UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE
APPLIQUÉES

PAR
MEZHOUD BILEL

ANALYSE COMPARATIVE DES MODÈLES DE VECTORISATION (BERT, WORD2VEC, FASTTEXT, GLOVE) ASSOCIÉS AU SVM POUR LA DÉTECTION DE SPAM

JUILLET 2024

RÉSUMÉ

Au quotidien, nos téléphones mobiles subissent une véritable invasion de SMS indésirables, générant une perturbation bien au-delà du simple inconvénient, parfois frôlant des risques potentiels. Malgré la mise en place de divers systèmes de détection, souvent basés sur des règles préétablies, les spammeurs persistent à contourner ces défenses. Ces mécanismes, rigides et incapables de s'adapter aux tactiques évolutives des spammeurs, reposent fréquemment sur des caractéristiques superficielles du spam, comme des motifs de mots-clés ou des comportements préétablis. Ainsi, ils peinent à identifier les nouvelles formes de spam pour lesquelles ils n'ont pas été spécifiquement programmés.

Dans cette bataille incessante, le Traitement Automatique du Langage Naturel (TALN) émerge comme un rayon d'espoir pour la détection automatique de messages suspects. Notre approche vise à conférer une adaptabilité accrue et une compréhension plus profonde des messages, réduisant ainsi les erreurs de classification. Notre méthode fusionne la technique SVM (Support Vector Machine), largement utilisée en classification de texte, avec des modèles de vectorisation de données tels que BERT, Word2Vec, FastText et GloVe. Ces derniers transforment les textes en représentations numériques, facilitant ainsi leur traitement et leur analyse informatique.

L'objectif principal de cette recherche consiste à évaluer et comparer la performance de ces méthodes : BERT+SVM, Word2Vec+SVM, FastText+SVM et GloVe+SVM. En utilisant des données issues de la collection de SMS Spam, ainsi que la création d'une base de données diversifiée, nous enrichissons notre évaluation du système face à de nouveaux échantillons de données.

Les résultats révèlent que les méthodes de vectorisation affichent des performances remarquables, avec une distinction particulière pour le modèle BERT, le positionnant légèrement au-dessus des autres. Cette étude contribue de manière significative au domaine du TALN en proposant une approche efficace pour la détection du spam, basée sur des modèles de vectorisation préalablement entraînés.

ABSTRACT

On a daily basis, our cell phones are subjected to a veritable invasion of unwanted SMS messages, generating disruption that goes far beyond mere inconvenience, sometimes bordering on potential risk. Despite the implementation of various detection systems, often based on pre-established rules, spammers persist in circumventing these defenses. These mechanisms are rigid and unable to adapt to spammers' evolving tactics, frequently relying on superficial characteristics of spam, such as keyword patterns or pre-established behaviors. As a result, they struggle to identify new forms of spam for which they have not been specifically programmed.

In this ongoing battle, Natural Language Processing (NLP) is emerging as a ray of hope for the automatic detection of suspicious messages. Our approach aims to confer greater adaptability and deeper understanding of messages, thereby reducing classification errors. Our method merges the SVM (Support Vector Machine) technique, widely used in text classification, with data vectorization models such as BERT, Word2Vec, FastText and GloVe. The latter transform texts into digital representations, facilitating their processing and computer analysis.

The main aim of this research is to evaluate and compare the performance of these methods: BERT+SVM, Word2Vec+SVM, FastText+SVM and GloVe+SVM. Using data from the SMS Spam collection, as well as the creation of a diversified database, we enrich our evaluation of the system in the face of new data samples.

The results show that the vectorization methods perform remarkably well, with a particular distinction for the BERT model, positioning it slightly above the others. This study makes a significant contribution to the field of NLP by proposing an effective approach to spam detection, based on previously trained vectorization models.

REMERCIEMENTS

Je tiens tout d'abord à exprimer ma profonde gratitude envers Dieu, qui a éclairé mon chemin tout au long de cette aventure académique. Sa guidance et Sa grâce m'ont permis de surmonter les obstacles et de persévérer dans mes études.

Je souhaite également adresser mes remerciements les plus sincères à mon encadrant, Ismaïl Biskri, qui a joué un rôle essentiel en guidant mon travail tout en me laissant une liberté appréciable.

À l'ensemble des membres du jury, Nadia Ghazzali et Boucif Amar Bensaber, je souhaite exprimer ma reconnaissance pour le temps et l'expertise précieux qu'ils ont consacrés à l'évaluation de mon mémoire.

À mes chers parents, Mezhoud Messaoud et Dakkich Nouara, ainsi qu'à toute ma famille, je vous suis profondément reconnaissant pour votre soutien inconditionnel, vos encouragements constants et votre foi en moi. Votre amour et votre confiance ont été des piliers essentiels qui ont nourri ma persévérance.

Je souhaite également exprimer ma gratitude envers tous mes amis et collègues qui ont partagé ce parcours académique avec moi.

Ce travail a été enrichi par la bienveillance et le soutien de chacun d'entre vous, et je suis reconnaissant pour les moments partagés et les leçons apprises tout au long de ce voyage académique.

Merci du fond du cœur.

Table des matières

Liste des figures :

Liste des tableaux :

CHAPITRE 1: INTRODUCTION GÉNÉRALE

CHAPITRE 1: INTRODUCTION GÉNÉRALE

La communication numérique se voit affectée par une fréquence constante de spam, une déferlante d'échanges électroniques massifs et indésirables qui vient perturber nos messageries électroniques. C'est une perturbation gênante, parfois préjudiciable, émergent de la nature indésirable et malveillante de ces messages [1].

À chaque ouverture de nos boîtes de réception électroniques, une multitude de messages publicitaires non sollicités nous accueille. Ils se présentent comme les ambassadeurs de produits douteux, proposent des remèdes miracles ou miroitent des gains extraordinaires en échange de quelques détails personnels. Certains arborent des liens suspects ou des pièces jointes porteuses de malveillance.

Le spam ne se confine pas à l'e-mail, il s'étend tentaculaire dans diverses formes de communication numérique. Le SMS (short message service), un service de messagerie textuelle pris en charge par smartphones et téléphones mobiles, devient un terrain fertile où sont stockées des données privées et sensibles, telle une mine d'informations bancaires, listes de contacts, courriels, dossiers médicaux, et autres données délicates. L'utilisation intensive de ce service par les réseaux mobiles a amplifié de manière spectaculaire le trafic de SMS. Aux États-Unis, en 2023, la Federal trade commission (FTC) a été confrontée par 54 208 plaintes liées à des messages texte non sollicités [2]. Un chiffre qui jette une lumière crue sur la persistance de ce problème, révélant une inquiétude croissante chez les utilisateurs face à cette forme envahissante de communication.

Les pertes attribuées à la fraude par SMS en 2023 culminent à une somme considérable avoisinant les 13 milliards de dollars. Une augmentation vertigineuse de 4 milliards par rapport à l'année précédente, représentant une croissance préoccupante de 18% [3]. Cette augmentation substantielle met en exergue la menace grandissante que représente la fraude par SMS dans notre paysage numérique en perpétuel

changement, requérant des mesures proactives pour faire barrage à cette tendance inquiétante.

Cette intrusion non désirée soulève des préoccupations majeures en matière de sécurité et de vie privée. Les spams SMS peuvent être exploités à des fins d'escroqueries, de fraudes, voire de violations de données personnelles. Ces incidents ne touchent pas seulement les finances, mais aussi la qualité de nos échanges numériques, nous obligeant à consacrer du temps à trier et éliminer ces messages indésirables, sapant ainsi la productivité et altérant notre expérience en ligne de manière négative.

Le problème du spam évolue dans notre paysage numérique, nécessitant une attention particulière. Un exemple concret de ce phénomène se dévoile dans la figure 1.1, mettant en scène un message indésirable se faisant passer pour Hydro-Québec, promettant des remboursements aux citoyens. Derrière cette façade généreuse se cache en réalité une tentative frauduleuse visant le détournement de fonds. L'impact palpable de cette illustration souligne de manière criante l'urgence de mettre en place des solutions efficaces pour contrer ces problèmes persistants.

Figure 1.1 : Exemple de message frauduleux sur téléphone

1.1 Contexte et Pertinence

Assurer la sécurisation des échanges en ligne requiert une détection précise des messages indésirables, constituant l'épine dorsale de nos investigations au sein de

ce paysage numérique dynamique. Malgré la pléthore d'outils de filtrage de pourriels disponibles, la persistance des individus malveillants dans le développement de nouvelles stratégies met constamment à l'épreuve les systèmes traditionnels. Ces derniers, fondés sur des critères rudimentaires tels que les listes noires et les règles heuristiques, font face à des défis de taille [4]. Leur adaptation restreinte aux tactiques émergentes et leur manque d'universalité les rendent vulnérables aux subterfuges sophistiqués des spammeurs.

La gestion des faux positifs/négatifs constitue également un enjeu crucial, attribuable à leur dépendance à des règles prédéfinies. Certains systèmes nécessitent des mises à jour fréquentes pour maintenir leur efficacité, imposant une contrainte significative. En plongeant dans cette problématique, une lueur d'espoir émerge grâce à l'intelligence artificielle, mettant en avant l'apprentissage automatique qui peut catégoriser les messages en deux entités distinctes : SPAM et Non SPAM (Ham).

Cependant, dès lors que nous abordons la classification du langage humain, les algorithmes d'apprentissage automatique exigent une mise en œuvre avancée des techniques sur les données textuelles pour appréhender, interpréter et manipuler le langage humain. Cela se réalise par le biais du Traitement du Langage Naturel, une branche de l'intelligence artificielle. Ces modèles d'apprentissage s'entraînent sur des corpus de données textuelles vastes, leur permettant de saisir et produire du langage humain avec une précision élevée et une compétence avancée. Cette synergie crée des perspectives passionnantes pour l'amélioration de la classification des messages non sollicités.

L'apprentissage automatique, de manière tangible, vise à élaborer des algorithmes exploitant les données pour appliquer ces connaissances à des situations futures. Néanmoins, il est impératif de souligner la nécessité d'avoir des ensembles de données de formation fiables afin de garantir une détection de pourriels efficace.

1.2 Objectifs de la recherche et motivation

Nous présenterons dans ce document une approche novatrice de la classification de texte en combinant l'apprentissage automatique et le traitement du langage naturel (TALN). Nous utiliserons la vectorisation des données pour transformer le texte et les mots en représentations numériques en vue de simplifier leur manipulation et leur traitement par les systèmes informatiques, et cela produit une grande efficacité d'analyse subséquente.

Nous évaluerons les performances des techniques de vectorisation : BERT, Word2Vec, FastText et GloVe combiné avec SVM. L'objectif de cette évaluation est de sélectionner le meilleur algorithme pour classifier les SMS spam et d'obtenir des meilleurs résultats.

Pour ce faire, nous utiliserons une base de données disponible sur Kaggle comprenant des messages SMS en anglais étiquetés comme "ham" (légitimes) ou "spam". De plus, nous constituerons notre propre base de données, incluant un ensemble diversifié de messages spam et non spam, afin de fournir une évaluation exhaustive du système sur de nouvelles données.

