Ministère de l'Enseignement Supérieur de la Recherche et de l'Innovation

Université Assane Seck de Ziguinchor

UFR Sciences et Technologies

Département Informatique

MEMOIRE DE FIN D'ETUDES

Pour l'obtention du diplôme de Master

Mention : Informatique

Spécialité : Génie Logiciel

Sujet

Approche à base d'apprentissage automatique de calcul de similarité entre paires de phrases dans le domaine clinique

Présenté par :
M. Papa Matar DIENG

Sous la supervision de :
Pr Youssou FAYE
Sous la direction de:
Dr Khadim DRAME

Soutenu le 29/10/2022 devant le jury composé de :

Dr Khadim DRAME	Maître de Conférences Titulaire	Encadrant	UASZ
Pr Youssou FAYE	Professeur Assimilé	Président	UASZ
Dr Gorgoumack SAMBE	Maître de Conférences Titulaire	Rapporteur	UASZ
Dr Mouhamadou GAYE	Maître de Conférences Titulaire	Rapporteur	UASZ

Année Universitaire 2021/2022

Dédicaces et Remerciements

Je dédié au fond de moi ce Mémoire
A mon ami frère défunt
Mamour DIOUF qui nous a quitté il y'a un an et à ma grand-mère Ndeye Fatou DIENG. Que la terre leur soit légère.

C'est avec un immense plaisir que je réserve cette page en signe de gratitude et de profonde reconnaissance à tous ceux qui, de prêt ou de loin, ont contribué à l'aboutissement de ce modeste travail.

Ma gratitude, ma reconnaissance et mes remerciements vont d'abord à mon encadreur. Je veux nommer Monsieur Khadim DRAMÉ, enseignant-chercheur à l'université Assane Seck de Ziguinchor (UASZ) pour sa disponibilité, sa patience et son efficacité tout au long de ce travail et pour la confiance qu'il m'a accordée ; ce qui m'a valu le titre de bénéficier d'un sujet de qualité et sans oublier les ressources de qualité qu'il n'a jamais cessé de mettre à notre disposition pour un travail de qualité .ils vont aussi aux membres du jury pour avoir accepté d'examiner notre document.

Je remercie tous les enseignants du département Informatique de l'UASZ pour la formation de qualité qu'ils m'ont procuré.

L'aboutissement de ce travail ne serait possible sans l'appui considérable et indispensable que m'a apporté ma famille. Je la remercie pour l'éducation et l'encadrement de qualité que m'ont donné mes parents durant mon enfance jusqu'à l'heure où nous sommes. Je tiens aussi à remercier profondément mon grand frère Diabel DIENG pour son soutien et ses sacrifices sans cesse pour notre réussite.

Je ne peux pas terminer sans remercier ma famille à Ziguinchor pour leur soutien (Kiné Diallo) et mes camarades avec qui je partage les meilleurs moments de ma vie estudiantine je veux nommer Bathie SENE et Abdou SECK.

Résumé

L'étude de la similarité sémantique entre concepts est une tâche importante dans le traitement de texte et dans de nombreux domaines du traitement automatique des langues comme l'extraction d'information, la recherche documentaire, la classification de textes et la traduction automatique. Du fait de manque de ressources linguistiques, cette tâche reste un défi dans beaucoup domaines.

Dans ce mémoire, nous proposons une approche de calcul de similarité entre paires de phrases écrites en Français dans le domaine clinique. Ce processus d'étude de données textuelles se rattache exclusivement au domaine du traitement automatique des langues qui propose un certain nombre d'applications pour aider à la compréhension et au traitement automatique du langage humain.

Beaucoup d'approches de calcul de similarité textuelle ont été proposées dans la littérature. Certaines approches couramment utilisées exploitent la structure syntaxique des phrases. D'autres tentent de prendre en compte les problèmes de synonymie et la sémantique des phrases en exploitant des ressources sémantiques ou des méthodes statistiques.

Nous proposons une approche basée sur l'apprentissage automatique pour calculer la similarité entre des paires de phrases dans le domaine clinique. Cette approche consiste à entrainer un algorithme à partir de données d'entraînement pour qu'il puisse apprendre à reconnaître les similitudes et les différences entre les paires de phrases. Ainsi, cinq algorithmes d'apprentissage automatique supervisé sont explorés et évalués sur des données standards. Les évaluations réalisées nous ont permis de choisir le meilleur modèle sur les données qui est obtenu avec le perceptron multicouche avec un score d'exactitude de 59%.

Abstract

The study of semantic similarity between concepts is an important task in text processing and in many areas of automatic language processing such as information retrieval, document retrieval, text classification and machine translation. Due to the lack of linguistic resources, this task remains problematic in some areas.

In this thesis, we propose an approach to calculating similarity between pairs of sentences written in French in the clinical field. This process of studying textual data is exclusively related to the field of automatic language processing, which offers a number of applications to help the understanding and automatic processing of human language.

Many approaches to calculating textual similarity have been proposed in the literature. Some commonly used approaches exploit the syntactic structure of sentences. Others try to take into account the problems of synonymy and the semantics of sentences by exploiting semantic resources or statistical methods.

We propose a machine learning-based approach to calculate the similarity between sentence pairs in the clinical field. This approach involves training an algorithm with training data so that it can learn to recognize similarities and differences between sentence pairs. Thus, five supervised machine learning algorithms are explored and evaluated on standard data. The evaluations carried out allowed us to choose the best model on our data that is obtained with the multilayer perceptron with an accuracy score of 59%.

Table des Matières

Liste des figures

Chapitre I

Chapitre II

Chapitre III

Chapitre IV

Liste des captures

Chapitre I

Chapitre IV

Liste des tableaux

Chapitre III

Chapitre IV

Liste des équations

Introduction générale

La sémantique fait partie intégrante des langues naturelles. Elle est définie selon « Wikipédia » comme une branche de la linguistique qui étudie le sens des expressions linguistiques (mots, phrases, textes, documents, etc.) c'est-à-dire ce dont on parle, ce qu'on veut transmettre par un énoncé [1]. Deux expressions linguistiques ayant la même signification sont dites similaires sémantiquement (signifient la même chose).

L'étude de la similarité sémantique entre expressions linguistiques a toujours été le sujet au quotidien dans plusieurs domaines de la communauté scientifique et a toujours reçu une attention particulière de cette dite communauté. C'est une métrique définie sur un ensemble de documents ou de termes correspondants à la distance entre les éléments en se basant sur la ressemblance de leur signification ou de leur contenu sémantique. La similarité sémantique est utilisée dans plusieurs domaines [2].

Dans le domaine du traitement automatique des langues (TAL), la similarité sémantique est utilisée dans plusieurs applications telles que l'analyse des sentiments, la traduction automatique, la compréhension du langage naturel, etc.

En s'aidant du contexte, l'être humain a la capacité dans la plupart des cas de mesurer la similarité sémantique entre deux phrases écrites dans son langage naturel sans fournir trop d'efforts. Cependant, en traitement automatique des langues, cette tâche reste problématique du fait du manque de structuration des textes exprimés en langue naturelle.

Face à cette situation, beaucoup d'approches ont été proposées dans la littérature. Certaines utilisent des taxonomies telles que **WordNet** qui est une base de données lexicale contenant des mots en langue anglaise associés à leurs synonymes et leurs définitions. Ces approches présentent quelques limites notamment dans le domaine biomédical et pour les langues comme le Français où les ressources sont rares.

Ainsi, dans le cadre de notre mémoire de Master, nous proposons une approche supervisée de calcul de similarité textuelle basée sur des algorithmes d'apprentissage automatique (machine learning en anglais).

Ce mémoire est subdivisé en quatre chapitres.

Dans le chapitre I, nous faisons une présentation globale des concepts de base du traitement automatique des langues (TAL) et particulièrement ceux utilisés pour le calcul de similarité textuelle.

Dans le chapitre II, nous faisons un état de l'art sur les approches de calcul de similarité entre textes proposées dans la littérature.

Dans le chapitre III, nous décrivons notre approche dans sa globalité en présentant toutes ses étapes.

Enfin, nous terminons avec le chapitre IV où nous évaluons les méthodes proposées sur des jeux de donnée standards, puis analyser et discuter leurs résultats.

Chapitre I : Concepts de base

I.1 Introduction

Dans ce chapitre, nous allons décrire des concepts fondamentaux du TAL. Dans un premier temps, il sera question de comprendre en quoi consiste le TAL. Ainsi, il sera nécessaire au préalable d'appréhender le contexte de sa création et les principes sur lesquels repose son fonctionnement. Dans un deuxième temps, quelques domaines d'application du TAL seront présentés. Le TAL étant un domaine pluridisciplinaire, la liste ne pourra être exhaustive mais va permettre de décrire quelques-uns d'entre eux pour mieux situer notre travail dans le TAL.

Enfin, dans la troisième et dernière partie de ce chapitre, une description de quelques outils du TAL utilisés dans le cadre de notre travail est faite. Pour rappel notre travail porte sur le calcul de similarité sémantique entre paires de phrases. Nous allons décrire les fonctionnalités de chacun de ces outils en les illustrant par des exemples pratiques.

I.2 Contexte et définition du TAL

Depuis son existence, l'être humain est confronté à un certain nombre de problèmes qu'il cherche à résoudre. Ainsi, il inventa l'ordinateur pour s'en procurer d'une aide sur la résolution de ses problèmes. Cette aide qui peut se magnifier en gain de temps dans la réalisation des tâches ou même la précision de la résolution des problèmes nécessitant toujours la présence de l'homme. De ce fait il doit interagir avec l'ordinateur pour une telle réalisation.

Le moyen de communication à l'aide duquel les humains peuvent échanger (parler, lire et écrire), c'est le langage. En d'autres termes, les humains peuvent penser, faire des plans, prendre des décisions dans leur langage naturel. La grande question est à l'ère de l'intelligence artificielle, les humains peuvent-ils communiquer en langage naturel avec des ordinateurs/machines ?

Le traitement automatique du langage naturel (TALN) ou traitement automatique des langues (TAL) (natural language processing (NLP) en anglais) s'intéresse à cette question ; il a pour objectif de développer des méthodes et outils qui permettent aux ordinateurs de traiter des données non structurées notamment des textes exprimées dans des langues naturelles.

Le développement du TAL a réellement commencé lors de la guerre froide marquant l'avènement du premier traducteur automatique avec 250 mots et 6 règles grammaticales [3].

Le TAL est un sous domaine de l'informatique plus spécifiquement de l'intelligence artificielle (IA) qui permet aux ordinateurs de comprendre, traiter et manipuler le langage humain [4].

C' est un domaine multidisciplinaire impliquant la linguistique, l'informatique et l'intelligence artificielle, qui vise à créer des outils de traitement de la langue naturelle pour diverses applications [5]. Du fait de son caractère pluridisciplinaire, il fait collaborer plusieurs

spécialistes appartenant à des domaines différents: linguistes, informaticiens, logiciens, psychologues, documentaliste, lexicographes, traducteurs, etc.

Le TAL est aussi une réalité socio-économique, avec des entreprises et des produits spécialisés tels que les correcteurs d'orthographes, les logiciels de traduction, la dictée vocale, etc. [6]**.** En d'autres termes, le TAL est un moyen pour les machines/ordinateurs d'analyser, de comprendre et de tirer du sens des langues humaines comme le français, l'anglais, l'espagnol, etc. [4].

Pour se faire, il existe deux principales catégories de méthodes de traitement automatique des langues qui peuvent être employées soit de manière indépendante ou hybride [7]:

- Les méthodes statistiques

Les méthodes statistiques considèrent la langue comme un objet mathématique pour permettre d'appliquer différentes formules probabilistes et statistiques aux données de la langue (mots, phrases, etc.).

- Les méthodes linguistiques

Les méthodes linguistiques s'appuient sur différents niveaux d'analyse utilisés en linguistique comme la phonétique, la syntaxe ou la sémantique. Elles ont l'avantage d'être très performantes mais sont très coûteuses en temps.

Des études ont montré qu'en 2019 le TAL restait un des domaines de recherche les plus actifs en sciences de données. C'est un domaine en interaction avec le machine learning (apprentissage automatique) et la linguistique. Son but est d'extraire des informations et une signification d'un contenu textuel [8].

I.3 Composants du TAL

Le TAL est un domaine regroupant les techniques permettant à un programme informatique d'analyser et d'interpréter le langage naturel [9]. Il comprend de multiples techniques diverses qui permettent d'interpréter le langage humain, allant des méthodes statistiques et d'apprentissage automatique aux approches fondées sur des règles. Un vaste choix d'approches est utilisé car les données textuelles et vocales varient considérablement. Ses différentes techniques ou tâches comprennent entre autres : la tokenisation, l'étiquetage morpho-syntaxique, la lemmatisation (recherche du radical), la racinisation, la détection des parties d'un discours, la détection de la langue et l'identification des relations sémantiques.

Ces techniques permettent de décomposer le langage en éléments essentiels courts, et tentent de comprendre les relations entre ces éléments et comment ils s'imbriquent pour créer du sens [10]**.**

Dans cette partie, nous présentons quelques composants du TAL qui sont considérés comme un ensemble de tâches ou d'étapes réalisable lors du traitement de données textuelles. La figure ci-après (Figure I-1) donne une approche sur ces composants et leur séquencement.

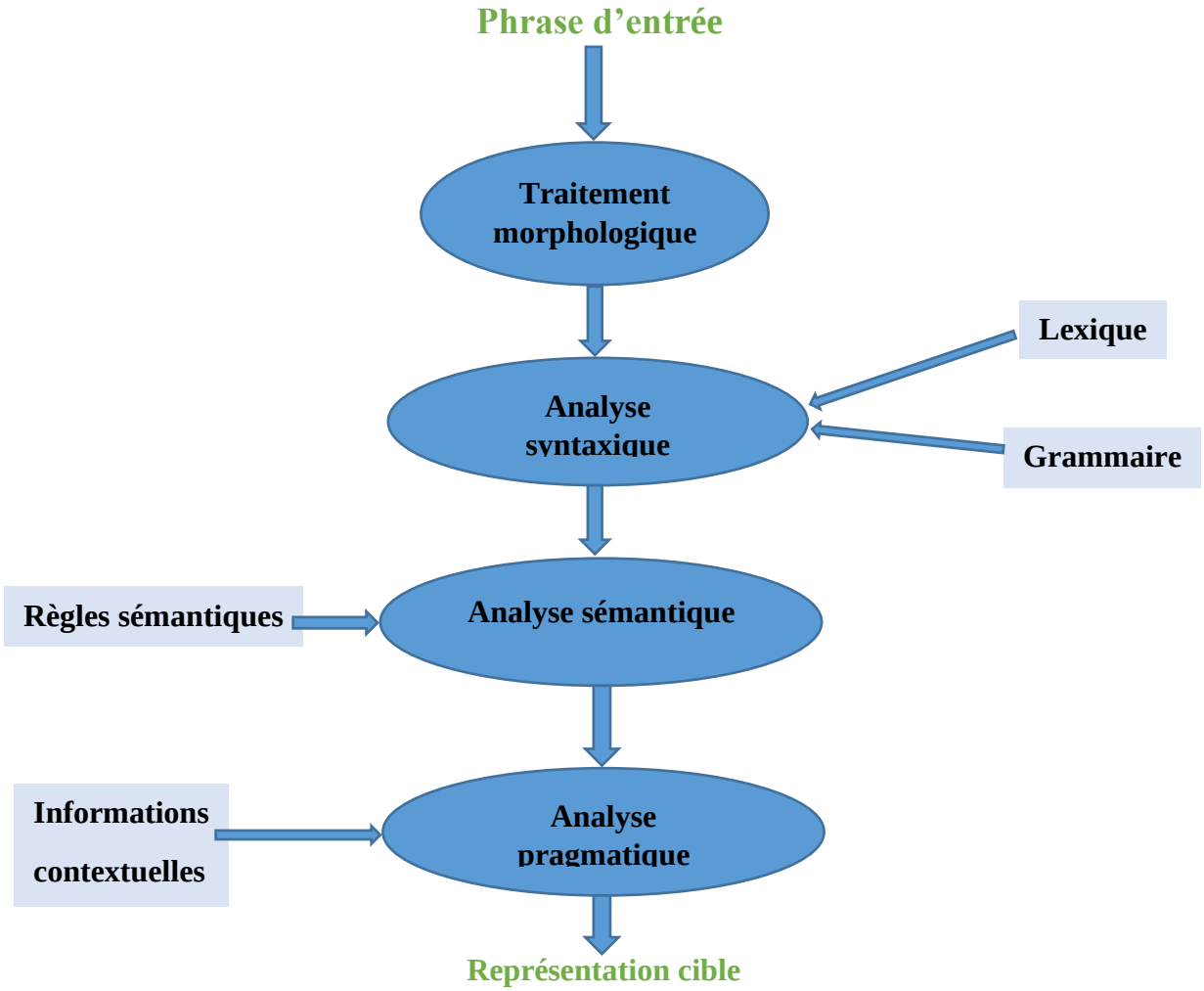

Figure I-1 : Composants du TAL [2]

- Le traitement morphologique

Le traitement morphologique est le premier composant du TAL. Il comprend la décomposition de morceaux de langue d'entrée en ensembles de jetons correspondant à des paragraphes, des phrases et des mots. Par exemple, un mot comme **« tous les jours »** peut être **«** tous les jours **».**

- L'analyse syntaxique

L'analyse syntaxique, le deuxième composant est l'un des composants les plus importants du TAL. Les objectifs de ce composant sont les suivants :

- vérifier si une phrase est bien formée ou non.
- la décomposer en une structure qui montre les relations syntaxiques entre les différents mots.
- Par exemple, les phrases comme **« L'école va a l'élève »** seraient rejetées par l'analyseur de syntaxe.

❖ L'analyse sémantique

L'analyse sémantique est le troisième composant du TAL qui est utilisé pour vérifier la signification du texte. Cela comprend le dessin de la signification exacte, où nous pouvons dire la signification du dictionnaire à partir du texte. Par exemple, les phrases comme "**C'est une glace chaude**" seraient rejetées par l'analyseur sémantique.

❖ L'analyse pragmatique

L'analyse pragmatique est le quatrième composant du TAL. Cela comprend l'ajustement des objets ou événements réels qui existent dans chaque contexte avec des références d'objet obtenues par le composant précédent, c'est-à-dire l'analyse sémantique. Par exemple, les phrases comme "Mettez les fruits dans le panier sur la table " peuvent avoir deux interprétations sémantiques, donc l'analyseur pragmatique choisira entre ces deux possibilités [4].

Par la suite, nous allons nous intéresser sur les deux derniers composants (sémantique et pragmatique) pour introduire la similarité sémantique.

I.4 Quelques domaines d'applications du TAL

Le traitement automatique des langues (TAL), du fait de son caractère incontournable, englobe plusieurs domaines d'application qui à leur tour se servent du TAL pour réaliser certaines tâches indispensables à leur évolution. Parmi les applications les plus connues, on peut citer : la traduction automatique, la correction orthographique, la recherche d'information et la fouille de textes, le résumé automatique de texte, etc.

Dans la suite, nous allons apporter des descriptions en guise d'approche pour mieux les scinder.

Outre celles citées, il existe tant d'autres telles que la recherche et l'extraction d'informations sur le web, la synthèse automatique de textes, les systèmes de questions-réponses, l'analyse des sentiments, la détection de spams, etc. [4]**.**

I.4.1 La traduction automatique

Le TAL est né des besoins de traduction. Bien qu'aujourd'hui son emploi soit beaucoup plus diversifié, son usage premier a été le développement de technologies de traduction automatique (TA) assez performantes pour décoder les messages Russes lors de la guerre froide. Ainsi la traduction automatique est considéré comme mère des activités du TAL [11]. Elle est l'une des

applications les plus importantes du traitement du langage naturel. La TA est essentiellement un processus de traduction d'une langue source ou d'un texte dans une autre langue.

La traduction automatique utilise un dispositif informatique pour traduire un texte d'une langue dite « source », dans laquelle est écrit le texte vers une langue dite « cible » dans laquelle on souhaite obtenir une traduction [12]. La figure ci-après (Figure I-3) donne un exemple expliquant le mécanisme de traduction automatique d'un texte en langue française vers la langue anglaise.

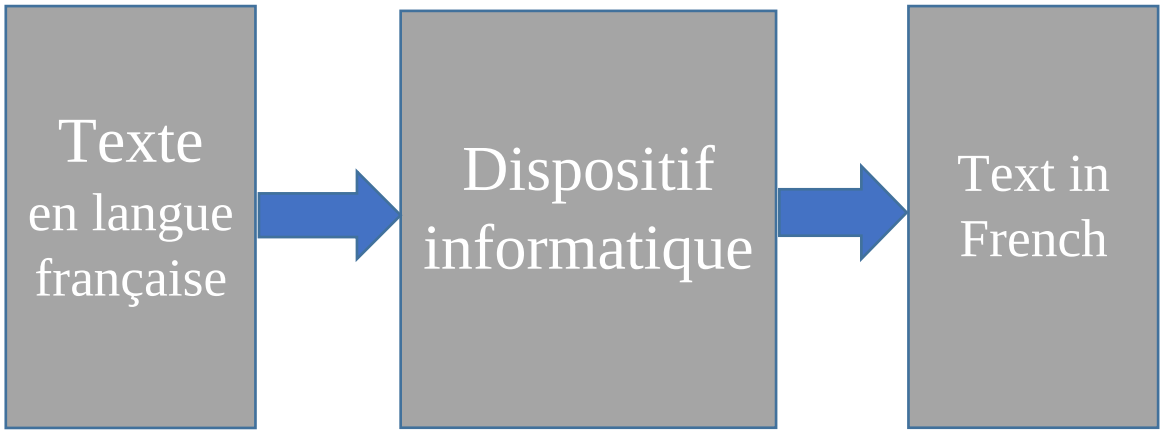

Figure I-2 : Mécanisme de la traduction automatique

Pour traduire un texte d'une langue vers une autre, il faut avoir dans ces deux langues une bonne connaissance de la grammaire en particulier, de la syntaxe c'est-à-dire la structure des phrases ainsi que de la sémantique, le sens des mots. Mais il est aussi nécessaire de connaitre la culture implicite, les connotations etc. L'objectif des systèmes de traduction automatique n'est pas de fournir une seule « bonne réponse » mais plutôt de produire des traductions compréhensibles et conservant le sens des textes d'origine [12].

Différentes approches ont été proposées, comme les systèmes de traduction fondés sur des règles ou sur des exemples. Actuellement, les méthodes les plus fréquemment utilisées pour la traduction automatique sur le web sont les méthodes statistiques [12].

I.4.2 La correction grammaticale et orthographique

La correction orthographique et grammaticale est une fonctionnalité très utile des logiciels de traitement de textes comme Microsoft Word. Le traitement du langage naturel est largement utilisé à cette fin [4].

Trouver une solution extérieure au rédacteur pour corriger d'éventuelles erreurs orthographiques relève d'une préoccupation ancienne qui date d'avant l'informatique à proprement parler. En 1918, puis en 1922, **Odell** et **Russell** [13] avaient déjà proposé des

méthodes pour la correction orthographique. Mais c'est à partir de 1957 qu'ont été développés des programmes destinés à corriger des erreurs orthographiques :

- dans des mots isolés, c'est-à-dire des mots figurant sur des listes, par exemple les noms mentionnés sur les listes de passagers des compagnies aériennes ;
- dans les mots clés des bases de données.

Ces premiers programmes ont permis la conception de correcteurs dits de première génération, dont le processus correspond à celui de « correcteurs lexicaux » stricto sensu[1]. Leur base de données est une liste de mots ordonnée alphabétiquement à laquelle est comparée la liste non ordonnée alphabétiquement des mots du texte. La comparaison se fait entre des chaînes de caractères : si le mot du texte n'est pas dans la liste de référence, le correcteur signale une erreur. Ce type de correcteurs est aujourd'hui le fondement de la plupart des correcteurs intégrés aux logiciels de traitement de textes.

Les correcteurs dits de deuxième génération sont des « correcteurs grammaticaux heuristiques[2] ». Ils représentent un progrès considérable sur les précédents dans la mesure où ils font intervenir des grammaires locales, qui analysent le contexte immédiat du mot et permettent de déterminer par exemple les règles d'accord propres à la langue pour laquelle ils ont été élaborés. Toutefois ces règles ne sont appliquées qu'à des contextes simples et localisés. C'est ce type de correcteur qui est encore utilisé dans les correcteurs grammaticaux intégrés dans les traitements de textes.

Enfin, les correcteurs dits de troisième génération fonctionnent sur une analyse grammaticale complète de phrases complexes. Autonomes, beaucoup plus élaborés, multifonctionnels, ces correcteurs sont maintenant intégrables dans plusieurs logiciels d'aide à la rédaction (traitement de textes, messageries, moteurs de recherche, Publication Assistée par Ordinateur (PAO), …) parmi les plus courants prédéfinis par le concepteur du logiciel (Word, Wordperfect, ClarisWorks,…). Ils emploient entre autres des technologies relevant de l'intelligence artificielle. Mais la complexité même qui en permet les performances est la cause de nombreuses fausses détections. Leur fonctionnement fait appel à des technologies innovatrices en matière d'intelligence artificielle [13].

I.4.3 La recherche et l'extraction d'informations sur le web

Internet est devenu un outil de recherche de plus en plus privilégié, voire indispensable. Le volume phénoménal d'information désormais disponible (plus de trois milliards de pages Web en 2004) laisse croire que la communication approche la perfection c'est-à-dire tout ce qu'on désire savoir se trouverait à portée de la main, à quelques clics de soi. La présence croissante du www (world wide web) dans les universités confirme son utilité pour la recherche universitaire et gagnera en importance dans les années à venir [14].

[1] Au sens littéral, au sens propre du mot ou de l'expression employés.
[2] Partie de la science qui a pour objet les procédures de recherche et de découverte.

La recherche d'une information précise sur internet est devenue un enjeu important. C'est pourquoi il est capital de bien préparer sa recherche, de se familiariser avec les outils disponibles, de connaître les principales ressources propres à son domaine d'étude et de disposer d'un esprit suffisamment critique pour choisir la bonne information [14].

Sur internet, les utilisateurs quand ils ont besoins d'informations font une recherche d'informations (RI). En d'autres termes on peut dire que cette recherche est déclenchée suite à un besoin d'informations que l'utilisateur exprime. Pour se faire, il utilise des navigateurs web qui à leur tour intègrent des moteurs de recherches leur permettant de réaliser les recherches afin de récupérer l'information nécessaire pour l'utilisateur.

La plupart des moteurs de recherche comme Google, Yahoo, Bing, sont basés sur les modèles d'apprentissage en profondeur. Ces modèles d'apprentissage en profondeur permettent aux algorithmes de lire du texte sur une page web, d'interpréter sa signification et de le traduire dans une autre langue [4].

Wikipédia définie la recherche d'information sur le web à l'aide d'un moteur de recherche comme une technique de l'information et de la communication, désormais massivement adoptée par les usagers [15].

