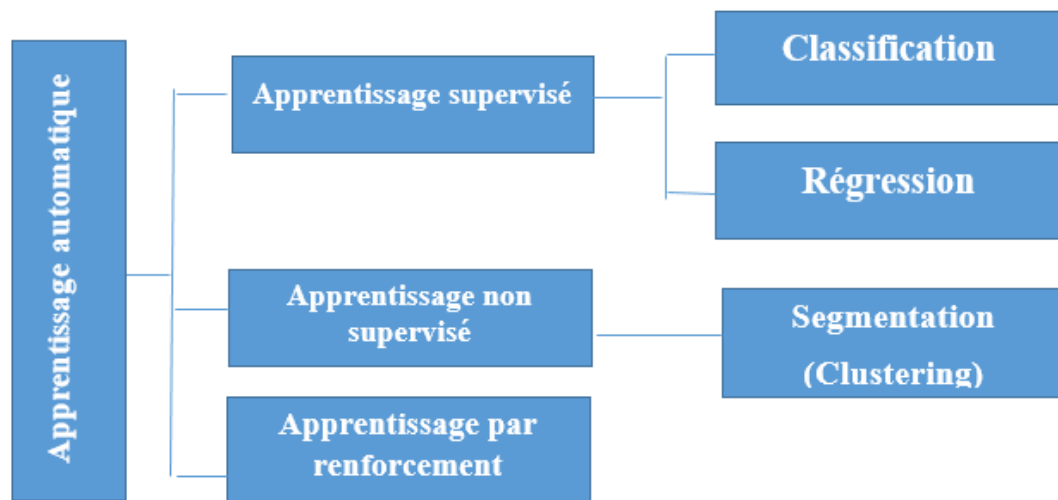


Figure 5: Méthodes d'apprentissage automatique.

## II.4 Les types de données utilisées dans l'apprentissage automatique

Avant de détailler les algorithmes d'apprentissage automatique, il est nécessaire de connaître les types de données utilisées. En effet, les algorithmes d'apprentissage automatique sont basés sur des données.

Selon Stanford, la machine Learning est une technologie de l'intelligence artificielle permettant aux ordinateurs d'apprendre sans avoir été programmés. Pour apprendre et se développer, les ordinateurs ont besoin de données pour s'entraîner. Les données sont donc très importantes pour l'apprentissage. Il existe différents types de données : [28]

### II.4.1 Données structurées

Elles résident généralement dans des bases de données relationnelles (SGBDR) et sont constituées de champs avec des types de données prédéfinis. Les données structurées sont facilement analysées par la plupart des algorithmes d'apprentissage automatique fonctionnent avec des données structurées.

### II.4.2 Données semi-structurées

Elles conservent un type de données contenant des balises sémantiques qui identifient des éléments de données distincts, ce qui permet le regroupement d'informations et les hiérarchies.

### II.4.3 Données non-structurées

Elles sont composées de types de fichiers, tels que des fichiers texte, audio, vidéo, et des publications sur les réseaux sociaux, qui peuvent être plus difficiles à analyser et à exploiter. Les données non structurées sont généralement prétraitées pour en extraire des données structurées.[28]

## II.5 Modèles de l'apprentissage automatique

### II.5.1 Apprentissage supervisé

L'apprentissage supervisé (Supervised Learning en anglais) permet aux utilisateurs de préciser les classes voulues selon les besoins spécifiques de l'application. Il consiste à apprendre une fonction de prédiction à partir d'exemples annotés [102], en d'autres termes à partir d'une base de données contenant des exemples de cas déjà étiquetés. L'objectif est de trouver des corrélations entre les données d'entrée (variables explicatives) et les données de sorties (variables à prédire), pour ensuite inférer la connaissance extraite sur des entrées avec des sorties inconnues. [25]

On appelle alors fonction d'apprentissage la fonction notée :  $\gamma$  qui associe un résultat supervisé à toute nouvelle entrée qui pourrait lui être présentée.

En apprentissage supervisé, on distingue entre deux types de tâches :

- **La classification** : quand la variable cible (à prédire) est discrète,  $\gamma = \{1, \dots, I\}$ . Ce qui revient à attribuer une classe (ou étiquette) à chaque entrée. C'est le cas si on cherche à prédire la tendance d'un mouvement futur d'un actif (haut, neutre, bas).
- **La régression** : quand la variable cible à prédire est continue,  $\gamma \in \mathbb{R}$  Exemple : prédire le prix futur en dollars de l'actif en question. [28]

Voici des exemples d'applications de l'apprentissage supervisé :

- Diagnostic médical : diagnostiquer un patient comme un malade ou non malade d'une maladie – classification.
- Prévision météo : prédire, par exemple : si oui ou non il va pleuvoir demain – classification.
- Prédire le prix de la location dans une région – régression.

### **II.5.2 L'apprentissage non supervisé**

L'apprentissage non supervisé consiste à apprendre sans superviseur. Cela signifie qu'il ne nécessite pas d'étiquetage et essaie plutôt de découvrir des clusters (groupes de données proches entre elles) dans un ensemble de données non étiquetées. Mais les clusters découverts ne correspondent pas toujours aux classes spécifiques voulues par l'utilisateur. Pour ce type d'apprentissage la base de données d'apprentissage ne contient pas de variable cible (comme on l'a vu en apprentissage superviser). Il y a seulement un ensemble de données collectées en entrée. L'algorithme doit découvrir par lui-même la structure en fonction des données. On utilise cette technique pour partitionner les données en « groupes » d'éléments homogènes. [30]

On distingue l'apprentissage supervisé et non supervisé. Dans le premier apprentissage, il s'agit d'apprendre à classer un nouvel individu parmi un ensemble de classes prédéfinies : on connaît les classes a priori. Tandis que dans l'apprentissage non-supervisé, le nombre et la définition des classes ne sont pas donnés a priori. Ce type d'apprentissage est important car il est beaucoup plus commun dans le cerveau humain que l'apprentissage supervisé. [36,29,35]

Voici des exemples d'applications de l'apprentissage non-supervisé :

- Marketing : segmentation du marché en découvrant des groupes de clients distincts à partir de bases de données d'achats.
- Planification de villes : identification de groupes d'habitations suivant le type d'habitation, valeur, localisation géographique, etc

### **II.5.3 Apprentissage semi supervisé**

L'apprentissage semi supervisé utilise un ensemble de données étiquetées et non étiquetées. Il se situe ainsi entre l'apprentissage supervisé qui n'utilise que des données étiquetées et l'apprentissage non supervisé qui n'utilise que des données non étiquetées.

L'apprentissage semi-supervisé est souvent associé au concept d'apprentissage transductif. Cela signifie que l'apprentissage s'effectue sur les données de la base d'apprentissage (d'entraînement) dans le but de faire des prédictions sur les observations de la base de test. [30]

Un exemple de cas d'application est en médecine, où il peut constituer une aide au diagnostic.

### **II.5.4 Apprentissage par renforcement**

L'apprentissage se fait sans supervision, par interaction avec l'environnement (principe d'essai/erreur). En observant le résultat des actions prises. Chaque action de la séquence est associée à

une récompense. L'agent est plongé au sein d'un environnement et il prend ses décisions en fonction de son état courant. En retour, l'environnement procure à l'agent une récompense, qui peut être positive ou négative. L'agent cherche, au travers d'expériences itérées, à déterminer la stratégie comportementale optimale afin de maximiser la récompense totale. Pour cela, un simple retour des résultats est nécessaire pour apprendre comment la machine doit agir. Ceci est appelé le signal de renforcement. [36,29]

Exemple de cas d'application de l'apprentissage par renforcement :

- L'algorithme de E-learning ou Td-Learning (dynamique) : Ce type d'apprentissage est appliqué dans de nombreux domaines comme les jeux (dames), la robotique, pilotage automatique, etc.

## **II.6 Objectifs des méthodes d'apprentissage automatique.**

L'apprentissage automatique a pour objectif d'extraire et d'exploiter automatiquement l'information présente dans un jeu de données.

L'apprentissage automatique couvre un vaste champ d'objectifs comme la fouille de données, la classification non supervisée, la sélection de variables, la discrimination, la régression, la sélection de modèle, etc.

Dans ce qui suit nous allons présenter les principaux objectifs de l'apprentissage automatique

### **II.6.1 La classification**

La classification consiste à inférer, à partir d'un échantillon d'objets classés, une procédure de classification en d'autre terme cela consiste à attribuer une classe (aussi appelée étiquette ou label) pour chaque valeur d'entrée. [28]

### **II.6.2 La régression**

Similaire à la classification, sauf que la classification se rapporte à des événements discrets. L'estimation porte sur des variables continues. Par exemple en prévoyant les prix d'un logement en fonction de l'âge de la maison de l'emplacement du quartier, du nombre de pièces.

Prédire le nombre de clics sur un lien ou par exemple Prédire le nombre d'utilisateurs et utilisatrices d'un service en ligne à un moment donné.

### **II.6.3 La segmentation**

Consiste à former des sous-groupes (clusters) relativement homogènes à l'intérieur d'une population hétérogène. Dans ce cas, les classes ne sont pas prédéfinies et pour cette tâche, il n'y a pas de classe à expliquer, il appartient à un expert du domaine de déterminer l'intérêt et la signification des sous-groupes ainsi constitués. [28]

## **II.7 Introduction aux réseaux de neurones**

Dans cette deuxième partie nous présenterons les réseaux de neurones, les aspects liés à leur fondement, ainsi que leurs différentes architectures.

Le domaine des réseaux de neurones n'est pas nouveau, car il a son origine dans des travaux conduits durant les années 40 (modèle de Hebb pour l'évolution des connexions synaptiques). Ces travaux conduisirent au modèle du perceptron dans les années 60 (modèle qui a principalement été appliqué à la reconnaissance de caractères). Mais ce n'est qu'à partir de 1986 que la recherche dans ce domaine a connu une expansion importante du fait de la publication de modèles de réseaux et d'algorithmes d'apprentissage suffisamment efficaces pour résoudre des problèmes réalistes et complexes. [26]

## **II.8 Présentation d'un réseau de neurone**

Le réseau de neurones artificiels a été inspiré du modèle biologique, c'est-à-dire le modèle du cerveau humain. Le cerveau humain est composé d'un grand nombre de cellules nerveuses. La cellule nerveuse humaine contient des neurones. Ces derniers ont une partie cellulaire et une partie de noyau.

Un neurone est une cellule du système nerveux, il est composé de : [31]

- Les dendrites : ce sont des réseaux réceptifs arborescents de fibres nerveuses qui conduisent les signaux électriques dans la cellule.
- Le corps cellulaire : il somme efficacement les signaux reçus et en fixe le seuil.
- L'axone : c'est une simple fibre longue qui conduit le signal de la cellule aux autres neurones. Le point de contact entre un axone d'une cellule et une dendrite d'une autre cellule est appelé synapse.



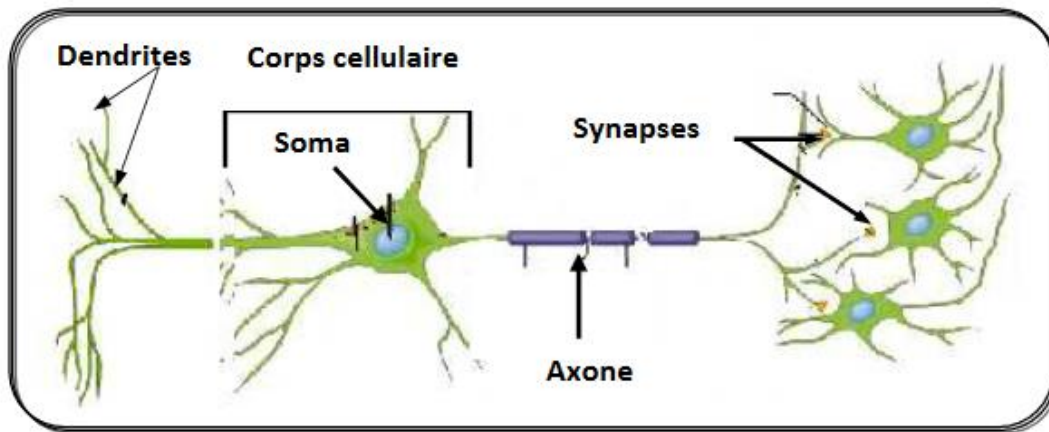


Figure 6: Architecture d'un neurone

Le premier neurone remonte à 1943 quand Pitts et MM. Mac Culloch ont tenté de modéliser d'une manière mathématique le cerveau humain, ils ont réalisé un neurone avec une sortie binaire. Ils ont démontré théoriquement qu'un réseau de ces neurones formels peut reproduire des fonctions logiques et mathématiques. [31]

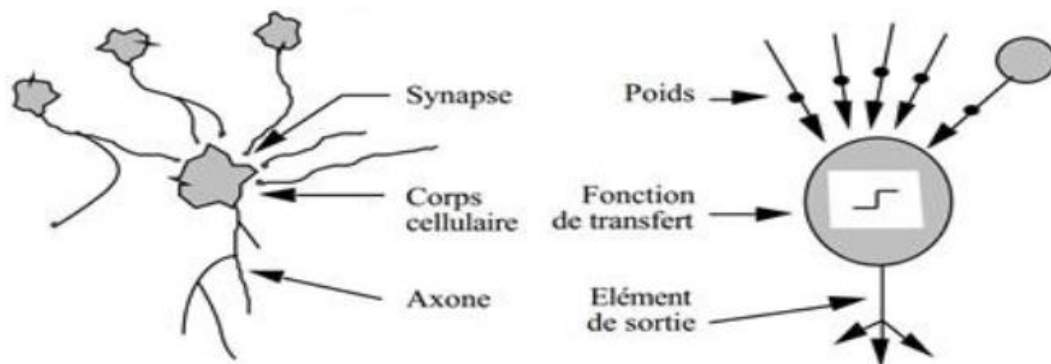


Figure 7: La différence entre un réseau de neurone biologique et un réseau de neurone artificiel

Le passage d'un réseau de neurones biologique à un réseau de neurones formels est modélisé dans le tableau ci-dessous

Neurone biologique	Neurone artificiel
Synapses	Poids de connexions
Axones	Signal de sortie
Dendrite	Signal d'entrée
Somma	Fonction d'activation

Tableau 1: Passage d'un réseau biologique à un réseau de neurone artificiel

## II.9 Topologie d'un réseau de neurone

On appelle topologie des réseaux la façon dont les neurones sont connectés entre eux.