1.3 Structure du mémoire

Notre mémoire est réparti en sept chapitres. Après une introduction générale sur notre sujet de recherche, le deuxième chapitre présente l'état de l'art de notre étude, nous décrirons les différentes techniques qui existent pour la détection du spam, nous introduirons le domaine d'apprentissage automatique, ainsi que le sous-domaine : le TALN. Enfin, nous citerons quelques études de littérature qui ont proposé des approches de détection des SMS frauduleux.

Dans le troisième chapitre, nous présenterons l'algorithme de classification SVM, nous expliquerons son fonctionnement et son principe de base ainsi les avantages et les inconvénients de cette approche.

Au quatrième chapitre, nous présenterons les méthodes de vectorisation de données que nous utiliserons dans notre étude pour convertir le texte en données numériques avant de les introduire à SVM.

Le cinquième chapitre décrit notre méthodologie, nous présenterons l'organigramme de flux de notre système, ensuite nous expliquerons en détails les différentes étapes de notre processus de développement, commençons par la préparation du jeu de données, la vectorisation des données et enfin l'entrainement et l'évaluation du modèle SVM.

Dans le sixième chapitre, nous mettrons en lumière les différentes métriques que nous utiliserons pour évaluer la performance de chaque modèle, ensuite nous conclurons avec les résultats obtenus.

CHAPITRE 2: ÉTAT DE L'ART

CHAPITRE 2: ETAT DE L'ART

2.1 Introduction

Dans ce chapitre, nous examinerons les différentes techniques de détection de spam. Ensuite, nous présenterons le domaine d'apprentissage automatique avec ces différentes méthodes d'apprentissage, le TALN avec ces domaines d'application dans la vie quotidienne. Enfin, nous passerons en revue quelques études précédentes et existantes sur le filtrage du spam SMS basé sur l'apprentissage automatique.

2.2 Techniques de détection de spam

Il existe deux catégories principales de techniques pour la détection de spam : celles basées sur l'origine et celles basées sur le contenu [5].

La figure 2.1 donne un aperçu de ces différentes techniques :

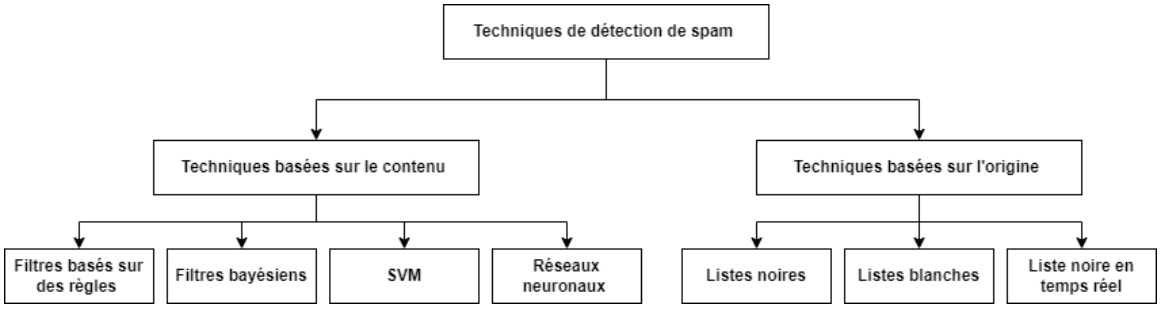


Figure 2.1 : Techniques de détection de spam

2.2.1 Techniques basées sur l'origine

Les filtres basés sur l'origine ou l'adresse sont des méthodes qui utilisent des informations réseau pour évaluer si un courrier électronique est du spam ou non. Les éléments clés de ces informations réseau sont l'adresse e-mail et l'adresse IP. Ces

filtres se regroupent généralement en deux principales catégories : les listes noires et les listes blanches [5].

a) **Listes noires :** Une liste noire contient des adresses e-mail ou des adresses IP que les fraudeurs ont déjà utilisées pour envoyer du spam [6]. Lors de la configuration des filtres, si l'expéditeur d'un e-mail est sur la liste noire, l'e-mail est classé comme indésirable et identifié comme spam. Par exemple, cette liste peut inclure des sites Web ayant des antécédents de fraude ou des vulnérabilités de navigateur. Le plus grand défi des listes noires est le maintien de leur exactitude et de leur actualité.
b) **Listes blanches :** À l'inverse des listes noires, les listes blanches répertorient les entrées autorisées à passer et à être acceptées. Les e-mails provenant de ces entrées sont considérés comme légitimes. Les listes blanches contiennent des URL et des noms de domaine réputés [7]. Contrairement aux listes noires qui bloquent les expéditeurs, les listes blanches autorisent spécifiquement ceux considérés comme des contacts de confiance, avec leurs adresses soigneusement répertoriées [6].
c) **Liste noire en temps réel (Realtime Blackhole List - RBL) :** Cette méthode de filtrage du spam fonctionne de manière similaire à une liste noire traditionnelle, mais nécessite moins de maintenance manuelle. Le Mail Abuse Prevention System et des administrateurs système tiers gèrent cette liste à l'aide d'outils de détection du spam [8]. Ce filtre nécessite essentiellement une connexion au système tiers à chaque réception d'e-mail afin de vérifier l'adresse IP de l'expéditeur par rapport à la liste. Étant donné que la liste est généralement entretenue par un tiers, l'utilisateur a moins de contrôle sur les adresses qui y figurent [6].

2.2.2 Techniques basées sur le contenu

Les techniques de détection de spam basées sur le contenu, dans lesquelles notre recherche se situe, se concentrent sur l'examen du contenu des courriels. Ces

filtres fonctionnent en utilisant des règles préétablies, parfois appelées filtres heuristiques, ou en apprenant à partir d'algorithmes d'apprentissage automatique [8]. L'objectif est d'analyser le texte du courriel pour prendre des décisions en fonction de son contenu. Ces filtres sont largement utilisés, allant des utilisateurs individuels sur leurs ordinateurs personnels aux grandes entreprises.

Ils sont si efficaces que les spammeurs ont dû développer des techniques de plus en plus complexes pour les contourner et atteindre les boîtes de réception des utilisateurs.

Les filtres basés sur le contenu incluent des catégories populaires telles que les filtres basés sur des règles, les filtres bayésiens, les machines à vecteurs de support (SVM) et les réseaux neuronaux artificiels (ANN).

a) **Filtres basés sur des règles :** Ces filtres utilisent un ensemble de règles appliquées aux mots du message pour déterminer s'il s'agit de spam ou d'un courrier légitime. Chaque courriel est comparé à un ensemble de règles, chacune ayant un poids spécifique. Le courriel commence avec un score de zéro, puis le message est analysé pour déterminer si des règles s'appliquent. Si une règle est identifiée, son poids est ajouté au score final. Si le score dépasse un seuil prédéfini, le message est classé comme spam. Cependant, ces filtres sont statiques et vulnérables à des techniques simples de modification de mots par les spammeurs [9].
b) **Filtres bayésiens :** Les filtres bayésiens sont une forme avancée de filtrage basé sur le contenu, utilisant les lois de probabilité pour distinguer les messages légitimes du spam. Les utilisateurs doivent former manuellement le filtre en marquant les messages comme spam ou courrier légitime. Le filtre analyse les mots et les phrases présents dans les courriels pour calculer la probabilité que le message soit du spam [10].
c) **Machines à vecteurs de support :** Les machines à vecteurs de support (SVM) sont utilisées pour classer les documents textuels, y compris les courriels. Son idée de base est d'intégrer les données spécifiées d’un document texte dans un

espace vectoriel dans lequel la géométrie et l'algèbre linéaire peuvent être réalisées [10]. SVM tente de créer une séparation linéaire entre deux classes dans l'espace vectoriel [5].

d) **Réseaux neuronaux artificiels :** Les réseaux neuronaux artificiels (ANN) sont des modèles d'apprentissage automatique inspirés du cerveau humain, utilisées pour modéliser des relations complexes entre les entrées et les sorties. Les ANN doivent être formés sur des ensembles de données représentatifs pour catégoriser les courriels en spam ou courrier légitime [5].

2.3 Apprentissage automatique

L'apprentissage automatique, une branche de l'intelligence artificielle, permet aux machines de s'améliorer grâce à l'analyse de données. En utilisant des ensembles d'apprentissage, on crée des modèles pour prendre des décisions ou faire des prédictions sur de nouvelles données [11]. Ses applications sont vastes, du traitement du langage naturel à la reconnaissance vocale en passant par la classification d'images [12, 13, 14]. L'objectif ultime est de doter les machines de la capacité de prendre des décisions de manière autonome.

Il existe deux principaux types d'apprentissage automatique en fonction des données d'entrée. D'abord, l'apprentissage supervisé, où les machines utilisent des données préexistantes pour apprendre à prendre des décisions éclairées. Ensuite, il y a l'apprentissage non supervisé, où les machines explorent des données sans guides préalables. Chaque type regroupe divers algorithmes spécialement conçus pour résoudre des problèmes particuliers [15]. Cette approche permet d'amplifier les capacités des machines, les rendant aptes à naviguer efficacement dans le monde des données informatiques.

La figure 2.2 présente les différents types d'apprentissage automatique :

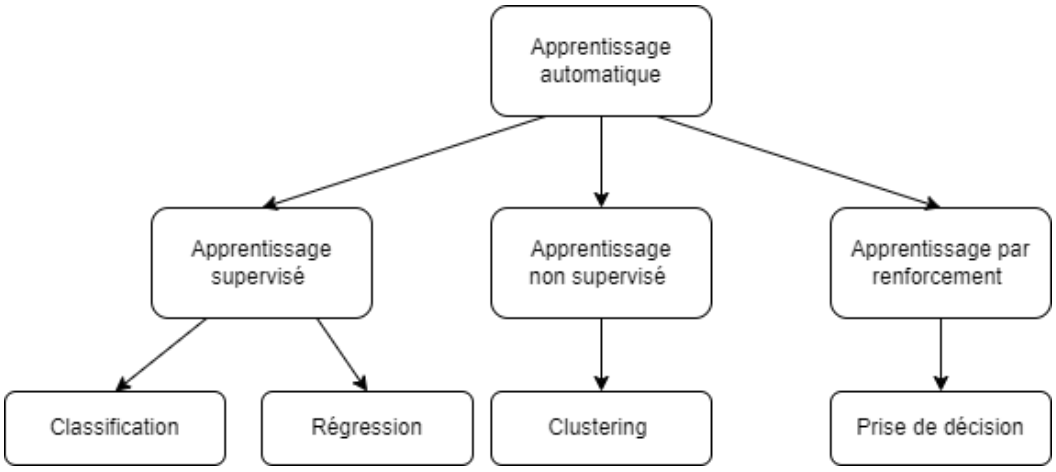


Figure 2.2 : Types d'apprentissage automatique

2.3.1 L'apprentissage supervisé

Les algorithmes d'apprentissage supervisé sont cruciaux pour résoudre divers problèmes tels que le filtrage de spams, la détection d'intrusions, la reconnaissance vocale et la catégorisation de texte. Ils apprennent à partir de données connues pour anticiper de nouvelles réponses. Deux approches dominantes sont la classification, qui trie les données en réponses discrètes comme le spam, et la régression, qui prédit des réponses continues comme les prix des actions. Parmi les algorithmes largement utilisés figurent la régression logistique, le naïfe de Bayes (NB), les machines à vecteurs de support, les réseaux de neurones et les arbres de décision [15].