I.4.4 La synthèse automatique de textes

La synthèse automatique de textes ou encore résumé automatique de textes fait référence à la technique de raccourcissement de longues portions de textes. L'intention est de créer un résumé cohérent et fluide ne contenant que les principaux points exposés dans le document. La synthèse automatique de textes est donc un problème courant dans l'apprentissage automatique et le traitement du langage naturel (TAL). Le but est de remplacer le texte original afin d'inciter le lecteur à lire le contenu en ayant une idée claire [16]. En d'autres termes c'est une technique qui crée un court résumé précis de documents textes plus longs. Par conséquent, cela nous aide à obtenir des informations pertinentes en moins de temps. En cette ère numérique, nous avons un besoin sérieux de récapitulation automatique des textes vu le flot d'informations sur Internet qui ne va pas s'arrêter. Le TAL et ses fonctionnalités jouent un rôle important dans le développement d'un résumé de textes automatique [4]. Selon de nombreuses analyses menées sur des articles, la fréquence de lecture est étroitement liée à la taille du texte. Souvent le résumé est le seul élément à être lu. De plus, la rédaction de résumé représente un travail humain assez important et engendre un coût [16].

Il existe de nombreux cas d'usage pour le résumé automatique dont voici quelques exemples. Le plus général est tout simplement la simplification de la lecture.

Dans le domaine scolaire, cette technique peut promouvoir le e-learning[3]. Par exemple, un cas courant est d'utiliser cette méthode comme aide aux enseignants pour la conception de nouveaux sujets.

[3] L'e-learning (parfois orthographié « e-learning »), ou formation en ligne, permet de suivre une formation à distance.

Dans le contexte éditorial, on peut retrouver la rédaction du résumé de livres, romans ou autres. Cette tâche est encore manuelle et difficile à réaliser. En automatisant le résumé en le rendant encore plus qualitatif, on poussera le consommateur à acheter plus rapidement car il en saura davantage sur le sujet à l'avance.

Dans le domaine scientifique, on retrouve beaucoup d'usages spécifiques. Un usage très pertinent serait le ciblage d'informations clés dans des rapports pour une rapide prise de connaissance des idées d'un texte (médical, patients, marché financier, etc.). Exemple : recenser tout l'historique médical d'un patient à destination d'un médecin. Le praticien pourra prendre en charge le patient avec une vision globale de son dossier [16].

I.4.5 Les systèmes de questions-réponses

La capacité d'une machine à comprendre le langage humain et à répondre de manière pertinente à une question posée est l'un des principaux défis en matière d'intelligence artificielle.

Le traitement du langage naturel a connu une progression rapide grâce aux récents développements en **deep learning** et **transfer learning**, en particulier les technologies de questions-réponses qu'elles soient ouvertes (réponse à des questions d'ordre général) ou fermées (réponse à des questions dans un domaine spécifique) [17]. La réponse aux questions, une autre application principale du TAL se concentre sur la construction de systèmes qui répondent automatiquement à la question posée par l'utilisateur dans son langage naturel [4]. Pour une telle réalisation, le modèle ne pouvant pas répondre à une question par lui-même se base sur le contexte en exploitant un morceau de texte passé avec la question à répondre afin de trouver des éléments de réponses [18].

Les systèmes de questions-réponses sont très utiles dans le monde réel. Aujourd'hui les, **chatbots**[4] sont de plus en plus utilisés au quotidien à la fois pour répondre aux questions des clients mais aussi des collaborateurs internes. Par exemple un client qui pose une question juridique sur son contrat. Le système de Q/R peut facilement répondre à sa question en passant le contrat comme contexte. La mise en production de la réponse aux questions nécessite la construction d'une API[5] d'inférence qui n'est pas facile car elle inclut la construction d'une infrastructure hautement disponible, rapide et évolutive pour servir les modèles derrière [17].

I.4.6 L'analyse de sentiments

L'analyse de sentiments (Sentiment Analysis ou Opinion Mining) est l'interprétation et la classification des émotions (positives, négatives ou neutres) dans les données textuelles à l'aide de techniques d'analyse de textes. Elle permet aux entreprises d'identifier l'opinion des clients à l'égard des produits, des marques ou des services dans les conversations et les commentaires

[4] Agent conversationnel.
Un **chatbot** est un robot logiciel pouvant dialoguer avec un individu ou consommateur par le biais d'un service de conversations automatisées.

[5] Application Programming Interface. Interface de programmation d'applications contenant un ensemble de fonctions courantes de « bas niveau », bien documentées permettant de programmer des applications » de haut niveau »

en ligne et se concentre sur la polarité (positive, négative, neutre), les sentiments et émotions (colère, joie, tristesse, etc.) et même sur les intentions (par exemple, intéressé contre non intéressé). Cela permet donc différents types d'utilisation de cette méthode d'analyse [19].

- Exemple d'analyse sur la polarité :

Le système de notation d'un site de critique de cinéma.

- Exemple d'analyse sur les émotions :

Les retours clients sur le support d'un site internet.

- Exemple d'analyse basée sur l'aspect ou les caractéristiques :

Une entreprise de construction de vélos peut chercher à connaître les composants du vélo faisant l'objet de critiques (positives ou négatives).

L'analyse des sentiments fait partie des autres applications importantes du TAL.

Les sociétés de commerce électronique en ligne comme Amazon, ebay, etc. utilisent l'analyse de sentiments pour identifier les opinions et sentiments de leurs clients en ligne. Cela les aidera à comprendre ce que leurs clients pensent de leurs produits et services [4].

L'analyse des sentiments est utilisée dans plusieurs cas parmi lesquels on peut citer :

- La supervision des réseaux sociaux

En tant qu'entreprise, il est essentiel de connaître la façon dont l'extérieur nous perçoit. Utiliser l'analyse de sentiments pour obtenir ces informations peut amener à communiquer différemment ou à changer de ligne de conduite. Il peut être tout aussi intéressant d'effectuer ces analyses sans déclencheur particulier pour connaître l'opinion « en temps normal » ou encore quand l'entreprise fait une annonce.

- L'analyse des commentaires des clients

Prenons l'exemple d'une entreprise de prêt-à-porter qui vend ses vêtements en ligne par livraison. Lorsqu'un client effectue une commande sur son site internet, il est intéressant d'obtenir son feedback[6]. Cela peut se faire via un questionnaire ou tout simplement par un texte libre. Utiliser l'analyse des sentiments dans ce cas-là permettra de connaître quel article plaît ou non. En poussant l'analyse, on pourrait même en connaître les raisons [19].

Plusieurs algorithmes et méthodes de TAL sont utilisés pour l'analyse de sentiments qu'on peut regrouper en trois catégories :

- Un ensemble de règles élaborées manuellement :

[6] Retour sur une expérience, sorte de bilan tiré à partir d'un événement. Exemple : J'attends vos **feedbacks**

Exemple : la racisation de mots, étiquette d'un mot, analyse syntaxique, lexique de mots (i.e. liste des mots et expressions) etc.

- Des techniques d'apprentissage automatique (Machine Learning) à partir de données :

Exemple : l'analyse de sentiment peut être modélisée comme un problème de classification où l'on donne au modèle un texte et où il nous retourne une catégorie (e.g. positif, négatif ou neutre).

- Les systèmes hybride :

Exemple : les systèmes hybrides combinent les éléments des méthodes basées sur des règles et des techniques automatiques dans un seul système. L'un des grands avantages est une plus grande précision des résultats [19].

I.4.7 La détection de spams

En raison de l'augmentation considérable des e-mails indésirables, les filtres anti-spam sont devenus importants car c'est la première ligne de défense contre ce problème. En considérant ses problèmes de faux positifs et de faux négatifs comme les principaux problèmes, une technique de TAL peut être utilisée pour développer un système de filtrage des spams [4].

La modélisation N-gram, le mot **Stemming** et la classification bayésienne sont des techniques utilisés pour le filtrage anti-spam.

I.5 Quelques concepts du TAL

Le calcul de similarité entre paires de phrases requiert la combinaison d'un certain nombre de fonctionnalités offert par le TAL. Ces dernières viennent en complément lors du suivi des différents principes du TAL. En d'autres termes, ils sont utilisés pour la réalisation des tâches pour les différentes étapes qui définissent ces principes.

Dans cette partie, nous nous intéressons à des outils que nous utilisons pour la réalisation de notre travail c'est à dire calculer la similarité sémantique entre paires de phrases. L'objectif de cette partie n'est pas de situer les concepts mais de donner une description au mieux possible de ces derniers avec des exemples précis.

La construction d'applications de TAL nécessite des compétences spécifiques avec une grande compréhension de la langue et des outils pour traiter la langue efficacement. Ainsi, différents outils existent pour faire du TAL. Certains d'entre eux sont open-source tandis que d'autres sont développés par des organisations pour créer leurs propres applications de TAL. Parmi ces outils

on peut citer la **Boite à outil de Standford**[7], **Gensim**[8], **UIMA**[9], **Open NLP**[10], **GATE**[11], **Mallet** [12]et le **la boite à outil NLTK** [4]**.** La plupart des outils sont écrits en JAVA.

I.5.1 Le Natural Language ToolKit (NLTK)

NLTK (Naturel Language Toolkit) est une boîte-à-outil permettant la création de programmes pour l'analyse de textes. Cet ensemble a été créé à l'origine par **Steven Bird** et **Edward Loper** en relation avec des cours de linguistique informatique à l'Université de Pennsylvanie en 2001 [20]. Il intègre la plupart des tâches telles que la création de jetons, la racinisation, la lemmatisation, la détection de la ponctuation, le nombre de caractères et le nombre de mots et bien d'autres tâches. Il est très élégant et facile à utiliser.

Il est écrit en langage python. C'est une plate-forme de premier plan pour la création de programmes Python pour travailler sur des données textuelles. Il fournit des interfaces faciles à utiliser pour plus de 50 corpus et ressources lexicales telles que WordNet[13], ainsi qu'une suite de bibliothèques de traitement de textes pour la classification, la tokenisation, la racinisation, le marquage, l'analyse et le raisonnement sémantique, des wrappers[14] pour des bibliothèques de TAL de niveau industriel et un forum de discussion actif [21].

I.5.1.1 Le corpus NLTK

Un corpus est une collection de données linguistiques, parfois une compilation de textes écrits, ou de transcriptions d'enregistrements de discours. La raison principale d'un corpus est de vérifier une hypothèse sur le langage par exemple : déterminer comment l'utilisation d'un son particulier, d'un mot ou d'une construction syntaxique varie. Les corpus linguistiques agissant avec les lois et les pratiques d'utilisation de corpus dans l'étude du langage. Un corpus informatique contient un ensemble vaste de textes traduits en langage machine [20].

Le corpus NLTK contient l'ensemble des données et paquages de la bibliothèque NLTK. Parmi ces ensembles de données on a : les **StopWords, Gutenberg, Framenet_v15,** etc.

Une fois le corpus téléchargé, on peut réaliser certaines tâches telles que :

I.5.1.2 La suppression des mots vides (stops word)

Parfois, nous avons besoin de supprimer certains éléments inutiles afin que les données soient davantage traduisibles pour l'ordinateur. En TAL, de telles données sont qualifiées de mots

[7] https://nlp.stanford.edu/
[8] https://radimrehurek.com/gensim/
[9] https://uima.apache.org/
[10] https://opennlp.apache.org/docs/
[11] https://gate.ac.uk/
[12] https://mimno.github.io/Mallet/
[13] Une grande base de données lexicale de l'anglais. Les noms, verbes, adjectifs et adverbes sont regroupés en ensembles de synonymes cognitifs (synsets),
[14] Dans le domaine informatique, le terme désigne un programme ou code qui « enveloppe », ou englobe, d'autres composants logiciels.

vides (stop word en anglais). Par conséquent, ces mots n'ont aucune signification pour nous et nous souhaiterions les retirer.

La capture ci-après (Capture I-1) donne le script python et le résultat du script correspondant à l'ensemble des mots vides en langage français.

```
Entrée [2]: from nltk.corpus import stopwords
            print(set(stopwords.words('French')))

{'me', 'eusse', 'eu', 'aient', 'je', 'étée', 'auront', 'étais', 'seriez', 'été', 'sera', 'tu', 'nos', 'aie', 'ses', 'que', 'mo
n', 'aura', 'avaient', 'notre', 'étés', 'le', 'serions', 'l', 'seraient', 'avais', 'eussiez', 'étante', 't', 'pour', 'fusses',
'ayante', 'avions', 'te', 'eûtes', 'étantes', 'serai', 'eue', 'soyez', 'avons', 'ne', 'toi', 'vos', 'et', 'c', 'd', 'de', 'as',
'leur', 'serez', 'aies', 'auras', 'fusse', 'soient', 'auraient', 's', 'ayez', 'ai', 'la', 'était', 'auriez', 'ayants', 'êtes',
'au', 'mes', 'sa', 'étées', 'seront', 'fûtes', 'lui', 'fût', 'ou', 'aurions', 'est', 'vous', 'elle', 'es', 'étant', 'eussions',
'une', 'fut', 'ils', 'ayant', 'les', 'eûmes', 'ont', 'son', 'votre', 'm', 'aurait', 'pas', 'se', 'moi', 'ton', 'serais', 'furen
t', 'fûmes', 'fussions', 'eurent', 'étiez', 'ce', 'étaient', 'aviez', 'y', 'soyons', 'eût', 'aux', 'aurais', 'j', 'suis', 'ave
c', 'étions', 'fus', 'un', 'ma', 'eusses', 'eus', 'par', 'sont', 'qu', 'sur', 'nous', 'eut', 'ayons', 'du', 'ayantes', 'mais',
'ait', 'tes', 'ta', 'eux', 'serait', 'fussent', 'n', 'étants', 'eues', 'on', 'fussiez', 'aurez', 'qui', 'en', 'même', 'seras',
'soit', 'avez', 'sois', 'aurons', 'des', 'dans', 'eussent', 'sommes', 'avait', 'aurai', 'à', 'serons', 'il', 'ces'}
```

Capture I-1 : Les mots vides en français

Ainsi, nous avons listé une collection non ordonnée des mots vides en langue française. Pour rappel, le but principal est de pouvoir supprimer ces mots de notre texte. Pour se faire, on réalise ce qu'on appelle la tokenisation.

I.5.1.3 La tokenisation (décomposition de phrases en mots)

La tokenisation telle que définie dans Wikipédia est : un processus consistant à briser un flux de textes en plusieurs phrases, symboles ou tout autre élément significatif dit signes (tokens) en anglais. C'est une tâche courante dans le TAL. C'est une étape fondamentale à la fois dans les méthodes traditionnelles de TAL et dans les architectures avancées basées sur l'apprentissage en profondeur comme les transformers[15]. Les jetons sont les éléments constitutifs du langage naturel. C'est aussi un moyen de séparer un morceau de texte en unités plus petites appelées jetons. Les jetons peuvent être des mots, des caractères ou des sous-mots. Par conséquent, la tokenisation peut être largement classée en 3 types : tokenisation de mots, de caractères et de sous-mots (caractères de n-gramme) [22].

[15] Un transformateur est un modèle d'apprentissage en profondeur qui adopte le mécanisme de l'auto-attention , pondérant différemment l'importance de chaque partie des données d'entrée. Il est principalement utilisé dans les domaines du traitement du langage naturel (TAL) et de la vision par ordinateur (CV).

Dans la suite, nous allons nous limiter à la tokenisation des mots avant de passer au calcul de similarité.

Le script suivant (Capture I-2) permet d'enlever les mots vides de nos phrases après tokenisation.

```
Entrée [13]: from nltk.corpus import stopwords
             from nltk.tokenize import RegexpTokenizer

             # Avoir l'ensembles des mots vides en langage francais

             mots_vides_francais = set(stopwords.words('French'))
             chaine_1 = "En l'absence d'amélioration comme en cas de persistance des symptômes, prendre un avis médical."
             chaine_2 ="En outre, la patiente sera avertie de la nécessité d'une consultation rapide devant tout saignement vaginal anormal."
             chaine_3 = "Almaty compte de nombreuses salles de spectacle et musées dont les suivants."
             # tokenisation des chaines 1n 2 et 3
             tokenizer = RegexpTokenizer(r'\w+')

             chaine_1_tokeniser = tokenizer.tokenize(chaine_1)
             chaine_2_tokeniser = tokenizer.tokenize(chaine_2)
             chaine_3_tokeniser = tokenizer.tokenize(chaine_3)

             '''suppression des mots vides de ma chaine et creation
                 d'une liste pour stocker le reste de notre chaine'''

             chaine_normale_1 = []
             chaine_normale_2 = []
             chaine_normale_3 = []

             for mot_1, mot_2, mot_3 in zip(chaine_1_tokeniser,chaine_2_tokeniser,chaine_3_tokeniser):
                 if mot_1 not in mots_vides_francais:
                     chaine_normale_1.append(mot_1)
                 if mot_2 not in mots_vides_francais:
                     chaine_normale_2.append(mot_2)
                 if mot_3 not in mots_vides_francais:
                     chaine_normale_3.append(mot_3)
             print(chaine_normale_1)
             print(chaine_normale_2)
             print(chaine_normale_3)
```

Capture I-2 : Script de tokenisation et suppression des mots vides

La capture I-3 donne le résultat après suppression des mots d'arrêts.

```
['En', 'absence', 'amélioration', 'comme', 'cas', 'persistance', 'symptômes']
['En', 'outre', 'patiente', 'avertie', 'nécessité', 'consultation']
['Almaty', 'compte', 'nombreuses', 'salles', 'spectacle', 'musées', 'dont', 'suivants']
```

Capture I-3 : Résultats de la tokenisation et suppression des mots vides

I.5.1.4 La lemmatisation

Le processus de « lemmatisation » consiste à représenter les mots sous leur forme canonique (lemmes). Par exemple pour un verbe, ce sera son infinitif. Pour un nom, son masculin singulier. L'idée étant encore une fois de ne conserver que le sens des mots utilisés dans le corpus [23]. Le résultat obtenu après une lemmatisation est appelé « lemme » qui est un mot valide et qui signifie la même chose.

I.5.1.5 La racinisation

La racinisation (Stemming en anglais) est une technique destinée à retourner la racine ou le radical d'un mot. C'est un processus un peu similaire à la lemmatisation. Elle consiste à ne conserver que la racine des mots étudiés. L'idée étant de supprimer les suffixes, préfixes et autres des mots afin de ne conserver que leur origine. C'est un procédé plus simple que la lemmatisation et plus rapide à effectuer puisqu'on tronque les mots essentiellement contrairement à la lemmatisation qui nécessite d'utiliser un dictionnaire [23].

Le but du stemming est de regrouper de nombreuses variantes d'un mot comme un seul et même mot. Par exemple, une fois que l'on applique un stemming sur « Chiens » ou « Chien », le mot résultant est le même. Cela permet notamment de réduire la taille du vocabulaire dans les approches de type sac de mots ou tf-idf (term frequency–inverse document frequency).[16]

La capture suivante (Capture I-4) donne le résultat de la racinisation des mots « chiens » et « chien »

```
Entrée [6]: from nltk.stem  import PorterStemmer
racine = PorterStemmer()
print(racine.stem('chien'))
print(racine.stem('chiens'))
print(racine.stem('grand'))
print(racine.stem('grandes'))

chien
chien
grand
grand
```

Capture I-4 : Racisation de mots

I.5.2 La bibliothèque Spacy

Spacy est l'une des principales bibliothèques du langage python pour le traitement automatique des langues. Elle est gratuite et open source publiée sous la licence MIT pour le traitement naturel du langage. Elle est écrite en Cython[17], et conçue pour l'usage en production grâce à une API concise et simple d'utilisation.

Avec la bibliothèque Spacy, il est possible de créer des applications permettant de traiter et de comprendre de larges volumes de textes**.** Il peut être utilisé notamment pour développer des systèmes d'extraction d'informations, de compréhension du langage naturel ou encore pour prétraiter des textes pour le Deep Learning [24].

[16] Abréviation du terme fréquence–fréquence de document inverse , est une statistique numérique destinée à refléter l'importance d'un mot pour un document dans une collection ou un corpus.
[17] Un langage sur-ensemble du langage Python qui prend également en charge l'appel de fonctions C et la déclaration de types C sur les variables et les attributs de classe.

I.5.2.1 Les outils et fonctionnement de Spacy

Comme la bibliothèque NLTK, Spacy peut être utilisé pour une large variété de tâches liées à des projets de TAL. Il permet par exemple la tokenization, la lemmatisation, le tagging POS, la reconnaissance de phrase ou d'entité, l'analyse des dépendances, la transformation mot/vecteur et d'autres techniques de normalisation et de nettoyage [24]. Certaines tâches telles que la tokenization et la lemmatisation ont été décrites dans la section précédente.

- Le tagging part-of-speech (POS)

Le **Tagging part-of-speech (POS)** aussi réalisable avec la NLTK bibliothèque est un procédé visant à assigner des propriétés grammaticales telles que des noms, des verbes, des adverbes ou des adjectifs à des mots. Les mots partageant les mêmes étiquettes POS suivent généralement la même structure syntaxique et sont utiles pour les processus basés sur des règles [24].

Certains mots peuvent fonctionner de plusieurs manières lorsqu'ils sont utilisés dans différentes circonstances. Le POS Tagging joue ici un rôle crucial pour comprendre dans quel contexte le mot est utilisé dans la phrase. Le balisage POS est utile pour l'analyse de phrases, l'extraction d'informations, l'analyse de sentiments, etc. [25]

- La reconnaissance d'entités nommées

La reconnaissance d'entité est un processus visant à classifier les entités nommées dans un texte dans différentes catégories prédéfinies. Il peut s'agir par exemple de personnes, de lieux, ou encore des dates. Le modèle statistique de spaCy permet de classifier une large variété d'entités notamment des personnes, des entités, des œuvres d'art ou encore des nationalités [24].

- Analyse de la dépendance

C'est une méthode permettant de piloter l'analyse de dépendance d'une phrase. Ceci permet de révéler son format grammatical. Cette technique met en lumière les relations entre les mots principaux et leurs dépendances.

- La représentation mot-vecteur

Elle aide les machines à comprendre et à interpréter les liens entre les mots à la manière d'un humain. La représentation numérique d'un mot met en lumière ses relations avec les autres mots [24].

I.5.3 Comparaison entre Spacy et NLTK

NLTK et Spacy sont les deux bibliothèques python les plus populaires pour faire le traitement automatique des langues (TAL). Ces deux outils présentent toutefois des différences [24]. Tout d'abord, Spacy regroupe divers algorithmes adaptés à différents problèmes dans sa boite

à outils. Ces algorithmes sont gérés et rénovés par la bibliothèque. De son côté, NLTK permet de choisir parmi de nombreux algorithmes en fonction du problème à résoudre. Une autre différence majeure est que Spacy utilise des modèles statistiques pour sept langues : le français, l'anglais, l'allemand, l'espagnol, l'italien, le portugais et le néerlandais tandis que NLTK prend en charge beaucoup plus de langues.

Lors d'une analyse de textes, comme l'analyse de sentiments, Spacy déploie une stratégie orientée objet**.** Les mots et les phrases sont considérés comme des objets. À l'inverse, NLTK est une bibliothèque de traitement de lignes. Elle reçoit les inputs et retourne des outputs sous forme de lignes de code.
Enfin, chacune de ces deux bibliothèques a sa propre spécialité. Pour la tokenization et le tagging POS, Spacy offre de meilleurs résultats et propose les algorithmes les plus récents et les plus performants. En revanche, NLTK est supérieur pour la tokenization de phrase [24]**.**

I.6 Conclusion

Dans ce chapitre, nous avons essayé de faire focus sur le terme TAL et sur quelques concepts fondamentaux sur lesquels repose son fonctionnement.

En résumé, les concepts clés lies au traitement automatique des langues permettent de réaliser certaines tâches importantes et sont indispensables au traitement de documents textuels.

Ces tâches sont le prétraitement, la vectorisation de texte et la classification qui est une tâche optionnelle. La réalisation de ces tâches nécessite l'utilisation de technologies les plus adaptées vu la complexité d'une tâche de traitement automatique des langues qui à son tour nécessite beaucoup de ressources pour son aboutissement. Ainsi, le langage python fournit beaucoup d'outils très appropries qu'on peut utiliser. Parmi lesquels NLTK et Spacy qui contiennent la quasi-totalité des outils dont nous avons besoin. La réalisation de ces différentes tâches est une étape préalable pour le calcul de similarité textuelle.

Chapitre II : Etat de l'art sur les approches de calcul de similarité

II.1 Introduction

Mesurer la similarité entre textes est à la base d'un grand nombre d'applications du traitement automatiques des langues. Il joue un rôle important dans divers domaines tels que la recherche d'informations, la traduction automatique, l'extraction d'informations, la détection de plagia, la catégorisation de textes, le résume de textes, la désambiguïsation lexicale, etc. [26]. Le terme texte renvoie à toutes ressources qui peut être représenter sous forme textuelle que ce soit des documents, des phrases, des mots ou de simples caractères dans tous les cas, le calcul de la similarité peut s'appliquer pour quelques raisons que ce soit.

Le calcul de similarité entre paires de phrases représente une tâche primordiale dans plusieurs applications du traitement automatique des langues et consiste à évaluer jusqu'à quel point ces phrases sont proches. Ainsi, il ressort de cette tâche la capacité de pouvoir identifier si deux phrases quelconques sont similaires ou pas. Si les phrases sont similaires alors elles contiennent des informations identiques ou très proches sémantiquement et offrent des indications importantes sur le fonctionnement de la langue [27].

En effet, ils existent plusieurs approches de calcul de similarité textuelle. Certaines approches couramment utilisées exploitent la structure syntaxique des phrases, le nombre de jetons ou n-grams en commun entre la phrase source et la phrase cible est généralement calculé. D'autres tentent de prendre en compte les problèmes de synonymie et la sémantique des phrases en exploitant des ressources sémantiques ou des méthodes statistiques [28].

Ainsi, pour mieux appréhender la notion de similarité, on se propose dans un premier temps d'identifier et de définir les concepts clés qui lui sont liés. Ensuite dans une deuxième et troisième étape, on essayera d'élaborer quelques approches de calcul de similarité avec en amont une brève description des fonctionnalités (mesures) qui sont utilisées pour chacune d'elles. Pour finir, on s'intéressera à la présentation de quelques algorithmes de machine Learning avant de réaliser une brève synthèse des différentes approches qui seront présentées.

II.2 Définition des concepts clés

L'indentification et la définition des mots clés nous paraissent une étape primordiale pour mieux comprendre notre sujet.

- Similarité sémantique

Wikipédia définit la similarité sémantique comme étant une notion définie entre deux concepts soit au sein d'une même hiérarchie conceptuelle, soit dans le cas d'alignement d'ontologies entre deux concepts appartenant respectivement à deux hiérarchies conceptuelles distinctes [29].