Selon le type du problème à résoudre, Le nombre de ces couches est défini. Les couches d'entrée et de sortie sont reliées par une couche appelée couche intermédiaire.

En général on peut distinguer deux grandes classes de réseaux de neurones artificiels selon leurs topologies :

- Les réseaux à couches qui sont des réseaux de neurones dans lesquels l'information se propage couche par couche sans retour en arrière possible,
- Et les réseaux récurrents qui sont des réseaux de neurones dans lesquels il y a une liaison vers l'arrière.

Ces réseaux de neurones sauvegardent les résultats produit par les nœuds de traitement puis ils nourrissent le modèle avec ces résultats

Il existe dans un réseau de neurone une fonction qui permet de transformer le signal entrant dans un neurone en un signal de sortie (réponse). Cette fonction est appelée fonction d'activation.

## II.10 Les couches du réseau de neurones

- **Couche d'entrée** : les neurones de cette couche reçoivent les valeurs d'entrée du réseau et les transmettent aux neurones cachés. Chaque neurone reçoit une valeur, il ne fait pas donc de sommation.
- **Couches cachées** : chaque neurone de cette couche reçoit l'information de plusieurs couches précédentes, effectue la sommation pondérée par les poids, puis la transforme selon sa fonction d'activation. Par la suite, il envoie cette réponse aux neurones de la couche suivante.
- **Couche de sortie** : elle joue le même rôle que les couches cachées, la seule différence entre ces deux types de couches est que la sortie des neurones de la couche de sortie n'est liée à aucun autre neurone.

### II.10.1 Comportement et structure d'un réseau de neurones artificiels

Le réseau reçoit des entrées qu'il vient moduler selon plusieurs paramètres (taux d'apprentissage, nombre de neurones), afin de fournir un résultat.

La figure ci-dessous montre la structure d'un réseau de neurones artificiels :

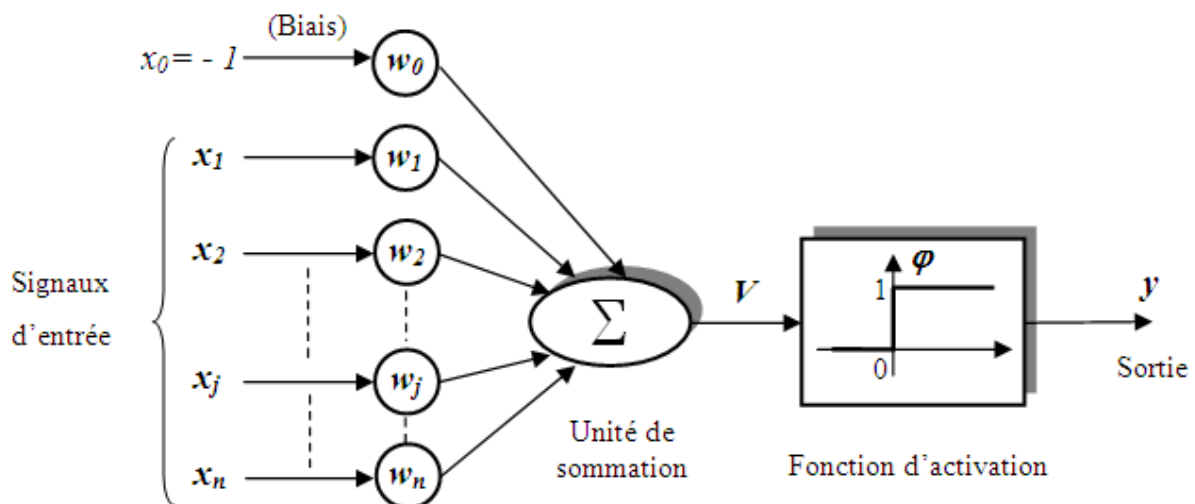


Figure 8: Structure d'un réseau de neurones artificiels.

Où :

$\{x_1, x_2, x_3, \dots, x_n\}$  sont les entrées externes.

$y$  Est la sortie.

$\{ \omega_1, \omega_2, \omega_3 \dots \omega_n \}$  Sont les poids associés à chaque connexion.

$\chi$  Est associé au vecteur d'entrée  $v$ ,

$\omega'$  Est le vecteur poids,

$\phi$  Est Appelé le biais

La fonction  $\phi$  : appelée fonction d'activation.

Dans un réseau de neurones artificiels, chaque neurone est un processeur élémentaire, il reçoit un nombre variable d'entrée en provenance de neurones amont. A chacune de ces entrées est associé un poids représentatif de la force de connexion. Le neurone est doté d'une sortie unique. [68]

## **II.11 Architecture de réseaux de neurones**

Selon la topologie de connexion et la propagation des informations entre les différents tiers (nœuds) on peut les classer en deux catégories : Réseaux non bouclés (statique ou feed forward) et réseaux bouclés (Dynamique, feed back ou récurrent). [34 ,27]

### **II.11.1 Réseaux statiques "feed-forward "**

Un réseau de neurones non bouclé (appelé aussi statique) est représenté comme un Graphe dont les nœuds sont les neurones. L'information circule des entrées vers les sorties sans retour en arrière. Dans ce type de réseaux de neurones, le signal se propage dans le réseau d'une manière linéaire. [34]

### **II.11.2 Réseaux multicouches (ou Perceptron Multi Couche PMC)**

C'est le réseau de neurones statique le plus utilisé, ce genre de réseau possèdent plusieurs couches cachées. Les neurones sont arrangés par couche. Les neurones de la première couche reçoivent le vecteur d'entrée, ils calculent leurs sorties qui sont transmises aux neurones de la seconde couche qui calculent eux-mêmes leurs sorties et ainsi de suite de couche en couche jusqu'à celle de sortie. Chaque neurone dans la couche cachée est connecté à tous les neurones de la couche précédente et de la couche suivante. [37,33]

### **II.11.3 Réseau récurrents "Feed-back"**

Ils sont aussi appelés aussi réseaux récurrents, ce sont des réseaux dans lesquels le retour en arrière est possible, parmi ces réseaux on retrouve :

- Cartes auto-organisatrices de Kohonen : C'est un réseau à apprentissage non-supervisée, il est sous forme d'une grille dont chaque nœud représente un neurone associé à un vecteur de poids. [32]
- Réseaux de Hopfield : ce sont des réseaux entièrement connectés. Dans ce type de réseau, chaque neurone est connecté à chaque autre neurone et il n'y a aucune différenciation entre les neurones d'entrée et de sortie. Ils sont capables de trouver un objet stocké en fonction de représentations. [32]

## **Conclusion**

Les Réseaux de Neurones Artificiels sont considérés comme des approches très intéressantes dans le domaine de l'Intelligence Artificielle. Ils sont connus par leur puissance d'apprentissage et généralisation. Ils constituent donc un domaine à ne pas négliger.

Au cours de ce deuxième chapitre on a vu l'apprentissage automatique avec des exemples concrets de leurs cas d'utilisations, on a aussi expliqué les différents types de données utilisés dans l'apprentissage avec les différents algorithmes. Dans la deuxième partie de ce chapitre on a présenté les réseaux de neurones, leurs topologies ainsi que leurs différentes architectures, et certaines méthodologies d'apprentissage, pour enfin finir avec une conclusion.

## CHAPITRE 3: Le Word Embedding

## Introduction

Les recherches récentes en traitement automatique du langage ont permis d'atteindre des performances proches des capacités humaines, notamment en ce qui concerne la détection de paraphrase ou l'évaluation de la similarité sémantique entre deux phrases. Cependant, ces avancées, fondées sur l'entraînement de réseaux de neurones sur de très vastes corpus de textes, sont à nuancer. En effet, malgré des progrès rapides ces dernières années dans l'adaptabilité des modèles de traitement du langage il reste difficile d'adapter ces modèles à de nouvelles tâches. Dans ce chapitre nous allons présenter l'un de ces modèles qui est le Word Emebeddings.

### III.1 Présentation des Word Emebeddings

Word Emebeddings désigne un ensemble de techniques de machines Learning qui visent à représenter les mots ou les phrases d'un texte par des vecteurs de nombres réels appelé « vecteur dense », où les mots qui ont le même sens ont une représentation similaire.

Le Word Emebeddings possède différentes appellations, neural Emebeddings ou prédiction-based Emebeddings, ou en français représentation vectorielle du mot, représentation continue du mot et aussi plongement de mot [39]. Ce modèle consiste à apprendre un réseau de neurones feed-forward pour estimer la probabilité du prochain mot, en s'appuyant sur la représentation continue des mots précédents. Cette représentation est apprise au fur et à mesure au moment de l'apprentissage du réseau de neurones.

Cette incorporation de mots permet de transformer le langage humain en une forme numérique. L'idée principale est que chaque mot peut être converti en un ensemble de nombres vecteur à N dimensions. Bien que chaque mot reçoive un vecteur par incorporation unique, des mots similaires finissent par avoir des valeurs plus proches les uns des autres, par exemple, les vecteurs pour les mots « femme » et « fille » auraient une similitude plus élevée que les vecteurs pour « fille » et « chat ». Le sens du mot peut également être pris en considération par exemple le mot « nourriture » et « faim » sont plus susceptibles d'être utilisés dans le même contexte que les mots « faim » et « logiciel ».

### III.2 Un bref historique sur le Word Embedding

Le Word Embeddings remonte aux années 1950, lorsque la distribution hypothèse [43] a été discutée en linguistique, cette hypothèse suggère que les mots utilisés dans les mêmes contextes ont tendance à prétendre à des significations similaires [44].

Le concept et les modèles de formation sur « l'intégration de mots » évoluent avec les progrès du, traitement du langage naturel, dans le traitement du langage naturel, une idée similaire appelée Vector Space Model (VSM) a été d'abord introduite dans la recherche d'informations [51, 52] pour déterminer documenter les similitudes. Sur la base de la matrice terme-document, dans ce vecteur les points qui sont rapprochés sont sémantiquement liés et les autres sont sémantiquement éloignés.

La notion de représentation sémantique distribuée est un autre élément important de Word Emebeddings, il vise à réduire le nombre de dimensions en utilisant des techniques comme la décomposition en valeurs singulières (SVD) consiste à la factorisation de la matrice et la sémantique latente (LSA).

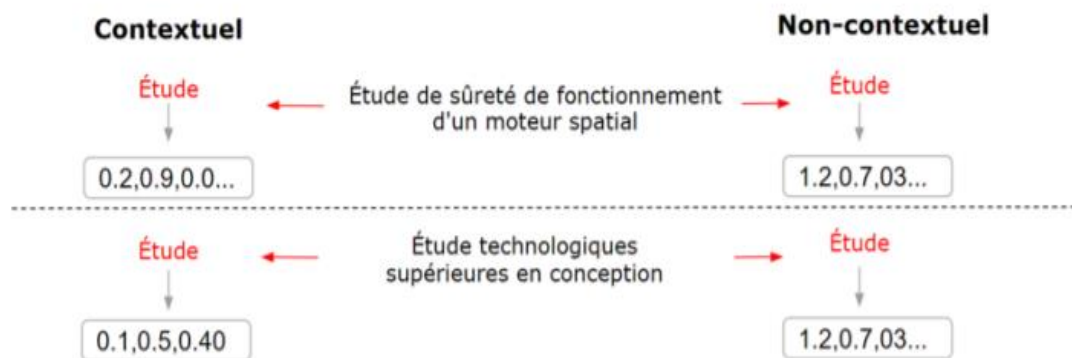
L'utilisation de la sémantique distribuée dans la représentation des mots ont apporté de nombreuses améliorations dans différentes tâches de NLP dans les années 2000.