2.3.2 L'apprentissage non supervisé

L'apprentissage non supervisé est une approche qui permet de découvrir des structures cachées dans les données sans étiquettes. Elle utilise des méthodes telles que le clustering pour regrouper les données en fonction de leurs similarités. Un exemple concret est l'utilisation du clustering par Google News pour organiser les actualités en catégories distinctes [11]. Ce type d'apprentissage est particulièrement utile pour l'analyse exploratoire des données et trouve des applications dans des domaines tels que la recherche génétique et la reconnaissance d'objets. Les algorithmes couramment utilisés pour effectuer ce type de regroupement incluent le

regroupement k-means, le regroupement hiérarchique, les modèles de Markov cachés et les modèles de mélange gaussien [15].

2.3.3 L'apprentissage par renforcement

L'apprentissage par renforcement (RL) représente une méthode fascinante dans l'apprentissage automatique, où un agent autonome explore la meilleure manière de choisir ses actions afin d'optimiser les récompenses au sein de son environnement interactif. Parmi les algorithmiques populaires, le Q-Learning se distingue en calculant avec précision la « valeur Q », guidant ainsi les décisions de l'agent à travers différentes situations. Un autre algorithme est le Deep Q Network (DQN), qui, avec ses réseaux neuronaux profonds, confère une perspective élargie à cette méthode d'apprentissage machine. Ces méthodes influent sur divers domaines tels que la robotique, les jeux et la gestion des ressources [16] [17].

2.4 Traitement automatique du langage naturel

Selon les propos de (François Yvon, 2010) [18], « le TALN regroupe toutes les recherches visant à reproduire, avec l'aide de machines, notre capacité humaine à créer et comprendre des énoncés linguistiques dans un but de communication ». Il insiste sur le fait que l'on parle ici du langage humain, qualifié de "naturel", excluant ainsi les langages formels tels que le C ou ADA. L'idée centrale est de recréer la richesse et la complexité du langage humain dans le cadre de la communication, en faisant clairement la distinction avec des langages plus formels et structurés.

Des avancées majeures ont été accomplies récemment dans ce secteur. En effet, diverses applications que nous utilisons quotidiennement s'appuient sur les progrès du traitement du langage naturel [19]. Parmi ces applications, on peut citer quelques domaines :

2.4.1 Traduction automatique

La richesse des langues dans le monde représente un défi stimulant pour les chercheurs s'engageant dans l'écriture multilingue. Les traducteurs conventionnels, tels que Google Translate, rencontrent des difficultés à transmettre de manière précise le sens des phrases. Cette complexité découle de l'entrelacement délicat du sens, de la grammaire et des subtilités contextuelles [19]. En 2016, Google a déployé son Système de Traduction Neuronale (GNMT) [20], exploitant l'intelligence artificielle pour une compréhension linguistique approfondie, tentant de surmonter les défis des expressions idiomatiques. Néanmoins, malgré ces progrès, atteindre une traduction automatique parfaitement fidèle demeure complexe en raison de la nature intrinsèque du langage.

2.4.2 Catégorisation de texte

La catégorisation textuelle constitue un processus essentiel dans lequel on établit une relation fonctionnelle entre un ensemble de textes et une série de catégories ou d'étiquettes. Ce lien fonctionnel, souvent désigné comme modèle de prédiction, est couramment développé à l'aide de techniques d'apprentissage automatique. La réalisation de cette tâche cruciale implique impérativement l'utilisation d'un ensemble de textes déjà annotés, également appelé ensemble d'apprentissage. En tirant parti de cet ensemble, nous sommes en mesure d'estimer les paramètres du modèle de prédiction, cherchant ainsi à créer un modèle performant visant à minimiser les erreurs de prédiction [21].

Un exemple concret mettant en lumière cette approche est le système Construe du Carnegie Group, conçu en 1991. Ce système de traitement de texte ingère des articles de Reuters, automatisant des tâches normalement dévolues aux indexeurs humains [22]. Cette application pratique illustre l'efficacité de la catégorisation textuelle dans l'optimisation des processus, soulignant le potentiel des méthodes d'apprentissage automatique dans le domaine de la compréhension et de la classification des textes.

2.4.3 Extraction d'informations

L'évolution de l'extraction d'informations (IE) dans le traitement du langage naturel constitue une avancée prometteuse dans le vaste champ de la fouille de textes, cherchant à apporter de la structure à des données issues de textes non structurés [23]. Cette technologie revêt une importance cruciale dans divers secteurs, tels que la science des matériaux, le traitement des documents médicaux et l'analyse de textes scientifiques, insufflant ainsi une dimension humaine à la transformation numérique [24].

Parmi les systèmes d'IE sophistiqués, le MITA (MetLife's Intelligent Text Analyzer) se distingue en tant qu'acteur notable. Sa mission est de rendre l'analyse des textes libres dans les demandes d'assurance-vie plus automatisée. Face au défi imposant du traitement d'un grand volume de demandes, souvent accompagnées d'une multitude de champs de texte libre [25], le MITA se distingue par sa capacité à aborder ces demandes avec une automatisation intelligente, offrant ainsi une solution efficace et centrée sur l'humain pour faire face à la complexité inhérente aux données non structurées. Cette avancée marque un pas significatif vers une intégration harmonieuse de la technologie tout en respectant les besoins humains au sein du paysage de la transformation numérique.

2.4.4 Résumer du texte

Dans l'ère numérique actuelle, le défi prédominant réside dans la surcharge d'informations, excédant notre capacité à assimiler l'accroissement exponentiel de données à notre disposition. L'explosion de données textuelles issues de diverses sources d'information et de connaissances exige une gestion judicieuse. Les techniques de résumé émergent comme un élément fondamental, pour rendre ces informations utilisables et accessibles. L'objectif fondamental demeure la préservation de la signification du texte tout en réduisant sa taille. Un exemple concret de l'application de ces techniques se retrouve dans le domaine des entreprises, où elles

exploitent cette capacité pour évaluer le sentiment sur les réseaux sociaux. Cette analyse fine joue un rôle déterminant dans le raffinement de leur stratégie marketing.

2.4.5 Systèmes de dialogue

Les systèmes de dialogue ou les agents conversationnelle, sont une tâche passionnante où des techniques de traitement du langage naturel sont utilisées pour concevoir des robots tels que des assistants personnels ou des chatbots. Ces entités sont capables d'interagir avec les individus en utilisant différents modes de communication tels que la parole, le texte, les gestes et les graphiques, à travers divers canaux de sortie et d'entrée [19]. Dans cette sphère, plusieurs systèmes de dialogue ont récemment été intégrés dans le processus industriel, notamment Siri, Cortana, Alexa [26]. Ces avancées dans les systèmes de dialogue représentent un progrès significatif dans le domaine de l'informatique.

2.5 Revue de la littérature

Les recherches antérieures dans le domaine de la classification des SMS spam ont apporté des contributions significatives, chacune avec ses propres forces et limitations.

Alper Uysal et al. (2013) [27] ont mené une étude approfondie sur l'impact de différentes méthodes d'extraction et de sélection de caractéristiques sur le filtrage des spams SMS en utilisant les langues turque et anglaise comme cas d'étude. Leur recherche a porté sur l'efficacité des combinaisons de caractéristiques basées sur le modèle de sacs de mots (BoW) et les caractéristiques structurelles pour améliorer les performances de classification des messages SMS. Bien que l'efficacité des méthodes de sélection de caractéristiques ait été évaluée, d'autres approches de sélection de caractéristiques pourraient être explorées pour une comparaison plus approfondie.

Foozy et al. (2014) [28] ont abordé la détection du spam SMS en examinant 179 SMS en malais avec une double classification en utilisant des techniques de

classification supervisée telles que NB et J48. Cette classification en deux étapes, a réussi à classifier les SMS en ham, spam et phishing avec une précision de 100%. Cependant, il est important de noter que cette étude était basée sur un ensemble de données spécifique collecté à partir de diverses sources. Il pourrait être nécessaire d'étendre l'étude à un ensemble de données plus large et diversifié pour évaluer la généralisabilité et la robustesse du cadre de détection proposé.

Tiago A. Almeida et al. (2016) [29] ont travaillé sur la normalisation de texte et l'indexation sémantique pour améliorer la détection de spam dans les messages SMS. Leurs résultats ont montré que l'approche de prétraitement du texte peut fournir des améliorations de performance pour les classificateurs dans ce contexte. Les limites de cette étude pourraient inclure le fait que les messages SMS sont souvent courts et contiennent des idiomes, des symboles et des abréviations, ce qui peut rendre la classification plus difficile. De plus, la normalisation et l'expansion du texte peuvent ne pas capturer toutes les subtilités du langage utilisé dans les messages, ce qui pourrait limiter l'efficacité de l'approche proposée.

Reaves et al. (2016) [30] ont évalué les performances des détecteurs de spam SMS existants sur un ensemble de données de près de 400 000 messages SMS, comprenant à la fois des messages en masse légitimes et des messages de spam. Les auteurs ont constaté que les détecteurs de spam existants ont des performances médiocres sur les messages en masse légitimes, et que de nouvelles techniques de détection de spam sont nécessaires pour tenir compte de ce nouveau paradigme de messagerie.

Dea Delvia Arifin et al. (2016) [31] ont mené une étude qui porte sur l'amélioration de la détection de spam SMS sur les téléphones mobiles en utilisant une combinaison de l'algorithme FP-Growth et du classificateur NB. Ils ont exploré l'utilisation de l'association et de la classification dans l'exploration de données pour améliorer les performances de filtrage du spam SMS. Cette collaboration a permis d'atteindre une précision moyenne de 98,506%.

Les travaux de Singh et Batra en 2018 [32] se concentrent sur la détection de spam dans les médias sociaux, en particulier sur Twitter, en utilisant un cadre basé sur un ensemble de classifieurs. Leur approche semi-supervisée intègre des structures de données probabilistes telles que le Quotient Filter et le Locality Sensitive Hashing pour une détection efficace du spam. Les mesures d'évaluation telles que la précision, le rappel et le F-score ont démontré l'efficacité de cette approche à détecter efficacement le spam dans les tweets de Twitter. Cependant, la limite de cette étude réside dans le fait qu'elle a été testée uniquement sur les tweets générés par Twitter, et il est suggéré qu'elle pourrait être étendue à d'autres plateformes de médias sociaux telles que Facebook et LinkedIn pour une validation plus large.

Gupta et al. (2018) [33] ont réalisé une étude comparative évaluant différentes techniques de classification. Les résultats révèlent que le CNN a atteint la plus haute performance, avec une exactitude de 99.10% et 98.25% pour les deux ensembles de données évalués, respectivement.

Nilam Nur Amir Sjarif et al. (2019) [34] ont utilisé la méthode de fréquence des termes - fréquence inverse du document (TF-IDF) avec Random Forest. Les résultas obtenus montrent que le Random Forest a surpassé les autres algorithmes évalués, avec une précision de 97,50%. L'article mentionne également que l'algorithme TF-IDF avec Random Forest a obtenu des performances supérieures en termes de précision et de mesure F par rapport à d'autres algorithmes évalués, tels que NB, KNN, SVM et Decision Tree. Cependant, une des limites de cette étude est que l'évaluation peut être insuffisante en raison du déséquilibre des données.

Prasanna Bharathi et al. (2021) [35] ont démontré l'efficacité de NB et SVM pour la classification des SMS avec une exactitude de 96.19% pour l'algorithme NB et 98.77% pour l'algorithme SVM. De plus, leur projection vers des algorithmes d'apprentissage profond pour des ensembles de données plus vastes indique une voie prometteuse.