C'est une tâche dans le domaine du traitement automatique des langues qui évalue la relation entre des textes ou des documents à l'aide d'une métrique définie. Elle a diverses applications dans le TAL telles que la recherche d'informations, la synthèse de textes, l'analyse de sentiments etc. [30]. Elle indique que ces deux concepts possèdent un grand nombre d'éléments en communs (propriétés, termes, instances).

- Phrases parallèles

La seule et unique vocation lors qu'on réalise l'étude de la similarité entre phrases est de voir si ces dernières sont parallèles ou pas. Dans le contexte de notre travail on considère deux phrases parallèles comme étant similaires sémantiquement. Elles sont utilisées dans de nombreuses application du traitement automatique des langues en particulier pour l'extraction terminologique automatique, le traduction automatique statistique etc. [31].

- Degré de similarité

En mathématique et en informatique théorique, une mesure de similarité, plus exactement une mesure de distance entre mots, est une façon de représenter par un nombre la différence entre deux mots, ou plus généralement deux chaines de caractères. Cela permet de comparer des mots ou chaines de façon simple et pratique. C'est donc une forme de distance mathématique et de métrique pour les chaînes de caractères.

Le degré de similarité représente l'élément central sur lequel on peut se baser pour émettre une conclusion sur la similarité ou dissimilaire entre deux concepts ou phrase. Lorsqu'on mesure la similarité entre deux phrases, on se met dans un contexte d'exploration de données ou d'apprentissage automatique et les phrases sont représentées par des distances avec des dimensions représentants les caractéristiques de ces dernières. Si la distance est petite, les caractéristiques présentent un degré élevé de similitude traduisant une similarité entre les phrases. Alors qu'une grande distance sera un faible degré de similitude qui correspond bien à une dissimilarité [32].

II.3 Quelques approches de calcul de similarité textuelle

Le calcul de similarité a connu un engouement dans plusieurs applications du TAL. Pour sa réalisation, beaucoup d'approches ont été proposés. Ces dernières se focalisent toutes sur le calcul de similarité mais différent par la manière dont ils conçoivent les phrases pour réaliser ce calcul.

Ainsi, nous allons essayer dans cette partie de faire une brève présentation de quelques-unes d'entre elles avec une description des fonctionnalités utilisées. La figure (Figure II-1) ci-après donne quelques approches de calcul qu'on présentera dans la suite de cette partie.

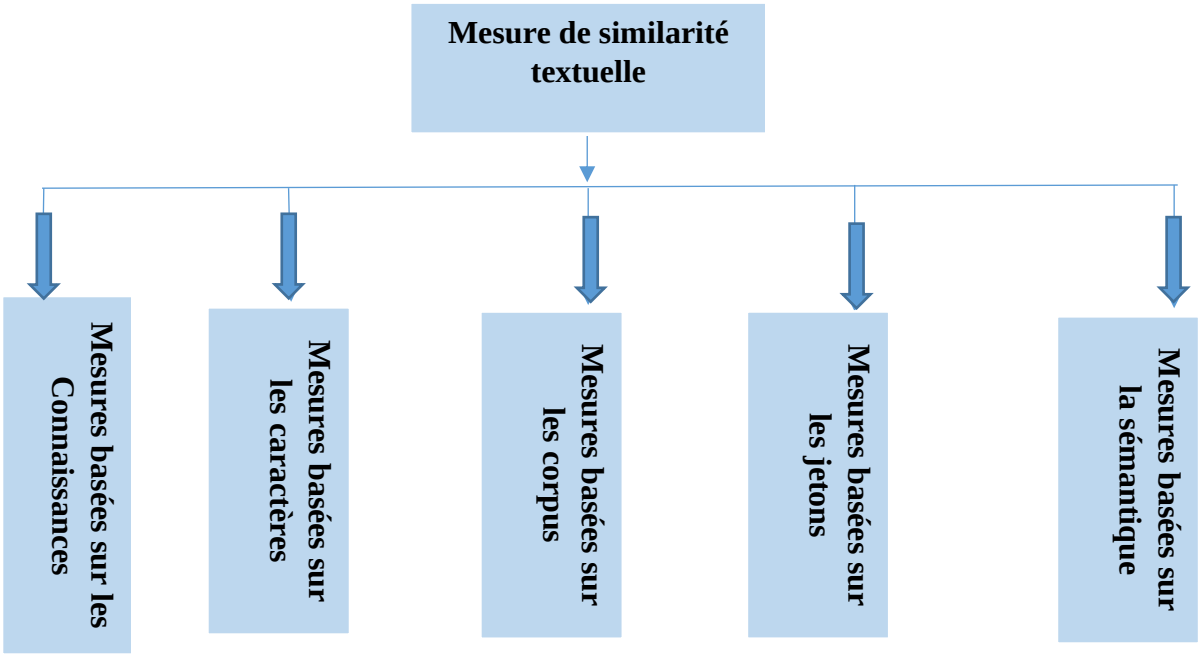

Figure II-1 : Quelques approches de calculs de similarité textuelle

II.3.1 Les mesures basées sur les jetons

Les algorithmes entrant dans cette catégorie sont, plus ou moins des algorithmes de similarité d'ensemble où les ensembles sont constitués de jetons (tokens) de chaînes [33]. Pour mesurer la similarité entre deux phrases, ils les considèrent comme des listes non ordonnées de jetons. La similitude entre une paire de phrases est mesurée efficacement par la similitude entre les listes de jetons correspondantes. Il existe beaucoup de mesures basées sur des jetons : la mesure de **Jaccard**, la similarité de **Jaccard généralisée**, similaire à la similarité Jaccard qui considère que deux jetons sont identiques si leur similitude est supérieure à un seuil prédéfini. Par ailleurs on a les mesures de similarité de **Jaro** qui trouvent effectivement un texte court similaire et fixé empiriquement le seuil à 0,6, la similarité de **Dice**, la similarité **d'Ochiai**, le coefficient de chevauchement (**overlap**) et la similarité TF-IDF, l'une des mesures les plus utilisées dans la recherche d'informations.

❖ La similarité de Jaccard

L'indice de Jaccard, également connu sous le nom de coefficient de Jaccard (à l'origine du nom Coefficient de Communauté par Paul Jaccard), est un indice statistique utilisé pour comparer la similarité et la diversité des ensembles d'échantillons.

En d'autres termes la mesure de similarité de **Jaccard** se fait en utilisant le nombre d'éléments partagés divisé par le nombre d'éléments distincts au total. Soient X et Y deux ensembles de jetons, l'indice de Jaccard entre X et Y est donnée par l'équation 1 ci-après **[**35] :

$$\text{Jaccard}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

Équation 1 : Mesure de similarité de Jaccard

La distance de **Jaccard** est la mesure dans laquelle la dissimilitude entre des ensembles d'échantillons est complémentaire du coefficient de Jaccard et est obtenue en soustrayant 1 du coefficient de Jaccard ou de manière équivalente en divisant la différence de la taille de l'union et de l'intersection des deux ensembles pour la dimension de la caisse.

❖ La similarité de Dice

Le coefficient **Sørensen-Dice** est une statistique utilisée pour évaluer la similitude de deux échantillons. Il a été développé indépendamment par les botanistes **Thorvald Sørensen** et **Lee Raymond Dice**, qui ont publié respectivement en **1948** et en **1945** [34].

La formule originale de Sørensen était destinée à être appliquée à des données discrètes. Étant donné deux ensembles, X et Y, il est défini comme :

$$\text{Dice}(X, Y) = \frac{2 * |X \cap Y|}{|X| + |Y|}$$

Équation 2 : Mesure de similarité de Dice

Où |X| et |Y| sont les cardinalités des deux ensembles (c'est-à-dire le nombre d'éléments dans chaque ensemble).

L'indice de Sørensen est égal au double du nombre d'éléments communs aux deux ensembles divisés par la somme du nombre d'éléments de chaque ensemble.

Lorsqu'il est appliqué aux données booléennes, en utilisant la définition de vrai positif (TP), faux positif (FP) et faux négatif (FN), il peut être écrit comme [34]**:**

$$DSC = \frac{2TP}{2TP+FP+FN}$$

Équation 3 : Mesure de similarité de Dice booléen

Il est différent de l'indice de Jaccard qui ne compte les vrais positifs qu'une seule fois au numérateur et au dénominateur. DSC est le quotient de similarité et varie entre 0 et 1. Il peut être considéré comme une mesure de similarité sur des ensembles [34].

❖ La similarité de Ochiai

En biologie, il existe un concept connu sous le nom de coefficient **Otsuka-Ochiai** nommé d'après **Yanosuke Otsuka** (également orthographié comme **Ōtsuka, Ootsuka** ou **Okuta**) et **Akira Ochiai** également connu sous le nom de coefficient d'**Ochiai-Barkman** ou **d'Ochiai,** qui peut être représenté par **[35]** :

$$\text{Ochiai}(X, Y) = \frac{|X \cap Y|}{\sqrt{|X| * |Y|}}$$

Équation 4 : Mesure de similarité de Ochiai

Ici X et Y sont des ensembles et |X| est le cardinal de X. Si les ensembles sont représentés sous forme de vecteurs binaires, le coefficient de Ochiai peut être considéré comme le même que la similarité cosinus que nous verrons plus tard [35].

❖ La similarité Overlap

Les mesures de similarité de chevauchement sont définies comme la taille de l'intersection des deux ensembles divisée par la taille du plus petit des deux ensembles [36].

Soient X et Y deux ensembles d'échantillons, le coefficient de chevauchement entre X et Y est donnée par l'équation 5 :

$$\text{Overlap}(X, Y) = \frac{|X \cap Y|}{\min(|X|, |Y|)}$$

Équation 5 : Mesure de similarité Overlap

L'algorithme de similarité de chevauchement est important pour déterminer quelles choses sont sous-ensembles d'autres.

L'ensemble des mesures de similarité entre textes basées sur les jetons comporte essentiellement trois étapes [33] que sont :

- Identification des jetons : examiner les chaines de textes à comparer et définir un ensemble de jetons. C'est-à-dire un ensemble des chaines de caractères.
- Décompte : compter le nombre de jetons dans chacune des chaines à comparer.
- Mesure : utiliser ces nombres pour calculer une mesure de distance ou de similarité.

❖ La similarité Cosinus

Dans ce type de mesures, la similarité **cosinus** est l'une des plus couramment utilisées. Son principe est de collecter des ensembles ordonnés de tous les mots des deux phrases (tweets, documents, ...), A et B, c'est-à-dire en fait deux « **vecteurs de** phrases » $\vec{a}$ et $\vec{b}$ et mesurer l'angle entre les deux vecteurs [37].

$$\alpha_{A,B} = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|} = \frac{\sum a_i b_i}{\sqrt{\sum {a_i}^2}_i \sqrt{\sum {b_i}^2}}$$

Équation 6 : Mesure de similarité cosinus

II.3.2 Les mesures basées sur les caractères

Les mesures basées sur les caractères sont également appelées mesures basées sur les séquences [38] : distance de Levenshtein [39] et Q-gram. Les mesures basées sur les jetons se concentrent sur la similitude au niveau des mots ou tokens. On les utilise également au niveau des caractères, en particulier la mesure **Q-gram**. Chaque phrase est transformée en une liste de sous-chaînes de longueur Q (Q-grammes) en faisant glisser une fenêtre de caractères Q sur la phrase. Ces approches, pour calculer la similarité entre deux phrases prennent les mesures à base de jetons sur les Q-gram.

Les mesures basées sur des jetons ignorent l'ordre des jetons. Celles basées sur la distance d'édition permettent de remédier à cette limitation. Elles se concentrent sur la façon de transformer une phrase en une autre en utilisant trois types de modifications : les insertions, les suppressions et les substitutions. En fait, les mesures basées sur les séquences sont très efficaces en informatique clinique et biomédicale. Par exemple, ce sont les principales mesures utilisées sur la détection de la redondance dans les notes cliniques et les enregistrements en double dans les bases de données biologiques.

Quelques exemples de mesures utilisant cette approche sont :

➢ **La distance de Hamming**

La distance de Hamming doit son nom à Richard Hamming (1915 1998). Elle est décrite dans un article fondateur pour la théorie des codes. Elle est utilisée en télécommunication pour compter le nombre de bits altérés dans la transmission d'un message d'une longueur donnée. Le poids de Hamming correspond au nombre de bits différents de zéro, il est utilisé dans plusieurs

disciplines comme la théorie de l'information, la théorie des codes et la cryptographie. Néanmoins, pour comparer des séquences de longueurs variables, ou des chaines de caractères pouvant subir non seulement des substitutions, mais aussi des insertions ou des effacements, des métriques plus sophistiquées comme la distance de Levenshtein sont plus adaptées.

Soit A un alphabet et F l'ensemble des suites de longueur n à valeur dans A. La **distance de Hamming** entre deux éléments a et b de F est le cardinal de l'ensemble des images de a qui diffèrent de celle de b [39].

Formellement, si d(.,.) désigne la distance de Hamming :

$$\forall\, a, b \in F \;\; a = (a_i)_{i \in [0,n-1]} \; et \; b = (b_i)_{i \in [0,n-1]} \;\; d(a,b) = \#\{i : a_i \neq b_i\}$$

La notation #E désigne la cardinale de E.

Un cas important dans la pratique est celui des symboles binaires. Autrement dit *A*= {0,1}, On peut alors écrire, si $\oplus$ désigne le ou exclusif.

$$d(a,b) = \sum_{i=0}^{n-1} (a_i \oplus b_i)$$

Équation 7 : Distance de Hamming

- **La distance Levensthien**

La distance de Levenshtein est une distance, au sens mathématique du terme, donnant une mesure de la différence entre deux chaines de caractères. Elle est égale au nombre minimal de caractères qu'il faut supprimer, insérer ou remplacer pour passer d'une chaîne à l'autre.

Elle a été proposée par Vladimir Levenshtein en 1965. Elle est également connue sous les noms de **distance d'édition** ou de **déformation dynamique temporelle** notamment en reconnaissance des formes et particulièrement en reconnaissance vocale.

Cette distance est d'autant plus grande que le nombre de différences entre les deux chaînes est grand. La distance de Levenshtein peut être considérée comme une généralisation de la distance de Hamming [40].

Formellement cette distance est définie avec deux chaines a et b, |a| le cardinal de a (ou son nombre de lettre) et a-1 la chaine de a tronquée de sa 1ère lettre.

$$Lev(a,b) = \begin{cases} \max(|a|,|b|) & si \min(|a|,|b|) = 0 \\ lev(a-1,b-1) \;\; si\; a[0] = b[0] \\ 1 + min \begin{cases} lev(a-1,b) \\ lev(a,b-1) \;\; sinon \\ lev(a-1,b-1) \end{cases} \end{cases}$$

Équation 8 : Distance de Levenshtein

- **La similarité de Needleman-Wunsch** généralise la similitude de Lenvenshtein en effectuant un alignement de séquences global dynamique (par exemple, elle permet d'attribuer un coût différent pour différentes opérations).
- **La similarité de Smith Waterman** se concentre sur l'alignement de séquence dynamique au niveau des sous-chaînes au lieu de globalement.
- **La similarité Q-gram**

Un **Q-gramme** est une sous-séquence de Q éléments construite à partir d'une séquence donnée. L'idée provient des travaux de Claude Shannon en théorie de l'information. Son idée était qu' à partir d'une séquence de lettres donnée il est possible d'obtenir la fonction de vraisemblance de l'apparition de la lettre suivante. Cette mesure compare les Q-grammes entre les deux textes et renvoie un score indiquant le degré de similarité entre les deux textes.

Les approches décrites précédemment mesurent la similarité entre des phrases en termes de structure syntaxique. Cependant, en langage naturel, des mots distincts peuvent avoir des significations proches. Les phrases contenant des mots ou des structures différents peuvent représenter une sémantique similaire. Par exemple, les phrases "chaque parent devrait porter une copie non fonctionnelle du gène CF afin d'être à risque d'avoir un enfant atteint de mucoviscidose " et " un individu doit avoir une mutation dans les deux copies du gène CFTR (une de la mère et un du père) à être affecté" contiennent des mots distincts mais sont sémantiquement similaires. L'utilisation de mesures basées sur des jetons et des séquences restent limitée dans ce cas.

II.3.3 Les mesures basées sur les corpus

Cette approche de calcul de similarité détermine la similarité entre deux concepts à partir des informations extraites d'un large corpus. En d'autres termes la similarité basée sur le corpus détermine la similarité sémantique entre les mots en fonction des informations obtenues à partir d'un grand corpus [41].

Il existe beaucoup de mesures de similarité basées sur des corpus. La figure ci-dessous (Figure II-2) donne les mesures basées sur des corpus parmi lesquelles on peut citer quelques-unes en guise d'exemples :

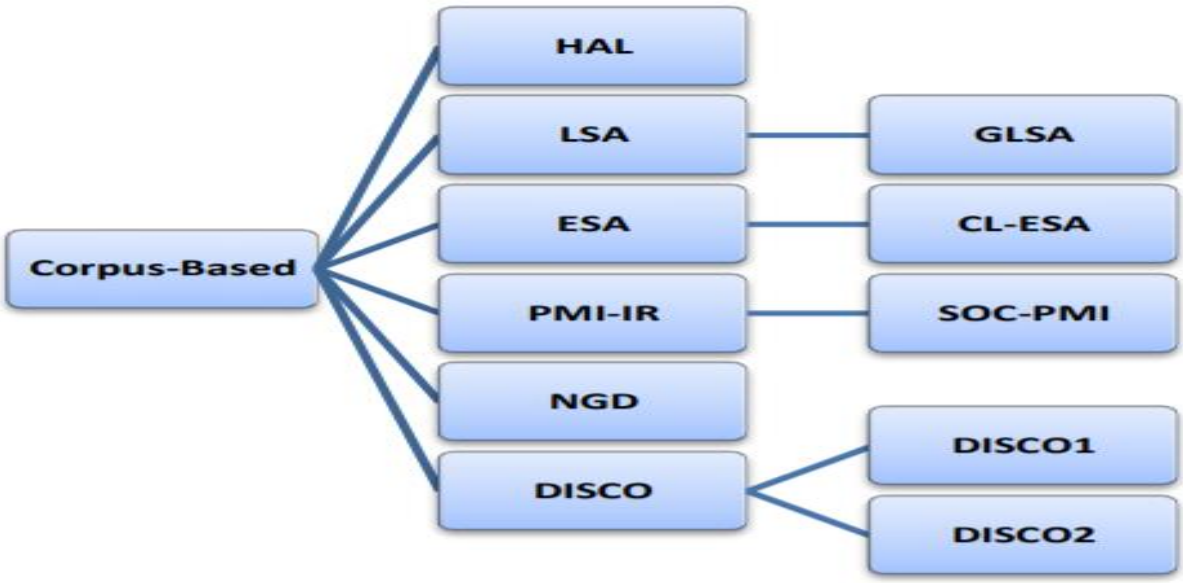

Figure II-2 : Mesures basées sur des corpus [41]

❖ Hyperspace Analogue to Language (HAL)

L'Hyperspace Analogue to Language (HAL) est un modèle d'espace sémantique validé et motivé sur le plan cognitif qui capture les dépendances statistiques entre les mots en considérant leurs cooccurrences dans une fenêtre de texte environnante. HAL a été appliqué avec succès à l'expansion de requêtes dans la recherche d'informations, mais présente plusieurs limitations, notamment un coût de traitement élevé et l'utilisation de statistiques de distribution qui n'exploitent pas la syntaxe [42]**.**

❖ Analyse sémantique latente (LSA)

L'analyse sémantique latente (ASL ou LSA) n'est pas un concept spécifique au web. Elle est en revanche à la base des techniques actuelles d'indexation et d'évaluation des contenus par les moteurs de recherche : la comprendre permet notamment d'optimiser son site web pour le référencement naturel.

Elle a pour but, à partir d'un ensemble de documents, par exemple des pages web, d'établir automatiquement des relations entre les termes contenus dans ces documents, les documents eux-mêmes et des « concepts » associés aux termes. Elle est notamment utilisée pour :

- établir des similitudes entre des termes (recherche des synonymes) ;
- associer des documents à des « concepts » à partir de l'analyse de leurs termes et donc établir une éventuelle proximité sémantique entre eux ;
- associer un concept à une requête de recherche en analysant ses termes.

L'analyse sémantique latente se base sur une matrice mathématique à deux dimensions. Dit plus simplement, elle utilise un tableau avec des lignes contenant les termes utilisés dans les différents documents (une colonne par document). Les cellules du tableau contiennent les occurrences des différents termes dans chaque document [43]. La figure (Figure II-3) ci-après donne un exemple de matrice utilisée en LSA.

	Rapide	Brun	Renard	Saute	Par-dessus	Paresseux	Chien
Le renard brun rapide saute par-dessus le chien paresseux	1	1	1	1	1	1	1
Si le renard est rapide, il peut sauter par-dessus le chien.	1	0	1	0	1	0	1
Les renards sont rapides. Les chiens sont paresseux.	0	1	1	0	0	1	1
Un renard peut-il sauter par-dessus un chien ?	0	0	1	1	1	0	1

Figure II-3 : Exemple de matrice en LSA [44]

- Analyse sémantique explicite (ESA pour explicit semantic analysis)

Dans le traitement du langage naturel et la recherche d'informations, l'analyse sémantique explicite (ESA) est une représentation vectorielle de textes (mots individuels ou documents entiers) qui utilise un corpus de documents comme base de connaissances. Plus précisément, dans le modèle ESA, un mot est représenté comme un vecteur dans la matrice tf-idf du corpus de textes et un document (chaîne de mots) est représenté comme le centroïde[18] des vecteurs représentant ses mots. En général, le corpus de textes utilisé est Wikipédia, bien que d'autres corpus peuvent être utilisés.

L'ESA a été conçue par **Evgeniy Gabrilovich** et **Shaul Markovitch** comme un moyen d'améliorer la catégorisation des textes et a été utilisée par des chercheurs pour calculer ce qu'ils appellent la « relation sémantique » au moyen du cosinus entre les vecteurs susmentionnés, collectivement interprété comme un espace de « concepts explicitement définis et décrits par des humains ». Le nom « analyse sémantique explicite » contraste avec l'analyse sémantique latente (LSA), car l'utilisation d'une base de connaissances permet d'attribuer des étiquettes lisibles par l'homme aux concepts qui composent l'espace vectoriel [45]**.**

- La distance Google normalisée (NGD)

[18] Intersection des différents vecteurs des mots

La distance Google normalisée est une mesure de similarité sémantique dérivée du nombre de résultats renvoyés par le moteur de recherche Google pour un ensemble donné de mots clés. Les mots-clés ayant des significations identiques ou similaires dans le sens d'un langage naturel ont tendance à être « proches » dans les unités de distance Google normalisée, tandis que les mots ayant des significations différentes ont tendance à être plus éloignés.

Plus précisément, la distance Google normalisée (NGD) entre deux termes de recherche *x* et *y* est :

$$\mathrm{NGD(x,y)} = \frac{max\{\log f(x), \log f(y)\} - \log f(x,y)}{\log N - \min\{\log f(x), \log f(y)\}}$$

Équation 9 : Distance de Google Normalisée

Où *N* est le nombre total de pages Web recherchées par Google multiplié par le nombre moyen de termes de recherche singleton apparaissant sur les pages, *f* (x) et *f* (*y*) sont respectivement le nombre de résultats pour les termes de x et y et f (x, y) est le nombre de pages web sur lesquelles x et y apparaissent à la fois [46].

- ❖ Latent Dirichlet Allocation (LDA)

L'allocation de Dirichlet latente est un modèle probabiliste génératif pour des collections de données discrètes telles que des corpus de textes. C'est un modèle bayésien hiérarchique à trois niveaux, dans lequel chaque élément d'une collection est modélisé comme un mélange fini sur un ensemble sous-jacent de sujets. Chaque sujet à son tour est modélisé comme un mélange infini sur un ensemble sous-jacent de probabilité de sujet [47].

La figure ci-après (figure II-4) est un exemple d'application de l'allocation de Dirichlet latente sur un ensemble de sujets.

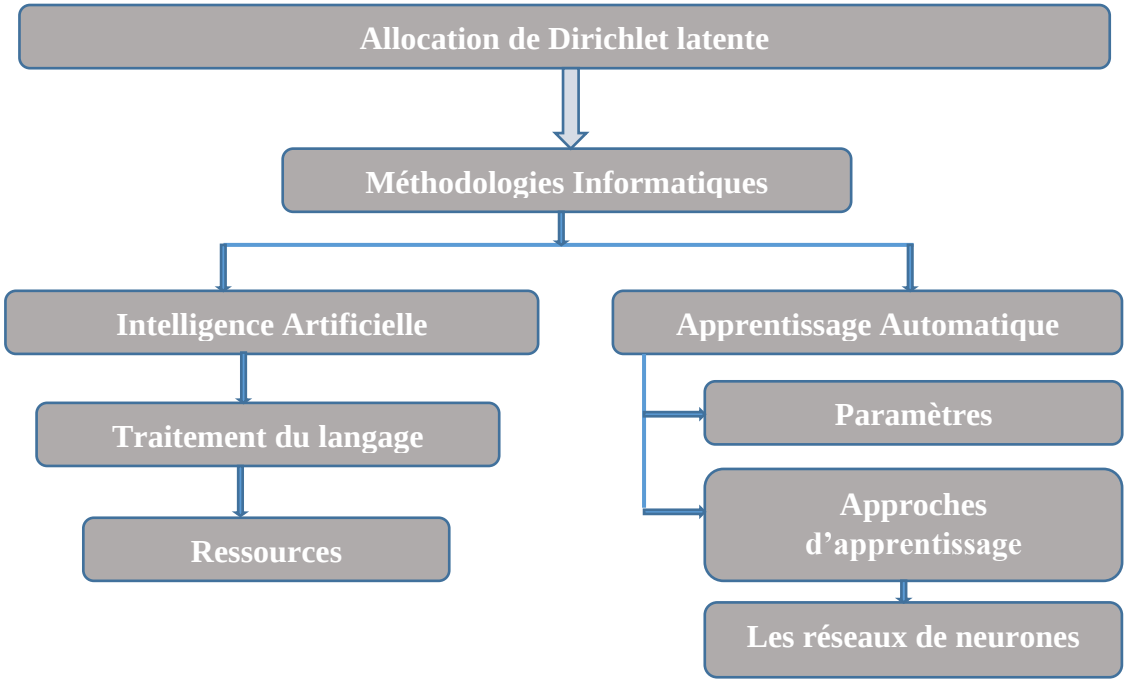

Figure II-4 : Exemple d'allocation de Dirichlet latente

II.3.4 Les mesures basées sur les connaissances

La similarité basée sur la connaissance mesure le degré de similarité entre les mots, termes ou concepts à l'aide d'informations dérivées de ressources sémantiques. **WordNet** est la ressource sémantique la plus couramment utilisée. Il s'agit d'une grande base de données lexicale de mots anglais étiquetés comme noms, verbes, adjectifs et adverbes. Les mots sont regroupés en ensembles de synonymes (synsets), chacun exprimant un concept distinct. La figure ci-après (figure II-5) montre les mesures utilisées avec l'approche basée sur les connaissances pour la mesure de similarité sémantique.