En 2013 d'autres algorithmes ont vu le jour tel qu'ELMo (Embeddings from Language Models) [45] et BERT (Bidirectionnel Représentations des codeurs des transformateurs). [40]

### **III.3 Les modèles Word Embeddings**

Le domaine de Traitement automatique des langues ne cesse d'évoluer, notamment sur l'aspect de la transformation vectorielle du texte. Les modèles de Word Emebeddings sont des modèles mathématiques basés généralement sur des algorithmes de Deep Learning. Chaque modèle a sa propre architecture qui lui permet de transformer le texte en vecteurs, deux exemples sont présentés sur la figure 1. Nous pouvons remarquer que le mot Étude possède deux sens différents : dans la première phrase, il fait référence à l'analyse d'un sujet particulier, tandis que dans la deuxième, il fait référence à une formation supérieure. Dans ce sens, l'approche contextuelle va prendre ceci en considération et va donc générer deux vecteurs différents pour le mot Étude, contrairement à l'approche non contextuelle qui va générer un seul et même vecteur de ce mot quel que soit son contexte.





**Figure 9: Les modèles contextuels et non-contextuels**

Il existe plusieurs modèles de « Word Emebeddings ». Nous pouvons citer les modèles suivants : [41]

Version contextuelle	Version non contextuelle
Flair (Zalando Research 2018)	Word2vec (Google 2013-2014)
Bert (Google 2018)	Glove (Stanford 2015)
Elmo (Google 2018)	Fasttext (Facebook 2017)

Dans ce chapitre nous décrivons les modèles d’Emebeddings : Word2Vec

### III.3.1 Le model word2vec

Word2Vec est un outil d'apprentissage non supervisé basé sur les modèles probabilistes et des réseaux de neurones. *Il a été développé par une équipe de recherche de Google sous la direction de Tomas Mikolov et son équipe [50] ayant pour but d'apprendre les relations sémantiques entre les mots.* Cet outil permet de créer une représentation vectorielle de l'ensemble des mots d'un texte. [49].

Avec word2vec Les mots sont représentés sous forme de vecteurs numérique, de telle manière que les mots de sens similaire apparaissent ensemble et que les mots différents sont situés de manière éloignée. Word2vec permet de reconstruire le contexte linguistique des mots. Pour expliquer la signification d’un contexte linguistique en bref, l'objectif principal d'une phrase est

le contexte. Un mot ou une phrase composant le langage parlé ou écrit aide à déterminer le sens du contexte. Word2vec apprend la représentation vectorielle des mots à travers les contextes. En d'autres termes, c'est à partir du contexte que le modèle word2vec sera créé. [41]

Word2Vec se présente sous deux architectures neuronales :

### **Le model CBOW (Continuous Bag Of Word)**

III.3.1.1 L'apprentissage des Word Emebeddings consiste à prédire un mot en fonction de son contexte. La couche d'entrée représente la présence ou l'absence des mots dans le contexte de manière binaire. Si le mot est présent dans le contexte alors on lui attribue un 1 et 0 s'il est absent. Ensuite ce mot est projeté dans la matrice des poids dans une couche appelée couche cachée. Puis, dans un troisième temps, la somme ou moyenne de ces représentations, est passée dans la couche de sortie. Il va également procéder à la rétro-propagation du gradient pour « s'autocorriger ». La rétro-propagation du gradient est une méthode qu'utilise word2vec automatiquement, pour rééquilibrer les poids de la première matrice, afin d'obtenir les meilleures prédictions possibles. Le but étant de se rapprocher le plus possible du bon contexte pour un mot, pour cela elle va soit augmenter, soit diminuer les poids de chacun des vecteurs. Ainsi, la méthode va se baser sur ce qui s'appelle une fonction de coût, c'est-à-dire que plus elle va être proche de zéro, plus le modèle sera performant et aura donc les bonnes prédictions .[41]

La figure 2 ci-dessous illustre l'architecture du modèle CBOW.

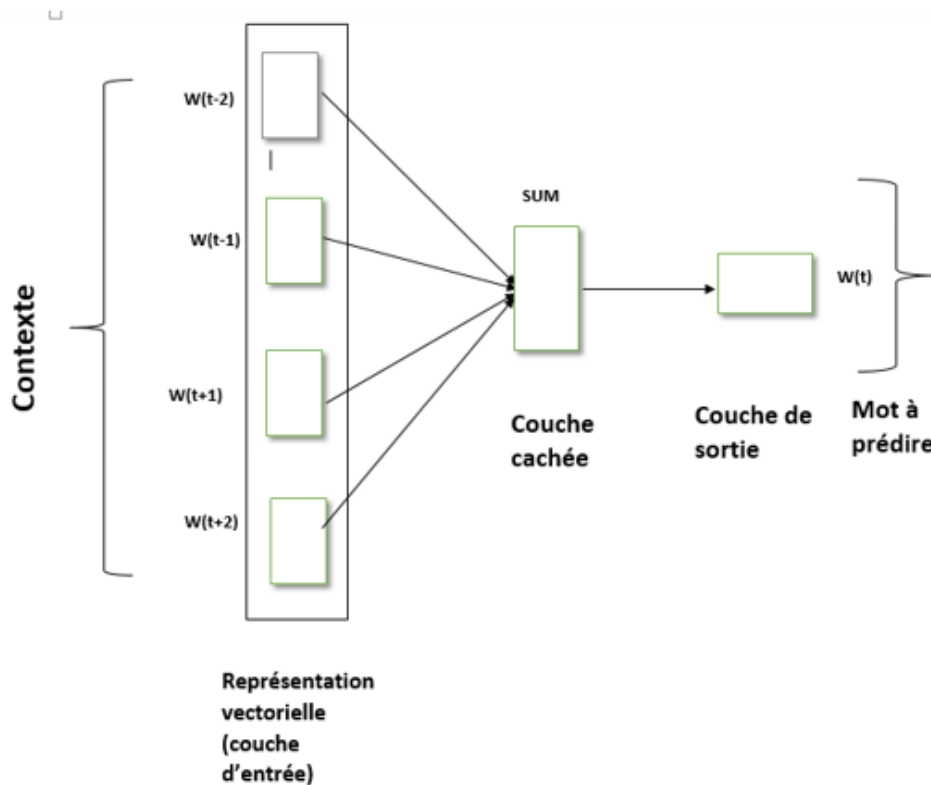


Figure 10: Architecture du modèle CBOW

### III.3.1.2

#### Le modèle skip gram

Skip-gram essaie de réaliser l'inverse de ce que fait le modèle CBOW, il essaie de prédire les mots du contexte source (mots environnants) à partir d'un mot cible (le mot central). Pour ce faire, dans la couche d'entrée, nous avons un vecteur contenant un seul mot. dans un premier temps, il va être projeté dans la couche cachée au moyen d'une matrice de poids. Cette matrice va attribuer des poids à chacun des vecteurs afin d'être projetés dans les neurones de la couche cachée. Ensuite, avec une seconde matrice, ils vont être projetés dans la couche de sortie. [41]

Le vecteur d'entrée va être comparé à chacun des mots du contexte afin de savoir si les prédictions étaient correctes ou non. Pour cela, il va également utiliser la rétro-propagation du gradient. La figure 3 ci-dessous illustre l'architecture du modèle Skip gram.

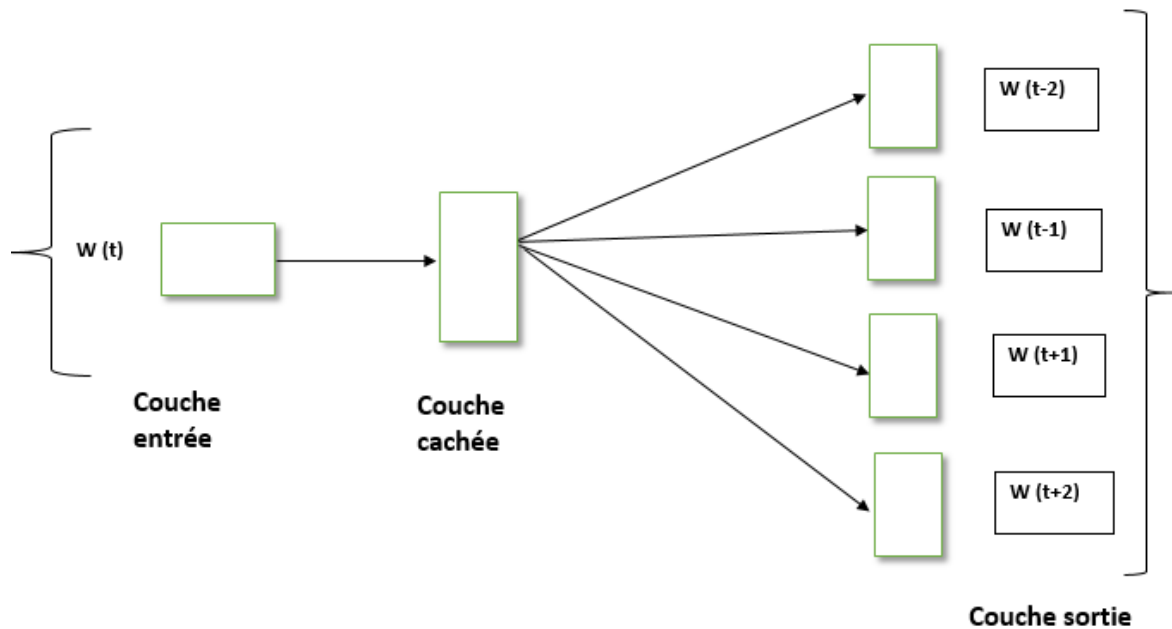


Figure 11: Architecture du modèle Skip gram

### III.4 Les couches du modèle Word2vec

Ces réseaux de neurones sont entraînés de façon à prédire un mot central étant donné un contexte (CBOW) ou vice-versa (Skip-gram). Ces modèles sont structurés en trois couches : [53]

- **Couche d'entrée** : contient le mot central pour le modèle Skip-Gram et un sac-de mots Contenant la fenêtre de contexte du mot centrale pour le modèle CBOW.
- **Couche intermédiaire** : correspond à la projection des mots d'entrée dans la matrice des poids afin d'attribuer des poids à chacun des vecteurs.
- **Couche de sortie** : correspond à la prédiction du modèle par l'utilisation de la fonction « Softmax », soit le mot central pour le modèle CBOW, soit un sac-de-mots contenant la fenêtre de contexte du mot central pour le modèle Skip-Gram.

### III.5 Paramètres de Word2vec

Word2Vec accepte plusieurs paramètres qui affectent à la fois la vitesse et la qualité de l'entraînement parmi ces paramètres on cite : [41]

#### 1. min\_count

Le paramètre min\_count permet d'ignorer les mots qui n'apparaissent qu'une ou deux fois dans un corpus d'un milliard de mots puisqu'ils sont probablement des fautes de frappe et des ordures inintéressantes. De plus, il n'y a pas assez de données pour faire une formation significative sur ces mots, il est donc préférable de les ignorer. La valeur par défaut de min\_count = 5.

#### 2. vector\_size

vector\_size est le nombre de dimensions (N) de l'espace à N dimensions sur lequel gensim (une des bibliothèque python que nous allons expliquer dans l'implémentation) Word2Vec mappe les mots.

Des valeurs de taille plus élevées nécessitent plus de données d'entraînement, mais peuvent conduire à de meilleurs modèles (plus précis). Les valeurs raisonnables se situent entre dizaines et centaines

#### 3. Workers

workers, l'un des paramètres majeurs qui permet la parallélisations de la formation word2vec, pour accélérer la formation, il est préférable d'utiliser de nombreux threads de travail pour entraîner le modèle (entraînement plus rapide avec des machines multicœurs).

#### 4. Epochs : C'est le nombre d'itérations (époques) sur le corpus.

#### 5. window - Distance maximale entre le mot actuel et le mot prédit dans une phrase

### III.6 Les domaines d'application du Word Embedding

Le modèle du Word Emebeddings a ouvert plusieurs perspectives dans de nombreuses taches dans le domaine du TALN. Plus particulièrement, la méthode Word2Vec qui est une méthode récente et qui a prouvé son efficacité à travers plusieurs travaux et applications, parmi elles on peut citer :

### **L'analyse des sentiments**

L'analyse des sentiments est devenue depuis quelque temps un sujet chaud, à la fois dans le monde académique et dans celui des applications pratiques. Pour les chercheurs en TALN, les travaux sur « Opining mining » ou « Sentiment Analysis » constituent un des champs d'investigation les plus actifs en ce moment. Une première utilisation de Word2Vec a été introduite en représentant une phrase par un vecteur en faisant la moyenne des vecteurs des mots la composant. Les vecteurs obtenus sont utilisés comme caractéristiques pour l'apprentissage de classificateur automatiques pour l'analyse de sentiments [50].