Yanhui Guo et al. (2022) [36] ont intégré BERT avec des classificateurs d'apprentissage supervisé pour détecter les emails de spam. Les résultats obtenus ont montré que l'algorithme de régression logistique a atteint la meilleure performance de classification, avec une précision de 97,86%, un rappel de 97,83% et un score F1 de 97,84% sur le premier ensemble de données, et une précision de 95,95%, un rappel de 96% et un score F1 de 95,92% dans deux ensembles de données publics. Par contre, ils ont souligné que des recherches futures pourraient explorer des moyens d'améliorer le modèle proposé en combinant des couches plus complètes à l'intérieur de BERT pour valider davantage le cadre proposé.

Prashob Joseph et Suleiman Y. Yerima (2022) [37] ont comparé diverses techniques de vectorisation, notamment le Bag of Words (BOW), les N-grams, TF-IDF, Word2Vec et Doc2Vec, à l'aide de cinq classificateurs d'apprentissage automatique : NB multinomial (NBM), KNN, SVM, Random Forest (RF) et Extra Trees (ET). Les résultats montrent que les méthodes d'incorporation de mots statistiques (TF-IDF, BOW et N-grams), associées aux classificateurs NBM, RF ou ET, offrent les meilleures performances globales pour la détection de spam SMS. Bien que Doc2Vec soit prometteur, Word2Vec a affiché des performances inférieures aux autres méthodes. Les auteurs ont également noté que TF-IDF a bénéficié du sur-échantillonnage pour équilibrer les ensembles d'entraînement, tandis que les N-grams et BOW n'ont pas montré d'améliorations significatives avec cette technique.

De même, Humaira Yasmin Aliza et al. (2022) [38] ont évalué diverses méthodes, dont le SVM, le KNN, le NB, RF, la régression logistique, et d'autres. Les résultats ont révélé que le SVM a atteint une précision remarquable de 99%, suggérant son utilité comme système d'apprentissage automatique efficace pour les recherches futures.

Sireesha et al. (2023) [39] ont comparé les performances de différentes méthodes de classification pour la détection des spams SMS, telles que RF, le SVM et le NB, et ont obtenu un taux de précision élevé de 97,90% avec le classificateur

RF. Cependant, l'étude se concentre uniquement sur l'utilisation de techniques d'apprentissage automatique et ne considère pas d'autres approches potentiellement efficaces telles que le traitement du langage naturel.

Suparna et Vuddaraju (2023) [40] ont comparé différents modèles de classification, tels que NB et Long Short-Term Memory (LSTM). Ils ont obtenu de bons résultats, en particulier pour le modèle basé sur LSTM, qui a atteint une précision de 98,18%. Cependant, les limites de l'étude comprennent le manque de prise en compte de l'ensemble de données déséquilibré et la nécessité d'une analyse plus approfondie des caractéristiques des messages SMS pour améliorer la détection de spam. Une évaluation plus approfondie est recommandée.

Mandar Shivaji Hanchate (2023) [41] s'est engagé dans l'exploration approfondie de l'apprentissage machine et du deep learning, en optant spécifiquement pour une approche qui combine le modèle BERT à d'autres. Les combinaisons BERT+SVM et BERT+BiLSTM, affichent des performances remarquables avec des taux de précision atteignant respectivement 99.10% et 99.19%. Cette étude souligne la puissance des modèles de deep learning, notamment ceux basés sur BERT, pour une classification SMS précise distinguant le spam du non-spam.

Ces recherches ont accompli une avancée significative dans le domaine de la détection de spam. Cependant, afin d'améliorer la classification, nous considérons impérative une comparaison approfondie avec d'autres techniques.

Dans notre recherche, nous adoptons une approche novatrice en comparant BERT, Word2Vec, FastText et GloVe avec le classifieur SVM, dans le but d'enrichir la classification textuelle et d'améliorer la précision des modèles.

2.6 Conclusion

Dans ce chapitre, nous avons exploré la détection de spams, évoquant les différentes méthodes utilisées. Nous avons abordé l'apprentissage machine, expliquant comment les ordinateurs peuvent apprendre, ainsi que la compréhension du langage naturel et ses utilisations. Enfin, nous avons présenté une analyse concise des études antérieures sur l'apprentissage machine appliqué à la lutte contre le SMS Spam. Ces investigations se connectent naturellement à notre projet, axé sur l'utilisation du Support Vector Machine (SVM) pour améliorer la classification des spams par SMS, résultant d'une analyse approfondie des approches existantes.

CHAPITRE 3: LES MACHINES À VECTEURS DE SUPPORT (SVM)

CHAPITRE 3 : MACHINES À VECTEURS DE SUPPORT

3.1 Introduction

Les machines à vecteur de support (SVM) sont une classe de techniques d'apprentissage introduite par Vladimir Vapnik dans les années 90 [42, 43]. Contrairement aux réseaux de neurones, elles reposent sur une base théorique solide, évoluant de la théorie vers l'application. Les SVM sont initialement utilisées pour la classification binaire et la régression, elles sont aujourd'hui employées dans divers domaines tels que le diagnostic médical, le marketing et la biologie [44].

Dans ce chapitre, nous présenterons une explication détaillée de l'approche SVM en nous basant sur la thèse présentée par (Abdelhamid DJEFFAL, 2012) [44]. Cette référence nous offre un cadre solide pour explorer les concepts fondamentaux et les applications de l'algorithme SVM.

3.2 Fonctionnement des SVM

Le fonctionnement de SVM repose sur trois idées principales : l'hyperplan optimal, la marge maximale et les vecteurs de support. Ces derniers sont les exemples les plus proches de la frontière de séparation entre les deux classes ($y = -1$ et $y = +1$), comme illustré dans la figure 3.1.

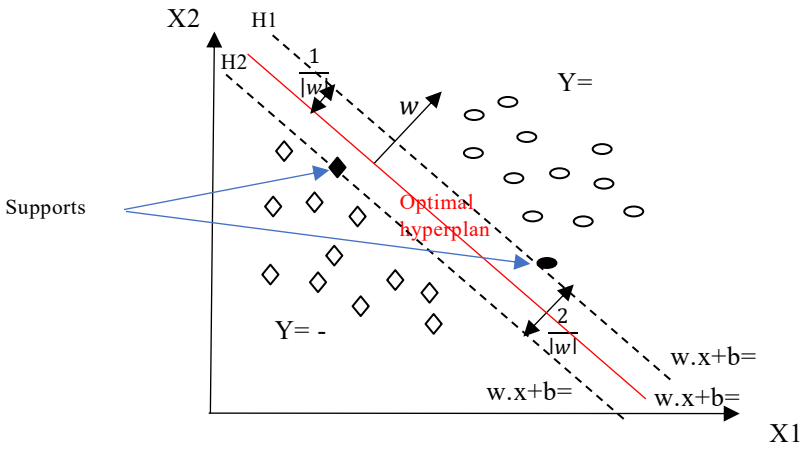


Figure 3.1 : Classification des données par la Machine à Vecteurs de Support (SVM).

3.2.1 SVM à marge souple (Soft Margin)

Lorsqu'on dispose d'un ensemble de n exemples d'entraînement répartis en deux classes étiquetées y appartenant à {+1, -1}, l'objectif est de trouver un hyperplan $w \cdot x + b = 0$ qui permet de séparer ces classes avec la marge la plus grande possible, comme le montre la figure 3.2.

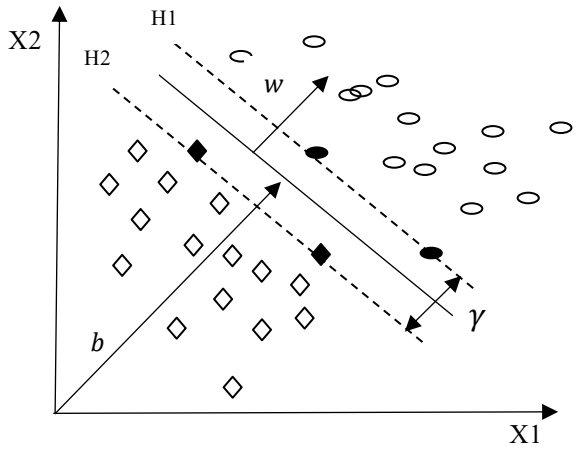


Figure 3.2 : Utilisation d'un hyperplan pour la classification binaire

Ici, w est un vecteur de dimension m dans $\mathbb{R}$, b est un scalaire dans $\mathbb{R}$, et $w \cdot x$ représente le produit scalaire entre w et x. La recherche de cette marge maximale, afin de déterminer les paramètres w et b de l'hyperplan, se traduit par un problème d'optimisation quadratique. En d'autres termes, on cherche un point qui minimise ou

maximise une certaine fonction, tout en respectant certaines contraintes. La fonction discriminante h est obtenue en combinant linéairement un vecteur d'entrée x, et s'exprime de la manière suivante :

$$h(x) = w^T x_i + b \tag{3.1}$$

La classe étant donnée par le signe de $h(x)$:

$$\hat{y} = sign\big(h(x)\big) = \begin{cases} +1 \, si \, h(x) \geq 0, \\ -1 \, si \; h(x) < 0 \end{cases} \tag{3.2}$$

L'équation qui définit l'hyperplan séparateur est : $w \cdot x + b = 0$. En prenant un des couples ($x^{(i)}$; $y^{(i)}$) parmi les n éléments de l'ensemble d'apprentissage D, l'objectif est de découvrir le classifieur h de manière à assurer que:

$$y^{(i)} \left(w \cdot x^{(i)} + b\right) \geq 0 \tag{3.3}$$

Comme précédemment discuté, la méthode SVM cherche à identifier l'hyperplan optimal parmi différentes options pour effectuer une classification précise des données. Lorsqu'on examine la figure 3.2, on observe que les vecteurs de support sont positionnés le long des hyperplans H1 et H2, qui sont parallèles à l'hyperplan optimal représenté par $w \cdot x + b = 0$. Ces hyperplans sont déterminés par les équations $w \cdot x + b = -1$ et $w \cdot x + b = +1$. Dans ce contexte, il devient impossible de repérer des exemples d'apprentissage au sein de la marge, car ils doivent respecter des contraintes spécifiques :

$$w \cdot x^{(i)} + b \geq +1 \; si \; y^{(i)} = +1 \tag{3.4}$$

$$w \cdot x^{(i)} + b \leq -1 \, si \; y^{(i)} = -1 \tag{3.5}$$

On peut regrouper les deux contraintes (3.4) et (3.5) en une seule inégalité, de la manière suivante:

$$y^{(i)}\big(w \cdot x^{(i)} + b\big) - 1 \geq 0 \tag{3.6}$$

En géométrie vectorielle, la marge est définie comme l'inverse de la norme euclidienne du vecteur w, symbolisée par $\|w\|$. Pour déterminer l'hyperplan optimal qui maximise cette marge, nous recherchons le vecteur w ayant la plus petite norme euclidienne, soit la distance minimale par rapport à l'origine, $\min\|w\|$. Cette condition de minimisation assure une classification adéquate des exemples d'apprentissage. Par conséquent, en résolvant l'équation ci-dessus, nous obtenons l'hyperplan séparateur optimal.

$$\begin{cases} min\frac{1}{2}\|w\|^2 \\ y^{(i)}\left(w \cdot x^{(i)} + b\right) - 1 \geq 0 \ \ \forall\, i = 1, \ldots, n \end{cases} \tag{3.7}$$

Pour calculer les variables w et b, on peut recourir à des méthodes d'optimisation adaptées telles que le principe de dualité et les multiplicateurs de Lagrange. Ces méthodes permettent de démontrer que le vecteur w*, qui atteint l'optimum, peut être exprimé sous la forme suivante :

$$w^* = \sum_{i=1}^{n} a_i^* y^{(i)} x^{(i)}, \tag{3.8}$$

Les multiplicateurs de Lagrange α_i^* sont non nuls uniquement pour les exemples $x^{(i)}$ qui se trouvent exactement sur les bords de la marge, c'est-à-dire les vecteurs de support. En désignant par l'ensemble des indices des vecteurs de support $vs = \{\, j \in \{1, .., l\} \mid \alpha_i^* > 0 \,\}$ la règle de décision pour une nouvelle observation x, basée sur l'hyperplan avec la marge maximale, peut être formulée comme suit :

$$\hat{y} = sign\left(\sum_{j \epsilon vs} a_i^* y^{(i)} x \cdot\ x^{(i)} + b^*\right) \tag{3.9}$$

La valeur de b^* peut être calculée à partir de n'importe quel vecteur de support en utilisant l'équation :

$$b^* = y^{(i)} - w \cdot\ x^{(i)} \tag{3.10}$$

Pour obtenir une valeur robuste de b^*, on peut calculer sa moyenne à partir de tous les vecteurs de support.