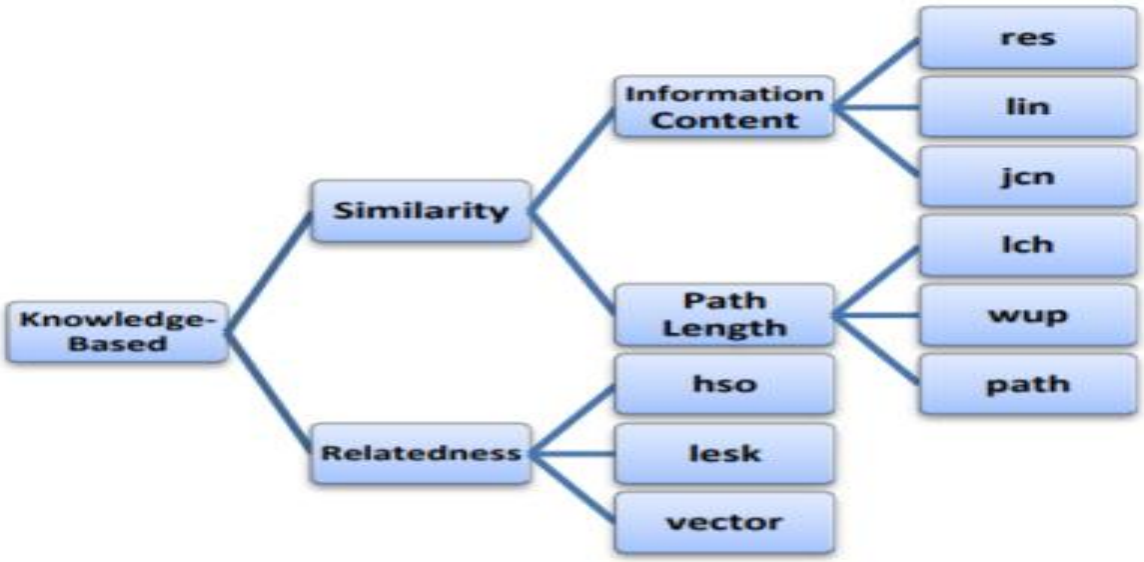

Figure II-5 : Mesures basées sur la connaissance [41]

- **La mesure de similarité de Resnik :**

Resnik définit la similarité sémantique entre deux concepts par la quantité d'information qu'ils partagent. Cette information partagée est égale au contenu informationnel du plus petit généralisant (PPG) c'est à dire le concept le plus spécifique qui subsume les deux concepts dans l'ontologie**.**

$$\mathrm{Sim}(c1, c2) = \mathrm{CI}\ (\mathrm{ppg}(c1, c2))$$

Équation 10 : Mesure de similarité de Resnik

La notion de contenu informationnel (CI) a été pour la première fois introduite par **Resnik**. Elle utilise conjointement l'ontologie et le corpus et traduit la pertinence d'un concept dans le corpus en tenant compte de sa spécificité ou généralité. On dit qu'un concept général subsume un concept plus spécifique. La fréquence de concepts dans le corpus est calculée pour retrouver le contenu informationnel. Cette fréquence regroupe la fréquence d'apparition du concept lui-même ainsi que des concepts qu'il subsume [48]. Le contenu informationnel d'un concept c est donné par :.

$$\mathrm{CI}(c) = -\log\ (\mathrm{P}(c))$$

Équation 11 : Contenu informationnel [48]

Avec P la probabilité de trouver une instance du concept c.

- **La similarité Jiang-Conrath** :

La mesure de **Jiang-Conrath** pallie aux limites de la mesure de **Resnik** en combinant le contenu informationnel du PPG à ceux des concepts. Elle prend en considération aussi le nombre d'arcs. Ainsi une distance est définie :

$$\mathrm{distance}(c1, c2) = \mathrm{C\,I}(c1) + \mathrm{CI}(c2) - (2\mathrm{CI}(\mathrm{ppg}(c1, c2)))$$

Équation 12 : Distance entre deux concepts

La mesure de similarité devient donc :

$$\mathrm{Sim}(c1, c2) = {}^{1}/_{\mathrm{distance}\ (c1, c2)}$$

Équation 13 : Mesure de similarité de Jiang-Conrath [48]

- **La similarité Lin :**

Lin a défini une mesure de similarité légèrement différente de celle de Resnik. Elle utilise une approche hybride qui combine deux sources de connaissances différentes (Thesaurus, corpus) [49].

II.3.5 Les approches de Word Embedding

Le **Word embedding** (plongement de mots ou plongement lexical en français) désigne un ensemble de techniques de machine learning qui vise à représenter les mots ou les phrases d'un texte par des vecteurs de nombres réels, décrits dans un modèle vectoriel.

Ces nouvelles représentations de données textuelles ont permis d'améliorer les performances des méthodes de traitement automatique des langues. Il repose sur la théorie linguistique fondée par **Zelling Harris** et connue sous le nom de sémantique distributionnelle. Cette théorie considère qu'un mot est caractérisé par son contexte, c'est à dire par les mots qui l'entourent. Ainsi, des mots qui partagent des contextes similaires partagent également des significations similaires. Les algorithmes de word embedding sont le plus souvent employés pour décrire des mots à travers de vecteurs numériques, mais ils peuvent également être utilisés pour construire des représentations vectorielles de phrases entières, de données biologiques comme les séquence d'ADN, ou encore des réseaux représentés comme des graphes (analyse de sentiment) [50].

Il existe plusieurs approches de word embedding. Les premières remontent aux années **1960** et reposent sur des méthodes de réduction de dimensionnalité. Plus récemment, de nouvelles techniques basées sur des modèles probabilistes et des réseaux de neurones, comme Word2Vec, ont permis d'obtenir de meilleures performances (analyse de sentiment) [50].

❖ Les méthodes de réduction de dimensionnalité (RD)

La RD désigne toute méthode permettant de projeter des données issues d'un espace de grande dimension dans un espace de plus petite dimension. Cette opération est cruciale en apprentissage automatique pour lutter contre ce qu'on appelle le fléau des grandes dimensions (le fait que les grandes dimensions altèrent l'efficacité des méthodes) [51].

Le mot dimension est utilisé au sens algébrique c'est-à-dire la dimension de l'espace vectoriel sous-jacent aux valeurs des vecteurs de descripteurs. La réduction de dimensionnalité permet de réduire la complexité d'un problème d'apprentissage automatique à plusieurs niveaux :

- ✓ D'un point de vue théorique, cela entraîne automatiquement une amélioration des propriétés de stabilité et de robustesse des algorithmes [51].
- ✓ D'un point de vue pratique, cela simplifie la résolution du problème d'optimisation associé en réduisant l'espace des solutions. En d'autres termes, réduire la dimensionnalité limite le nombre de possibilités à tester, ce qui permet de traiter les données plus rapidement. Ce gain de temps est en fonction de la dépendance de la complexité temporelle de l'algorithme par rapport à la dimension. Ainsi dans le cas d'un algorithme comme le **Kernel Ridge Regression**, la dépendance vis-à-vis de la dimension étant linéaire, une réduction de moitié de la dimension double la vitesse de l'algorithme [51].

Pour réduire la dimension, on peut agir de deux manières :

- ✓ Supprimer des dimensions ou (descripteurs) ;
  Combiner les variables afin d'obtenir un plus petit nombre de nouvelles variables plus expressives et/ou moins redondantes [51].

❖ La technique Word2vec

W**ord2vec** est parmi les méthodes les plus courantes. Il a été développé par une équipe de recherche de Google sous la direction de **Tomas Mikolov**. Il repose sur des réseaux de neurones à deux couches et cherche à apprendre les représentations vectorielles des mots composant un texte, de telle sorte que les mots qui partagent des contextes similaires soient représentés par des vecteurs numériques proches. Il possède deux architectures neuronales, appelées **CBOW** et **Skip-Gram**, parmi lesquelles l'utilisateur peut choisir.

- CBOW reçoit en entrée le contexte d'un mot, c'est à dire les termes qui l'entourent dans une phrase, et essaye de prédire le mot en question.
- Skip-Gram fait exactement le contraire : elle prend en entrée un mot et essaye de prédire son contexte [52].

Dans les deux cas, l'entraînement du réseau se fait en parcourant le texte fourni et en modifiant les poids neuronaux afin de réduire l'erreur de prédiction de l'algorithme.

Il possède différents paramètres dont les plus importants sont :

- ❖ La dimensionnalité de l'espace vectoriel à construire, c'est à dire le nombre de descripteurs numériques utilisés pour décrire les mots (entre 100 et 1000 en général).
- ❖ La taille du contexte d'un mot, c'est à dire le nombre de termes entourant le mot en question (les auteurs suggèrent d'utiliser des contextes de taille 10 avec l'architecture Skip-Gram et 5 avec l'architecture CBOW).

L'algorithme word2vec est rapide à entrainer et à exécuter vu qu'il n'est composé que de deux couches. Ce qui révèle un avantage important par rapport à d'autres méthodes de word embedding [52].

Le plus souvent l'entrainement se fait en utilisant un algorithme de descente de gradient.

Le schéma (figure II-6) suivant illustre l'architecture CBOW, appliquée au dernier vers du poème de Guillaume Apollinaire « Nuit rhénane ». Dans cet exemple, les mots considérés comme trop communs, appelés aussi « mots outils » ou «stop words » sont éliminés des données fournies au réseau de neurones. La première couche du réseau projette chaque mot du contexte (en gris) vers sa représentation vectorielle (en bleu). Puis la couche cachée (en vert) analyse ces représentations vectorielles afin de tenter de prédire le mot central (encadré en rouge)[52].

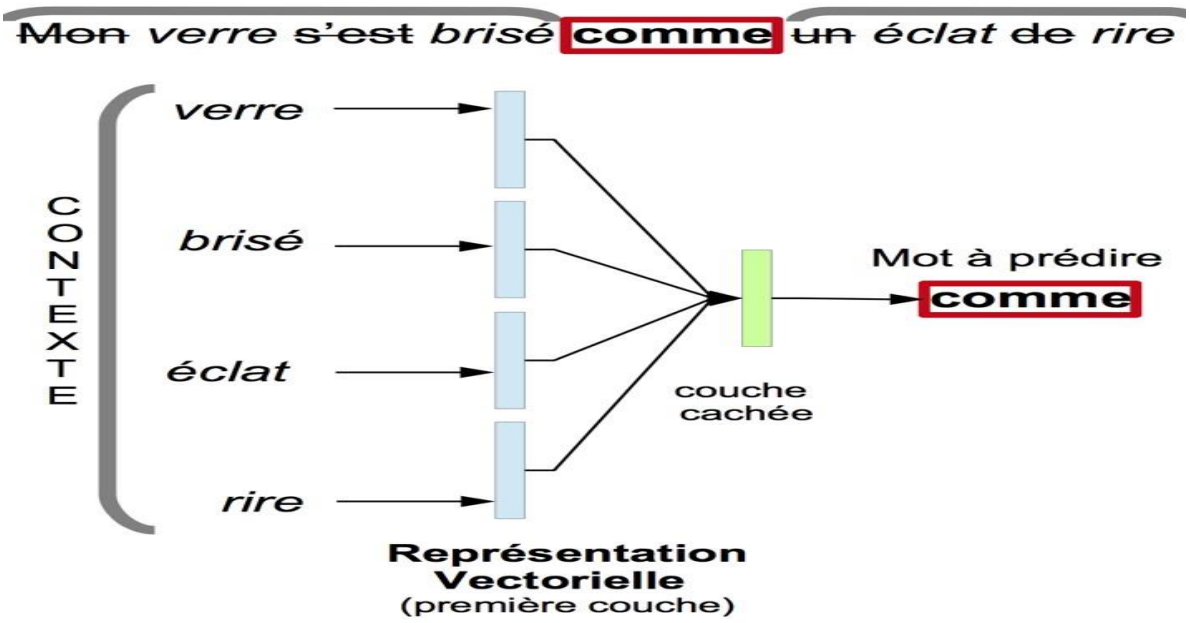

Figure II-6 : Architecture CBOW [52]

Il existe aussi d'autres outils pour la représentation des mots en vecteurs dont parmi lesquelles nous pouvons citer **Glove** [19] qui utilise une approche d'apprentissage non supervisée et **FastText**[20] qui est une bibliothèque open-source, gratuite et légère qui permet aux utilisateurs d'apprendre les représentations de textes.

II.4 Algorithmes d'apprentissage automatique

Dans cette partie, nous décrivons quelques algorithmes d'apprentissage automatique que nous allons utiliser par la suite dans notre proposition de méthode de calcul de similarité entre paires de phrases.

II.4.1 La régression linéaire (Linear Regression)

La régression linéaire est un algorithme de machine Learning (apprentissage automatique) basé sur l'apprentissage supervisée c'est-à-dire qu'à partir de variables explicatives ou prédictives x = (x_1, .. , x_n), le modèle fait une prédiction d'une variable cible (y) appelée variable expliquée. L'objectif est de trouver une fonction dite de prédiction ou une fonction coût qui décrit la relation entre x et y c'est-à-dire à partir des valeurs connues de x on arrive à donner une prédiction des valeurs de y. La relation est donnée par : y = f(x) avec f(x) une fonction linéaire [53]. En d'autres termes, il modélise une valeur de prédiction cible basée sur des variables indépendantes et est principalement utilisé pour découvrir la relation entre les variables et les prévisions [54].

[19] https://nlp.stanford.edu/projects/glove/
[20] https://fasttext.cc/

C'est l'un des algorithmes les plus utilisés en apprentissage automatique. Il permet de résoudre des problématiques de tous genres pour de multiples entités et secteurs. Encore connu sous le nom de modèle linéaire, la régression linéaire est un modèle statistique spécialisé dans la mise en œuvre des fonctions prédictives avec un minimum d'erreurs.

Son principe repose sur l'exploitation de valeurs numériques pour dégager une tendance ou une évolution prévisible dans le temps [55]. Il ajuste un modèle linéaire avec des coefficients w= (w1, .. , wn) pour minimiser la somme résiduelle des carrées entre les cibles observées dans l'ensemble des données et les cibles prédites par l'approximation linéaire.

Du point de vue de l'implémentation, il s'agit simplement de moindres carrées ordinaires ou de moindres carrées non négatifs enveloppés comme un objet prédicteur [56].

Les modèles de régression linéaires sont des modèles paramétriques c'est à dire qu'ils peuvent être assimilés à une fonction avec un nombre bien défini de paramètres qui peuvent être connu à l'avance. Les paramètres sont des variables qui permettent de faire des configurations et dont les valeurs peuvent être estimées à partir des données [57].

Par exemple avec la régression linéaire ou on a **y = ax + b, a** et **b** sont appellés les paramètres du modèle et la relation qui lie **y** à **x** est une relation affine.

On distingue le plus souvent deux types d'algorithmes de régression linéaire : la régression linéaire simple et la régression linéaire multiple. On parle de régression linéaire simple lorsqu'on a une seule variable indépendante sur laquelle on se base pour faire des prédictions. Par contre la régression linéaire multiple correspond à l'utilisation d'une ou de plusieurs variables explicatives. Ces variables sont appelées **features** et constituent les entrées principales de notre modèle [55].

II.4.2 Les arbres de décision (Decision Tree)

Un arbre de décision est un outil d'aide à la décision représentant un ensemble de choix sous la forme graphique d'un arbre. Les différentes décisions possibles sont situées aux extrémités des branches (les « feuilles » de l'arbre), et sont atteintes en fonction de décisions prises à chaque étape. C'est un outil utilisé dans des domaines variés tels que la sécurité, la fouille de texte (text mining), la médecine, l'apprentissage automatique etc. [58].

L'avantage des arbres de décision en apprentissage automatique est qu'ils peuvent être calculés automatiquement à partir de bases de données par des algorithmes d'apprentissage supervisé. Ces algorithmes sélectionnent automatiquement les variables discriminantes[21] à partir de données non structurées et potentiellement volumineuses. Ils peuvent ainsi permettre d'extraire des règles logiques de cause à effet (des déterminismes) qui n'apparaissaient pas initialement

[21] Des variables déterminées par l'extérieur du modèle et qui détermine les valeurs des autres variables sans être déterminé par les autres variables.

dans les données brutes [58]. La figure ci-dessous (figure II-7) donne dans le cadre général en quoi ressemble un arbre de décision.

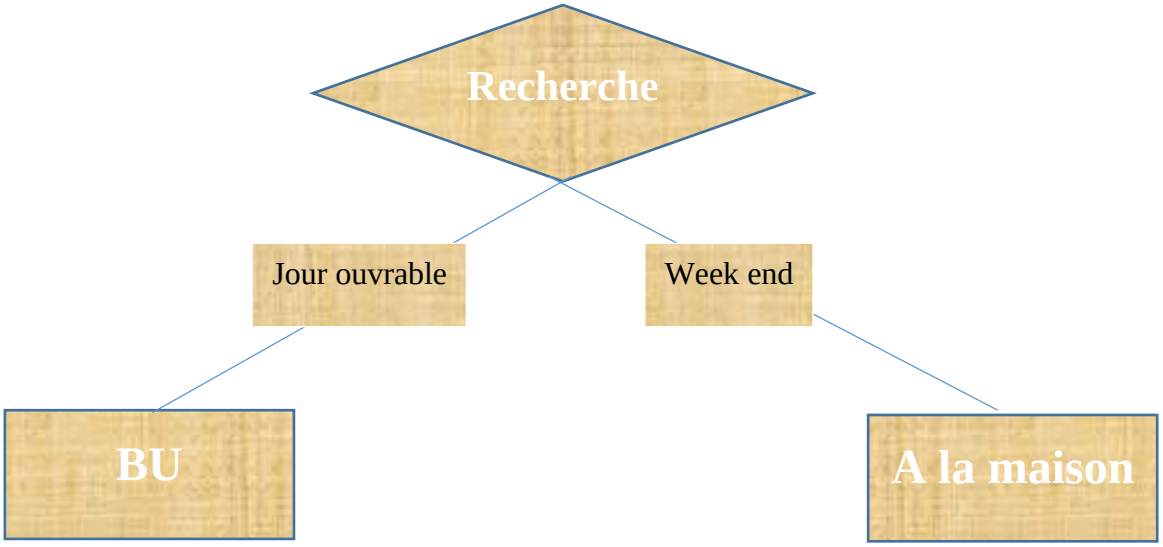

Figure II-7 : Exemple d'arbres de décisions

C'est une méthode d'apprentissage automatique supervisée non paramétrique utilisée pour la classification et la régression.

Un modèle basé sur les arbres de décision est très simple. Etant donnée plusieurs caractéristiques, la prise de décision commence par une de ces caractéristiques, si ce n'est pas suffisant on utilise une autre, ainsi de suite. Il est largement connu et utilisé dans de nombreuses entreprises pour faciliter le processus de prise de décision et l'analyse des risques.

Il a été largement utilisé dans les années 1960-1980 pour la construction de systèmes experts. Les règles sont introduites manuellement, pour cette raison ce modèle a perdu sa popularité après les années 80. L'apparition des méthodes mathématiques pour construire les arbres de décision fait revenir ce modèle à la bataille des algorithmes de l'apprentissage automatique [59].

L'objectif des arbres de décision est de créer un modèle qui prédit la valeur d'une variable cible en apprenant des règles de décisions simples déduites des caractéristiques des données.

Par exemple, dans l'exemple ci-dessous (figure II-8) les arbres de décision apprennent à partir des données pour approximer une courbe sinusoïdale avec un ensemble de règles de décision **if-then-else.** Plus l'arbre est profond, plus les règles de décision sont complexes et plus le modèle est adapté [60].

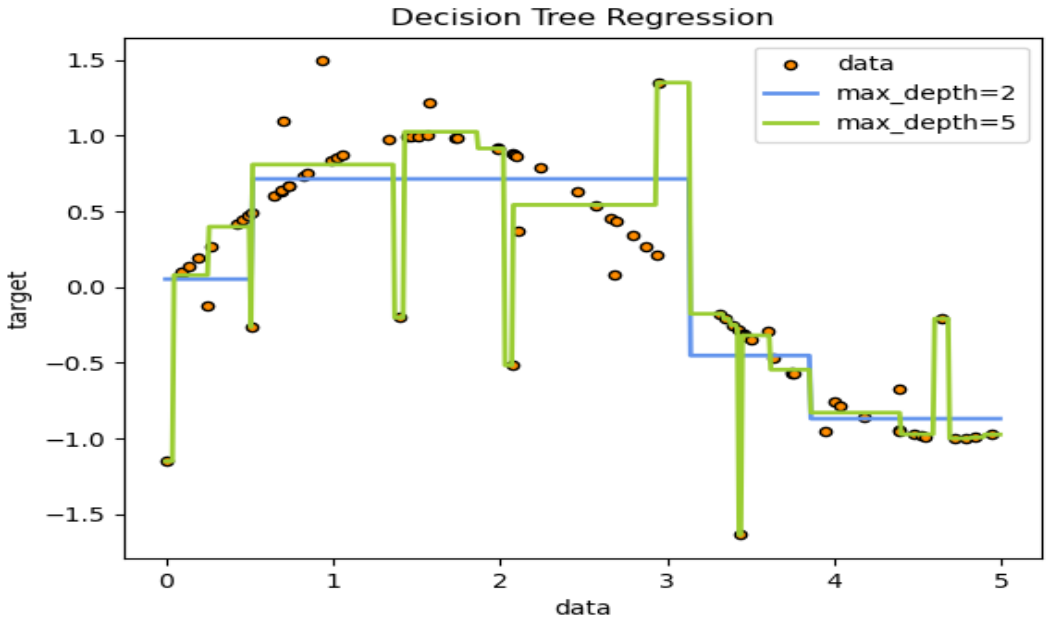

Figure II-8 : Exemple d'arbres de décisions [60]

Un autre avantage avec les arbres de décision est qu'ils sont simples à comprendre, à interpréter et les arbres peuvent être visualisés.

En classification, les arbres de décisions sont des classes capables d'effectuer une classification multi-classe sur un jeu de données. Ils prennent en entrée deux tableaux : un tableau X, clairsemé ou dense, de forme contenant les échantillons d'apprentissage, et un tableau Y de valeurs entières contenant les étiquettes de classe pour les échantillons d'apprentissage [60].

Pour un problème de classification, les arbres de décision utilisent couramment deux (02) critères de classement que sont :

- Gini

Le coefficient de Gini, ou indice de Gini est une mesure statistique permettant de rendre compte de la répartition d'une variable (salaire, revenus, patrimoine) au sein d'une population. Autrement dit, il mesure le niveau d'inégalité de la répartition d'une variable dans la population. Il a été développé par le statisticien **italien Corrado Gini**. Le coefficient de Gini est un nombre variant de 0 à 1, où 0 signifie l'égalité parfaite et 1, qui ne peut être atteint, signifierait une inégalité parfaite (une seule personne dispose de tous les revenus et une infinité d'autres n'ont aucun revenu**)** [61].

- Entropie

L'entropie de Shannon, due à Claude Shannon, est une fonction mathématique qui, intuitivement, correspond à la quantité d'information contenue ou délivrée par une source

d'information. Cette source peut être un texte écrit dans une langue donnée, un signal électrique ou encore un fichier informatique quelconque (collection d'octets).

Pour une source, qui est une variable aléatoire discrète X comportant n symboles, chaque symbole x_i ayant une probabilité P_i d'apparaître, l'entropie H de la source X est définie comme

$$H_b(X) = -E = [log_b P(X)] = \sum_{i=1}^{n} P_i log_b\left(\frac{1}{P_i}\right) = -\sum_{i=1}^{n} P_i log_b(P_i)$$

Équation 14 : Entropie de Shannon [62]**.**

Où E désigne l'espérance mathématique et $\log_b$ le logarithme en base b. On utilise en général un logarithme à base 2 car l'entropie possède alors les unités de bit/symbole. Les symboles représentent les réalisations possibles de la variable aléatoire X. Dans ce cas, on peut interpréter H(X) comme le nombre de questions à réponse oui/non que doit poser en moyenne le récepteur à la source, ou la quantité d'information en bits que la source doit fournir au récepteur pour que ce dernier puisse déterminer sans ambiguïté la valeur de X.

$$H(X) = H_2(X) = -\sum_{i=1}^{n} P_i log_2(P_i)$$

Équation 15 : Entropie de Shannon en base 2 [62]**.**

Il existe différents algorithmes d'arbres de décisions pour la résolution de problèmes en machine Learning :

II.4.3 Les forets aléatoires (Random Forest)

Les forets aléatoires sont des algorithmes d'apprentissage supervisé. Ils peuvent être utilisés à la fois pour la classification et la régression. Ils sont composés d'arbres. On dit que plus il y a d'arbres, plus une forêt est robuste. Les forêts aléatoires créent des arbres de décision sur des échantillons de données sélectionnés au hasard, obtiennent une prédiction de chaque arbre et sélectionnent la meilleure solution au moyen d'un vote. Il fournit également un assez bon indicateur de l'importance de la fonctionnalité [63].

Par ailleurs, c'est un méta estimateur qui adapte un certain nombre de classificateurs d'arbre de décision sur divers sous-échantillons de l'ensemble de données et utilise la moyenne pour améliorer la précision prédictive et contrôler le sur ajustement [64]. Le modèle de random Forest est un modèle paramétrique.

C'est un algorithme créé en 1995 par **HO,** puis formellement proposé par les scientifiques **Adele Culter** et **Leo Breiman** en 2001. Il est particulièrement efficace en termes de prédictions dans le domaine du machine Learning.

Il est composé de plusieurs arbres de décision, travaillant de manière indépendante sur une vision d'un problème. Chacun produit une estimation, et c'est l'assemblage des arbres de décision et de leurs analyses, qui va donner une estimation globale. En somme, il s'agit de s'inspirer de différents avis traitant un même problème pour mieux l'appréhender. Chaque modèle est distribué de façon aléatoire aux sous-ensembles d'arbres décisionnels. L'efficacité du modèle de random Forest dépend fortement de la qualité de l'échantillon de données de départ.

Un random Forest fonctionne sur le principe du **bagging**[22]. La première étape consiste à découper un jeu de données en sous-ensembles (arbres de décision), puis de proposer un modèle d'entraînement à chacun de ses groupes. Enfin, on combine les résultats de ces arbres afin d'obtenir la prévision la plus solide.

Il existe deux méthodes pour déterminer le résultat final. Un random Forest de régression consiste à calculer la moyenne des prévisions obtenues. On prend donc en compte l'ensemble des prédictions provenant des arbres décisionnels.

De même qu'un random Forest de régression, un random Forest de classification aussi fonctionne sur le système du **bagging**. La différence est que l'estimation se réalise à partir d'une méthode de classification. Ainsi, on choisit la catégorie de réponse la plus fréquente. Pour faire la sélection, on recherche la prévision qui revient le plus souvent au lieu d'utiliser tous les résultats obtenus.