### **III.6.2 Reconnaissances d'entités nommées**

La tâche de reconnaissance d'entités nommées (REN) a été introduite à la fin des années 90, elle a suscité un grand intérêt dans les communautés scientifiques. La tâche de reconnaissance d'EN est une sous tâche du domaine d'extraction d'information consistant à repérer toutes les formes linguistiques bien identifiées telles que les expressions temporelles (dates, durées,), les quantités (monétaires, unités de mesure, pourcentages...) et à leur affecter une étiquette sémantique choisie dans une liste .[46]

Cette technique peut utiliser deux méthodes différentes. La première approche s'appuie sur l'utilisation de grammaires formelles construites à la main. Cela en utilisant des marqueurs lexicaux, des dictionnaires de noms propres et parfois un étiquetage syntaxique.

La seconde démarche fait usage de techniques statistiques ou encore dites à base d'apprentissage pour apprendre des spécifies sur de larges corpus de textes où les entités-cibles ont été au paravent étiqueté créant ainsi un corpus d'apprentissage, par la suite adapter un algorithme d'apprentissage qui va permettre d'élaborer automatiquement une base de connaissances à l'aide de plusieurs modèles numériques [46,38].

### **III.6.3 La traduction automatique**

La traduction consiste à porter un texte écrit dans une langue naturelle, la langue source, vers une autre langue, la langue cible. L'intégration des méthodes de traduction dans l'apprentissage automatique se basent sur des règles grammaticales et syntaxiques.

La traduction automatique est également utilisée dans les modèles de prolongements lexicaux. En effet des spécialistes ont entraîné le modèle word2vec dans chaque langue [50], une transformation géométrique est par la suite calculée entre les espaces vectoriels issus des deux langages. Plus récemment ils [42] ont proposé d'entraîner un modèle Word2Vec directement à partir de couple de mot, par exemple étant donnés les couples de mots « un\_\_uno » « cinco—five ».>>. Le but principal de l'utilisation de la méthode de traduction à partir de couple de données est de pouvoir détecter les erreurs qui sont faites par les ordinateurs lors de la traduction automatique. La figure suivante représente le regroupement des vecteurs Word2vec de mots en « anglais » et en « espagnol »

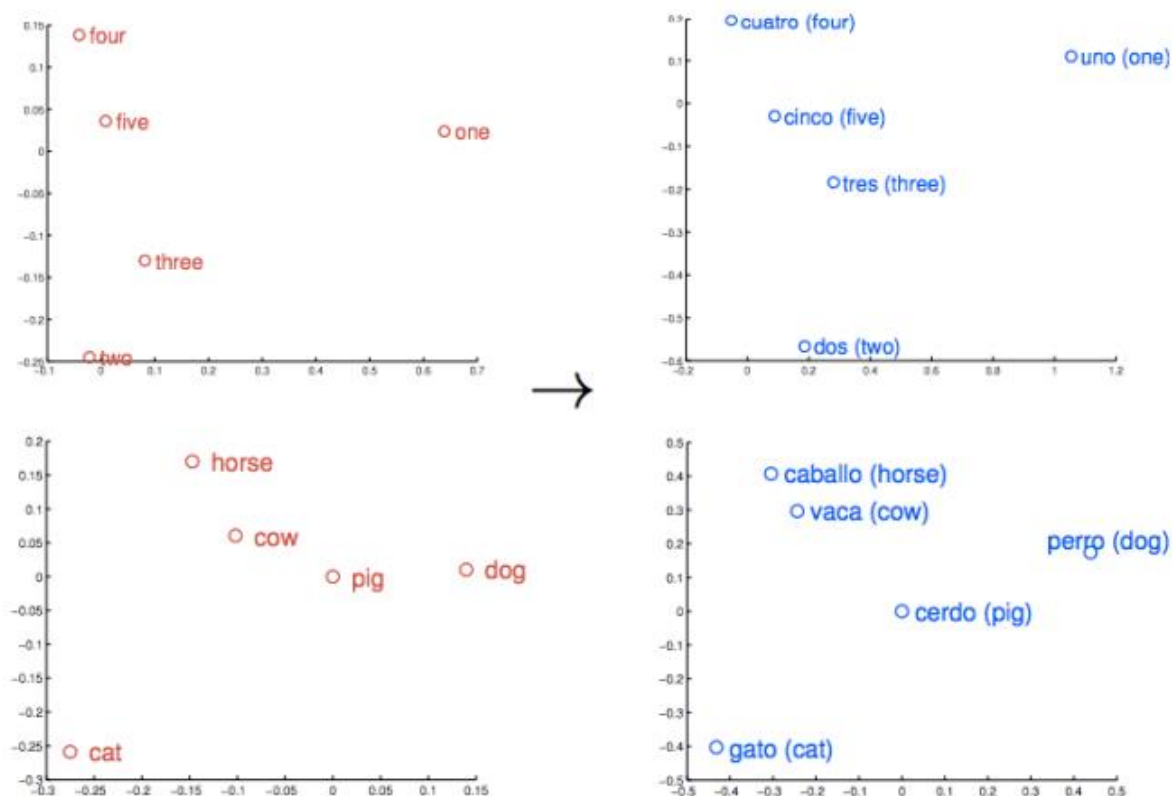


Figure 12: Exemple d'entrainement du modèle Word2vec à partir de couple de mot

## Conclusion

Nous nous sommes intéressés dans ce chapitre à la présentation des Word Embeddings ainsi qu'à son mode de fonctionnement, nous avons aussi présenté un bref historique sur sa création ainsi qu'à son évolution mais également sur ses différents types d'architectures neuronales, tout en exposant les fondements théoriques des algorithmes d'apprentissage de réseau de neurones.

Ensuite, nous avons montré les deux Techniques de visualisation de ces représentations de mots, nous avons aussi illustré les domaines d'application du Word Embedding pour finir le chapitre avec une conclusion



## CHAPITRE 4 : LA CONCEPTION

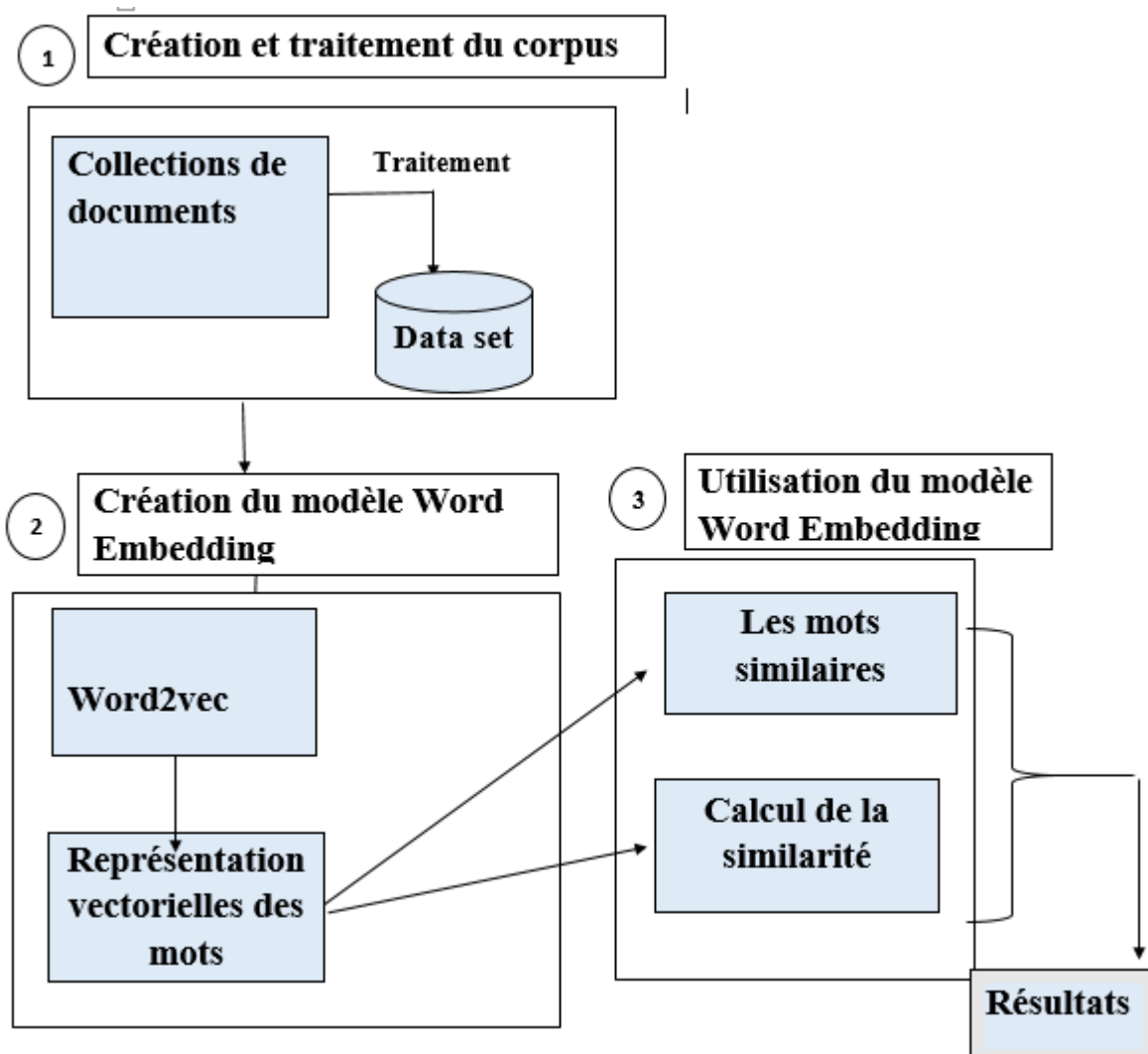
## Introduction

Afin de pouvoir aborder notre projet de la meilleure façon possible, il est nécessaire de revenir sur les étapes à suivre et délimiter le travail que nous souhaitons mettre en œuvre. Dans ce chapitre, nous présenterons en détails les différentes étapes qui nous permettront de créer notre corpus jusqu'à la génération de la représentation vectorielle des mots qui composent notre corpus. Pour enfin, terminer avec l'utilisation de notre modèle de word embedding pour extraire la similarité entre concepts.

### IV.1 Approche proposée

Notre approche consiste à construire un corpus dense en relation avec des concepts d'enseignements, préalablement établis. Puis de générer la représentation vectorielle (Word Embedding) des mots qui composent ce corpus. Cette représentation vectorielle sera ensuite utilisée pour extraire le degré de similarité entre des concepts d'enseignement, ainsi que d'afficher l'ensemble des mots avoisinant un concept donné.

Pour ce faire, notre approche a été divisée en 3 tâches principales : la construction du corpus, la création du modèle de Word Embedding, et l'utilisation du modèle pour afficher la similarité entre des concepts ainsi que les voisins d'un concept donné. Comme l'illustre la figure ci-dessous.



**Figure 1 : Architecture générale de notre approche proposée.**

Notre approche consiste à créer un corpus dense de thème d'enseignements depuis des collections de documents et de thèses issues de Wikipédia. L'API Wikipédia nous permet de faire une recherche de plusieurs articles de même domaine depuis Wikipédia, puis nous allons appliquer la méthode word2vec sur notre corpus pour entrainer un modèle qui nous permettra de calculer la similarité entre ces termes ainsi qu'afficher les mots similaire a un terme donner.

Les étapes suivies pour la conception de notre approche sont détaillées dans ce qui suit :

### **IV.1.1 Tâche 1 : Construction du corpus**

La manipulation et le traitement des modèles de word embeddings nécessitent des collections de données volumineuses. Pour obtenir un vocabulaire suffisant qui permettra d'appliquer les modèles des word embeddings performant, nous allons construire notre corpus sur la base de documents extraits de différents articles de Wikipédia<sup>1</sup>.

Pour ce faire, les étapes suivies pour la construction de notre corpus sont illustrées dans la figure et détaillées dans ce qui suit :

#### **A. Acquisition des données**

Nous avons choisi la plateforme Wikipédia pour extraire des textes en relation avec différents domaines d'enseignement. Notre choix s'est porté sur cette plateforme, car selon nous elle contient des articles fiables et de qualité.

Pour acquérir les textes issus de Wikipédia, il a fallu procéder comme suit :

1. Établir une liste de disciplines scientifiques (telles que l'informatique, la biologie la chimie ...etc) en langue française à rechercher sur wikipédia.
2. Utiliser l'API Wikipédia qui reçoit le nom de la thématique recherchée et qui retourne les articles en relation avec cette dernière.
3. Récupérer le texte en brut, les regrouper par domaine puis les enregistrer sous format zip (pour gagner de l'espace en mémoire).

Une fois l'ensemble des données acquises, nous passons à l'étape de nettoyage et de traitement du texte.

#### **B. Traitement des données**

Pour les besoins de notre travail, nous devons effectuer un traitement sur les données collectées sur Wikipédia afin qu'elles puissent correspondre aux données passées en entrée à l'outil utilisé

---

<sup>1</sup> <https://fr.wikipedia.org/>

pour la génération du modèle Word Embedding.

Pour ce faire, nous devons diviser tous les articles en phrases, puis tokeniser chaque séparément (token=mot). Ensuite, nous transformons tous les mots en minuscule, et supprimons les mots non alphabétiques.