3.2.2 SVM à marge dure

En présence de données non linéairement séparables, les applications complexes du monde réel exigent des fonctions hypothétiques plus complexes que les simples fonctions linéaires. Il est donc nécessaire d'introduire des fonctions plus expressives pour traiter efficacement les subtilités des données réelles. De plus, il convient de noter que la recherche d'un hyperplan séparateur parfait n'est pas toujours possible, et les erreurs d'étiquetage dans les données d'entraînement peuvent considérablement affecter la détermination de l'hyperplan optimal. En cas de données non linéairement séparables ou de données contenant des valeurs aberrantes, il est essentiel de permettre une certaine marge d'erreur dans la classification des données, ce qui est caractéristique de l'approche « SVM à marge souple ».

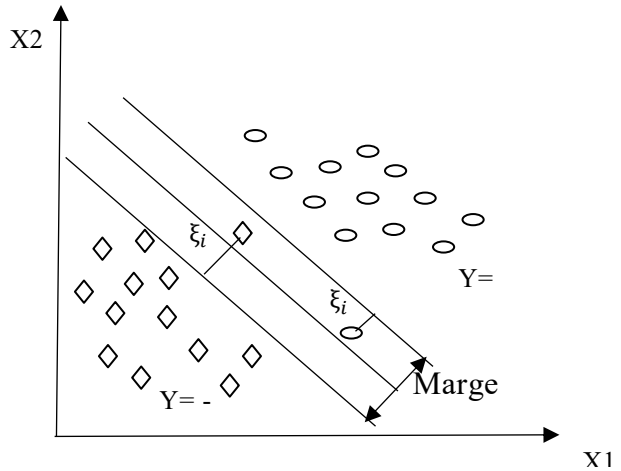


Figure 3.3: Classification binaire avec SVM à marge souple

Pour résoudre le problème complexe des données qui ne peuvent pas être séparées linéairement, nous adaptons les contraintes en utilisant des variables appelées "variables de relaxation," notées ξ_i (voir Figure 3.3).

Ces ajustements modifient les contraintes énoncées dans l'équation (3.11) de la manière suivante :

$$y_i(w^T x_i + b) \geq 1 - \xi_i, i = 1..n \tag{3.11}$$

Grâce à l'introduction de ces variables de relaxation, qui sont toutes des valeurs non négatives, nous nous assurons qu'un hyperplan séparateur est toujours possible, même lorsque $\xi_i < 1$. Cela signifie que le point x_i ne respecte pas entièrement la marge, mais il reste correctement classé. Dans cette configuration, notre objectif n'est pas seulement de rechercher un hyperplan séparateur qui maximise la marge, mais aussi de minimiser la somme des erreurs autorisées, représentée par la fonction $Q(w) = \sum_{i=1}^{n} \xi i$

En combinant cette nouvelle contrainte avec l'équation (3.7), nous aboutissons au problème primal formulé dans l'équation (3.12) suivante :

$$\begin{cases} Minimiser & \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{n} \xi_i \\ sous\ contraintes & \\ y_i(w^T x_i + b) \geq 1 - \xi_i & i = 1...n \\ \xi_i \geq 0 & \end{cases} \tag{3.12}$$

Le paramètre C est un coefficient positif qui agit comme un facteur d'équilibrage entre les deux composantes de la fonction objectif, à savoir la maximisation de la marge et la minimisation de l'erreur de classification. Nous pouvons maintenant formuler le problème dual de l'équation (3.13) en introduisant les multiplicateurs de Lagrange α_i et β_i :

$$Q(w, b, \alpha, \xi, \beta) = \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i y_i(w^T x_i + b) - 1 + \xi_i - \sum_{i=1}^{n} \beta_i \xi_i \tag{3.13}$$

Lorsque nous atteignons la solution optimale, les dérivées par rapport aux variables w, b, β, α s'annulent, et le produit des contraintes aux multiplicateurs est également respecté. Cela se traduit par les conditions suivantes :

$$\begin{cases} \frac{\partial Q(w,b,\xi,\alpha,\beta)}{\partial w} = 0 & (a) \\ \frac{\partial Q(w,b,\xi,\alpha,\beta)}{\partial b} = 0 & (b) \\ \alpha_i\{y_i(w^T x_i + b) - 1 + \xi_i\} = 0 & (c) \\ \beta_i \xi_i = 0 & (d) \\ \alpha_i \geq 0;\ \beta_i \geq 0;\ \xi_i \geq 0 & (e) \end{cases} \tag{3.14}$$

On déduit:

$$\begin{cases} w = \sum_{i=1}^{n} \alpha_i y_i x_i \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \alpha_i + \beta_i = 0 \end{cases} \tag{3.15}$$

En substituant l'équation (3.15) dans l'équation (3.13), nous arrivons au problème dual (3.16) suivant

$$\begin{cases} Maximiser & Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j \\ sous\ contraintes & \\ & \sum_{i=1}^{n} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \end{cases} \tag{3.16}$$

La principale distinction par rapport à la SVM à marge dure réside dans les valeurs possibles pour les α_i. Elles peuvent se trouver dans l'une des trois situations suivantes:

Cas où $\alpha_i = 0$:

- Conditions : $\beta_i = C$ et $\xi_i = 0$,
- Interprétation : le point de données x_i est correctement classé.

Cas où $0 < \alpha_i < C$:

- Conditions : $\beta_i > 0$ et $\xi_i = 0$,
- Interprétation : x_i est un vecteur support non borné, et la condition $y_i(w^T x_i + b) = 1$ est satisfaite.

Cas où $\alpha_i = C$:

- Conditions : $\beta_i = 0$ et $\xi_i > 0$,
- Interprétation : x_i est un vecteur support non borné, et la condition $y_i(w^T x_i + b) = 1 - \xi_i$ est satisfaite.
    - ♦ Sous-condition : Si $\xi i \geq 1$, cela signifie que x_i est correctement classé.
    - ♦ Sous-condition : Si $0 \leq \xi_i < 1$, alors x_i est mal classé.

Ces conditions concernant les valeurs de i sont connues sous le nom de conditions de Karush-Kuhn-Tucker (KKT) [45, 46]. Elles sont couramment utilisées dans les algorithmes d'optimisation pour déterminer les valeurs optimales de i et, par conséquent, l'hyperplan optimal. La fonction de décision est calculée de la même manière que dans le cas des SVM à marge dure, en se basant uniquement sur les vecteurs supports non bornés et en utilisant la fonction suivante :

$$H(x) = \sum_{i \in U} a_i y_i x_i^T x + b \tag{3.17}$$

En ce qui concerne les vecteurs supports non bornés, nous pouvons déterminer les valeurs des paramètres comme suit :

$$b = y_i - w^T x_i \tag{3.18}$$

Afin d'assurer une précision optimale, nous calculons la moyenne de la valeur de «b» pour l'ensemble des vecteurs supports non bornés de la manière suivante :

$$b = \frac{1}{|U|} \sum_{i \in U} y_i - w^T x_i \tag{3.19}$$

3.2.3 L'utilisation des noyaux

L'acceptation de la mauvaise classification de certains exemples ne garantit pas toujours une généralisation efficace, même si l'hyperplan est optimisé (voir figure 3.4).

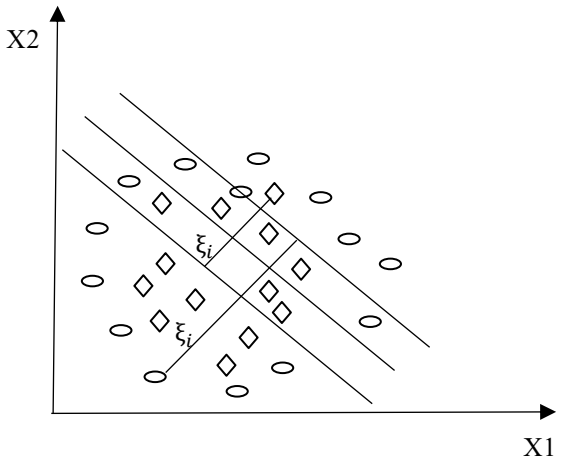


Figure 3.4 : Inadéquation de l'hyperplan face aux problèmes concrets de classification

Au lieu d'une simple droite, une représentation idéale de la fonction de décision serait celle qui s'ajuste le mieux aux données d'entraînement (Voir Figure 3.5).

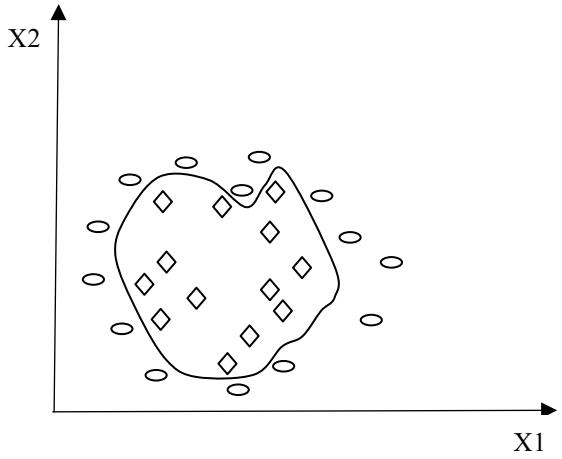


Figure 3.5: La représentation optimale de la fonction de décision

La détermination d'une fonction non linéaire idéale peut être extrêmement complexe, voire impossible. Pour surmonter cette difficulté, il est courant de projeter les données dans un espace où la fonction devient linéaire. Cette transformation permet de maintenir la structure de modélisation utilisée dans les sections précédentes, notamment les SVM basées sur la notion de séparation linéaire. Cette transformation est généralement réalisée à l'aide d'une fonction $F = f(x)$ pour x

appartenant à l'espace d'origine X, et ce processus crée un nouvel espace appelé espace de caractéristiques (Voir Figure 3.6).