Etant un modèle paramétrique, le random Forest utilise un **grid search** pour tester une série de paramètres et identifier les plus utiles. Par définition, le **grid search** est un outil d'optimisation très utile pour paramétrer au mieux la forêt d'arbres décisionnels. Parmi les paramètres on peut citer le nombre d'arbres décisionnels et le nombre de variables à mettre en œuvre dont leur choix est nécessaire [65]. Il existe aussi d'autres paramètres du modèle qui peuvent être utiles lors de la classification. Parmi ces derniers on peut citer [66]:

- Critère de division : c'est une mesure pour déterminer sur quelle caractéristique un arbre doit être divisé. Il peut être déterminé par deux méthodes en calculant l'impureté Gini ou par entropie. Par défaut le critère Gini est utilisé.
- Profondeur maximale : c'est la mesure de combien l'arborescence doit être étendue de chaque nœud jusqu'au nœud feuille. Par défaut ce paramètre n'est pas défini. Etant un algorithme se basant exclusivement sur des arbres, plus la profondeur est élevée plus il y' a de chances qu'il adapte sur les données. Donc random Forest est généralement admis d'avoir des arbres profonds.
- Fractionnement minimum d'échantillons : avec ce paramètre, on définit le nombre minimum d'éléments ou enregistrement devant être présent dans chaque nœud. Ainsi,

[22] Le bootstrap aggregating, également appelé bagging (de bootstrap aggregating), est un meta-algorithme d'apprentissage ensembliste conçu pour améliorer la stabilité et la précision des algorithmes d'apprentissage automatique.

on peut déterminer si l'algorithme peut arrêter de se fractionner davantage. Par exemple si ce paramètre est défini à 20, cela signifie qu'après 4 divisions si le nœud a encore plus de 20 éléments ou enregistrements, il peut être fractionné davantage. C'est le fractionnement continue tant qu'il y'a plus de 20 enregistrements.

La figure suivante (figure III-3) fournit un résume du fonctionnement du modèle de random Forest.

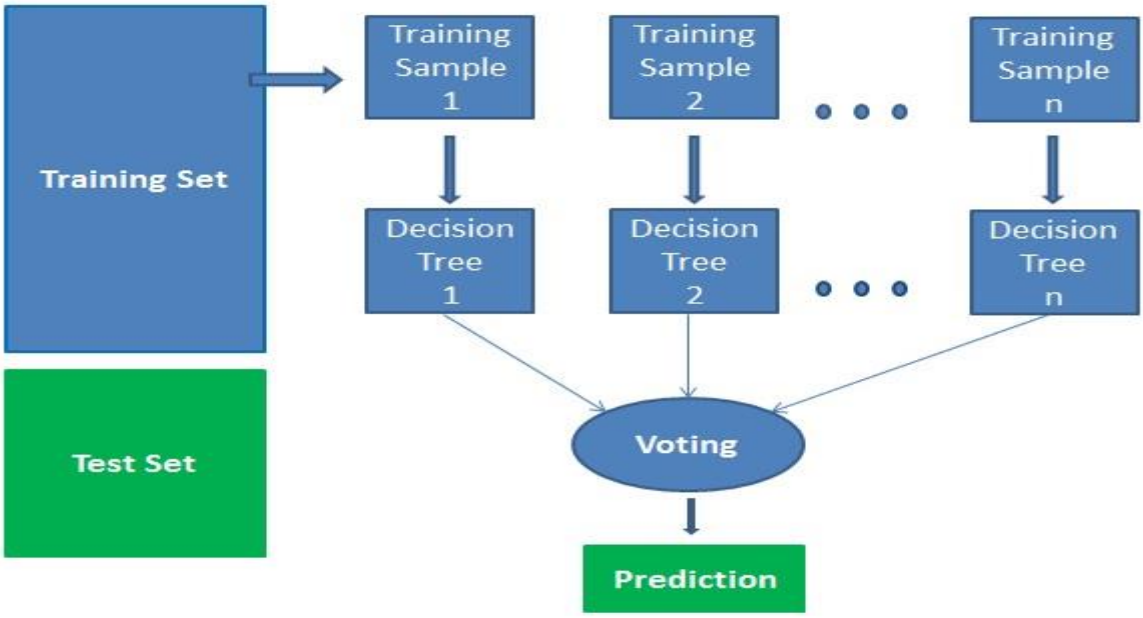

Figure II-9 : Fonctionnement de Random Forest [63]

L'algorithme de **Random Forest** fonctionne en quatre (04) étapes [63]:

- Sélectionner des échantillons à partir d'un ensemble de données ;
- Construire un arbre de décision pour chaque échantillon et obtenir un résultat de prédiction à partir de chaque arbre de décision ;
- Effectuer un vote pour chaque résultat prédit ;
- Sélectionner le résultat avec le plus de votes comme prédiction finale.

Parmi les avantages des forêts aléatoires on peut citer [63]:

- ✓ La précision et la robustesse des forêts en raison du nombre d'arbres de décision participant au processus ;
- ✓ La possibilité d'utiliser la régression ou la classification ;
- ✓ La gestion des valeurs manquantes ;
- ✓ La possibilité de sélectionner les fonctionnalités les plus contributives pour le classificateur.

II.4.4 Le perceptron multicouche (Multilayer Perceptron)

Les réseaux de neurones artificiels sont simplement des systèmes inspirés du fonctionnement des neurones biologiques. Le perceptron multicouche est le plus célèbre de tous les réseaux de neurones. C'est un système artificiel capable d'apprendre par l'expérience. Introduit en 1957 par **Franck Rosenblatt** et utilisé en 1982, le perceptron multicouche est de plus en plus utilisé [67].

Par définition, un perceptron multicouche (Multilayer Perceptron - MLP) est un modèle qui comprend plusieurs couches cachées et permet de produire un séparateur non linéaire. Il est constitué de plusieurs entrées et sorties. Il s'agit d'un réseau à propagation directe (**feed forward**) disposant d'un nombre de neurones artificiels variables qui compose plusieurs couches du système neuronal. On dépasse alors le machine Learning pour entrer dans l'ère du deep Learning, avec des systèmes d'apprentissage pouvant traiter des données en profondeur [68]**.** C'est un algorithme d'apprentissage supervisée qui apprend une fonction $\mathbf{f} : \mathbf{R^m} \rightarrow \mathbf{R^o}$ en s'entrainant sur un ensemble de données où **m** est le nombre de dimensions pour l'entrée et **o** le nombre de dimension pour la sortie. Etant donné un ensemble de fonctionnalités $X = \mathbf{x_1, x_2, .. ., x_m}$ et une cible y, l'algorithme peut apprendre un approximateur de fonction non linéaire pour la classification ou la régression. Elle est différente de la régression logistique, en ce sens qu'entre la couche d'entrée et la couche de sortie il peut y avoir une ou plusieurs couches non linéaires appelées couches cachées. Le perceptron multicouche est le premier réseau de neurones à avoir trouvé de nombreuses applications pratiques telles que la reconnaissance de fleurs, la détection de fraude, etc. Il peut être utilisé pour toutes tâches de classification supervisée. De nos jours il est l'un des modèles les plus populaires en machine Learning et deep Learning[23]. La figure II.10 montre un MLP à une couche cachée avec une sortie scalaire [69].

[23] Le deep Learning ou apprentissage profond consiste à ce qu'une intelligence artificielle parvienne à assimiler de nouvelles connaissances à travers un réseau de neurones artificiels

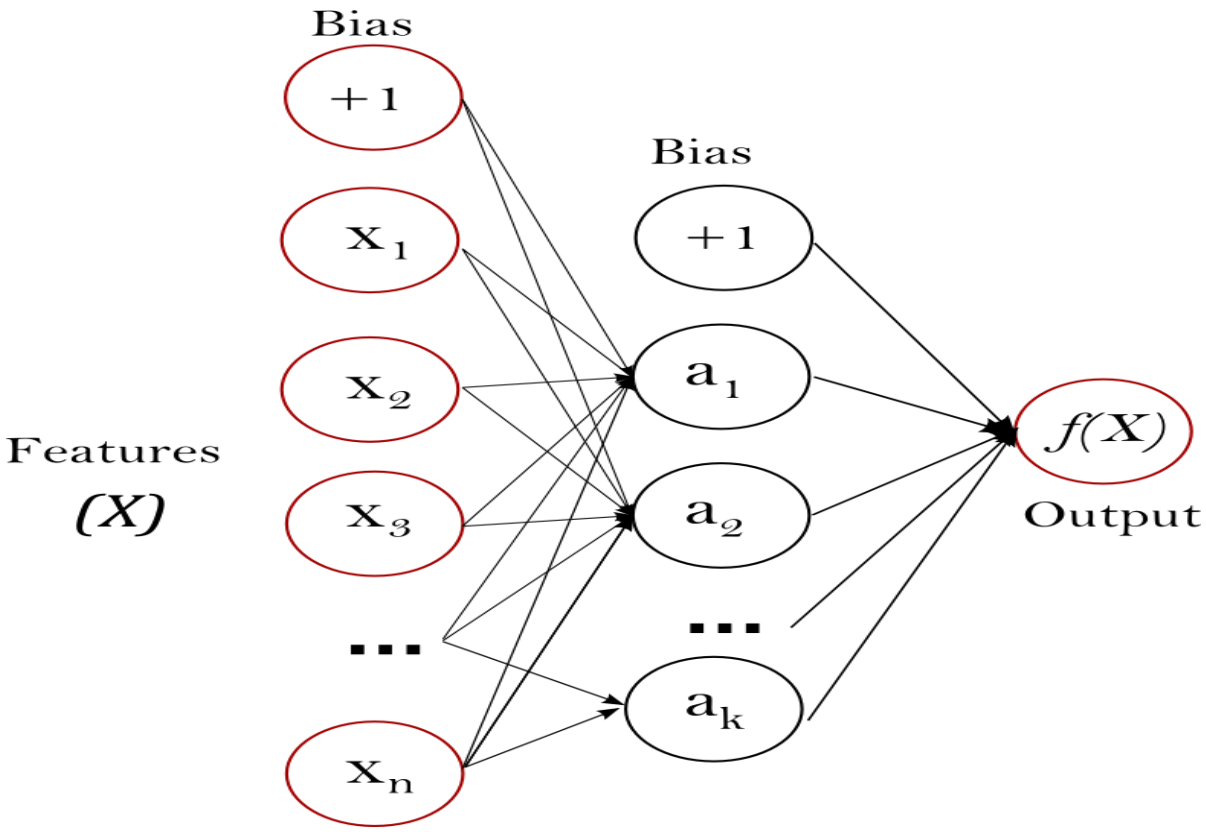

Figure II-10 : Le perceptron multicouche [69]

La couche la plus à gauche appelée couche d'entrée est constituée d'un ensemble de neurones $\{x_i \mid x_1, x_2, \ldots, x_m\}$ représentant les entités d'entrée. La couche de sortie reçoit les valeurs de la dernière couche masquée et les transforme en valeurs de sortie.

Le perceptron prend en entrée un vecteur à plusieurs dimensions (1 par réseau) et opère une séparation entre ces données pour fournir une sortie grâce à cette séparation qu'il a construite entre les données, il sait pour un nouvel exemple, quelle doit être la réponse [67].

II.4.5 La machine à vecteur de support (Support Vector Machine)

Les machines à vecteurs de support (Support Vector Machine - SVM) sont un ensemble de méthodes d'apprentissage supervisé utilisées pour la classification, la régression et la détection d'anomalies par exemple [70]. Ils sont connus pour leur solide garantie théorique, leur grande flexibilité ainsi que leur facilité d'utilisation même sans grande connaissance en data mining.

Ils ont été développés dans les années 1990 à partir d'une théorie d'apprentissage statistique développé par les informaticiens russes **Vladimir Vapnik** et **Alexey Chervonenkis** : la théorie de **Vapnik-Chervonenkis**. Ce modèle a été rapidement adopté en raison de sa capacité à travailler avec des données de grandes dimensions, ses garanties théoriques et les bons résultats réalisés en pratique. Requérant un faible nombre de paramètres, les SVM sont appréciées pour leur simplicité d'usage [71].

Comme le montre la figure ci-après (figure II-11), leur principe est simple : il consiste à séparer les données en classes à l'aide d'une frontière aussi simple que possible, de telle sorte que la distance entre les différents groupes de données et la frontière qui les sépare soit maximale. Cette distance est aussi appelée marge ce qui fait que les SVMs sont aussi appelés séparateurs à vaste marge, les vecteurs de support étant les données les plus proches de la frontière [72].

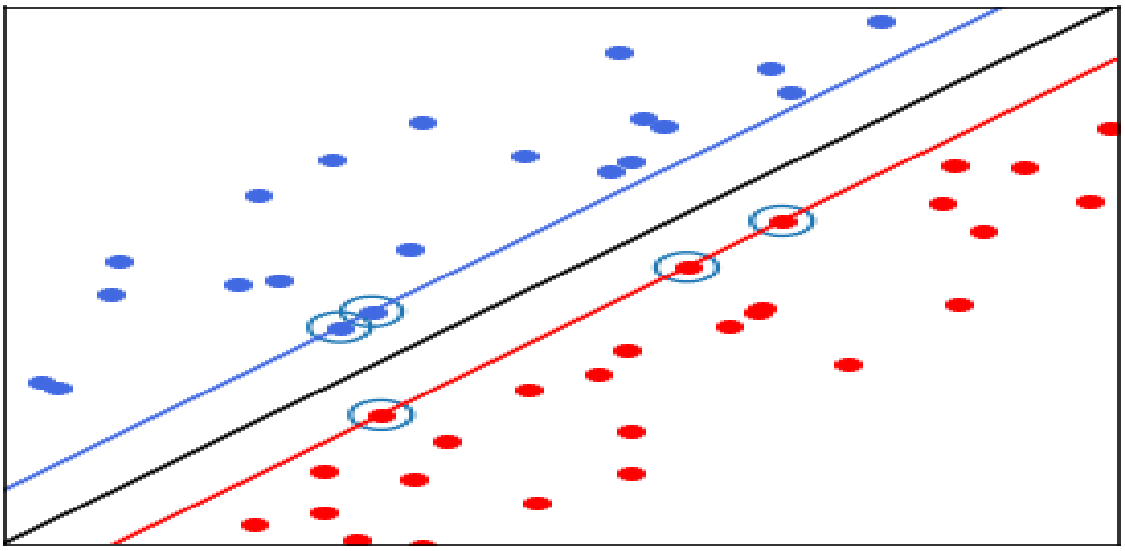

Figure II-11 : Principe de fonctionnement des SVMs [72]

La notion de frontière suppose que les données soient linéairement séparables, ce qui est rarement le cas. Pour y remédier, les SVMs reposent souvent sur l'utilisation de noyaux. Ces fonctions mathématiques permettent de séparer les données en les projetant dans un **feature space** (un espace vectoriel de plus grande dimension, voir figure ci-dessous). La technique de maximisation de marge permet, quant à elle de garantir une meilleure robustesse face au bruit et donc un modèle plus généralisable [72].

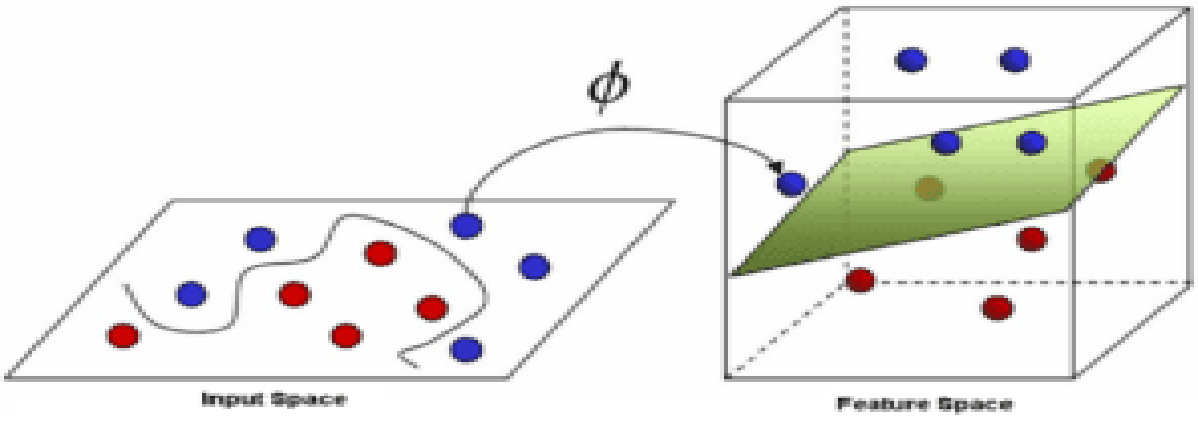

Figure II-12 : SVM : projection des données dans un feature space [72]

Pour la classification, on parle de **SVC**, **NuSVC** et **LinearSVC**. Ils correspondent à des classes capables d'effectuer une classification binaire ou multi-classe sur un jeu de données.

SVC et **NuSVC** sont des méthodes similaires mais acceptent des ensembles de paramètres légèrement différents et ont des formulations mathématiques différentes. D'autre part avec **LinearSVC** il existe une autre implémentation (plus rapide) de la classification des vecteurs de support pour le cas d'un noyau linéaire [70].

Les SVMs sont utilisés pour plusieurs applications comme la bio-informatique, la recherche d'informations, la vision par ordinateur, etc.

Par ailleurs les performances des SVMs sont en général de même ordre voire supérieures à celles des réseaux de neurones ou celles de mélanges gaussiens selon les données. Certains cas notables comme la classification d'images sont des exceptions [72].

Les algorithmes d'apprentissage automatique sont nombreux et variés. Dans cette section, nous avons présenté quelques-uns parmi les plus courants. Tous les algorithmes décrits sont des algorithmes d'apprentissage supervisée, donc utilisent un ensemble de données bien étiquetées.

La liste d'approches de calcul de similarité présentée dans cette partie n'est pas exhaustive mais couvrent les différentes catégories. Certaines sont limitées à notre niveau car se focalisant seulement sur l'aspect syntaxique des textes et ne permettent pas de prendre en compte la sémantique des phrases. Ainsi pour remédier à ça, d'autres approches utilisent des méthodes statistiques et/ou des ressources sémantiques. .

II.5 Conclusion

En définitive, on peut dire que le calcul de similarité reste une tâche complexe. Cette complexité peut s'expliquer par le fait qu'il existe une pléthore d'approches et qu'on rencontre d'énormes difficultés pour en choisir. Le choix d'une approche dépend exclusivement du problème à résoudre c'est-à-dire la nature des données, la quantité de données disponible ainsi que leur type.

Lors du calcul de similarité sémantique, le choix d'une approche est fondamental vu que certaines ne gèrent pas la sémantique, d'autres font recours à la sémantique mais ne permettent pas de gérer le problème de langue. Par exemple, les approches à base de connaissances utilisant **Wordnet** ne gère que la langue anglaise.

Dans le prochain chapitre, nous présentons une approche supervisée que nous proposons pour le calcul de similarité sémantique entre paires de phrases.

Chapitre III : Proposition d'une approche supervisée de calcul de similarité sémantique

III.1 Introduction

Dans ce chapitre, nous décrivons l'approche que nous proposons pour le calcul de similarité sémantique entre des paires de phrases. Nous proposons une approche supervisée de calcul de similarité sémantique basée sur des algorithmes d'apprentissage automatique. Cette approche, comme décrite dans [28]**,** consiste à représenter chacune des paires de phrases par un ensemble d'attributs. Différentes mesures de similarité textuelle et différents algorithmes d'apprentissage automatiques sont utilisés.

Dans la suite, nous allons présenter l'approche proposée en partant d'abord du prétraitement des textes qui constitue la première étape de notre méthode. Ensuite les mesures de similarité textuelle utilisées sont exposées. Enfin les algorithmes d'apprentissage automatique sont utilisés pour générer les modèles de calcul de similarité. La figure ci-dessous (figure III.1) présente notre sous forme d'une architecture en énumérant toutes les étapes réalisable pour le calcul de la similarité sémantique entre une paires de phrases.

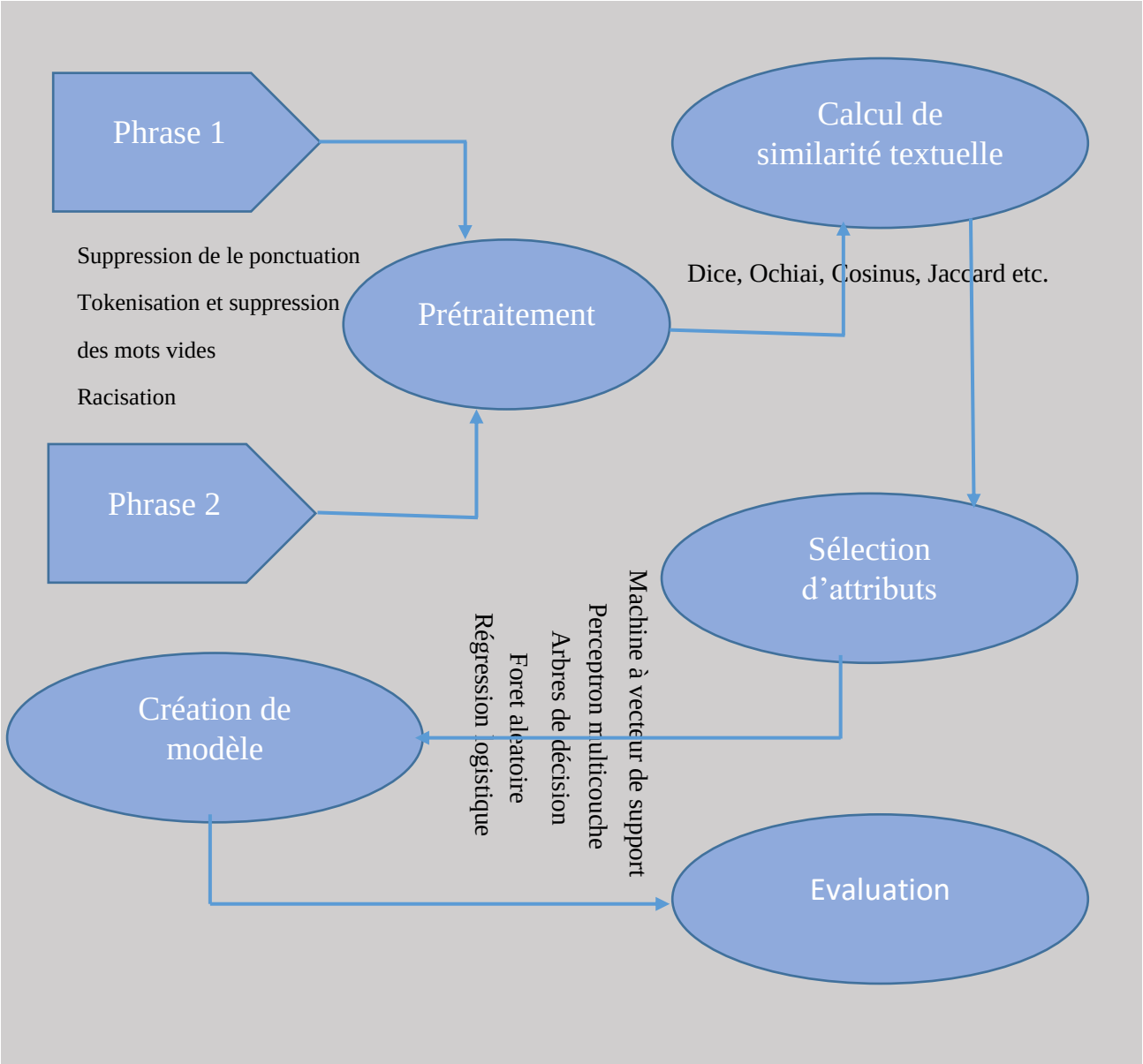

Figure III-1 : Représentation architecturale de notre approche de calcul de similarité entre paires de phrases

III.2 Prétraitement de données

Le prétraitement des données est une étape très importante dans le traitement des textes. Elle constitue la première étape à réaliser pour notre méthode.

Dans le cadre de notre travail et vu les données que nous traitons (corpus composé d'un ensemble de paires de phrases en Français), nous avons réalisé les tâches suivantes : la suppression de la ponctuation, la suppression des mots vides, la tokenisation et la racinisation

afin de permettre au mieux possible le calcul de similarité textuelle avec les mesures abordées dans le chapitre II.

❖ La suppression de la ponctuation

La suppression de la ponctuation est une étape importante du processus de prétraitement et entre dans le cadre du nettoyage des données textuelles pour les rendre plus propre et utilisable par la machine. Ainsi, pour chaque phrase du corpus, les points, points-virgules, points d'interrogation, d'exclamation etc. sont supprimés.

Le tableau ci-après (tableau III-1) donne un extrait de quelques phrases de notre corpus dans lesquelles on va supprimer toutes les ponctuations. Le résultat ainsi obtenu est stocké dans le tableau III.2.

Tableau III-1 : Extrait avant suppression de la ponctuation

Id=1	- En l'absence d'amélioration comme en cas de persistance des symptômes, prendre un avis médical.
Id=2	- En outre, la patiente sera avertie de la nécessité d'une consultation rapide devant tout saignement vaginal anormal.
Id=3	En cas de saignement vaginal anormal, il importe de consulter au plus vite votre médecin.
Id=4	- L'administration d'huile de paraffine chez les jeunes enfants, les personnes débilitées, allongées ou ayant des difficultés de déglutition doit être prudente en raison du risque d'inhalation bronchique et de pneumopathie lipoïde.
Id=5	- Ne pas utiliser chez les personnes présentant des difficultés pour avaler en raison du risque d'inhalation de la paraffine liquide qui entraîne une pneumopathie lipoïde.
Id=6	Aucun essai étudiant les effets bénéfiques potentiels des hormones oestrogènes chez les femmes atteintes d'athérosclérose des membres inférieurs n'était disponible.

Tableau III-2 *:* Extrait après suppression de la ponctuation

Id=1	En l absence d amélioration comme en cas de persistance des symptômes prendre un avis médical
Id=2	En outre la patiente sera avertie de la nécessité d une consultation rapide devant tout saignement vaginal anormal
Id=3	En cas de saignement vaginal anormal, il importe de consulter au plus vite votre médecin.
Id=4	L administration d huile de paraffine chez les jeunes enfants les personnes débilitées allongées ou ayant des difficultés de déglutition doit être prudente en raison du risque d inhalation bronchique et de pneumopathie lipoïde
Id=5	Ne pas utiliser chez les personnes présentant des difficultés pour avaler en raison du risque d inhalation de la paraffine liquide qui entraîne une pneumopathie lipoïde
Id=6	Aucun essai étudiant les effets bénéfiques potentiels des hormones oestrogènes chez les femmes atteintes d'athérosclérose des membres inférieurs n'était disponible

❖ Tokenisation et suppression des mots vides

Pour chaque paire de phrases de notre corpus, on réalise la tokenisation et la suppression des mots vides avant de passer au calcul de similarité. Cette étape intervient juste après la tâche de suppression de la ponctuation. Ainsi, chacune des phrases de notre corpus est subdivisée en mots appelés jetons. Le tableau III-3 donne le résultat obtenu après tokenisation et suppression des mots vides des phrases du tableau III.2.