Au final, nous obtenons corpus qui est formé de listes de phrases qui sont elles-mêmes des listes de tokens.

Les étapes de construction et de traitement de notre corpus sont modélisées dans la figure 2 ci-dessous :

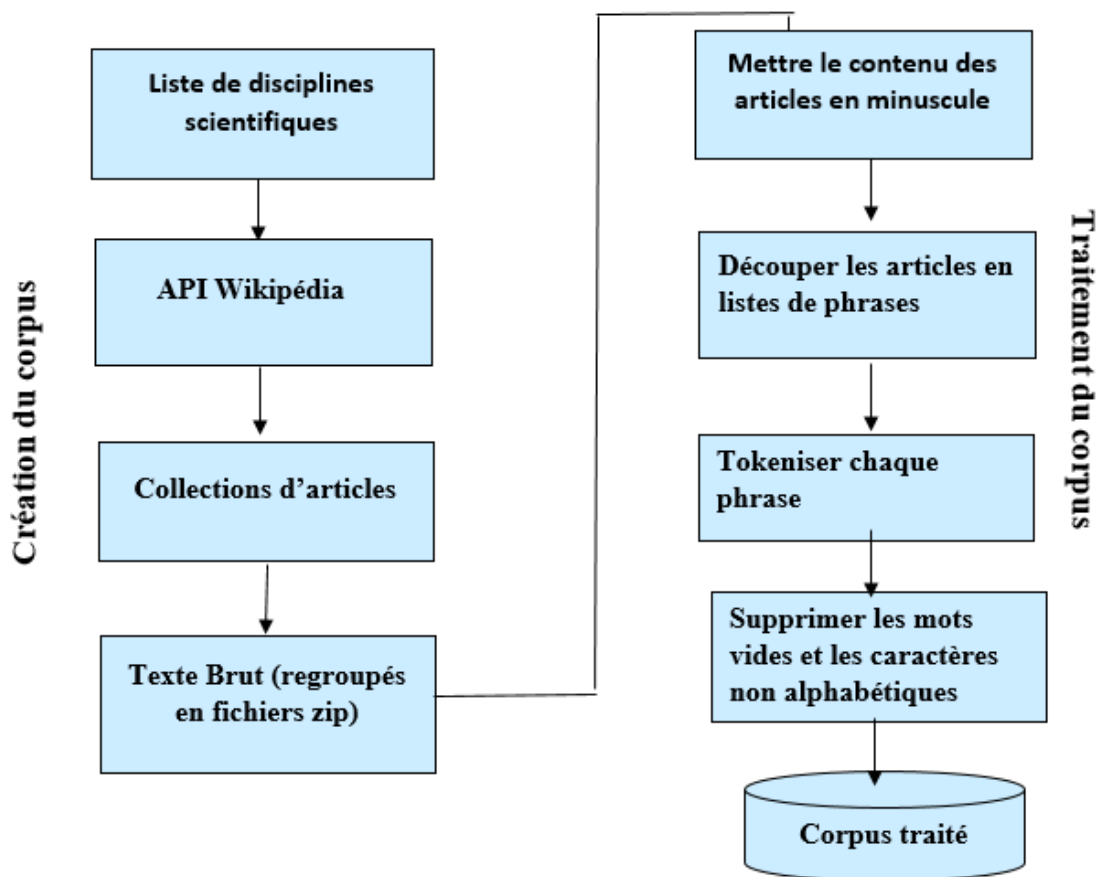


Figure 2 : Étapes de construction et traitement du corpus.

### C. Quelques statistiques du corpus

Le corpus construit est constitué d'articles scientifiques, extraits de Wikipédia. Il recouvre plus de 31 disciplines scientifiques en langue française qui sont : l'informatique, biologie, économie,

histoire, architecture électronique, mécanique, automatique, chimie, physique, anglais agronomie, géographie, la musique, science de gestion, électrotechnique et bien d'autres encore. Chaque discipline compte environ des millions de mots.

Notre corpus est constitué de 153 000 000 de mots bruts.

Le tableau 1 ci-dessous illustre le nombre de mots pour quelques-unes de ces disciplines.

Disciplines	Nombre de mots
Informatique	1 159 306
Économie	1 607 530
Biologie	1 40 133
Architecture	1 503 630

**Figure Tableau 1 : Nombre de mots dans certaines disciplines**

Chaque collection (domaine) est constituée d'un ensemble de documents texte(.txt). Nous avons utilisé seulement les collections de documents en langue française se rapportant à des disciplines d'enseignement.

#### **IV.1.2 Tâche 2 : Création du Modèle Word Embedding**

Pour générer notre modèle de Word Embedding, qui pour rappel est une représentation vectorielle des mots qui composent notre corpus, nous avons choisi d'utiliser le modèle **Skip-gram de Word2vec**. Notre choix s'est porté sur Skip-gram et non sur CBOW, car il a été prouvé que Skip-gram arrivait à fournir des bons résultats même avec des corpus de taille plus au moins petite.

### IV.1.3 Tâche 3 : Utilisation du modèle pour extraire la similarité entre concepts

Le nouvel espace de représentation est constitué alors d'un vocabulaire de termes uniques. Chaque mot est projeté dans le nouvel espace de représentation afin de construire sa nouvelle représentation,

Pour calculer la similarité, nous avons utilisé la bibliothèque `gensim` qui est chargée d'extraire des mots similaires, pour chaque mot contenu dans notre corpus et dans le vocabulaire il va calculer les N mots similaires avec leurs valeurs de similarité. `Gensim` utilise la fonction cosinus pour calculer la similarité par défaut.

Pour le calcul de la similarité nous avons donc utilisé la similarité cosinus  
similarité cosinus :

Après avoir transformé un mot en vecteurs, le cosinus de l'angle entre les vecteurs de mots peut être utilisé pour comparer le degré de ressemblance des mots. Plus un angle est petit, plus la valeur du cosinus sera grande.

Le produit scalaire et la similarité de cosinus sont des mesures de similarité, mais le produit scalaire est sensible à la magnitude, contrairement à la similarité cosinus. Selon le nombre d'occurrences d'un mot, il peut avoir un produit scalaire petit ou grand avec un autre mot. Nous normalisons notre vecteur pour éviter cet effet, de sorte que tous les vecteurs ont une magnitude unitaire. La solution pour laquelle nous avons opté permet de calculer la similarité en cosinus qui les normalisera.

Pour calculer la similarité du cosinus on doit d'abord calculer le produit scalaire des deux vecteurs par la suite le diviser par le produit des normes des deux vecteurs

La figure ci-dessous montre la méthode de calcul de similarité

Formules mathématiques du produit scalaire et de cosinus

$$\text{Similarité} = \cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|} = \frac{\sum_{i=1}^h A_i \cdot B_i}{\sqrt{\sum_{i=1}^h A_i^2} \cdot \sqrt{\sum_{i=1}^h B_i^2}}$$

Skip-Gram peut créer une représentation distribuée très riche de mots qui réserve également les relations sémantiques que les mots ont avec d'autres mots. Skip-Gram peut comprendre fondamentalement la signification d'un mot basé sur les apparences passées.

Cela signifie que nous pouvons maintenant regrouper des mots, trouver des mots similaires et rechercher des relations similaires de mots différents. [106]



## CHAPITRE 5 : Implémentation

## **Introduction**

Il ne fait aucun doute que le plus important pour résoudre un problème dans le domaine d'informatique est un bon choix pour le langage de développement. Le développement du domaine d'informatique est devenu un problème multiple, et l'accès à la résolution des méthodes les plus simples est très important. Nous avons donc choisi le langage de programmation optimal qui nous a considérablement aidés à obtenir de bons résultats. Dans ce chapitre on va définir le langage choisi pour résoudre notre problème, l'environnement de développement, et on va détailler l'implémentation de chaque partie de notre application.

### **V.1 Choix du langage de programmation**

Lorsque vous choisissez un langage de programmation qui se spécialise dans l'apprentissage automatique de langage naturel, il doit considérer les compétences répertoriées dans les offres d'emploi actuelles ainsi que les bibliothèques disponibles dans différentes langues qui peuvent être utilisées pour les processus d'apprentissage automatique. Python est le langage de programmation le plus recherché dans le domaine de l'apprentissage automatique et l'apprentissage profond. Python est suivi par Java, puis par le R, puis C ++. Python [35].

#### **Python (langage)**

Python est un langage de programmation, interprété car, avant de pouvoir exécuter un programme, un logiciel spécialisé se charge de transformer le code du programme en langage machine, multi-paradigme et multiplateformes, est placé sous une licence libre, qui vous permet de travailler rapidement et d'intégrer les systèmes plus efficacement. Python peut être utilisé pour gérer des données volumineuses et effectuer des calculs complexes. Il existe ce qu'on appelle des bibliothèques qui aident le développeur à travailler sur des projets particuliers. Plusieurs bibliothèques peuvent ainsi être installées pour, par exemple, développer des interfaces graphiques en Python.



**Figure 1 Logo python.**

Ce choix a été motivé par les raisons suivantes :

- L'une des principales langues parmi les langues appropriées pour la programmation de problèmes d'apprentissage profond.
- Il dispose un grand nombre de bibliothèques pour le traitement du langage naturel, telles que NLPnet, NLTK, ... .
- Un langage simple, productif et utilisable dans presque tous les domaines et systèmes.

## **V.2 Environnement de développement**

Pour l'environnement de développement, nous avons utilisé l'environnement du Visual Studio Code.

Visual studio code est un éditeur de code extensible développé par Microsoft, il se présente sous la forme d'un environnement multi langages léger.il est adapté pour la manipulation de fichiers avec des très gros volumes en effet il possède un moteur de gestion de code très puissant. Il exploite toute la puissance de VS Code pour fournir l'auto-complétion, le débogage et les tests unitaires, ainsi que la possibilité de basculer facilement entre les environnements Python, y compris les environnements virtuels et conda.



**Figure 2 l'environnement du Visual Studio Code**

### **V.3 Bibliothèques Python utilisées**

Les communautés NLP et Machine Learning se sont principalement concentrées sur le langage Python pour développer différents outils bibliothèques qui facilitent aujourd'hui l'utilisation de ces techniques, même pour les plus novices dans ce domaine.

Les bibliothèques auxquelles nous avons eu recours pour la mise en œuvre de notre projet sont :

#### **A. Gensim**

Gensim est une bibliothèque libre de Python conçue pour le traitement du langage naturel (NLP) open source populaire utilisée pour modélisation de sujets non supervisée. Il utilise des modèles académiques de pointe et un apprentissage automatique statistique moderne pour effectuer :

- Création de documents ou de vecteurs de mots
- Corpora
- Exécution de l'identification de sujets
- Exécution de la comparaison de documents (récupération de documents sémantiquement similaires)
- Analyse de documents en texte brut pour la structure sémantique[101] SRINIVASA-DESIKAN, Bhargav. *Traitement du langage naturel et linguistique computationnelle: un guide pratique pour l'analyse de texte avec Python, Gensim, spaCy et Keras* . Packt Publishing Ltd, 2018.

## NumPy

NumPy est une bibliothèque python de bas niveau écrite en C (et FORTRAN) pour les fonctions mathématiques de haut niveau. NumPy dépasse habilement le problème d'exécuter des algorithmes plus lents sur Python en utilisant des tableaux multidimensionnels et des fonctions qui opèrent sur des tableaux. [60]

## B. NLTK

NLTK est une plate-forme leader pour la création de programmes Python compatibles avec les données de langage humain. Il fournit des interfaces faciles à utiliser pour les corpus et ressources lexicales, ainsi qu'une suite de bibliothèques de traitement de texte pour la classification, la tokenisation. Python standard pour le NLP. Elle regroupe des algorithmes de classifications, de stemming, de tokenisation en (mots) et en (phrases). Elle contient également des corpora de données et permet de faire de la reconnaissance d'entités nommées (NER) et de l'analyse de sentiments. [57]

## API wikipédia

Wikipedia est une bibliothèque Python qui facilite l'accès et l'analyse des données de Wikipedia.

Recherchez sur Wikipédia, obtenez des résumés d'articles, obtenez des données comme des liens et des images à partir d'une page, etc.[100] SEN, Shilad, LI, Toby Jia-Jun, TEAM, WikiBrain, *et al.* Wikibrain: democratizing computation on wikipedia. In : *Proceedings of The International Symposium on Open Collaboration*. 2014. p. 1-10.

## V.4 Matériel utilisé

Pour le développement de notre application, nous avons utilisé une machine avec les configurations suivantes :

- Nom de l'ordinateur : Dell XPS

- Processeur : Intel(R) Core (TM) i7-3537U CPU @ 2.00GHz 2.00 GHz
- Mémoire Ram installer: 8,00 Go (7,88 Go utilisable)
- Type du système : Système d'exploitation 64 bits, processeur x64

## V.5 Mise en Œuvre

Dans cette section nous présentons et expliquons les différentes phases de notre travail, qui commence par créer notre corpus et l'entraîner puis à présenter notre expérimentation sur une plateforme qu'on a créé.