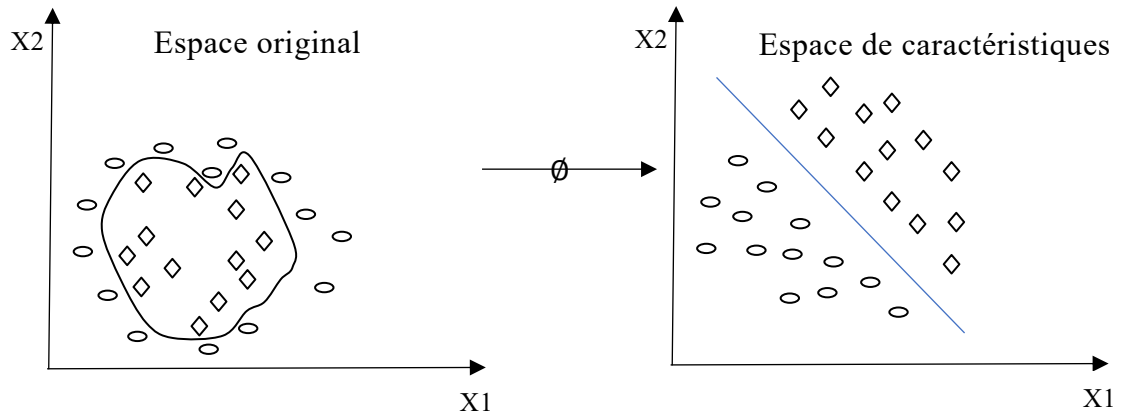


Figure 3.6: Transformation d'espace

Dans ce nouvel espace de caractéristiques, la fonction objectif à optimiser de l'équation (3.16) se transforme en :

$$Q(\alpha) = \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i \, y_j \left(\varphi(x_i), \varphi\left(x_j\right)\right) \qquad (3.20)$$

Dans le contexte des noyaux, au lieu de calculer explicitement les transformations $\varphi(x_i)$ et $\varphi\left(x_j\right)$ ainsi que leur produit scalaire, on utilise une fonction $k = \left(x_i, x_j\right)$ qui encapsule à la fois ces transformations (qui peuvent être inconnues) et le produit scalaire. Cette approche permet de résoudre le problème de détermination de la transformation φ et permet d'apprendre des relations non linéaires avec des machines linéaires. La fonction $k = \left(x_i, x_j\right)$ peut être envisagée comme une matrice $G[n, n]$ connue sous le nom de matrice de Gram [47], qui représente les distances entre tous les exemples :

$$G\,(n,\, n) = \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_n) \\ \vdots & \ddots & \vdots \\ K(x_n, x_n) & \cdots & K(x_n, x_n) \end{bmatrix}$$

Afin qu'une matrice G (fonction k) puisse être considérée comme un noyau, elle doit satisfaire aux conditions de Mercer [48, 49], ce qui signifie qu'elle doit être symétrique et semi-définie positive, c'est-à-dire qu'elle ne doit pas avoir de valeurs propres négatives. La construction de tels noyaux est un domaine de recherche bien étudié [50, 51]. Néanmoins, il existe des noyaux couramment utilisés et considérés comme des normes. Une fois le noyau choisi, la fonction objective (3.20) peut être calculée comme suit :

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_{i,} x_j) \tag{3.21}$$

Et la fonction de décision est désormais la suivante :

$$H(x) = \sum_{i \in S} \alpha_i y_i K(x_{i,} x) + b \tag{3.22}$$

S représente ici l'ensemble des vecteurs supports.

Parmi les différentes options de noyaux utilisées en apprentissage automatique, voici quelques exemples significatifs :

a) **Noyau linéaire :** Dans le cas où les données d'entraînement sont linéairement séparables, il est judicieux d'utiliser un noyau linéaire. Ce noyau repose sur le produit scalaire entre les vecteurs d'entrée pour définir la fonction de décision.

$$K(x_i, x_J) = x_i^T x_J \tag{3.23}$$

b) **Noyau polynomial :** Les données non linéairement séparables peuvent être traitées à l'aide d'un noyau polynomial. Ce noyau élève le produit scalaire des vecteurs d'entrée à une puissance d, ce qui permet d'introduire des degrés de non-linéarité dans la classification. Lorsque d est égal à 1, le noyau est linéaire, mais en augmentant sa valeur, on peut mieux adapter la séparation aux données complexes.

$$K(x_i x_j) = (x_i^T x_J)^d \tag{3.24}$$

c) **Noyau RBF (Radial Basis Function) :** Les noyaux RBF offrent une grande flexibilité. Ils sont formulés sous la forme générale $K(x_i x_j) = f\ (d\ (x_i x_j))$, où d est une mesure de distance entre les vecteurs d'entrée et f est une fonction réelle. Un exemple courant de noyau RBF est le noyau Gaussien, qui utilise la distance euclidienne pour quantifier la proximité entre les données. La largeur de bande, représentée par le paramètre

$$K(x_i, x_j) = e^{(-\frac{||xi - xj||^2}{2\sigma^2})} \tag{3.25}$$

L'écart-type σ joue un rôle clé dans la capture de relations complexes entre les données.

Le choix du noyau dépend de la nature des données et du problème de classification. Il permet d'adapter le modèle SVM pour obtenir les meilleures performances en fonction des caractéristiques spécifiques des données.

3.3 Avantages et inconvénients des SVM

3.3.1 Avantages des SVM

Les SVM se distinguent par leur efficacité dans la gestion d'ensembles de données complexes, constituant ainsi une solution adaptée pour des problèmes aux dimensions étendues. Le paramètre C, que nous avons introduit précédemment dans la section 3.2.2 "SVM à marge dure", renforce leur résilience dans des contextes où le nombre d'observations est significativement inférieur à celui des variables, assurant une adaptation souple et prévenant le surapprentissage. Comparées à d'autres approches, les SVM démontrent souvent une performance supérieure. En outre, leur souplesse paramétrique, notamment la résistance au surapprentissage, les positionne comme une option privilégiée dans le domaine de l'apprentissage automatique [52].

3.3.2 Inconvénients des SVM

La sélection délicate des paramètres pose un défi, et les SVM sont sensibles à des ensembles de données volumineux avec un nombre élevé d'observations, entraînant des problèmes de capacité mémoire et des temps de calcul accrus. En présence de bruit dans les classes, les SVM peuvent rencontrer des difficultés avec une multiplication des points supports. Les noyaux non linéaires ne fournissent pas de modèle explicite, compliquant l'interprétation des relations entre les variables. La compréhension immédiate de la pertinence des variables pour la classification est limitée. En résolvant les problèmes multi-classes avec les SVM, des ajustements spécifiques sont parfois nécessaires, cette question demeure ouverte [52].

3.4 Conclusion :

En conclusion, ce chapitre nous a offert une vue d'ensemble des machines à vecteurs de support, débutant par la classification linéaire et progressant vers des concepts plus avancés, notamment la classification non linéaire avec l'utilisation de noyaux. Nous avons présenté aussi quelques avantages et inconvénients concernant l’approche SVM. Cette compréhension préliminaire des SVM établit une base solide pour notre exploration ultérieure, où nous nous plongerons plus profondément dans l'entraînement et l'évaluation de cette technologie.

CHAPITRE 4: MÉTHODES DE VECTORISATION

CHAPITRE 4 : MÉTHODES DE VECTORISATION

4.1 Introduction

Afin d'appliquer le Support Vector Machine (SVM) à la classification des spams, il est essentiel de transformer les documents en données numériques. Ce chapitre se consacre à détailler la notion de Word Embedding et à expliquer le fonctionnement des modèles de vectorisation que nous avons intégrés dans notre étude.

4.2 Word Embedding:

Le Word Embedding englobe diverses techniques d'apprentissage machine qui cherchent à exprimer les mots ou les phrases d'un texte à travers des vecteurs de nombres réels, définis dans un modèle vectoriel (Vector Space Model) [53]. Son objectif principal est d'établir des connexions entre les mots au sein d'une phrase, tout en saisissant les subtilités sémantiques et syntaxiques. En utilisant des vecteurs, ces représentations visent à préserver la signification des mots dans leur contexte, favorisant ainsi une compréhension plus précise du langage naturel par les machines. Cette approche d'embedding offre une perspective riche pour appréhender la structure et les nuances du langage, contribuant ainsi à l'amélioration de l'interaction entre les machines et le langage humain.

Supposons que nous avons un mini-corpus avec sept mots (abeille, aigle, oie, hélicoptère, drone, fusée et jet) et trois contextes (ailes, moteur et ciel). Chaque mot est caractérisé par trois coordonnées correspondant au nombre de fois que le mot apparaît dans chaque contexte. Par exemple, l'hélicoptère n'est pas trouvé dans le contexte des ailes, et il apparaît deux fois et quatre fois dans les contextes moteur et ciel, respectivement. Ses coordonnées sont donc (0,2,4). Chaque mot occupe une position spécifique dans l'espace vectoriel, comme le montre la figure 4.1.

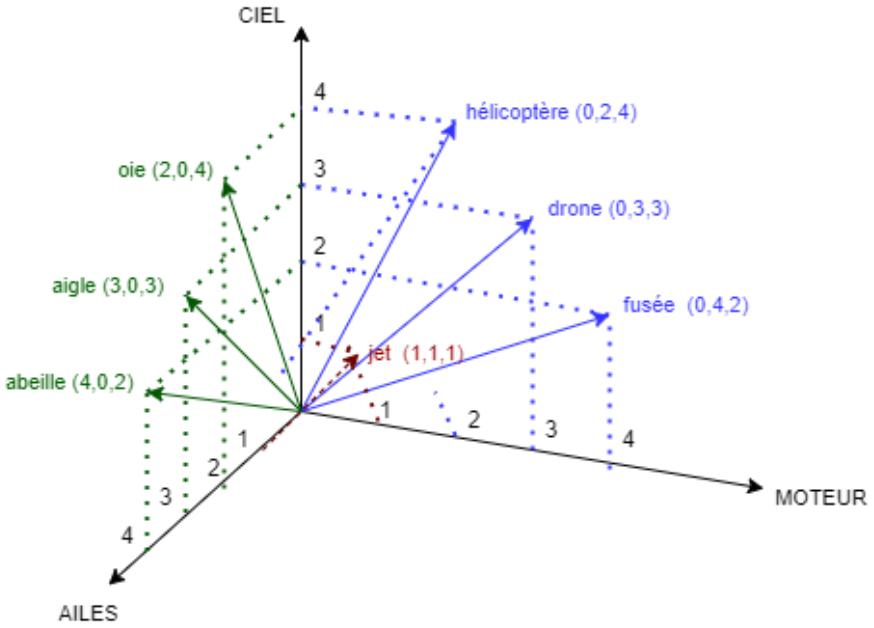


Figure 4.1 : Espace vectoriel de sept mots avec trois contextes. [54]

Le principe fondamental des embeddings de mots repose sur l'idée que les similarités sémantiques sont reflétées par les affinités contextuelles. Par exemple, les mots "hélicoptère" et "drone" sont proches dans l'espace vectoriel car ils apparaissent dans des contextes similaires, présentent des profils vectoriels comparables, et se trouvent donc rapprochés dans cet espace.