Tableau III-3 : Tokenization et suppression des mots vides

Id=1	'En', 'absence', 'amélioration', 'comme', 'cas', 'persistance', 'symptômes', 'prendre', 'avis', 'médical'
Id=2	'En', 'outre', 'patiente', 'avertie', 'nécessité', 'consultation', 'rapide', 'devant', 'tout', 'saignement', 'vaginal', 'anormal'
Id=3	En', 'cas', 'saignement', 'vaginal', 'anormal', 'importe', 'consulter', 'plus', 'vite', 'médecin'
Id=4	'L', 'administration', 'huile', 'paraffine', 'chez', 'jeunes', 'enfants', 'personnes', 'débilitées', 'allongées', 'difficultés', 'déglutition', 'doit', 'être', 'prudente', 'raison', 'risque', 'inhalation', 'bronchique', 'pneumopathie', 'lipoïde'
Id=5	'Ne', 'utiliser', 'chez', 'personnes', 'présentant', 'difficultés', 'avaler', 'raison', 'risque', 'inhalation', 'paraffine', 'liquide', 'entraîne', 'pneumopathie', 'lipoïde'
Id=6	'Aucun', 'essai', 'étudiant', 'effets', 'bénéfiques', 'potentiels', 'hormones', 'oestrogènes', 'chez', 'femmes', 'atteintes', 'athérosclérose', 'membres', 'inférieurs', 'disponible'

❖ La racinisation (Stemming)

La racinisation consiste à retourner le radical d'un mot. Après avoir fini de tokeniser et de supprimer les mots vides de notre corpus, nous procédons à la racinisation pour mieux faciliter leur exploitation. Ainsi cette étape constitue la toute dernière à réaliser dans cette phase de prétraitement des données textuelles. Le tableau III-4 donne les racines des différents mots des listes 1, 2 et 3 du tableau III-3.

Tableau III-4 : Racinisation de textes

Id=1	['en', 'absenc', 'amélior', 'comm', 'ca', 'persist', 'symptôm', 'prendr', 'avi']
Id=2	['en', 'outr', 'patient', 'averti', 'nécessité', 'consult', 'rapid', 'devant', 'tout', 'saignement', 'vagin']
Id=3	['en', 'ca', 'saignement', 'vagin', 'anorm', 'import', 'consult', 'plu', 'vite']

Outre ces étapes de prétraitement, il existe d'autres dont la réalisation est indispensable pour le calcul de similarité. Toutefois, certaines de ces étapes n'interviennent que pour certains cas spécifiques ; par exemple la vectorisation de textes est indispensable lors qu'on veut calculer la similarité avec la mesure de cosinus. Elle consiste à représenter un texte sous forme d'un vecteur.

La plupart des mesures que nous allons utiliser dans la suite pour le calcul de similarité textuelle prennent en entrée des ensembles. Ainsi, après la réalisation de cette dernière étape qui est la racinisation, nous transformons le résultat obtenu en ensemble avant de passer au calcul de similarité textuelle.

III.3 Le calcul de similarité textuelle

Nous utilisons les mesures de similarité textuelle présentées dans le chapitre II pour calculer les scores de similarité entre chaque paire de phrases et ces dernières vont constituer les attributs de nos méthodes.

Le tableau (tableau III-5) ci-après donne un extrait du résultat obtenu après le calcul de similarité textuelle avec nos différentes mesures.

Tableau III-5 : Extrait du tableau des mesures de similarité textuelle

id	Similarité de Dice	Similarité de Jaccard	Similarité de Ochiai	Similarité Cosinus
1	0.75	0.6	0.760638829255665	0.7892051872524705
2	0.9	0.8181818181818182	0.9	0.9375
3	0.36363636363636365	0.2222222222222222	0.3651483716701107	0.32539568672798425
4	0.42857142857142855	0.2727272727272727	0.42857142857142855	0.6310315028394047
5	0.7857142857142857	0.6470588235294118	0.7877263614433762	0.760608730574164
6	0.8333333333333334	0.7142857142857143	0.8451542547285166	0.7385489458759965
7	0.35294117647058826	0.21428571428571427	0.35856858280031806	0.5892374918291376
8	0.0	0.0	0.0	0.1289651760438322
9	0.5714285714285714	0.4	0.5735393346764044	0.6825236327899351

Les mesures utilisées pour le calcul de cette similarité textuelle sont nombreuses et celles mentionnées dans le tableau ci-dessus constituent quelques-unes d'entre elles. Comme le montre le tableau, les scores pour toutes les mesures utilisées sont compris entre 0 et 1. La valeur 0 signifie que les deux phrases sont distinctes c'est-à-dire aucune similarité entre les deux. La valeur 1 signifie les deux phrases sont complètement similaires.

III.4 Construction de modèles de calcul de similarités

La construction de modèles de calcul de similarité est l'étape la plus importante de la méthode que nous proposons. Elle repose sur les algorithmes d'apprentissage automatique présentés dans le deuxième chapitre. En d'autres termes, il sera question pour chaque algorithme de créer

un modèle, l'entrainer sur des données d'entrainement, le tester sur des données de tests différentes de données d'entrainements avant de passer à l'évaluation des différents modèles obtenus pour en sélectionner le ou les meilleurs. Ainsi il importe de comprendre la notion de modèle, d'entrainement et de prédiction. Avant de passer à la notion de modèle et de prédiction, une étape importante et indispensable à la création de modèles de qualité est le choix des attributs qui seront utilisés pour alimenter les modèles.

III.4.1 La sélection d'attributs

Le calcul de similarité textuelle implique un bon nombre de mesures. Ainsi, pour chaque paire de phrases de notre ensemble de données, dix-sept (17) mesures sont utilisées pour le calcul du score. Ainsi, on obtient un jeu de données avec 17 colonnes contenant les scores de similarité qui varient entre 0 et 1. L'utilisation des différentes mesures est possible dans la mesure où on note à chaque fois des scores différents pour une paire de phrases calculée en utilisant des mesures différentes. Néanmoins pour un bon nombre de mesures, la différence notée n'est pas toujours significative alors que pour d'autres elle est très importante. Malgré toutes ces remarques, aucune hypothèse ne peut être émise sur quelle mesure est plus pertinente qu'une autre. Ceci étant, on fait recours à des méthodes de sélection d'attributs.

La sélection d'attributs est l'une des étapes les plus importantes lors de la création d'un modèle d'apprentissage automatique. Son objectif est de trouver le meilleur ensemble d'attributs possible pour créer un modèle d'apprentissage automatique. Il existe différentes techniques de sélection d'attributs en apprentissage automatique parmi lesquelles nous avons :

- Les méthodes de filtrage

Ces méthodes sont généralement utilisées lors de l'étape de prétraitement. Elles sélectionnent des attributs dans l'ensemble des données indépendamment d'un algorithme d'apprentissage automatique. En termes de calcul, elles sont très rapides et peuvent être très efficaces pour supprimer les caractéristiques dupliquées, corrélées et redondantes. La figure ci-dessous (figure III-1) donne le principe général de cette méthode de sélection d'attributs.

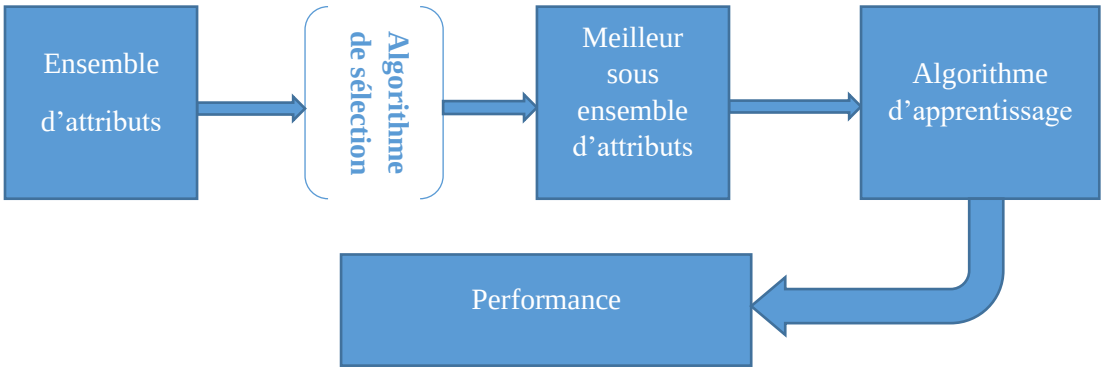

Figure III-2 : Les méthodes de filtrage

- Les méthodes d'emballage

Les méthodes d'emballages (wrapper en anglais), qualifiés de gloutonnes, entrainent l'algorithme en utilisant un sous ensemble de fonctionnalités de manière itérative. L'ajout et la suppression d'attributs ont lieu avant la formation du modèle et sur la base des conclusions tirées. Pour sélectionner le meilleur sous ensemble, les critères d'arrêt sont généralement prédéfinis par la personne qui entraine le modèle. Ces méthodes sont très coûteuses en termes de calcul. Le principe de fonctionnement de ces méthodes est résumé dans la figure ci-dessous (figure III-2).

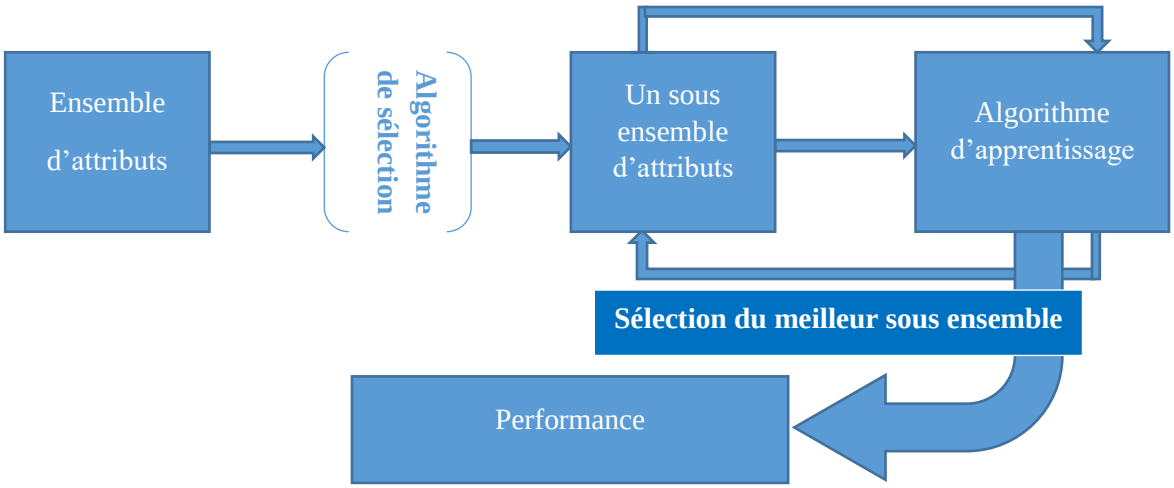

Figure III-3 : Les méthodes d'emballage

- Les méthodes embarquées (méthodes intégrées)

Avec ces méthodes, l'algorithme de sélection d'attributs est mélangé dans le cadre de l'algorithme d'apprentissage automatique, ayant aussi ses propres méthodes de sélections de caractéristiques intégrées. Les méthodes intégrées rencontrent les inconvénients des méthodes de filtrage et d'emballage et bénéficient de leurs avantages. Ces méthodes sont plus rapides et plus précises que les méthodes de filtrage et prennent également en considération une combinaison de fonctionnalité**s** [73]. Le principe de ces méthodes est résumé dans la figure III-3.

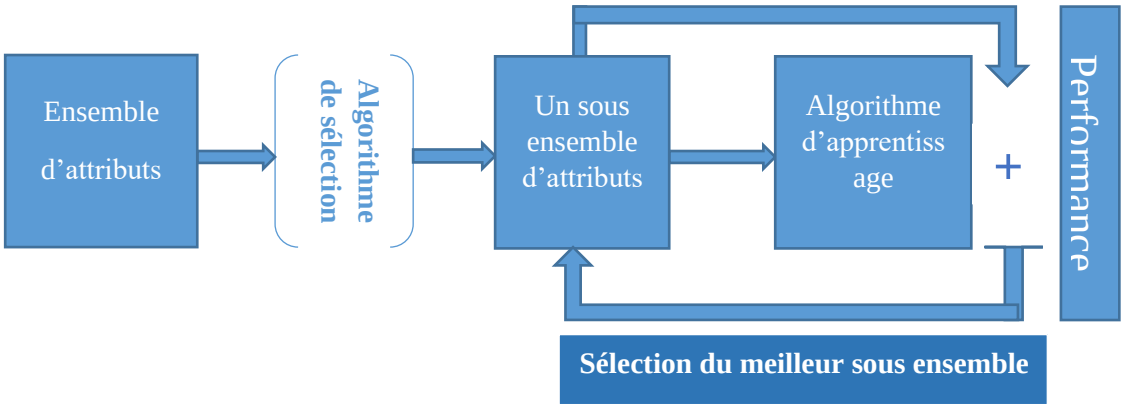

Figure III-4 : Les méthodes embarquées (méthodes intégrées)

Dans la suite de ce travail, nous allons nous limiter sur les méthodes de filtrage pour réaliser la sélection d'attributs. Beaucoup de techniques ou algorithmes sont utilisés avec ces méthodes de sélection. On va essayer d'en présenter quelques-unes qui aident à la réalisation de cette tâche et qui sont faciles à implémenter.

III.4.1.1 Le test de chi-deux (X^2)

Le test de **chi-deux** (X^2) est une méthode de sélection généralement utilisée pour tester la relation entre deux variables catégorielles. Il compare les valeurs observées de différents attributs de l'ensemble de données à leurs valeurs attendues [73].

En d'autres termes le test de chi-deux est utilisé pour tester l'hypothèse nulle c'est à dire l'absence de relation entre deux variables catégorielles. On peut également dire que ce test vérifie l'hypothèse de dépendance de ces variables. Si deux variables dépendent l'une de l'autre, elles partagent quelque chose, la variation de l'une influence la variation de l'autr**e** [74]. Sa formule s'exprime par :

$$X^2 = \sum \frac{(valeur\ observée - valeur\ attendue)^2}{valeur\ attendue}$$

Équation 16 : formule de chi-deux [73]

III.4.1.2 Le seuil de variance

Il s'agit d'une approche dans laquelle tous les attributs dont la variance n'atteint pas le seuil fixé sont supprimées. Par défaut, cette méthode supprime les entités ayant une variance nulle. L'hypothèse faite à l'aide de cette méthode est que les caractéristiques de variance plus élevée sont susceptibles de contenir plus d'informations [73].

La variance est une indication de dispersion des valeurs, c'est-à-dire qu'elle est toujours positive. Pour une série statistique dont tous les termes ont la même valeur, la variance est nulle. Elle augmente lors que les valeurs sont étalées et invariante lors de l'ajout d'une constante. Si on considère un ensemble de k séries statistiques, la variance globale peut être calculé avec

l'effectif n_i, la variance v_i d'une série et la moyenne de chaque série initiale par la formule suivante :

$$V = \frac{1}{N}\sum_{i=1}^{k} n_i(v_i + (\bar{x} - \bar{x_i})^2)$$

Équation 17 : Formule de la variance

Où N $= \sum_{i=1}^{k} n_i$ est l'effectif total et $\bar{x}$ $\frac{1}{N}\sum_{i=1}^{k} n_i\bar{x}_i$ est la moyenne globale. En d'autres termes la variance globale est la variance des moyennes et de la moyenne des variances.

L'analyse de la variance permet d'étudier le comportement d'une variable continue à expliquer en fonction d'une ou de plusieurs variables explicatives catégorielles. Lorsque l'on souhaite étudier le comportement de plusieurs variables à expliquer en même temps, on utilisera une analyse de la variance multiple (MANOVA) [75].

III.4.2 Le modèle d'apprentissage automatique

Après avoir procédé à la sélection de tous les attributs pertinents en utilisant une des méthodes de sélection décrites précédemment, nous passons à la construction de modèles d'apprentissage automatique. Le modèle d'apprentissage automatique (machine Learning) est un fichier entrainé pour reconnaitre certains types de modèles. L'entrainement se fait sur un ensemble de données fournissant au modèle un algorithme qu'il peut utiliser pour raisonner sur les données et apprendre de celles-ci [76]. Le type d'algorithme utilisé par un modèle de machine learning dépend exclusivement du type d'apprentissage souhaité.

Ainsi, pour chaque algorithme présenté dans cette section, nous créons un modèle que nous allons par la suite entrainer et tester respectivement sur un ensemble de données d'apprentissage ou données d'entrainement et un ensemble de données de tests.

Après la phase d'entrainement, nous passons à celle de test et d'évaluation. Cette étape est la toute dernière étape à réaliser en apprentissage automatique. Elle consiste à confronter le modèle à la réalité du terrain. Pour se faire, nous utilisons un sous ensemble de notre ensemble de données appelé ensemble de test. Cet ensemble est évidemment l'autre partie de notre dataset différente des données d'apprentissage utilisées pour l'entrainement du modèle. En d'autres termes les données de test sont un sous-ensemble de l'ensemble d'information qui cherche à affiner le modèle grâce à un certain nombre de scenarios ou données que l'ordinateur n'a jamais expérimenté lors de la phase d'apprentissage [77].

III.4.3 Evaluation des modèles

Après avoir entrainé et testé nos modèles, nous évaluons leurs performances sur des jeux de données. Ainsi, nous serons en mesure de déterminer le meilleur modèle pour notre problème.

Elle demande beaucoup d'attention, car évaluer un modèle sur les mêmes données qui ont servi à son apprentissage ne garantit pas de bons résultats sur de nouvelles données. Cela conduit à ce qu'on appelle le biais de sur-apprentissage. C'est-à-dire avoir un modèle qui semble être performant mais ne soit pas capable de généraliser ses performances sur de nouvelles données. La technique utilisée pour bien évaluer un modèle est de simuler le fait de ne connaitre que les étiquettes sur le jeu d'apprentissage dont les étiquettes sont connues. Ainsi, on sépare le jeu d'apprentissage en deux jeux de données : un pour l'apprentissage et un autre pour le test. Le modèle entrainé sur le jeu d'apprentissage est par la suite évalué sur le jeu de données de test[78].

Il existe plusieurs méthodes pour la validation de modèles d'apprentissage automatique.

III.4.3.1 La validation croisée

La validation croisée (cross validation en anglais) est une méthode d'évaluation se basant sur la formation de plusieurs modèles d'apprentissage machine sur des sous-ensembles de données d'entrées et leur évaluation sur des sous-ensembles complémentaires [79].

Le principe de la validation croisée repose sur la réalisation d'une opération à plusieurs reprises de telle sorte que les ensembles de données connues soient à tour de rôle utilisés comme données d'apprentissage et de test. Pour se faire, on subdivise les données en parties égales dans la mesure du possible (fold en anglais) et on utilise à chaque fois qu'on crée un modèle une partie comme jeu de test et le reste pour l'entrainement du modèle. Pour mieux illustrer ce principe, la figure ci-dessous donne un exemple de validation croisée où le nombre de parties (k folds) est égal à 5 [80].

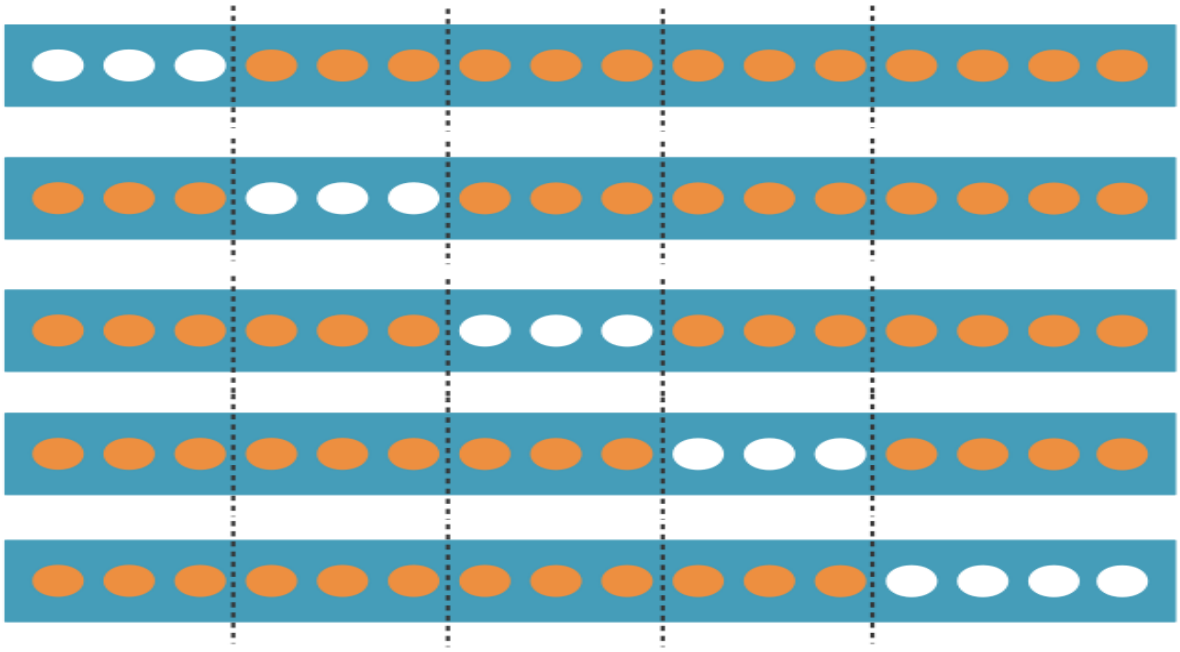

Figure III-5 : Validation croisée avec 5 folds [80]

En blanc on a les points appartenant au jeu de données de test et en orange les points appartenant au jeu de données d'entrainement.

L'évaluation de modèle de machine learning avec la validation croisée se fait en calculant la moyenne des performances et l'erreur de chaque fold ou en évaluant les prédictions faites sur l'ensemble des données. Pour gagner du temps on utilise généralement k = 5 ou k = 10.

Pour cette méthode, il est important d'appliquer la stratification. La stratification est un processus qui consiste à diviser les données connues en folds homogènes avant l'échantillonnage, c'est-à-dire répartir les étiquettes pour que chaque fold ressemble au maximum à un petit jeu de données connues [80].

L'importance d'utiliser cette méthode de validation est qu'on peut évaluer un modèle de machine Learning même avec des données limitées. Elle est à la fois simple à comprendre, simple à implémenter et moins biaisée que les autres méthodes.

III.4.3.2 La matrice de confusion

Comme la validation croisée, la matrice de confusion (confusion matrix en anglais) est un outil pour mesurer les performances d'un modèle de machine learning. Elle permet de vérifier à quel point les prédictions sont exactes par rapport à la réalité dans les problèmes de classification. En d'autres termes une matrice de confusion permet de savoir à quel point le modèle de machine learning est « confus », ou se trompe. Il s'agit d'un tableau avec en colonne les différents cas réels et en ligne les différents cas d'usage prédits.

Pour un problème de classification, une matrice de confusion représente un résumé des résultats de prédictions. Les prédictions correctes et incorrectes sont mises en exergue et reparties par classe et les résultats sont ensuite comparés avec les valeurs réelles.

Pour calculer une matrice de confusion, il est nécessaire de disposer d'un ensemble de données de test (test dataset) ou d'un ensemble de données de validation (validation dataset) avec les valeurs attendues. On fait ensuite une prédiction pour chaque ligne du « test dataset ». En fonction des résultats attendus et des prédictions, la matrice indique le nombre de prédictions correctes et le nombre de prédictions incorrectes pour chaque classe. Chaque ligne du tableau correspond à une classe prédite.

L'avantage de ces matrices est qu'elles sont très simples à lire et à comprendre. Elles permettent de visualiser très rapidement les données et les statistiques afin d'analyser les performances d'un modèle et d'identifier les tendances qui peuvent aider à modifier les paramètres. Pour **les problèmes de classification avec trois classes ou plus**, il est aussi possible d'utiliser une matrice de confusion en ajoutant des lignes et des colonnes.

Il existe quatre (04) terminologies dans une matrice de confusion qu'il convient de bien comprendre [81].

- ✓ VP (Vraie Positive) : le cas où la prédiction est positive, et où la valeur réelle est effectivement positive.
- ✓ VN (Vraie Négative) : le cas où la prédiction est négative, et où la valeur réelle est effectivement négative.
- ✓ FP (Faux Positive) : les cas où la prédiction est positive, et ù la valeur réelle est effectivement négative.
- ✓ FN (Faux Négative) : le cas où la prédiction est négative, et où la valeur réelle est effectivement positive.

Pour une illustration, prenons un exemple de classement d'étudiants selon qu'ils sont admis ou pas suite à un examen. On considère deux classes pour cet exemple : « est admis » et « n'est pas admis ». Le tableau suivant représente la matrice de confusion pour cet exemple.

Tableau III-5 : Exemple de matrice de confusion

		Cas réels **Si l'étudiant est admis ou non**	
		est admis	**n'est pas admis**
Prédiction Ce que le modèle prédit	**est admis**	Nombre de **Vrai positif** (VP)	Nombre de **Faux positif** (FP)
	n'est pas admis	Nombre de **Faux négatif** (FN)	Nombre de **Vrai négatif** (VN)

On obtient donc les quatre valeurs suivantes :

- o VP : Les valeurs réelles et prédites sont identiques et positives c'est-à-dire l'étudiant est admis et le modèle le prédit.
- o VN : les valeurs réelles et prédites sont identiques et négatives. C'est-à-dire l'étudiant n'est pas admis et le modèle prédit qu'il ne l'est pas.
- o FP : les valeurs réelles et prédites sont différentes. C'est-à-dire l'étudiant est admis mais le modèle prédit qu'il ne l'est pas.
- o FN : les valeurs réelles et prédites sont différentes. C'est-à-dire l'étudiant n'est pas admis mais le modèle prédit qu'il l'est.

L'étude de ces valeurs prédictives permet de définir si le modèle de machine learning est fiable, dans quels cas il commet des erreurs et dans quelle mesure [82].

III.4.3.3 Le coefficient de corrélation de Pearson

Le coefficient de corrélation de Pearson est une mesure de quantification de l'association entre les deux variables continues et la direction de la relation avec ses valeurs allant de -1 à 1 [73].