### V.5.1 Les étapes de création de notre corpus

La création de notre corpus est faite à l'aide de l'API Wikipédia, Nous avons constitué notre corpus de manière manuel depuis Wikipédia,

Les étapes de créations

- 1 étape :

On a utilisé la classe WikipédiaScraper qui est un wrapper de la librairie d'origine(Wikipédia)

```
import wikipedia as wiki
```

On a défini la langue en français

```
wiki.set_lang(lang)
```

pour chercher des articles en rapport avec l'enseignement sur Wikipédia. La figure ci-dessus montre une liste des domaines d'enseignements qu'on va récupérer en langue française. Pour chaque thème on va récupérer une liste d'articles.

La figure suivante montre les disciplines qui ont constitué notre corpus d'apprentissage :

```

from educ_wordembedded.corpus.core import CorpusMaker

themes = [
    "Agronomie",
    "Anglais",
    "Apprentissage",
    "Architecture",
    "Arts plastiques",
    "Automatique",
    "Biologie",
    "Chimie",
    "Développement",
    "Economie",
    "Educatif",
    "Education civique",
    "Electronique",
    "Enseignement",
    "Français",
    "Géographie",
    "Histoire",
    "Instruction",
    "Langage",
    "Musique",
    "Mécanique",
    "Mécanique",
    "Parole",
    "Philosophie",
    "Physique",
    "Programmation Informatique",
    "Pédagogie",
    "Recherche",
    "Savoir",
    "Sciences de gestion",
    "Sciences de la Vie et de la Terre",
    "Secteur d'activité",

```

Figure 3: Les domaines qui constituent notre corpus

## ➤ Etape 2

Corpus maker est l'objet permettant de télécharger massivement le contenu des articles en mode multithread (c'est-à-dire plusieurs articles en parallèle) et d'enregistrer chaque article en format zip

```

class CorpusMaker:
    """
    CorpusMaker is a corpus maker that scraping data from Wikipedia.

    CorpusMaker is multithreading ready.
    """

    def __init__(
        self,
        name,
        themes,
        path,
        depth=50,
        language="en",
        type_storage="zip",
        thread=32,
    ):
        self.name = name
        self.articles = WikipediaScraper.search_articles(
            themes, language, depth
        )
        self.lang = language
        self.type_storage = type_storage
        if type_storage == "csv":
            # path = os.path.join(path, "corpus_data")
            path = os.path.join(path, name.replace(" ", "-") + ".csv")
            self.file = Csv(path, ["theme", "title", "content"], mode="w")
            self.file.add_headers()
        if type_storage == "zip":
            self.path = path

        self.__nb_thread = thread

```

Figure 4: Le téléchargement des articles en mode multithread et leurs enregistrements en format zip

### ➤ Etape 3

Le prétraitement est fait à l'aide la bibliothèque NLTK, on va transformer les le contenu de notre corpus en minuscule, puis on a supprimé les mots non alphabétiques et aussi les stopwords qu'on a téléchargé avec la bibliothèque nltk.

```

nltk.download("stopwords")
nltk.download("punkt")

```



Au final, nous obtenons une liste de phrases qui sont elles mêmes des listes de tokens. Chaque article traité est enfin enregistré dans un fichier format zip.

Voici un exemple d'une liste de phrases extraites de l'article ...

La figure (ci-dessous) représente le processus de prétraitement d'un document avec quelques fonctions prédéfinies en python pour le traitement du langage naturel.

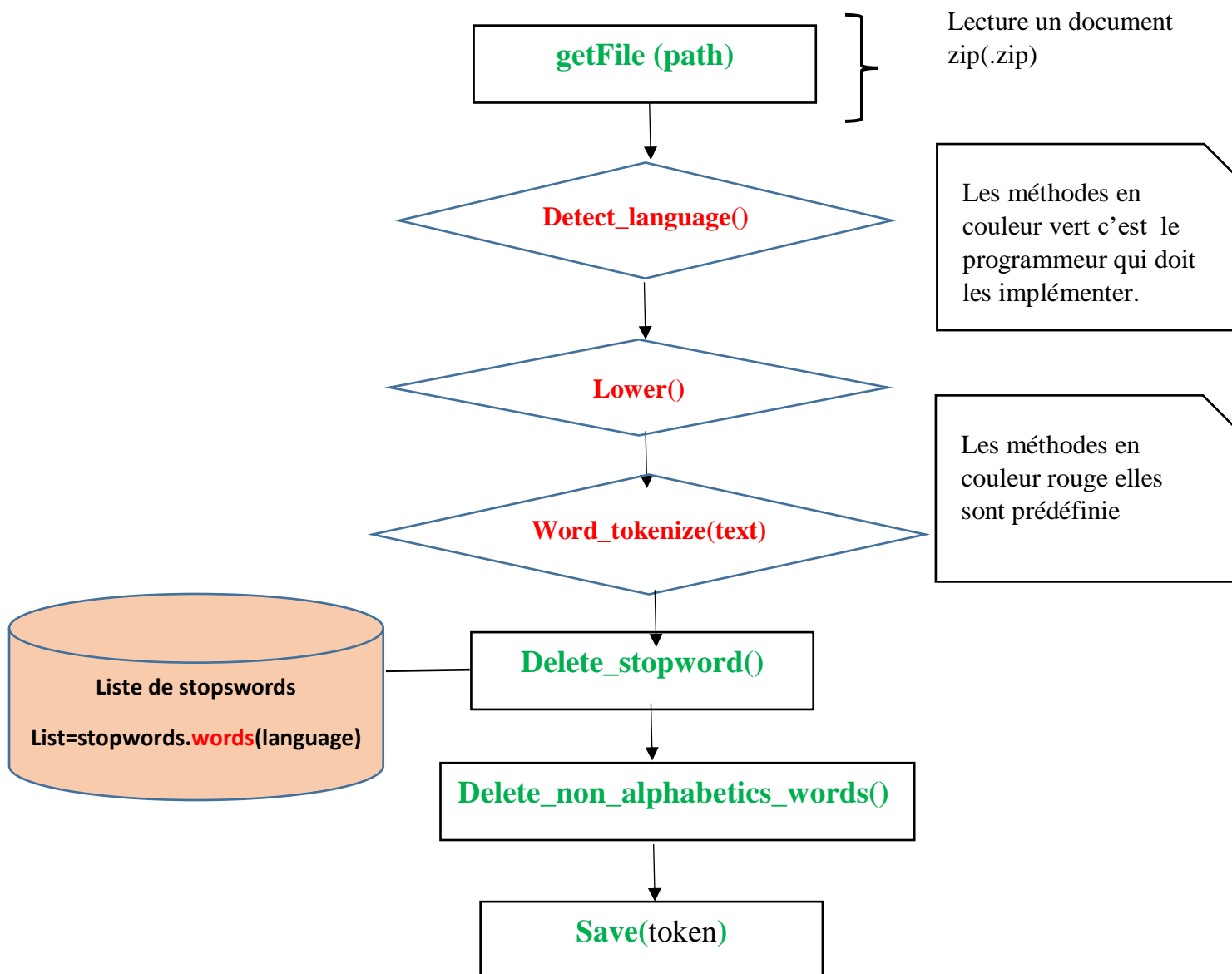


Figure 5 processus de l'exécution de traitement de corpus

## Avant la tokenisation

["1, rue Sésame (Sesame Street) est une émission de télévision éducative franco-américaine pour la jeunesse, basée sur le format Sesame Street. Coproduite par TF1 et The Children's Television Workshop, l'émission est diffusée du 3 avril 1978 au 24 juin 1982, sur TF1.", "En 1974, Jean-Louis Guillaud, directeur général de la troisième chaîne couleur de l'ORTF, confie à Christophe Izard l'adaptation pour la France de l'émission éducative américaine Sesame Street pour

la rentrée 1974. L'émission s'appelle Bonjour Sésame, doit durer vingt minutes et alterne des scènes avec marionnettes en mousse sculptée et des scènes avec personnages réels."],

## Après la tokenisation

[[ '1', ',', 'rue', 'sésame', '(', 'sesame', 'street', ')', 'est', 'une', 'émission', 'de', 'télévision', 'éducative', 'franco-américaine', 'pour', 'la', 'jeunesse', ',', 'basée', 'sur', 'le', 'format', 'sesame', 'street', ',', 'coproduite', 'par', 'tf1', 'et', 'the', 'children', '"s', 'television', 'workshop', ',', '"l'émission", 'est', 'diffusée', 'du', '3', 'avril', '1978', 'au', '24', 'juin', '1982', ',', 'sur', 'tf1', ''], ['en', '1974', ',', 'jean-louis', 'guillaud', ',', 'directeur', 'général', 'de', 'la', 'troisième', 'chaîne', 'couleur', 'de', 'l'ortf", ',', 'confie', 'à', 'christophe', 'izard', '"l'adaptation", 'pour', 'la', 'france', 'de', '"l'émission", 'éducative', 'américaine', 'sesame', 'street', 'pour', 'la', 'rentrée', '1974.', '"l'émission", "s'appelle", 'bonjour', 'sésame', ',', 'doit', 'durer', 'vingt', 'minutes', 'et', 'alterne', 'des', 'scènes', 'avec', 'marionnettes', 'en', 'mousse', 'sculptée', 'et', 'des', 'scènes', 'avec', 'personnages', 'réels', ''],

## Après la suppression des stop word et les caractere non alphanumérique

[[ 'rue', 'sésame', 'sesame', 'street', 'est', 'une', 'émission', 'de', 'télévision', 'éducative', 'pour', 'la', 'jeunesse', 'basée', 'sur', 'le', 'format', 'sesame', 'street', 'coproduite', 'par', 'et', 'the', 'children', 'television', 'workshop', 'est', 'diffusée', 'du', 'avril', 'au', 'juin', 'sur'],

['en', 'guillaud', 'directeur', 'général', 'de', 'la', 'troisième', 'chaîne', 'couleur', 'de', 'confie', 'à', 'christophe', 'izard', 'pour', 'la', 'france', 'de', 'éducative', 'américaine', 'sesame', 'street', 'pour', 'la', 'rentrée', 'bonjour', 'sésame', 'doit', 'durer', 'vingt', 'minutes', 'et', 'alterne', 'des', 'scènes', 'avec', 'marionnettes', 'en', 'mousse', 'sculptée', 'et', 'des', 'scènes', 'avec', 'personnages', 'réels'],

## V.6 Les étapes d'entraînement de model Word Emebeddings

Une fois que notre corpus est prêt nous allons exploiter les données contenues dans ce corpus est nous allons créer le modèle Word2vec.

Nous passons en entrée à notre programme notre corpus qui, pour rappel, est une collection de phrases tokenisées. Notre programme aura à générer une représentation vectorielle de chaque tokens (mot) qui compose notre corpus. Cet ensemble de représentations vectorielles formera notre modèle de word Embeddings de notre corpus.

Afin de générer un modèle word2vec nous avons utilisé la bibliothèque Gensim avec les paramètres suivants :

- Taille de vecteur = c'est le nombre de dimensions des incorporations
- Taille de fenêtre = la distance entre le mot cible et les mots environnants.
- Min count : c'est le nombre de mots à prendre en compte lors de la création du modèle, les occurrences des mots inférieurs au numéro min count, ne sont pas pris en compte.
- sg = 1 pour choisir skip-gram
- sg=0 pour CBOW

Le processus de creation d'un model word embeddings doit suivre les etapes suivantes (figure 6), en commençant par le chargement des donnes jusqu a la validation de model

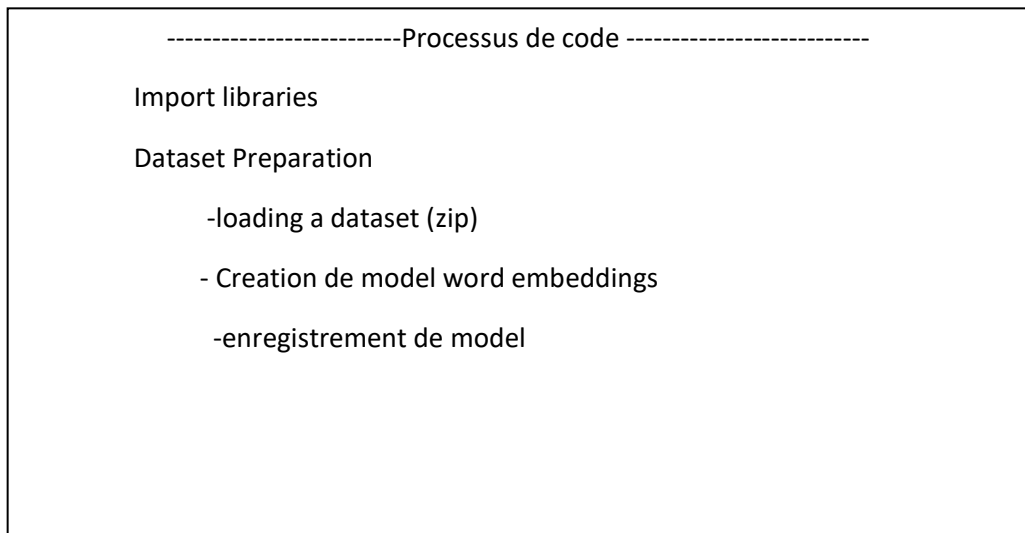


Figure 6 Le processus de creation d'un model word embeddings

### V.6.1 Importer les bibliothèques

Avant de créer le modèle, nous devons importer une bibliothèque disponible dans le package gensim en python. Tel que :

```
from gensim.models import Word2Vec
```

### V.6.2 Importer les données

Comme mentionné dans les chapitres précédents, le processus d'entraînement d'un modèle Word Emebeddings nécessite une grande quantité de données.