Les coordonnées vectorielles peuvent être utilisées pour calculer la proximité entre les mots. Cela se fait à l'aide de la similarité cosinus ($cos\,\theta$), c'est-à-dire le cosinus de l'angle entre deux vecteurs de mots.

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\| \vec{a} \| \| \vec{b} \|}$$

Où $\vec{a}$ et $\vec{b}$ représentent les coordonnées respectives des vecteurs. Les doubles barres indiquent les normes des vecteurs. Une fois dépliée, la formule se présente comme suit :

$$\cos\theta = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

4.3 Techniques de vectorisation

De nombreuses techniques ont été introduites par des chercheurs pour convertir les documents en données numériques. Le sac de mots, bien que considéré comme une méthode de base pour cette conversion, ne produit pas des résultats satisfaisants. Une version améliorée appelée TF-IDF est souvent utilisée, prenant en compte la pertinence des mots. Cependant, ces approches ne parviennent pas à capturer le sens des mots ni l'ordre dans les phrases [55].

Pour résoudre ces limitations, des modèles plus avancés tels que Word2Vec, FastText, GloVe et BERT peuvent être employés.

4.3.1 Word2Vec

Créé par Tomas Mikolov et son équipe chez Google (Mikolov et al., 2013) [56], ce modèle d'apprentissage automatique est largement utilisé en traitement du langage naturel. Il utilise des réseaux de neurones pour représenter les mots sous forme de vecteurs denses dans un espace de dimensions réduites, capturant ainsi les relations sémantiques et syntaxiques entre eux.

4.3.1.1 Architectures de Word2Vec

Le modèle utilise deux approches d'entraînement, Continuous Bag of Words (CBOW) et Skip-gram (SG). Dans l'approche CBOW, le modèle prédit un mot cible en utilisant les mots voisins comme contexte. En revanche, dans l'approche Skip-gram, le modèle fonctionne à l'inverse : il prédit les mots de contexte en se basant sur un mot d'entrée. Dans les deux situations, le réseau de neurone comprend deux couches; une couche cachée et une couche de sortie.

La figure 4.2 présente les deux architectures de Word2vec : CBOW et Skip-gram :

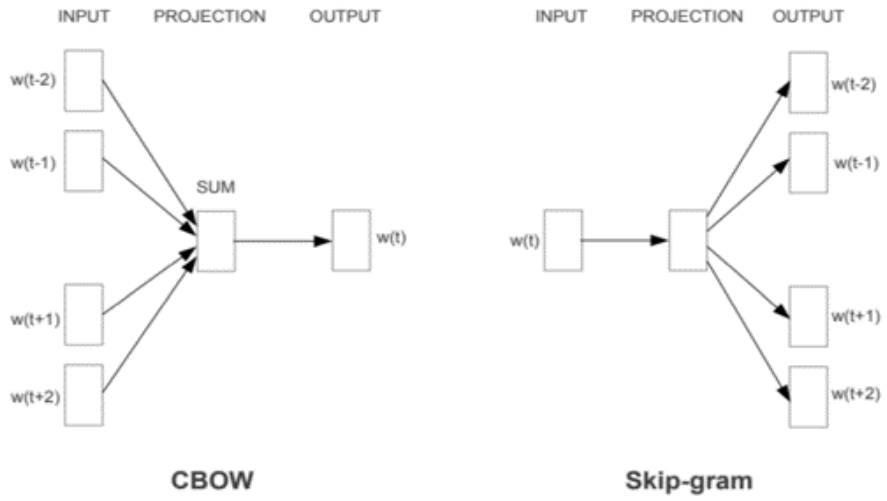


Figure 4.2 : Architectures CBOW et Skip-gram [56]

4.3.1.2 Entraînement du réseau Word2vec (CBOW)

Dans l'article de Meyer (2016) [57], l'auteur a expliqué en détail le processus d'entraînement de Word2Vec pour les deux architectures, à savoir CBOW et Skip-gram. Pour illustrer ce processus, nous pouvons prendre l'exemple de l'architecture CBOW. Son processus d'entraînement se déroule en plusieurs étapes :

- **Création d'un vocabulaire**

  D'abord, nous prenons comme exemple la phrase suivante : "Spam dans ma boîte mail", la première étape c'est de rassembler tous les mots uniques dans ce corpus d'entraînement pour créer un vocabulaire :

  ["Spam", "dans", "ma", "boîte", "mail"].

- **Représentation one-hot**

  Ensuite, on transforme chaque mot du vocabulaire en un vecteur one-hot de dimension V. En d'autres termes, on crée un vecteur binaire où la position correspondant au mot en question est indiquée par un 1, tandis que toutes les autres positions sont mises à 0.

Un exemple est donné dans le tableau 4.1.

Mots d'entrées	**Vecteur one-hot**
"Spam"	[1, 0, 0, 0, 0]
"dans"	[0, 1, 0, 0, 0]
"ma"	[0, 0, 1, 0, 0]
"boîte"	[0, 0, 0, 1, 0]
"mail"	[0, 0, 0, 0, 1]

Tableau 4.1 : Encodage one-hot des mots d'entrée

- **Création de paires de mots**

On forme des paires de mots en utilisant une fenêtre de contexte. Pour cet exemple, supposons une fenêtre de taille 1. Cela donne une liste de paires de mots possibles :

[("Spam", "dans"), ("dans", "Spam"), ("dans", "ma"), ("ma", "dans"), ("ma", "boîte"), ("boîte", "ma"), ("boîte", "mail"), ("mail", "boîte")].

- **Calcul du vecteur moyen**

Pour chaque paire de mots, on calcule la moyenne des vecteurs one-hot correspondants.

Exemple : Pour la paire ("dans" : [0, 1, 0, 0, 0] ; "Spam" : [1, 0, 0, 0, 0]).

Le vecteur moyen est obtenu en prenant la moyenne des valeurs.

$$Vecteur\ moyen = \left[\frac{0+1}{2}, \frac{1+0}{2}, \frac{0+0}{2}, \frac{0+0}{2}, \frac{0+0}{2}\right] = [0.5, 0.5, 0, 0, 0]$$

- **Transformation et estimation de probabilité**

Ensuite, On multiplie le vecteur moyen par une matrice de poids $W1$:

$$H = v_{m^*} w_1$$

La fonction ReLU est une fonction d'activation couramment utilisée dans les réseaux de neurones. Elle transforme les valeurs négatives en zéro tout en laissant les valeurs positives inchangées. Elle est définie comme suit :

$$ReLU(x) = max(0, x)$$

Où x est l'entrée de la fonction.

On applique Cette fonction d'activation pour obtenir la couche cachée.

$$H = ReLU(v_{m^*} w_1)$$

On multiplie la couche cachée (H) par une autre matrice de poids w_2 pour obtenir la sortie O.

$$O = H * w_2$$

Enfin, pour obtenir une distribution de probabilités et estimer la probabilité du mot cible, on applique une fonction softmax sur le vecteur de sortie O:

$$softmaxe(O)$$

4.3.2 FastText

Développée par Facebook AI Research (Joulin et al., 2016) [58], cette technique constitue une avancée majeure dans l'apprentissage automatique, spécifiquement dans le traitement du langage naturel. Elle se concentre sur la représentation des mots et la classification du texte, en exploitant de manière proactive les subword, fragmentant les mots en "n-grammes" afin de saisir précisément les aspects morphologiques et sémantiques des mots, même en présence d'inconnus dans le lexique.

4.3.2.1 Fonctionnement du modèle FastText

Pour résoudre les limites de Word2Vec, notamment pour les mots hors-vocabulaire [59], les chercheurs ont proposé cette nouvelle méthode d'embedding

appelée FastText. Leur idée principale était d'utiliser la structure interne d'un mot pour améliorer les représentations vectorielles obtenues à partir de la méthode Skip-gram.

C'est-à-dire, au lieu de considérer chaque mot comme une unité, FastText le traite comme une collection de sous-mots (n-grammes de caractères), comme illustré dans l'exemple présenté dans la figure 4.3 :

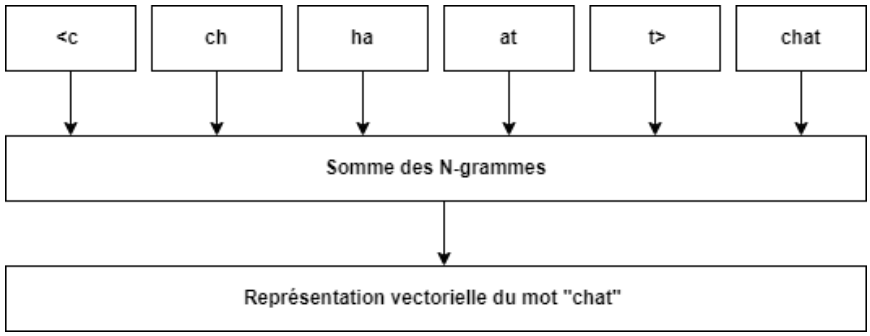


Figure 4.3 : SkipGram avec n-grammes de caractères=2

La figure 4.3 démontre le fonctionnement du modèle FastText en utilisant des sous-mots, également appelés n-grammes de caractères. Le mot 'chat' est décomposé en segments tels que « c », « ch », « ha », « at », « t », et « chat ». Chaque segment est associé à un vecteur d'embedding et la représentation finale du mot "chat" est obtenue en additionnant les vecteurs de ces segments.

Cette approche permet de représenter le mot dans l'espace vectoriel de manière plus précise, en capturant les similarités entre les mots partageant des sous-mots communs.

4.3.2.2 Entraînement du modèle FastText

Le processus de génération des embeddings de mots avec FastText comprend plusieurs étapes clés. Voici un résumé de ces étapes.

a) Extraction de mots et génération de n-grammes

La première phase du processus implique l'extraction individuelle des mots du corpus. Ensuite, pour chaque mot, des n-grammes de caractères sont générés en déplaçant une fenêtre de longueur n sur le mot.

La figure 4.4 illustre un exemple de corpus : "Spam dans ma boîte mail".

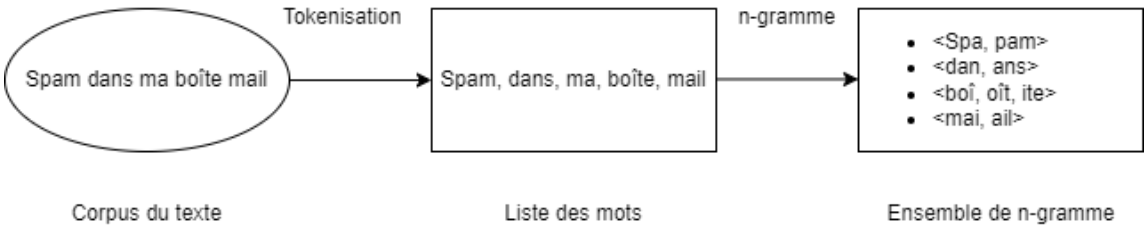


Figure 4.4 : Processus d'extraction de mots et génération de n-grammes.

b) Construction d'un vocabulaire limité en utilisant les n-grammes de caractères

Les caractères sont regroupés en n-grammes, fusionnés pour créer un vocabulaire unique, puis convertis en entiers via une fonction de hachage. Par exemple, les 3-grammes <spa, pam, am> deviennent <467, 378, 7>. Ces entiers servent d'indices de seau pour faciliter l'accès aux n-grammes dans un vocabulaire, simplifiant ainsi le traitement et l'analyse textuelle dans le domaine informatique.