Par ailleurs il est défini comme un indice reflétant une relation linéaire entre deux variables continues. La valeur 0 reflète une relation nulle entre les deux variables, une valeur négative (corrélation négative) signifiant que lorsqu'une des variables augmente, l'autre diminue tandis qu'une valeur positive (corrélation positive) indique que les deux variables varient ensemble dans le même sens. Voici des exemples (figure III-6) illustrant les 3 situations :

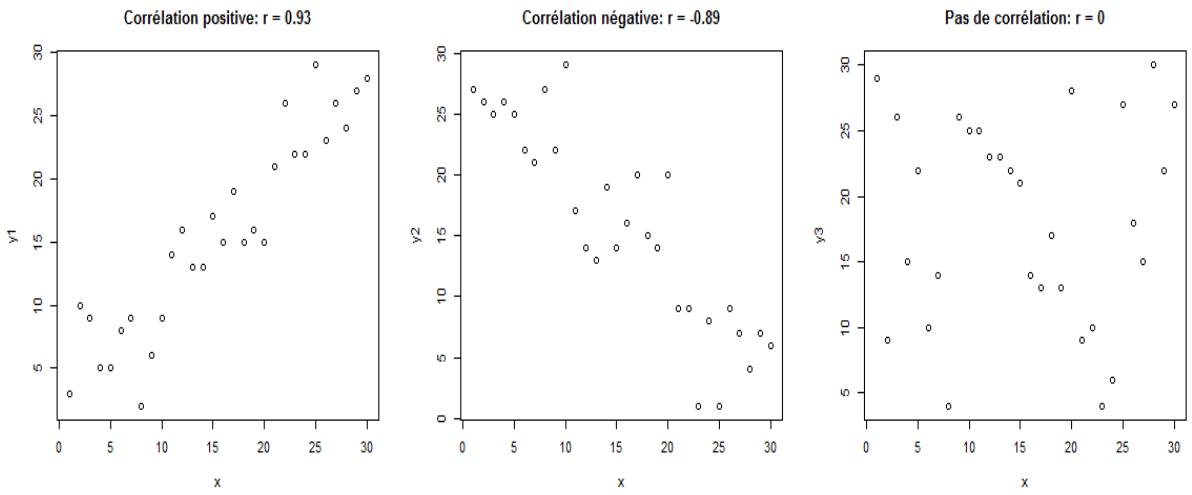

Figure III-6 : Exemples de corrélations de Pearson [83]

La corrélation de Pearson se mesure à l'aide de la formule suivante :

$$r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\ \sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}}$$

Équation 18 : Coefficient de corrélation de Pearson [30]

III.4.3.4 L'erreur de prédiction

Pourvu qu'un modèle ne soit jamais parfait, nous allons après chaque phase de test avec un modèle calculer le taux d'erreur de prédiction. Ce calcul n'est possible qu'après avoir répertorié toutes les valeurs obtenues après la phase de test de notre modèle c'est-à-dire prédiction de valeurs inconnues par le modèle. Pour se faire beaucoup de techniques existent dont **l'erreur quadratique moyenne** (Mean Squared Error en anglais) donnée par la formule suivante.

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Équation 19 : Erreur Quadratique Moyenne

Avec y_i la valeur prédite, $\hat{y}$ la valeur réelle attendue et le nombre de valeurs prédites.

Elle est calculée en prenant la moyenne, en particulier la moyenne des erreurs au carrée des données.

Plus la EQM est grande plus les données sont dispersées par rapport à la moyenne c'est à dire que les prédictions sont imprécises. Une EQM petite indique que les données sont étroitement dispersées par apport à la moyenne c'est-à-dire les prédictions sont mieux précises.

D'autres part nous avons la mesure de l'efficacité du modèle (**accuracy**) aussi appelée précision qui est un peu similaire à la EQM et permet de mesurer l'efficacité d'un modèle à prédire correctement à la fois les individus positifs et négatifs.

C'est une métrique pour évaluer les performances des modèles de classification à 2 classes ou plus. Comme toutes les métriques elle se base sur la matrice de confusion présentée plus haut et permet de mesurer le taux de prédictions correctes sur l'ensemble des individus.

Sa formule est donnée par [84]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Équation 20 : Taux d'accuracy

III.5 Conclusion

Dans ce chapitre, nous avons présenté notre approche de calcul de similarité sémantique entre paires de phrases basée sur l'apprentissage automatique supervisé. Les différentes étapes de cette approche sont décrites : le prétraitement des données, le calcul des mesures de similarité qui représentent les attributs, la sélection d'attributs, la construction et l'évaluation des modèles.

Dans le prochain chapitre, nous allons exposer les expérimentations réalisées avec cette approche sur des jeux de données standards.

Chapitre IV : Expérimentation de notre approche de calcul de similarité sémantique

IV.1 Introduction

Dans le chapitre précédant (Chapitre III) nous avons présenté l'approche que nous proposons pour le calcul de similarité sémantique entre paires de phrases. Dans ce présent chapitre, il sera question d'expérimenter (mettre en pratique) notre approche en utilisant des jeux de données standards.

Pour rappel notre approche se base sur des algorithmes d'apprentissage automatique supervisée. Ainsi il sera nécessaire d'utiliser des outils appropriés pour mener à bien cette tâche d'expérimentation. Pour se faire, nous avons utilisé le langage **python** avec sa bibliothèque **sklearn** qui implémente tous les algorithmes que nous utilisons dans notre travail.

Pour ce présent travail, nous avons fait une étude comparative des résultats de cinq (05) algorithmes: la régression linéaire, les arbres de décision, les forêts aléatoires, le perceptron multicouche et les machines à vecteur de support. Le choix sur ces derniers est qu'ils sont très connus dans la littérature et simples à utiliser.

Ceci étant, on va dans un premier temps procéder à la présentation des données qui seront utilisées dans ce travail, ensuite nous allons présenter les outils utilisés pour enfin terminer avec une analyse et une discussion sur les résultats de nos méthodes.

IV.2 Présentation des données utilisées

Les données sur lesquelles nous avons réalisé nos expérimentations sont issues de l'édition 2020 du défi fouille de texte (DEFT 2020) qui est une campagne d'évaluation visant à promouvoir le développement de méthodes et d'applications dans le domaine du traitement automatiques des langues (TAL) [28]. Elles concernent l'analyse des cas cliniques rédigés en langues française. Les données sont représentées en formant XML par paires de phrases et pour chaque paire, une phrase source et une phrase cible. Elles sont composées d'un jeu de données d'entrainement et d'un jeu de données de tests.

Le jeu de données d'entrainement est un corpus constitué de 600 paires de phrases avec pour chaque paire son score de similarité. Chaque paire de phrases est annotée manuellement avec un score indiquant le degré de similarité des phrases. Le score pour chaque paire est représenté sous forme d'attribut de la paire concernée. Pour chaque paire, on a un attribut « **id** » qui correspond à son numéro dans le fichier XML. Elles sont annotées indépendamment par deux experts qui attribuent des scores de similarité entre les paires de phrases allant de 0 (complètement différentes) à 5 (sémantiquement équivalentes). La figure ci-dessous (figure

IV.1) donne un histogramme qui montre les proportions de paires de phrases sur les cinq niveaux de vote sur notre ensemble de données.

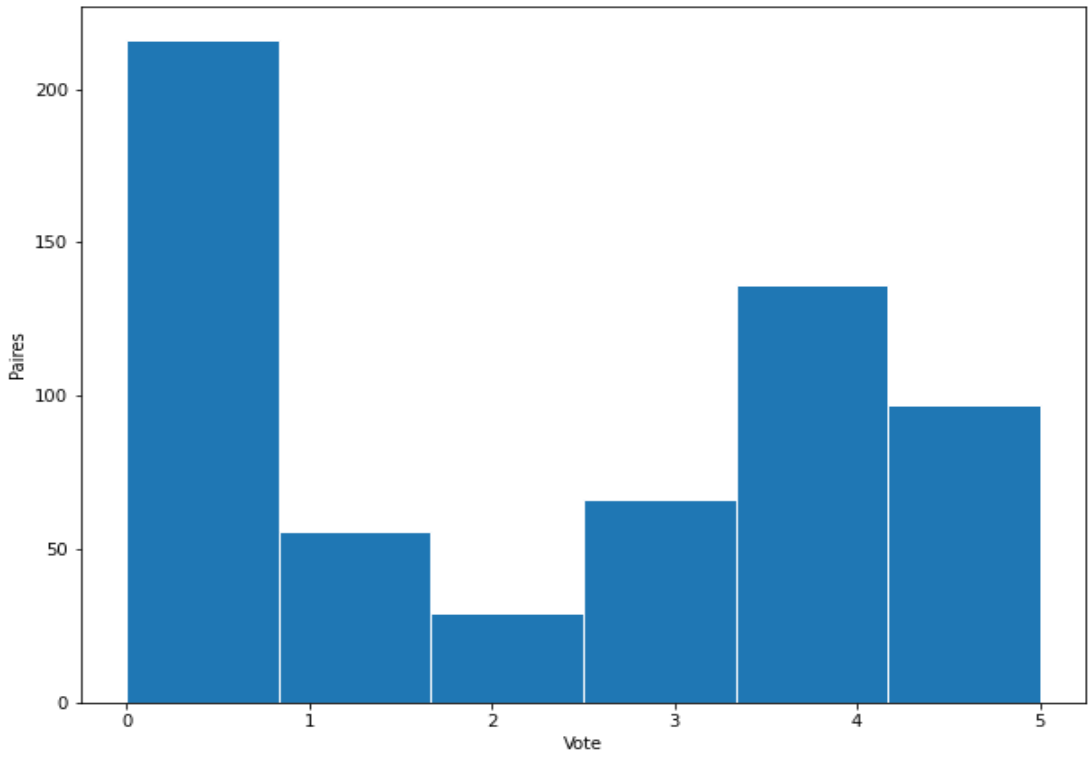

Figure IV-1 : Proportions de paires de phrases sur les cinq niveaux de vote

L'histogramme montre que dans notre ensemble de données, plus de 200 paires de phrases ont un score de similarité égal à 0 c'est à dire sont complétement différentes. De l'autre côté, on note que notre ensemble possède plus de 125 paires ayant un score égal à 4 c'est-à-dire quasiment identiques et presque 100 paires seulement sont exactement identiques qui correspondent à un score de 5.

Le corpus de test quant à lui est composé de 410 paires de phrases. Les captures ci-après (capture IV.1 et IV.2) donnent respectivement un extrait du corpus d'entrainement et celui de test [28].

```
</paire>
- <paire vote="5" id="3">
    <source>- En outre, la patiente sera avertie de la nécessité d'une consultation rapide devant tout saignement vaginal anormal.</source>
    <cible>En cas de saignement vaginal anormal, il importe de consulter au plus vite votre médecin.</cible>
</paire>
- <paire vote="4" id="4">
    <source>- L'administration d'huile de paraffine chez les jeunes enfants, les personnes débilitées, allongées ou ayant des difficultés de déglutition doit être prudente
      en raison du risque d'inhalation bronchique et de pneumopathie lipoïde.</source>
    <cible>- L'administration d'huile de paraffine chez le jeune enfant, les malades alités ou ayant des difficultés pour avaler doit être prudente car elle peut être à
      l'origine de passage dans les bronches et de troubles respiratoires.</cible>
</paire>
- <paire vote="5" id="5">
    <source>- Ne pas utiliser chez les personnes présentant des difficultés de déglutition en raison du risque d'inhalation bronchique et de pneumopathie
      lipoïde.</source>
    <cible>- Ne pas utiliser chez les personnes présentant des difficultés pour avaler en raison du risque d'inhalation de la paraffine liquide qui entraîne une
      pneumopathie lipoïde.</cible>
</paire>
- <paire vote="4" id="6">
    <source>- Refermez le flacon immédiatement après utilisation.</source>
    <cible>Refermez l'embout du flacon avec le bouchon immédiatement après utilisation.</cible>
</paire>
- <paire vote="4" id="7">
    <source>« Autant que possible, on choisit des biberons en verre ; ceux en métal, en bois ou en caoutchouc, contenant toujours à la longue une odeur plus ou moins
      forte, plus ou moins fétide, désagréable, repoussante pour l'enfant.</source>
    <cible>En effet, l'utilisation des biberons en bois, en métal ou encore en caoutchouc dégageait une mauvaise odeur qui faisait que l'enfant le refusait.</cible>
</paire>
```

Capture IV-1 : Extrait du jeu de données d'entrainement

```
<?xml version="1.0" encoding="UTF-8"?>
<doc>
- <paire vote="" id="1">
    <source>- Chez les patients présentant une absence complète d'activité de la dihydro-pyrimidine-déshydrogénase (DPD).</source>
    <cible>- Si vous savez que vous n'avez aucune activité de l'enzyme dihydropyrimidine déshydrogénase (DPD).</cible>
</paire>
- <paire vote="" id="2">
    <source>- Eviter le contact de l'embout avec l'œil ou les paupières.</source>
    <cible>Evitez le contact de l'embout du flacon avec l'œil ou les paupières.</cible>
</paire>
- <paire vote="" id="3">
    <source>- L'utilisation prolongée de l'huile de paraffine est susceptible de réduire l'absorption des vitamines liposolubles (A, D, E, K).</source>
    <cible>- L'utilisation prolongée de l'huile de paraffine est susceptible de diminuer l'absorption de certaines vitamines.</cible>
</paire>
- <paire vote="" id="4">
    <source>- Ne pas dépasser la posologie recommandée.</source>
    <cible>Ne dépassez pas la posologie recommandée.</cible>
</paire>
- <paire vote="" id="5">
    <source>- Refermez le flacon immédiatement après utilisation.</source>
    <cible>Refermez soigneusement le flacon après utilisation.</cible>
</paire>
- <paire vote="" id="6">
    <source>- Se laver soigneusement les mains avant de pratiquer l'instillation.</source>
    <cible>Lavez-vous soigneusement les mains avant de procéder à l'instillation.</cible>
</paire>
```

Capture IV-2 : Extrait du jeu de données de test

IV.3 Présentation des outils utilisés

Les outils dont nous feront usage dans la suite pour cette expérimentation seront présentés dans cette partie. Parmi ces outils on peut citer le langage python et l'éditeur spider.

IV.3.1 Le Langage python

Python est un langage de programmation open source créé par le programmeur **GUIDO VAN ROSSUM** en **1991**. C'est le langage le plus utilisé dans le domaine du machine learning, du Big Data et de la Data Science.

En tant que langage de programmation de haut niveau, python permet aux programmeurs de se focaliser sur ce qu'ils font plutôt que la façon dont ils le font. Ainsi écrire des programmes prend moins de temps que dans un autre langage.

Le langage Python doit sa popularité à plusieurs avantages qui profitent aussi bien aux débutants qu'aux experts. Tout d'abord, il est facile à apprendre et à utiliser. Ses caractéristiques sont peu nombreuses, ce qui permet de créer des programmes rapidement et avec peu d'efforts. De plus, sa syntaxe est conçue pour être lisible et directe [85].

La grande majorité des bibliothèques utilisées pour la science des données ou le machine learning ont des interfaces Python. Ainsi, ce langage est devenu l'interface de commande de haut niveau la plus populaire pour les bibliothèques de machine learning et du traitement de textes. Ces nombreuses bibliothèques telles que **Numpy**, **Pandas**, **Sklearn**, **NLTK**, **Spacy** permettent de réaliser une variété de tâches aussi bien en machine learning qu'en traitement de textes.

Pour notre travail, python constitue le langage principal qu'on va utiliser aussi bien pour le traitement de textes que pour la phase d'apprentissage. Son utilisation se fera via l'environnement de développement intégré (IDE) **Spyder** que nous présenterons plus tard.

IV.3.1.1 Coup d'œil sur quelques bibliothèques de python

Dans cette section nous présentons quelques librairies python indispensables en TAL et en ML dont parmi lesquelles nous avons Numpy, Pandas et sklearn

IV.3.1.1.1 La bibliothèque Numpy

Numpy est une bibliothèque du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux. Plus précisément, cette bibliothèque libre et open source fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes [86].

IV.3.1.1.2 La bibliothèque Pandas

Pandas est une bibliothèque du langage de programmation Python permettant la manipulation et l'analyse des données. Elle propose en particulier des structures de données et des opérations de manipulation de tableaux numériques et de séries temporelles [87]. C'est l'une des bibliothèques de science des données les plus populaires. Elle a été développée par des Data Scientists habitués au R et au Python, et est aujourd'hui utilisée par un grand nombre de scientifiques et d'analystes. Elle offre de nombreuses fonctionnalités natives très utiles. Il est notamment possible de lire des données en provenance de nombreuses sources, de créer de larges data frames à partir de ces sources et d'effectuer des analyses agrégées basées sur les questions auxquelles on souhaite obtenir des réponses. Des fonctionnalités de visualisation permettent également de générer des graphiques à partir des résultats des analyses, ou de les exporter au format Excel. On peut aussi s'en servir pour la manipulation de tableaux numériques et de séries temporelles [85].

Les tableaux utilisés pour présenter nos résultats découleront en grande partie de cette bibliothèque.

IV.3.1.1.3 La bibliothèque Sklearn

Sklearn est une bibliothèque python destinée à l'apprentissage automatique. Elle propose un Framework de nombreuses bibliothèques d'algorithmes à implémenter. Elle implémente tous les algorithmes dont nous ferons usage et elle est écrite en python avec quelques algorithmes en cython[24] pour optimiser les performances.

C'est la librairie la plus puissante et la plus robuste pour l'apprentissage automatique en python. Elle fournit un ensemble d'outils efficaces aussi bien pour l'apprentissage machine que pour la modélisation statistique, notamment la classification, la régression et le clustering[25] via une interface cohérente de python. Elle s'appuie sur Numpy, Scipy et Matplotlib [88]**.**

IV.3.1.1.4 La bibliothèque Matplotlib

Matplotlib est une librairie du langage de programmation python utilisée pour visualiser les données sous formes graphique. Elle peut être combinée avec Numpy et Scipy [89].Tous les graphiques que nous allons tracer dans la suite découleront de cette bibliothèque.

Pour la réalisation de cette expérimentation avec le langage python qui offre une bonne sélection de bibliothèques nous permettant de mener à bien notre travail, nous allons utiliser

[24] Un langage sur-ensemble du langage Python qui prend également en charge l'appel de fonctions C et la déclaration de types C sur les variables et les attributs de classe.

[25] Le clustering est une discipline particulière du Machine Learning ayant pour objectif de séparer des données en groupes homogènes ayant des caractéristiques communes. L'algorithme des K-moyennes (K-means) est un algorithme non supervisé très connu en matière de Clustering.

l'environnement de développement intégré (IDE en anglais pour développement Integrated Development Environment) **Spyder**.

IV.3.2 L'environnement Spyder

Spyder est l'acronyme de « Scientific Python Development EnviRonment ». Spyder est donc plutôt orienté calcul scientifique, bien qu'on puisse développer n'importe quel type d'applications. Spyder est un projet « open Source » et peut être téléchargé gratuitement. Il peut être installé directement sur Windows à partir du web ou via **Anaconda** [90]**.**

Anaconda est une distribution de python qui est orienté « data Science » (Sciences de données) et « machine Learning » (apprentissage automatique).

Elle regroupe un ensemble d'outils gravitant autour du langage de programmation python et R. Elle fournit notamment plusieurs environnements d'exécution.

Avec la distribution anaconda, l'utilisation de python devient plus simple dans la mesure où il fournit en amont toutes les bibliothèques nécessaires dont on a besoins pour faire du traitement de textes et de l'apprentissage automatique. Ainsi on enlève au programmeur la tâche d'installation de bibliothèques. Ceci justifie le choix de cet éditeur et cette façon de travailler avec python. En effet le langage python apporte une aide précieuse pour la réalisation de certaines tâches.

IV.4 Expérimentation de la méthode

Nous réalisons dans cette partie l'expérimentation de notre approche en utilisant un jeu de données stansard. Les données présentées dans la partie « présentation des données » de cette section sont de type textuel donc ne peuvent pas être traitées directement pour notre tâche d'expérimentation. Ainsi, nous nous servons de ces dernières pour construire un dataset bien structuré avec lequel nous réaliserons notre travail.

On rappelle que le corpus sur le quel portera cette expérimentation est celui obtenu après avoir réalisé la tâche de calcul de similarité textuelle en utilisant les mesures de similarité textuelle décrites dans le chapitre 2. Ces données ont été présentées dans la section serons chargées.

IV.4.1 Le chargement des données

Avant toute chose, on procède d'abord au chargement des données. Après le calcul de similarité textuelle sur chaque corpus (corpus d'entrainement et corpus de test), les données ont été regroupées dans un data frame Pandas où les colonnes représentent les attributs et les lignes les enregistrements correspondants aux valeurs obtenues en appliquant toutes les mesures sur chaque paire de phrases de notre corpus. Après le calcul de la similarité textuelle avec python en utilisant toutes les mesures, le résultat obtenu est exporté en format CSV grâce à la librairie Pandas. Les figures ci-dessous (figure IV.4 et figure IV.5) donnent respectivement un extrait du résultat obtenu après chargement de nos données obtenues après le calcul de similarité textuelle avec les données d'entrainement et de teste.

Tableau IV-1 : Extrait données d'entrainement après chargement

	Dice	Jaccard	Ochiai	Cosinus	Overlap	Un_Gram	...	Vote
0	0.727273	0.571429	0.739600	0.786214	0.888889	0.278689	...	4
1	0.888889	0.800000	0.888889	0.928571	0.888889	0.043478	...	5
2	0.380952	0.235294	0.384900	0.333849	0.444444	0.684211	...	5
3	0.526316	0.357143	0.526316	0.620633	0.526316	0.441964	...	4
4	0.769231	0.625000	0.771517	0.750652	0.833333	0.255952	...	5
...	...	...	...	...	...	...	...	...
595	0.051282	0.026316	0.054393	0.096225	0.076923	0.699659	...	0
596	0.137931	0.074074	0.138675	0.147442	0.153846	0.734694	...	0
597	0.000000	0.000000	0.000000	0.000000	0.000000	0.711957	...	0
598	0.400000	0.250000	0.408248	0.421637	0.500000	0.542373	...	2
599	0.000000	0.000000	0.000000	0.271448	0.000000	0.722581	...	0

Le tableau IV.1 représente un extrait des données d'entrainement. Comme le montre la figure, le dataset d'entrainement contient au total 600 enregistrements (lignes) obtenus en utilisant 17 mesures de similarité correspondant aux attributs de notre dataset. La colonne vote représente la cible ou Target (valeur à prédire). Elle a été renseignée par des experts du domaine.

Tableau IV-2 : Extrait données de test après chargement

	Dice	Jaccard	Ochiai	Cosinus	Overlap	Un_Gram
0	0.000000	0.000000	0.000000	0.149071	0.000000	0.441176
1	0.666667	0.500000	0.670820	0.821584	0.750000	0.184615
2	0.600000	0.428571	0.612372	0.800641	0.750000	0.241667
3	0.666667	0.500000	0.666667	0.833333	0.666667	0.243902
4	0.750000	0.600000	0.750000	0.833333	0.750000	0.470588
...	...	...	...	...	...	...
405	0.214286	0.120000	0.214834	0.123091	0.230769	0.687500
406	0.166667	0.090909	0.176777	0.136083	0.250000	0.696429
407	0.260870	0.150000	0.263117	0.448276	0.300000	0.625000
408	0.818182	0.692308	0.832050	0.820610	1.000000	0.355556
409	0.000000	0.000000	0.000000	0.302079	0.000000	0.709091

Cette figure correspond aux données de test après chargement. De la même manière que celles d'entrainement, elles sont obtenues en utilisant les 17 mesures de similarité mais la différence est que cet ensemble de test est plus court et ne contient que 410 enregistrements.

IV.4.2 La sélection d'attributs

Dans cette partie, nous allons essayer de trouver et de sélectionner les variables ou attributs les plus utiles de notre ensemble de données pour l'entrainement de nos modèles de machine learning. Pour se faire, nous utilisons une des méthodes de sélections d'attributs basées sur le filtrage. Ainsi nous utilisons le test de chi-deux (X^2) décrit dans le chapitre précédant.

Le test de chi-deux se base sur les tests de dépendance et permet de sélectionner les variables qui ont les scores de dépendance les plus élevés avec la variable cible c'est-à-dire les variables sur les quelles dépend plus la variable cible (vote). Après le calcul du score, on utilise un **transformer** pour réaliser la sélection.

Le module **Feature_selection** de la librairie sklearn de python contient tous les **Transformers** et toutes les fonctions de test de dépendance. En l'occurrence le **chi-deux** y est et nous allons faire une combinaison avec un **transformer** très célèbre nommé **SelectKBest** (pour sélectionner les K meilleures variables) pour réaliser cette tâche. Le tableau ci-après donne l'ensemble des attributs de notre ensemble de données avant sélection. Comme décrit plus haut, ils sont au nombre de 17.

Tableau IV-3 : Ensemble d'attributs avant sélection

Numéro			Libelle		
1	2	3	Dice	Jaccard	Ochiai
4	5	6	Cosinus	Overlap	Un_Gram
7	8	9	Un_Gram_Dice	Deux_Gram	Deux_Gram_Dice
10	11	12	Trois_Gram	Trois_Gram_Dice	Quatre_Gram
13	14	15	Quatre_Gram_Dice	Cinq_Gram	Cinq_Gram_Dice
16		17	Levenshtein		Manhanttan

Le tableau représente l'ensemble des attributs de notre ensemble de données. Ils y sont rangés par ordre d'apparition dans notre dataset. L'attribut Dice correspondant à la similarité de Dice étant le premier et Manhanttan correspondant à la distance de Manhanttan le dernier attribut. On signale que pour le calcul de similarité on a eu à combiner la similarité de Dice avec celle de **Ngram** où N est remplacé dans le tableau par un entier compris entre 1 et 5 et cette mesure est nommée similarité **N_gram_Dice**.

Par le biais du test de chi2, nous allons dans ce tableau sélectionner les variables avec lesquelles on va continuer ce travail. Pour se faire nous allons dans un premier temps calculer les scores

de dépendance entre nos variables et la cible. Le tableau ci-dessous donne les scores de dépendance entre nos variables.

Tableau IV-4 : Les différents scores de dépendance entre nos variables et la cible

Variables	Vote
Dice	95.01799768
Jaccard	91.18399945
Ochiai	94.03701673
Cosinus	63.32240764
Overlap	92.98497543
Un_Gram	21.58475133
Un_Gram_Dice	2.84441604
Deux_Gram	22.28002441
Deux_Gram_Dice	30.52539061
Trois_Gram	22.6466103
Trois_Gram_Dice	70.41068873
Quatre_Gram	22.87775158
Quatre_Gram_Dice	85.67971603
Cinq_Gram	22.99249878
Cinq_Gram_Dice	91.45558546
Levenshtein	20.29393786
Manhanttan	27.1310095

Les scores obtenus donnent une idée sur les attributs ou variables qui seront sélectionnées. Dans la suite, nous allons avec le test de chi2 sélectionner pour chaque algorithme l'ensemble minimal de variables les plus importantes et qui donnent les meilleures performances pour le modèle concerné. Ainsi, les ensembles de données d'entrainement qui seront considérés ne seront pas forcément les mêmes.