A l'aide de la bibliothèques Zlib ci-dessous on a chargé nos donnée (Preentraînées) en les décompressant

```
article = zlib.decompress(f.read()).decode("utf-8")
```

### V.6.3 La création du modèle

L'étape la plus importante de notre projet est l'étape de la création du modèle, la figure ci-dessous représente des quelques instructions de la création du modèle

```
model = Word2Vec(WikiCorpus("data"), workers=64)  
model.save("models_ia/wiki_model6.mm")
```

-workers est le nombre de thread

On peut changer de model ("[models\\_ia/wiki\\_model6.mm](#)") pour voir la différence qu'on peut avoir entre un model entrainer sur un grand nombre de donnée et moins de de données

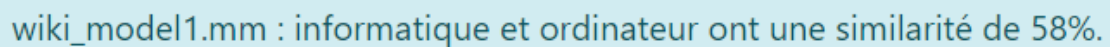
## V.6.4 Mise en application

### Calcul des similarités

Le calcul de similarité entre deux mots est obtenu par le cosinus de leurs vecteurs.

Méthode <sup>V.6.4.1</sup> de calcul entre deux vecteurs est représentée dans un espace à n dimensions

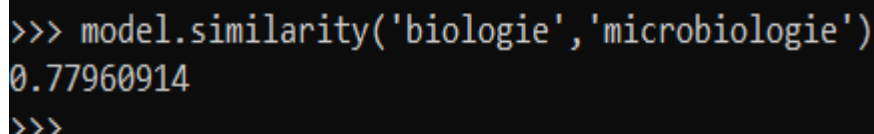
Prenons exemple de calcul de la similarité entre le mot « ordinateur » et le mot « informatique » le résultat retourné par notre programme sur le terminal est montré dans la figure ci-dessous :



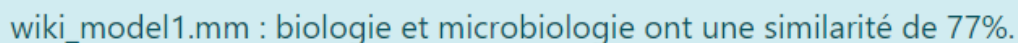
```
wiki_model1.mm : informatique et ordinateur ont une similarité de 58%.
```

Figure 7: Similarité entre mots « ordinateur » et « informatique »

On calcul la similarité entre le mot « biologie » et le mot « microbiologie »



```
>>> model.similarity('biologie','microbiologie')
0.77960914
>>>
```



```
wiki_model1.mm : biologie et microbiologie ont une similarité de 77%.
```

Figure 8: Similarité entre mots « biologie » et « microbiologie »

Maintenant on essaye de calculer la similarité entre le terme « informatique » et le terme « biologie »

Le résultat retourné est représenté dans la figure ci-dessous

```
>>> model.similarity('biologie','informatique')
<stdin>:1: DeprecationWarning: Call to deprecate
0.34405565
```

Figure 9: Similarité entre mots « biologie »et « informatique »

On constate que les résultats retournés lors du calcul de la similarité entre ces deux termes est très éloigné comparé aux résultats retournés dans les deux premières représentations. En effet le principe du modèle word2vec est que les mots similaires finissent par avoir des valeurs plus proches les uns des autres comparé aux mots éloignés

### Trouver les mots similaires

On peut chercher les mots similaires d'un mot en relation avec les thèmes de notre corpus, la figure ci-dessous nous montre un exemple de notre application.

Les mots les plus proches de "informatique" dans le modèle "wiki\_model1.mm" sont :

- virtualisation à 68%
- informatiques à 68%
- logicielle à 65%
- logiciel à 64%
- numérique à 63%

Figure 10 les mots similaires au mot informatique

Trouver les mots proches de trois mots données exemple :

# Comparaison de corpus

Étant donné plusieurs corpus de taille différente, cette démo montre une comparaison entre les model Word2vec obtenu à partir de chaque corpus.

Sélection du model

Wiki model1

Mot à comparer 1

ordinateur

Mot à comparer 2

histoire

Mot à comparer 3

automatique

TOPN

5

Envoyer

Effacer

## V.7 Présentation de notre application

Figure 13: Interface de notre application

Word2Vec Similarité Mot Proche Comparaison

### Démonstration de Word2Vec

Notre démo est basée sur des Embeddings de mots induits en utilisant la méthode Word2Vec. Formés sur 4,5 milliards de mots d'un corpus de différents concepts d'enseignement provenant de Wikipédia.



Fonction de la similarité.  
Étant donné deux mots dans notre corpus ,  
cette démo nous permet de calculer la  
valeur de similitude entre mot1 et mot2.

Go



Fonction de recherche des mots les plus  
proches.  
Étant donné un mot dans notre corpus,  
cette démo montre une liste d'autres mots  
qui lui sont similaires.

Go



Étant donné plusieurs corpus de taille  
différente, cette démo montre une  
comparaison entre les model Word2vec  
obtenus à partir de chaque corpus.

Go

Dans la fonction de similarité

L'utilisateur va choisir le corpus sur lesquels il veut travailler :

Il a le choix entre le modèle :

Skipgram et le modèle cbow entraînés avec un corpus nettoyé ou un corpus non nettoyé .

## Recherche de similarité

Fonction de la similarité.

Étant donné deux mots dans notre corpus , cette démo nous permet de calculer la valeur de similitude entre mot1 et mot2.

Sélection du modèle

Wiki model1

Mot 1

Enter un mot

Mot 2

Enter un mot

Envoyer

Effacer



L'utilisateur va saisir deux mots et il va calculé leurs similarités

Dans la fonction de recherche de mots les plus proches

## Recherche de mot proche

Fonction de cherche des mots les plus proches.

Étant donné un mot dans notre corpus, cette démo montre une liste d'autres mots qui lui sont similaires.

Sélection du model

Wiki corpus cbow clean

Mot à rechercher

Enter un mot

TOPN

1

☐ Afficher le nuage de point

Envoyer

Effacer

L'utilisateur va selectionner le model , puis saisir le mot pour lequel il veut calculer les mots les plus proches (mots similaires) et choisir le nombre de mots similaire qu'il veut afficher .

Pour afficher les mots proches dans un nuage de point il suffit de cocher cette option .

Dans la fonction comparaison de corpus :

## Comparaison de corpus

Étant donné plusieurs corpus de taille différente, cette démo montre une comparaison entre les model Word2vec obtenu à partir de chaque corpus.

Sélection du model

Wiki model1

Mot à comparer 1

Enter un mot

Mot à comparer 2

Enter un mot

Mot à comparer 3

Enter un mot

TOPN

1

Envoyer Effacer

L'utilisateur va introduire trois mots , pour afficher les mots les plus proches de ces trois mots

### V.7.1 Démonstration sur notre applicaton

- **Calcule des mots les plus proches**

La figure ci-dessous montre les mots similaires à un mot saisi par un utilisateur, ce dernier a le choix entre le nombre de mots qu'il souhaite afficher (Top N).

Le mot saisi est le mot « informatique » : on remarque que les mots retournés sont tous des mots très proches au mot informatique.

Les figures suivantes montrent quelques exemples de notre modèle et illustrent les capacités et la cohérence de ce modèle.

### Model cbow

La figue ci-dessous montre les mots les plus proches d' informatique avec le model cbow clean(avec corpus traiter)

Les mots les plus proches de "informatique" dans le modèle "wiki\_corpus\_CBOW\_clean.mm" sont :

- informatiques à 73%
- virtualisation à 71%
- logiciel à 70%
- logicielle à 69%
- logiciels à 67%

Figure 14:Top 5 des mots proches au mot informatique

Les mots les plus proches de "informatique" dans le modèle "wiki\_corpus\_CBOW\_unclean.mm" sont :

- électronique à 70%
- virtualisation à 68%
- robotique à 68%
- l'informatique à 67%
- logicielle à 66%

**Remarque :** On remarque que les résultats retournés avec le model cbow clean sont plus pertinents que ceux retournés avec cbow unclean.

V.7.1.2

### Model skipgram

## Recherche de mot proche

Fonction de recherche des mots les plus proches.

Étant donné un mot dans notre corpus, cette démo montre une liste d'autres mots qui lui sont similaires.

Sélection du model

Wiki corpus skipgram clean

Mot à rechercher

informatique

TOPN

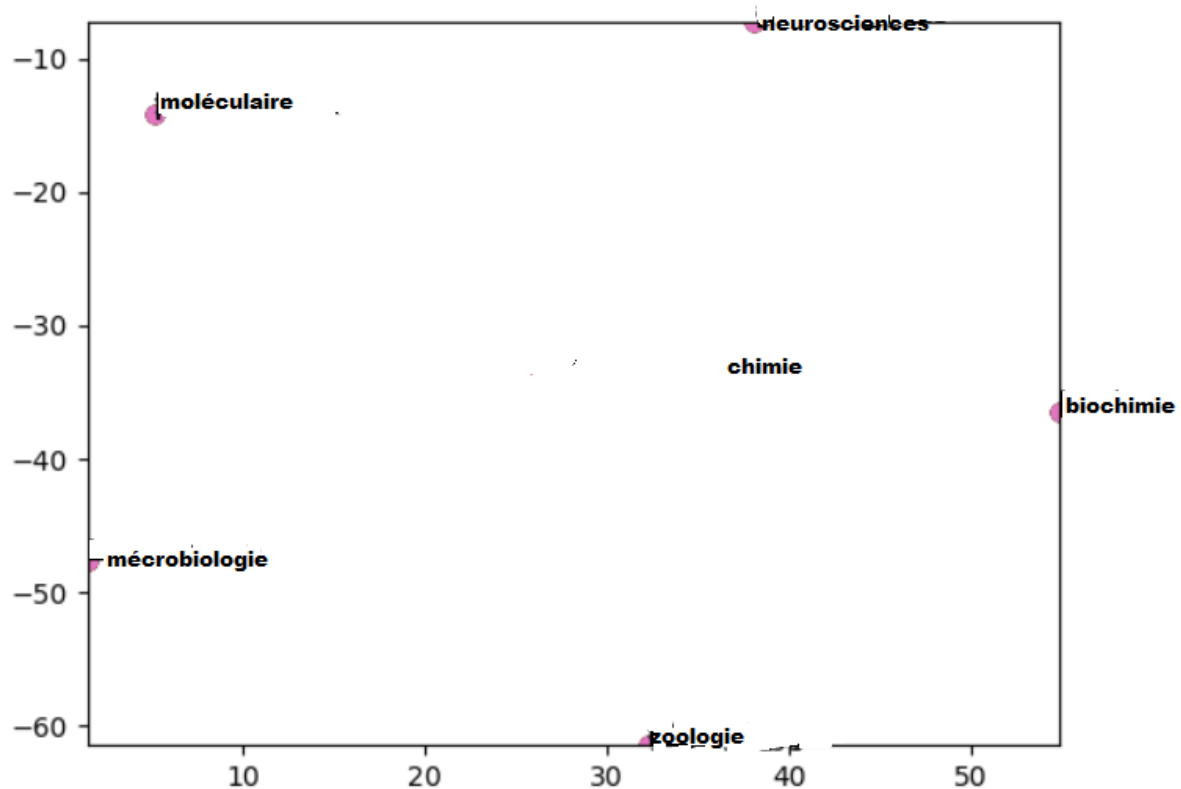
5

☐ Afficher le nuage de point

Envoyer

Effacer

La figure ci-dessous nous montre un nuage de points des résultats retournés plus haut :



La figure ci-dessous montre les mots les plus proches d'informatique avec le model skipgram clean(avec corpus traiter) et skipgram unclean.

Les mots les plus proches de "informatique" dans le modèle "wiki\_corpus\_skipgram\_clean.mm" sont :

- informatiques à 83%
- cryptologie à 80%
- logiciel à 79%
- hosting à 77%
- décisionnelle à 76%

Les mots les plus proches de "informatique" dans le modèle "wiki\_corpus\_skipgram\_unclean.mm" sont :

- informatiques à 79%
- logiciel à 78%
- l'informatique à 78%
- décisionnelle à 78%
- cryptologie à 76%

Remarque : On remarque que les résultats sont pertinents dans les deux cas.

- **Calcul de similarité entre deux mots**

La capture ci-dessous nous permet de calculer la similarité entre deux mots saisis par un utilisateur

## Recherche de similarité

Fonction de la similarité.

Étant donné deux mots dans notre corpus , cette démo nous permet de calculer la valeur de similitude entre mot1 et mot2.

Sélection du model

Wiki corpus skipgram clean

Mot 1

informatique

Mot 2

ordinateur

Envoyer

Effacer

wiki\_corpus\_skipgram\_clean.mm : informatique et ordinateur ont une similarité de 71%.

Figure 15: Similarité entre deux mots

Le word2vec Embeddings capture efficacement les propriétés sémantiques et arithmétiques d'un mot

- **Calcul de mots proche entre trois mots**

# Comparaison de corpus

Étant donné plusieurs corpus de taille différente, cette démo montre une comparaison entre les modèles Word2vec obtenus à partir de chaque corpus.

Sélection du modèle

Wiki corpus skipgram clean

Mot à comparer 1

informatique

Mot à comparer 2

logiciel

Mot à comparer 3

biologie

TOPN

5

Envoyer

Effacer

Les mots les plus proches de "informatique", "logiciel" et "biologie" dans le modèle "wiki\_corpus\_skipgram\_clean.mm" sont :

- utilisateur à 75%

Les mots les plus proches de "informatique", "logiciel" et "biologie" dans le modèle "wiki\_corpus\_skipgram\_clean.mm" sont :

- utilisateur à 75%
- logiciels à 75%
- serveur à 73%
- applicatif à 73%
- émulateur à 71%

On a choisi le modèle skipgram clean (vu qu'il est plus pertinent).

On a introduit trois mots qui sont : informatique, logiciel et biologie pour un Top N de 5.

La figure ci-dessus nous montre les résultats retournés.

- **Comparaison entre les modèles cbow et skipgram :**

Dans notre cas, le skipgram a donné de meilleurs résultats par rapport au cbow, si on peut faire une description assez simpliste et plutôt naïve de la différence : Comme nous le savons, CBOW apprend à prédire le mot en fonction du contexte. Ou maximisez la probabilité du mot cible en regardant le contexte. Et cela arrive à être un problème pour les mots rares. Par exemple, étant donné le contexte yesterday was a really [...] day le modèle CBOW vous dira probablement que le mot est beautiful ou nice. Des mots comme delightful auront beaucoup moins d'attention

dans le modèle, car il est conçu pour prédire le mot le plus probable. Ce mot sera lissé sur beaucoup d'exemples avec des mots plus fréquents. D'autre part, le modèle skip-gram est conçu pour prédire le contexte. Étant donné le mot delightful il doit le comprendre et nous dire qu'il existe une probabilité énorme que le contexte soit yesterday was a really [...] day ou un autre contexte pertinent. Avec skip-gram, le mot deslightful n'essaiera pas de rivaliser avec le mot beautiful mais les paires delightful+context seront traitées comme de nouvelles observations

## Conclusion

Au cours de ce chapitre, nous avons présenté l'environnement de développement afin de mettre en œuvre notre approche nous avons également présenté le langage de programmation utilisé ainsi que les différentes bibliothèques et matériel utilisés, nous avons également montré notre approche de mise en œuvre depuis la création de notre corpus jusqu'au calcul de la similarité. Nous avons également présenté notre application dans laquelle nous avons présenté les résultats de notre travail, pour finir nous avons clôturé ce chapitre par une conclusion

# Conclusion générale



Nous avons présenté dans ce mémoire notre travail qui s'inscrit dans le calcul de similarité et l'affichage des mots similaires d'un mot en utilisant word2vec.

Word2vec est un modèle de Word Embeddings et parmi les plus connus, et il a été développé par une équipe de recherche de Google sous la direction de Tomas Mikolov.

Cette approche joue un rôle important dans le développement de notre travail parce que word2vec gère les textes et est facile à appliquer en plus de donner des résultats impressionnants, il donne à chaque mot plusieurs mots ayant le même sens ou la même convergence de sens.

En perspectives, nous proposons de considérer entre autres les aspects suivants :

Ajouter une étape de correction grammaticale automatique au début de l'étape de traitement de texte afin de corriger les erreurs et les ambiguïtés dans un texte.

## □ Bibliographie

- [1] BENAÏSSA, Bedr-Eddine. Construction semi-automatique d'ontologies à partir de textes arabes. 2011.
- [2] Benoît TROUVILLIEZ, Traitement Automatique des Langues (TAL), Intelligence Artificielle (IA), Analyse sémantique et Clusterings, 31 mars 2010.
- [3] CERQUIGLINI, Jacqueline, DESBORDES, Françoise, EBBESSEN, Sten, et al. L'Ambiguïté: Cinq études historiques réunies par Irène Rosier. Presses Univ. Septentrion, 1988.
- [4] EL NAQA, Issam et MURPHY, Martin J. What is machine learning?. In : Machine Learning in Radiation Oncology. Springer, Cham, 2015. p. 3-11
- [5] FUCHS, Catherine et HABERT, Benoit. Le traitement automatique des langues: des modèles aux ressources. 2004.
- [6] HAMMO, Bassam, ABU-SALEM, Hani, LYTINEN, Steven L., et al. QARAB: A: Question Answering System to Support the Arabic Language. In : Proceedings of the ACL-02 workshop on Computational approaches to semitic languages. 2002
- [7] JÄNICKE, Stefan, FRANZINI, Greta, CHEEMA, Muhammad Faisal, et al. Visual text analysis in digital humanities. In : *Computer Graphics Forum*. 2017. p. 226-250
- [8] MÉVERGNIES, François-Xavier Nève. Le hasard et la nécessité en linguistique. Réflexions sur la téléonomie des langues naturelles. *La Linguistique*, 1976, vol. 12, no Fasc. 1, p. 35-49.
- [9] NOUVEL, Damien, EHRMANN, Maud, et ROSSET, Sophie. *Les entités nommées pour le traitement automatique des langues*. ISTE Group, 2015.
- [10] STANFORD NLP GROUP, et al. Stanford Named Entity Recognizer [Online] Available from: <http://www-nlp.stanford.edu/software.CRFNER.html>, 2006

- [11] SILBERZTEIN, Max. NooJ : à linguistic annotation system for corpus processing. In : Proceedings of HLT/EMNLP 2005 Interactive Demonstrations. 2005. p. 10-11
- [12] SILBERZTEIN, Max et TUTIN, Agnès. NooJ, un outil TAL pour l'enseignement des langues. Application pour l'étude de la morphologie lexicale en FLE. Alsic. Apprentissage des Langues et Systèmes d'Information et de Communication, 2005, vol. 8, no 2.
- [13] YVON, François. Des apprentis pour le traitement automatique des langues. Mémoire d'habilitation à diriger des recherches, Université Pierre et Marie Curie, Paris, 2006.
- [14] YVON, François. Une petite introduction au traitement automatique des langues naturelles. In : Conference on Knowledge discovery and data mining. 2010. p. 27-36.
- [25] BOCHRA, GHENNAM et SAFIA, SMARA. Les réseaux de neurone convolutionnel (CNN) pour la classification des images associées aux places de stationnement d'un parc de véhicule. 2019.
- [26] BUREL, Gilles, CATROS, Jean-Yves, POTTIER, Isabelle, et al. Réseaux de neurones en traitement de l'image et du signal. 1993.
- [27] CHAKRAVARTHY, Srinivasa V. et GHOSH, Joydeep. Scale-based clustering using the radial basis function network. *IEEE Transactions on Neural Networks*, 1996, vol. 7, no 5, p. 1250-1261.
- [28] CORNUÉJOLS, Antoine et MICLET, Laurent. Apprentissage artificiel : concepts et algorithmes. Editions Eyrolles, 2011
- [29] DJOKHRAB, Ala Eddine. Planification et Optimisation de Trajectoire d'un Robot Manipulateur à 6 DDL par des Techniques Neuro-Floues. 2015. Thèse de doctorat. Université Mohamed Khider-Biskra.
- [30] DREYFUS, G., MARTINEZ, J. M., SAMUELIDES, M., *et al.* Réseaux de neurones : méthodologies et applications. Édition EYROLLES Avril 2004. 2004
- [31] IDIR, Mellal. Implémentation d'un réseau de neurones d'un micro capteur sur un FPGA. 2010. Thèse de doctorat. Université Mouloud Mammeri
- [32] MASSINE, G. A. N. A. Implémentation d'un réseau de neurones dans un microcontrôleur. 2016. Thèse de doctorat. Université Mouloud Mammeri.
- [33] NAKAMOTO, Pat. Neural Networks and Deep Learning : Deep Learning explained to your granny A Visual introduction for beginners who want to make their own Deep Learning Neural Network. CreateSpace Independent Publishing Platform, 2017.

- [34] NERRAND, Olivier, ROUSSEL-RAGOT, Pierre, PERSONNAZ, Léon, *et al.* Neural networks and nonlinear adaptive filtering : Unifying concepts and new algorithms. *Neural computation*, 1993, vol. 5, no 2, p. 165-199
- [35] PARIZEAU, Marc. Réseaux de neurones. GIF-21140 et GIF-64326, 2004, vol. 124
- [36] ZACCONE, Giancarlo, KARIM, Md Rezaul, et MENSRAWY, Ahmed. Deep Learning with TensorFlow. Packt Publishing Ltd, 2017.
- [37] TOUZET, Claude. les réseaux de neurones artificiels, introduction au connexionnisme. 1992
- [38] BENGIO, Yoshua. Learning deep architectures for AI. Now Publishers Inc, 2009.
- [39] BENGIO, Yoshua, DUCHARME, Réjean, VINCENT, Pascal, et al. A neural probabilistic language model. *Journal of machine learning research*, 2003, vol. 3, no Feb, p. 1137-1155.
- [40] DEVLIN, Jacob, CHANG, Ming-Wei, LEE, Kenton, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [41] DUSSERRE, Emmanuelle. Utilisation de la méthode distributionnelle pour la constitution de classes sémantiques d'une liste de formes du lexique scientifique transdisciplinaire. 2016.
- [42] ELMAN, Jeffrey L. Finding structure in time. *Cognitive science*, 1990, vol. 14, no 2, p. 179-211.
- [43] FIRTH, John Rupert. A synopsis of linguistic theory 1930-55., volume 1952-59. The Philological Society. 1957.
- [44] HARRIS, Zellig S. Distributional structure. *Word*, 1954, vol. 10, no 2-3, p. 146-162.
- [45] HEWITT, John et MANNING, Christopher D. A structural probe for finding syntax in word representations. In : *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)*. 2019. p. 4129-4138.
- [46] KHODJA, Ala Eddine. Un système d'extraction d'information pour la langue arabe. 2017. Thèse de doctorat. FACULTE: Mathématique et Informatique-UNIVERSITE MOHAMED BOUDIAF-M'SILA.
- [47] LEVY, Omer et GOLDBERG, Yoav. Dependency-based word embeddings. In : *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2014. p. 302-308.

- [48] MAATEN, Laurens van der et HINTON, Geoffrey. Visualizing data using t-SNE. *Journal of machine learning research*, 2008, vol. 9, no Nov, p. 2579-2605
- [49] MIKOLOV, Tomas, SUTSKEVER, Ilya, CHEN, Kai, et al. Distributed representations of words and phrases and their compositionality. In : *Advances in neural information processing systems*. 2013. p. 3111-3119
- [50] MIKOLOV, Tomáš, YIH, Wen-tau, et ZWEIG, Geoffrey. Linguistic regularities in continuous space word representations. In : *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*. 2013. p. 746-751.
- [51] ROCCHIO, Joseph. Relevance feedback in information retrieval. *The Smart retrieval system-experiments in automatic document processing*, 1971, p. 313-323
- [52] SALTON, Gerard. Some experiments in the generation of word and document associations. In : *Proceedings of the December 4-6, 1962, fall joint computer conference*. 1962. p. 234-250.
- [53] SCHWENK, Holger, DÉCHELOTTE, Daniel, et GAUVAIN, Jean-Luc. Continuous space language models for statistical machine translation. In : *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*. 2006. p. 723-730.
- [58] KOUAHLA, Zineddine. Plateforme de développement pour l'internet des objets (IdO) avec un apprentissage automatique. 2019.
- [59] ROSSUM, Guido. *Python reference manual*. 1995.
- [61] VU, Hai-Hieu, VILLANEAU, Jeanne, SAÏD, Farida, *et al.* Mesurer la similarité entre phrases grâce à Wikipédia en utilisant une indexation aléatoire
- [96] VU, Hai-Hieu, VILLANEAU, Jeanne, SAÏD, Farida, *et al.* Mesurer la similarité entre phrases grâce à Wikipédia en utilisant une indexation aléatoire

# Webographies

[54] <https://fr.sciencewal.com/28496-what-the-heck-is-word-embedding-b30f67f01c81-86>

[57] <https://riptutorial.com/fr/nltk/topic/4077/demarrer-avec-nltk>

[60] <https://python-guide-pt-br.readthedocs.io/fr/latest/scenarios/scientific.html>.

[98] [http : //www.tresfacile.net/introduction-au-langage](http://www.tresfacile.net/introduction-au-langage)

[60] <https://python-guide-pt-br.readthedocs.io/fr/latest/scenarios/scientific.html>