Le principe abordé pour réaliser une telle tâche est de sélectionner toutes nos variables (K=17 variables) dans un premier temps. Ensuite on réalise l'apprentissage automatique sur cet ensemble en utilisant tous les algorithmes et ainsi on note les scores obtenus pour chacun d'eux. On continue à réaliser cette tâche en faisant varier la valeur de K et à chaque fois on note les

nouveaux scores obtenus s'ils donnent les meilleures performances pour nos modèles. Le tableau ci-dessous (tableau IV.5) donne les résultats obtenus après sélection pour chaque algorithme avec en amont la valeur de K qui donne avec la métrique d'exactitude (accuracy score) le meilleure score pour le modèle concerné.

Tableau IV-5 : scores des modèles après sélection d'attributs

Modèles	Meilleur score du modèle	K
Régression logistique	0.5853658536585366	13
Arbres de décisions	0.4975609756097561	12
Random Forest	0.5682926829268292	17
Perceptron Multicouche	0.5902439024390244	13
SVM	0.5780487804878048	16

Comme le montre le tableau, on a quatre (04) ensembles d'attributs qui, combinés, donnent les meilleures performances pour nos modèles. Les tableaux qui suivent donnent la liste par ordre d'apparition dans le dataset des attributs sélectionnés pour chaque algorithme.

- Pour la régression logistique et le perceptron Multicouche, les 13 meilleurs attributs qui ont été sélectionnés sont données dans le tableau suivant.

Tableau IV-6 : Ensemble de données sélectionnées pour les modèles de régression logistique et de perceptron multicouche

Dice	Jaccard	Ochiai	Cosinus
Overlap	Deux_Gram_Dice	Trois_gram	Trois_gram_Dice
Quatre_Gram	Quatre_Gram_Dice	Cinq_Gram	Cinq_Gram_Dice
Manhanttan			

- Pour l'algorithme des arbres de décision, les 12 meilleurs attributs qui ont été sélectionnés sont données dans le tableau suivant.

Tableau IV-7 : Ensemble de données sélectionné pour le modèle des arbres de décisions

Dice	Jaccard	Ochiai	Cosinus
Overlap	Deux_Gram_Dice	Trois_gram_Dice	Quare_Gram
Quatre_Gram_Dice	Cinq_Gram	Cinq_Gram_Dice	Manhanttan

- Avec l'algorithme de Random Forest, la combinaison de toutes les variables de notre dataset a donné les meilleures performances.
- Avec celui des machines à vecteur de support, les 16 meilleures variables ont été sélectionnés, donc seulement la variable Un_Gram_Dice n'a pas été sélectionnée et obtient le score de dépendance le moyen élevé.

Ainsi, dans la suite de notre travail on va continuer avec les attributs qui ont été sélectionnés pour chaque algorithme.

IV.4.3 Construction et évaluation dse modèles de machine Learning

Dans cette partie, nous allons créer des modèles de machine learning pour chaque algorithme. En utilisant les données obtenues après sélection, nous allons entrainer et tester nos modèles avec respectivement les données d'entrainement et de test. Pour chaque modèle on effectuera une évaluation en se basant sur la corrélation de Pearson, la corrélation de Spearman, l'exactitude (accuracy) et le taux d'erreur effectué pour choisir le meilleur modèle pour nos futures prédictions.

Apres construction, entrainement et test des modèles, on réalise quelques opérations pour s'assurer de leur qualité. Ainsi, en utilisant la corrélation de Pearson, la corrélation de Spearman, l'accuracy et l'EQM (Erreur Quadratique moyenne) avec chaque modèle sur les données obtenues et celles attendues, on réalise une évaluation de chacun des modèles. Le tableau ci-dessous (Tableau IV.13) donne les résultats obtenus pour l'évaluation.

Tableau IV-8 : Evaluation des modèles obtenus

Modèles	**Corrélation de Pearson**	**Corrélation de Spearman**	**Accuracy**	**EQM**
Régression Logistique	0.73	0.74	0.58	2.3
Arbre de décision	0.58	0.58	0.42	3.28
Forets aléatoires	0.77	0.76	0.56	1.89
Perceptron Multicouche	0.75	0.75	0.59	2.17
SVM	0.73	0.74	0.57	2.30

IV.4.4 Discussion des résultats obtenus

L'évaluation de nos différents modèles d'apprentissage automatique sur des jeux de données standards montre la bonne performance des algorithmes classiques d'apprentissage automatique pour réaliser la tâche de calcul de similarité sémantique entre paires de phrases.

Ainsi, nos modèles donnent des prédictions correctes pour la plupart des paires de phrases des données utilisées. Une évaluation de tous nos modèles a permis de constater que le perceptron multicouche donne les meilleures performances sur notre jeu de données.

Le tableau III-13 donne un aperçu sur les modèles qui donnent les meilleures prédictions. Ainsi, comme le montre notre tableau, le modèle de perceptron multicouche fait moins d'erreurs et obtient le taux d'exactitude le plus grand (59%).

IV.5 Conclusion

La réalisation de cette expérimentation nous a permis de transformer nos données qui étaient initialement de types textuels à un dataset structuré pour faire de l'apprentissage automatique. Ainsi, avec les données cliniques, nous avons pu réaliser de l'apprentissage machine supervisé en utilisant différents algorithmes.

En définitive, l'expérimentation sur les données mises à notre disposition constitue une porte pour opérer différents traitements sur ces dernières : la sélection des attributs qui est une tâche sous-jacente du prétraitement en machine learning et qui nous procure beaucoup d'informations sur leur comportement, la construction et l'évaluation de modèles d'apprentissage automatique.

Conclusion générale et perspectives

Le calcul de similarité est une tâche importante dans le traitement automatique des langues (TAL). Les langues naturelles n'étant pas structurées, et du fait de leur apport indispensable pour l'être humain, faire recours aux méthodes utilisant le TAL permet de gagner du temps.

Le TAL utilise des méthodes se basant sur des principes modernes telles que le calcul de la similarité qui intervient dans beaucoup d'applications telles que la recherche d'information sur le web, la traduction automatique de texte, etc.

Il a été question dans ce travail, après avoir fait focus sur le domaine du TAL qui couvre notre champ de recherche, de faire un état de l'art sur les approches de calcul similarité sémantique et de proposer une approche performante. L'état de l'art réalisé dans la deuxième partie de ce mémoire nous a permis de voir quelques études qui ont été réalisées dans ce domaine. En plus, il nous a permis de comprendre la différence entre ces différentes approches qui existent et leurs limites. Cette étude nous a permis de faire dans une troisième partie une proposition d'approche de calcul de similarité sémantique. L'approche proposée dans le chapitre III est basée sur des algorithmes d'apprentissage automatique supervisée.

Des méthodes de calcul de similarités textuelles sur notre ensemble de données textuelles sont utilisées afin d'obtenir un dataset prêt à être utilisé pour faire du machine learning. Un ensemble de 17 mesures de similarités textuelles est utilisé pour expérimenter notre méthode. L'expérimentation nous a permis de créer des modèles avec des algorithmes classiques d'apprentissage automatique, de les tester et d'évaluer leurs performances.

Bien qu'ils permettent de faire des prédictions correctes dans la plupart des cas sur des données nouvelles, les modèles obtenus sont à améliorer. Ceci peut être expliqué par le fait que le dataset obtenu est constitué en utilisant en grande partie des mesures de similarité textuelle (similarité syntaxique).

Ainsi, on se fixe comme perspectives d'augmenter les performances de nos modèles en se focalisant davantage sur des mesures de similarité sémantique.

Références

[1] « Sémantique », Wikipédia. 21 février 2022. Consulté le: 5 avril 2022. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=S%C3%A9mantique&oldid=191125702

[2] « Similitude sémantique ». http://stringfixer.com/fr/Semantic_relatedness (consulté le 5 avril 2022).

[3] « Traitement automatique du langage (TAL) : qu'est-ce que c'est ? », Inbenta, 7 juillet 2014. https://www.inbenta.com/fr/blog/quest-ce-que-le-tal/ (consulté le 1 décembre 2021).

[4] « Boîte à outils en langage naturel - Introduction ». https://www.hebergementwebs.com/tutoriel-sur-la-boite-a-outils-en-langage-naturel/boite-a-outils-en-langage-naturel-introduction (consulté le 27 mars 2021).

[5] « Traitement automatique des langues », Wikipédia. 30 août 2021. Consulté le: 10 novembre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Traitement_automatique_des_langues&oldid=185923476

[6] « Traitement automatique des langues - Cours - Université du Québec à Chicoutimi ». https://cours.uqac.ca/7INF517 (consulté le 11 novembre 2021).

[7] « Qu'est-ce que le Traitement Automatique du Langage ? », Inbenta, 7 juillet 2014. https://www.inbenta.com/fr/blog/quest-ce-que-le-tal/ (consulté le 11 novembre 2021).

[8] M. Fabien, « Traitement Automatique du Langage Naturel en français (TAL / NLP) », Stat4decision, 3 novembre 2019. https://www.stat4decision.com/fr/traitement-langage-naturel-francais-tal-nlp/ (consulté le 11 novembre 2021).

[9] « Parlez-vous NLP ? », Sqli digital experience, 26 mars 2019. https://www.sqli-digital-experience.com/blog-fr/parlez-vous-nlp (consulté le 2 décembre 2021).

[10] « Natural Langauage Processing (NLP), c'est quoi ? » https://www.sas.com/fr_ca/insights/analytics/what-is-natural-language-processing-nlp.html (consulté le 2 décembre 2021).

[11] « La traduction automatique, mère des activités TAL », MasterTSM@Lille, 11 avril 2021. https://mastertsmlille.wordpress.com/2021/04/11/la-traduction-automatique-mere-des-activites-tal/ (consulté le 1 décembre 2021).

[12] « La traduction automatique statistique, comment ça marche ? » https://interstices.info/la-traduction-automatique-statistique-comment-ca-marche/ (consulté le 1 décembre 2021).

[13] C. Jacquet-Pfau, « Abstract », Revue francaise de linguistique appliquee, no 2, p. 81-94, 2001.

[14] M. Gagnon et F. Farley-Chevrier, « Chapitre 4. La recherche par Internet », in Guide de la recherche documentaire, Montréal: Presses de l'Université de Montréal, 2018, p. 71-93. Consulté le: 1 décembre 2021. [En ligne]. Disponible sur: http://books.openedition.org/pum/14218

[15] « Recherche d'information », Wikipédia. 16 octobre 2021. Consulté le: 1 décembre 2021. [En ligne]. Disponible sur:

https://fr.wikipedia.org/w/index.php?title=Recherche_d%27information&oldid=187192748
[16] lcollet, « Text Mining : le résumé automatique de contenus », Blog Digital, 14 avril 2021. https://blogdigital.beijaflore.com/text-mining-le-resume-automatique-de-contenus/ (consulté le 1 décembre 2021).
[17] E. A. IA, « Atelier NLP : Construire un système de Question Answering - Intelligence artificielle », Actu IA. https://www.actuia.com/evenements/atelier-nlp-construire-un-systeme-de-question-answering/ (consulté le 1 décembre 2021).
[18] « API de réponse aux questions ». https://nlpcloud.io/fr/nlp-question-answering-api.html (consulté le 1 décembre 2021).
[19] lcollet, « Text Mining : l'analyse de sentiments », Blog Digital, 14 avril 2021. https://blogdigital.beijaflore.com/text-mining-analyse-de-sentiments/ (consulté le 1 décembre 2021).
[20] « Introduction au Natural Language Toolkit (NLTK) », Code Envato Tuts+. https://code.tutsplus.com/fr/tutorials/introducing-the-natural-language-toolkit-nltk--cms-28620 (consulté le 27 mars 2021).
[21] « Boîte à outils en langage naturel - documentation NLTK 3.5 ». https://www.nltk.org/ (consulté le 27 mars 2021).
[22] « What is Tokenization | Tokenization In NLP », Analytics Vidhya, 25 mai 2020. https://www.analyticsvidhya.com/blog/2020/05/what-is-tokenization-nlp/ (consulté le 15 novembre 2021).
[23] « Nettoyez et normalisez les données », OpenClassrooms. https://openclassrooms.com/fr/courses/4470541-analysez-vos-donnees-textuelles/4854971-nettoyez-et-normalisez-les-donnees (consulté le 2 décembre 2021).
[24] « spaCy : la bibliothèque Python Open Source de NLP », Formation Data Science | DataScientest.com, 14 juin 2021. https://datascientest.com/spacy (consulté le 3 décembre 2021).
[25] A. Fardeen, « Tutorial on Spacy Part of Speech (POS) Tagging », MLK - Machine Learning Knowledge, 12 août 2021. https://machinelearningknowledge.ai/tutorial-on-spacy-part-of-speech-pos-tagging/ (consulté le 3 décembre 2021).
[26] H.-H. Vu, J. Villaneau, F. Saïd, et P.-F. Marteau, « Mesurer la similarité entre phrases grâce à Wikipédia en utilisant une indexation aléatoire », in TALN 2015, Caen, France, juin 2015. Consulté le: 23 février 2021. [En ligne]. Disponible sur: https://hal.archives-ouvertes.fr/hal-01167929
[27] R. Cardon et N. Grabar, « Détection automatique de phrases parallèles dans un corpus biomédical comparable technique/simplifié », in TALN 2019, Toulouse, France, juill. 2019. Consulté le: 24 février 2021. [En ligne]. Disponible sur: https://hal.archives-ouvertes.fr/hal-02430446
[28] K. Dramé, G. Sambe, I. Diop, et L. Faty, « Approche supervisée de calcul de similarité sémantique entre paires de phrases (Supervised approach to compute semantic similarity between sentence pairs) », in Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Atelier DÉfi Fouille de Textes, Nancy, France, 2020, p. 49-54. Consulté le: 23 février 2021. [En ligne]. Disponible sur: https://www.aclweb.org/anthology/2020.jeptalnrecital-deft.5

[29] « Similarité sémantique », Wikipédia. 10 septembre 2020. Consulté le: 4 mars 2021. [En ligne].Disponible sur: https://fr.wikipedia.org/w/index.php?title=Similarit%C3%A9_s%C3%A9mantique&oldid=174608064
[30] « Similarité sémantique à l'aide de transformateurs », ICHI.PRO. https://ichi.pro/fr/similarite-semantique-a-l-aide-de-transformateurs-157490910029526 (consulté le 24 novembre 2021).
[31] Z. Zhang et P. Zweigenbaum, « zNLP: Identifying Parallel Sentences in Chinese-English Comparable Corpora », in Proceedings of the 10th Workshop on Building and Using Comparable Corpora, Vancouver, Canada, août 2017, p. 51-55. doi: 10.18653/v1/W17-2510.
[32] « Five most popular similarity measures implementation in python », Dataaspirant, 11 avril 2015. https://dataaspirant.com/five-most-popular-similarity-measures-implementation-in-python/ (consulté le 24 novembre 2021).
[33] par E. Blurock, « Métriques de similarité de chaîne : méthodes de jeton | Baeldung sur l'informatique », 9 février 2021. https://www.baeldung.com/cs/string-similarity-token-methods (consulté le 26 novembre 2021).
[34] « Coefficient de Sørensen–Dice ». http://stringfixer.com/fr/Sorenson_index (consulté le 4 décembre 2021).
[35] « Cosine similarity », Wikipédia. 6 novembre 2021. Consulté le: 4 décembre 2021. [En ligne]. Disponible sur: https://en.wikipedia.org/w/index.php?title=Cosine_similarity&oldid=1053914647
[36] « Overlap Similarity - Neo4j Graph Data Science », Neo4j Graph Database Platform. https://neo4j.com/docs/graph-data-science/1.8/alpha-algorithms/overlap/ (consulté le 4 décembre 2021).
[37] « langage naturel - Quantifier la similitude des sacs de mots », Cross Validated. https://stats.stackexchange.com/questions/289400/quantify-the-similarity-of-bags-of-words (consulté le 6 mars 2021).
[38] D. D. Prasetya, A. Prasetya Wibawa, et T. Hirashima, « The performance of text similarity algorithms », Int. J. Adv. Intell. Informatics, vol. 4, no 1, p. 63, mars 2018, doi: 10.26555/ijain.v4i1.152.
[39] « Distance de Hamming : définition et explications », Techno-Science.net. https://www.techno-science.net/definition/3856.html (consulté le 29 janvier 2023).
[40] « Distance de Levenshtein », Wikipédia. 8 novembre 2021. Consulté le: 27 novembre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Distance_de_Levenshtein&oldid=187819344
[41] ИОУА, « Different types of Text Similarity Approaches », Medium, 2 décembre 2018. https://medium.com/@ayon_biswas/different-types-of-text-similarity-approaches-9b33f00bed09 (consulté le 27 novembre 2021).
[42] T. Yan, T. Maxwell, D. Song, Y. Hou, et P. Zhang, « Event-based hyperspace analogue to language for query expansion », présenté à ACL 2010 Conference Short Papers, Uppsala, Sweden, juill. 2010. Consulté le: 27 novembre 2021. [En ligne]. Disponible sur: http://dl.acm.org/citation.cfm?id=1858864
[43] « Qu'est-ce qu'une Analyse Sémantique Latente ? Définition et conseils ». https://www.1min30.com/dictionnaire-du-web/analyse-semantique-latente-seo (consulté le 27 novembre 2021).

[44] « matrice LSA – Recherche Google ». https://www.google.com/search?q=matrice+LSA&source=lnms&tbm=isch&sa=X&ved=2ahUKEwid7bGx38j0AhUZg_0HHbH3BJ8Q_AUoAXoECAEQAw&biw=1366&bih=696&dpr=1 (consulté le 3 décembre 2021).
[45] « Explicit semantic analysis », Wikipédia. 12 novembre 2021. Consulté le: 3 décembre 2021. [En ligne]. Disponible sur: https://en.wikipedia.org/w/index.php?title=Explicit_semantic_analysis&oldid=1054921158
[46] « Normalized Google distance », Wikipédia. 3 mai 2021. Consulté le: 27 novembre 2021. [En ligne]. Disponible sur: https://en.wikipedia.org/w/index.php?title=Normalized_Google_distance&oldid=1021228071
[47] D. M. Blei, A. Y. Ng, et M. I. Jordan, « Latent dirichlet allocation », J. Mach. Learn. Res., vol. 3, no null, p. 993-1022, mars 2003.
[48] H. Zargayouna et S. Salotti, « Mesure de similarité sémantique pour l'indexation de documents semi-structurés », janv. 2004.
[49] T. Slimani, B. Yaghlane, et K. Mellouli, « Une extension de mesure de similarité entre les concepts d'une ontologie », oct. 2022.
[50] « Word embedding », Data Analytics Post. https://dataanalyticspost.com/Lexique/word-embedding/ (consulté le 4 décembre 2021).
[51] « Réduction de dimensionnalité », Data Analytics Post. https://dataanalyticspost.com/Lexique/reduction-de-dimensionnalite/ (consulté le 4 décembre 2021).
[52] « Word2Vec », Data Analytics Post. https://dataanalyticspost.com/Lexique/word2vec/ (consulté le 4 décembre 2021).
[53] « Concepts mathématiques derrière le machine learning : la régression linéaire », Actu IA, 25 juin 2019. https://www.actuia.com/tutoriel/concepts-mathematiques-derriere-le-machine-learning-la-regression-lineaire/ (consulté le 22 janvier 2022).
[54] « Python | Régression linéaire à l'aide de sklearn – Acervo Lima ». https://fr.acervolima.com/python-regression-lineaire-a-laide-de-sklearn/ (consulté le 19 janvier 2022).
[55] « La régression linéaire en Machine Learning | Jedha Bootcamp ». https://www.jedha.co/blog/la-regression-lineaire-en-machine-learning (consulté le 19 janvier 2022).
[56] « sklearn.linear_model.LinearRegression », scikit-learn. https://scikit-learn/stable/modules/generated/sklearn.linear_model.LinearRegression.html (consulté le 20 janvier 2022).
[57] « Modèles paramétriques et non paramétriques dans l'apprentissage automatique », ICHI.PRO. https://ichi.pro/fr/modeles-parametriques-et-non-parametriques-dans-l-apprentissage-automatique-186874993435187 (consulté le 20 janvier 2022).
[58] « Arbre de décision », Wikipédia. 20 janvier 2022. Consulté le: 16 février 2022. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Arbre_de_d%C3%A9cision&oldid=190057022
[59] « Chapitre V: Arbre de décision », Introduction à l'apprentissage automatique. https://projeduc.github.io/intro_apprentissage_automatique/arbres.html (consulté le 22 janvier 2022).

[60] « 1.10. Decision Trees », scikit-learn. https://scikit-learn/stable/modules/tree.html (consulté le 20 janvier 2022).
[61] « Coefficient de Gini », Wikipédia. 17 septembre 2021. Consulté le: 16 février 2022. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Coefficient_de_Gini&oldid=186412080
[62] « Entropie de Shannon », Wikipédia. 8 février 2022. Consulté le: 16 février 2022. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Entropie_de_Shannon&oldid=190645701
[63] « Sklearn Random Forest Classifiers in Python », DataCamp Community, 16 mai 2018. https://www.datacamp.com/community/tutorials/random-forests-classifier-python (consulté le 22 janvier 2022).
[64] « sklearn.ensemble.RandomForestClassifier », scikit-learn. https://scikit-learn/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (consulté le 20 janvier 2022).
[65] « Random forest (ou forêt aléatoire) : définition et cas d'usage ». https://www.journaldunet.fr/web-tech/guide-de-l-intelligence-artificielle/1501905-random-forest-ou-foret-aleatoire-definition-et-cas-d-usage/ (consulté le 19 février 2022).
[66] « Comprendre les paramètres de la fonction Random Forest dans scikit-learn », ICHI.PRO. https://ichi.pro/fr/comprendre-les-parametres-de-la-fonction-random-forest-dans-scikit-learn-10944379748609 (consulté le 19 février 2022).
[67] L. R, « Focus : Le Réseau de Neurones Artificiels ou Perceptron Multicouche », Pensée Artificielle, 11 février 2018. https://penseeartificielle.fr/focus-reseau-neurones-artificiels-perceptron-multicouche/ (consulté le 21 février 2022).
[68] « Perceptron : retour sur l'ancêtre du machine learning ». https://www.journaldunet.fr/web-tech/guide-de-l-intelligence-artificielle/1501903-perceptron-retour-sur-l-ancetre-du-machine-learning/ (consulté le 21 février 2022).
[69] « 1.17. Neural network models (supervised) », scikit-learn. https://scikit-learn/stable/modules/neural_networks_supervised.html (consulté le 25 janvier 2022).
[70] « 1.4. Support Vector Machines », scikit-learn. https://scikit-learn/stable/modules/svm.html (consulté le 25 janvier 2022).
[71] « Machine à vecteurs de support (SVM) : définition et cas d'usage ». https://www.journaldunet.fr/web-tech/guide-de-l-intelligence-artificielle/1501879-machine-a-vecteurs-de-support-svm-definition-et-cas-d-usage/ (consulté le 26 janvier 2022).
[72] « SVM », Data Analytics Post. https://dataanalyticspost.com/Lexique/svm/ (consulté le 26 janvier 2022).
[73] « Techniques de sélection de fonctionnalités dans l'apprentissage automatique – Acervo Lima ». https://fr.acervolima.com/techniques-de-selection-de-fonctionnalites-dans-l-apprentissage-automatique/ (consulté le 25 février 2022).
[74] « Test de Chi-2 | ». https://spss.espaceweb.usherbrooke.ca/test-de-chi-2/ (consulté le 25 février 2022).
[75] « Variance (mathématiques) », Wikipédia. 11 avril 2021. Consulté le: 2 mars 2022. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Variance_(math%C3%A9matiques)&oldid=181795743

[76] QuinnRadich, « Qu'est-ce qu'un modèle Machine Learning ? » https://docs.microsoft.com/fr-fr/windows/ai/windows-ml/what-is-a-machine-learning-model (consulté le 28 janvier 2022).
[77] « Etapes Machine Learning : comprendre le processus | Talend », Talend - A Leader in Data Integration & Data Integrity. https://www.talend.com/fr/resources/etapes-machine-learning/ (consulté le 3 mars 2022).
[78] « Machine learning : comment évaluer vos modèles ? Analyses et métriques! », Saagie, 12 octobre 2021. https://www.saagie.com/fr/blog/machine-learning-comment-evaluer-vos-modeles-analyses-et-metriques/ (consulté le 3 mars 2022).
[79] « Validation croisée - Amazon Machine Learning ». https://docs.aws.amazon.com/fr_fr/machine-learning/latest/dg/cross-validation.html (consulté le 4 mars 2022).
[80] « Machine learning : comment évaluer vos modèles ? Analyses et métriques! », Saagie, 12 octobre 2021. https://www.saagie.com/fr/blog/machine-learning-comment-evaluer-vos-modeles-analyses-et-metriques/ (consulté le 4 mars 2022).
[81] +Bastien L, « Confusion Matrix : l'outil de mesure de performances du Machine Learning », LeBigData.fr, 10 décembre 2018. https://www.lebigdata.fr/confusion-matrix-definition (consulté le 4 mars 2022).
[82] « Machine learning : comment évaluer vos modèles ? Analyses et métriques! », Saagie, 12 octobre 2021. https://www.saagie.com/fr/blog/machine-learning-comment-evaluer-vos-modeles-analyses-et-metriques/ (consulté le 4 mars 2022).
[83] « corr ». http://www.biostat.ulg.ac.be/pages/Site_r/corr_pearson.html (consulté le 17 décembre 2021).
[84] « Qu'est-ce que l'accuracy ? Métriques de performance en Machine Learning », Kobia, 17 novembre 2021. https://kobia.fr/classification-metrics-accuracy/ (consulté le 23 mars 2022).
[85] +Bastien L, « Python : tout savoir sur le principal langage Big Data et Machine Learning », LeBigData.fr, 14 décembre 2021. https://www.lebigdata.fr/python-langage-definition (consulté le 25 décembre 2021).
[86] « NumPy », Wikipédia. 10 novembre 2021. Consulté le: 25 décembre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=NumPy&oldid=187880025
[87] « Pandas », Wikipédia. 1 décembre 2021. Consulté le: 25 décembre 2021. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Pandas&oldid=188466798
[88] J. JVC, « Scikit-Learn : guide de démarrage rapide en Machine Learning avec Python », Data Transition Numérique, 20 septembre 2021. https://www.data-transitionnumerique.com/scikit-learn-python/ (consulté le 9 mars 2022).
[89] « Matplotlib », Wikipédia. 8 janvier 2022. Consulté le: 9 mars 2022. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Matplotlib&oldid=189692561
[90] KooR.fr, « KooR.fr - Présentation de l'IDE Spyder - Le tutoriel/cours sur le langage de programmation Python ». https://koor.fr/Python/Tutorial/python_ide_spyder.wp (consulté le 25 décembre 2021)

.