La figure 4.5 illustre l'application de la fonction de hachage sur les n-grammes du corpus donné :

Figure 4.5 : Application de la fonction de hachage sur les n-grammes.

c) Entraînement des embeddings de mots avec l'algorithme Skip-gram

L'algorithme Skip-gram est utilisé pour entraîner les embeddings de mots. Les n-grammes de caractères générés à partir des mots du corpus d'entraînement tels que <spa, pam, am>, sont utilisés comme contexte pour prédire les mots cibles, comme le mot "Spam".

Cette approche permet de capturer les relations sémantiques basées sur les similarités entre les n-grammes de caractères. Par exemple, si d'autres mots du corpus partagent des n-grammes similaires, ils seront considérés comme ayant une relation sémantique avec le mot "Spam".

La figure 4.6 représente le processus d'entraînement des embeddings de mots avec l'algorithme Skip-gram :

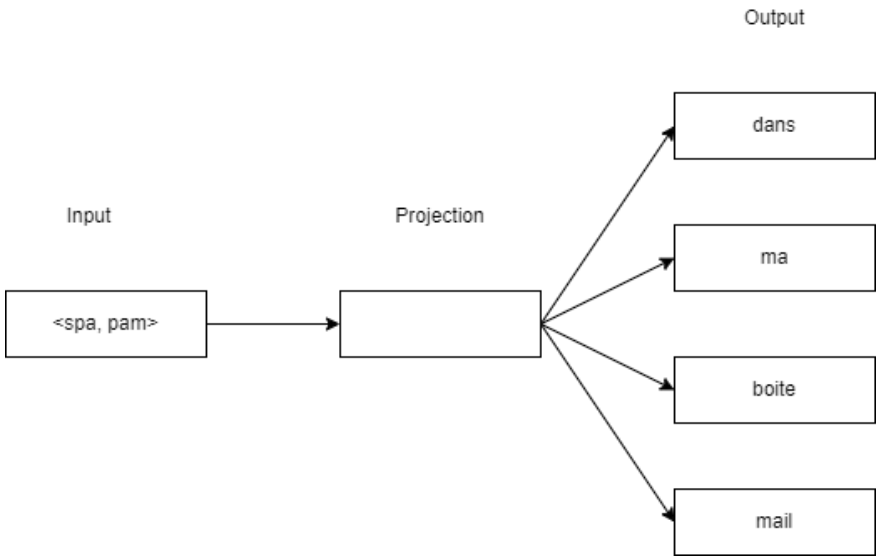


Figure 4.6 : Entraînement des embeddings de mots avec l'algorithme Skip-gram.

d) Optimisation des embeddings des n-grammes via les probabilités d'apparition dans le contexte mot.

FastText optimise les embeddings des n-grammes en utilisant l'algorithme Skip-gram, qui se base sur le calcul des probabilités conditionnelles d'apparition des

n-grammes dans le contexte des mots, exprimées par P(n-gramme|mot). Ensuite, les vecteurs de chaque n-gramme sont ajustés à l'aide de la descente de gradient, dans le but de rapprocher les n-grammes qui sont fréquemment associés aux mêmes mots voisins dans l'espace vectoriel.

Par exemple, pour le mot "Spam", nous calculons P(<spa|Spam>), P(<pam|Spam>), et P(<am|Spam>). Si les n-grammes <spa> et <pam> sont souvent liés aux mêmes mots voisins, tels que "dans", leurs vecteurs respectifs subissent des ajustements visant à les rapprocher dans l'espace vectoriel.

e) Utilisation des embeddings pour construire des représentations vectorielles des mots

Enfin, les embeddings de chaque n-gramme de caractères sont utilisés pour construire des représentations vectorielles des mots dans l'espace des embeddings. Cela signifie que chaque sous-groupe de lettres (n-gramme) dans un mot est associé à un vecteur numérique, et ces vecteurs sont ensuite combinés pour représenter le mot dans un espace vectoriel.

La figure 4.7 représente le processus de création d'une représentation vectorielle d'un mot (par exemple : "spam"), où V désigne un vecteur numérique.

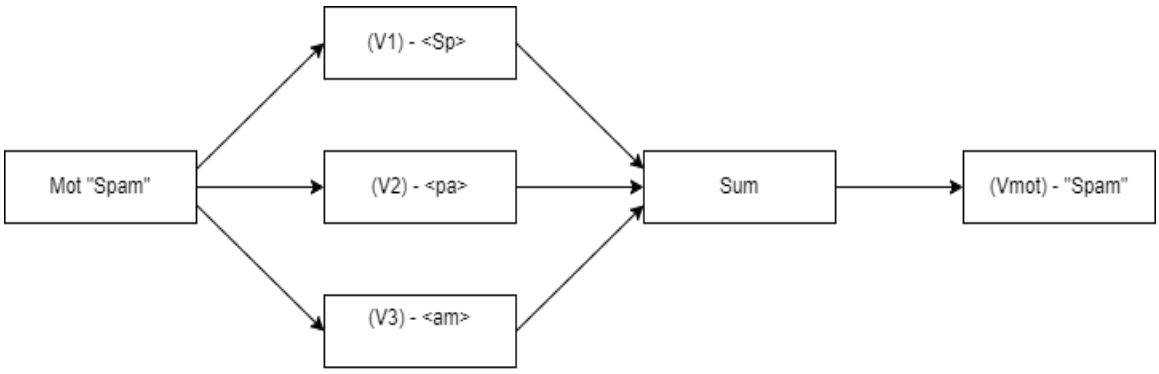


Figure 4.7 : Processus de création d'une représentation vectorielle

En suivant ces étapes, FastText permet de générer des embeddings de mots qui intègrent les informations des n-grammes de caractères et qui capturent efficacement les similitudes et les relations sémantiques entre les mots.

4.3.3 GloVe

Cette approche introduite par (Pennington et al., 2014) [60], consiste en un algorithme d'apprentissage non supervisé permettant d'obtenir des représentations vectorielles des mots en apprenant à partir des statistiques globales de cooccurrence mot-mot dans un corpus, révélant ainsi des sous-structures linéaires fascinantes de l'espace vectoriel des mots.

4.3.3.1 Fonctionnement de GloVe

La méthodologie du GloVe repose sur l'utilisation de l'algorithme de construction de la matrice de co-occurrence pour saisir les relations sémantiques entre les mots.

a) Construction de la matrice de co-occurrence X

Afin d'illustrer le fonctionnement de la matrice de co-occurrence, examinons l'exemple suivant avec une fenêtre de taille 2.

Considérons deux phrases :

- « Spams peuvent être ennuyeux. »
- « Certains spams peuvent être dangereux. »

La matrice résultante représente la co-occurrence des mots dans ces deux phrases. Chaque élément Xij de cette matrice exprime la fréquence avec laquelle le mot i apparaît dans le contexte du mot j.

Le tableau 4.2 offre une représentation visuelle de la matrice de co-occurrence pour l'exemple donné :

	Spams	*Peuvent*	*Être*	*Ennuyeux*	*Certains*	*Dangereux*
Spam	0	2	0	0	1	0
Peuvent	2	0	2	0	0	0
Être	0	2	0	1	0	1
Ennuyeux	0	0	1	0	0	0
Certains	1	0	0	0	0	0
Dangereux	0	0	1	0	0	0

Tableau 4.2 : Exemple d'une matrice de co-occurrence.

La valeur 2 dans la cellule (spam, peuvent) indique que les mots "spam" et "peuvent" apparaissent ensemble 2 fois dans la fenêtre de co-occurrence, tandis que "Ennuyeux" et "Dangereux" n'apparaissent jamais ensemble.

De manière similaire, ce principe s'applique à toutes les cellules de la matrice.

b) Utilisation des Probabilités de Co-occurrence

GloVe peut également utiliser des probabilités de co-occurrence pour capturer les préférences de co-occurrence entre les mots. Par exemple, en analysant les probabilités de co-occurrence des mots cibles "glace" et "vapeur" avec différents mots sondes provenant d'un corpus de 6 milliards de mots, nous pouvons obtenir des informations plus précises sur les relations sémantiques. Voici quelques probabilités réelles provenant de ce corpus.

Probabilités et ratio	k = solide	k = gaz	k = eau	k = mode
P(k|glace)	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
P(k|vapeur)	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
P(k|glace) / P(k|vapeur)	8.9	8.5×10^{-2}	1.36	0.96

Tableau 4.3 : Probabilités de co-occurrence des mots cibles "glace" et "vapeur"

Les probabilités conditionnelles des mots et de leur contexte, qui représentent la probabilité d'observer un mot donné après avoir observé un mot voisin, peuvent être difficiles à interpréter. Cependant, en analysant le rapport de ces probabilités conditionnelles, nous obtenons une information plus compréhensible.

Par exemple, le rapport nous indique que le mot "solide" est beaucoup plus susceptible d'apparaître dans le contexte de "glace" que dans celui de "vapeur". De même, le mot "gaz" est beaucoup plus susceptible d'apparaître dans le contexte de "vapeur" que dans celui de "glace".

Les colonnes finales, avec des ratios proches de 1, indiquent que les mots "eau" et "mode" ont des chances équivalentes et non équivalentes respectivement d'apparaître dans les contextes de "glace" et de "vapeur". Ainsi, GloVe vise à trouver des vecteurs de mots de faible dimension en utilisant la descente de gradient et une fonction de coût des moindres carrés afin de récupérer ce rapport de probabilités [60].

4.3.3.2 Entraînement du GloVe

Dans le processus d'apprentissage des vecteurs de mots avec GloVe, des contraintes souples sont définies pour chaque paire de mots. Ces contraintes douces sont incorporées dans la fonction de coût utilisée pour apprendre les vecteurs. Les contraintes douces sont exprimées par l'équation suivante :

$$w_i^T w_j + b_i + b_j = log(x_{ij})$$

Où :

- w_i et w_j sont les vecteurs de mots pour le mot i et j respectivement.
- b_i et b_j sont les biais scalaires pour les mots i et j
- x_{ij} est la fréquence d co-occurrence des mots i et j dans le texte analysé

La fonction de coût J mesure l'écart entre les produits scalaires de vecteurs de mots et les co-occurrences réelles des mots dans le corpus. La fonction de coût pour chaque paire de mots est donnée par :

$$J = \sum_{i=1}^{v} \sum_{j=1}^{v} f(x_{ij}) \, (w_i^T w_j + b_i + b_j - log\, x_{ij})^2$$

Où

- f est une fonction de pondération qui nous permet d'éviter d'apprendre uniquement à partir de paires de mots extrêmement courantes. Les auteurs de GloVe ont choisi la fonction suivante :

$$f(X_{ij}) = \begin{cases} \left(\frac{X_{ij}}{x_{max}}\right)^{\alpha} & \text{si } X_{ij} < XMAX \\ 1 & \text{sinon} \end{cases}$$

Où

- α est un paramètre de pondération qui ajuste
- $XMAX$: la fréquence de co-occurrence maximale observée dans le corpus.

4.3.4 BERT

BERT, abréviation de Bidirectional Encoder Representations from Transformers, est un modèle de représentation du langage naturel implémenté par Google (Devlin et al., 2018) [61]. Il se base sur l'architecture Transformer, qui utilise des mécanismes d'attention pour traiter les séquences de données. L'une des caractéristiques clés de BERT est sa représentation contextuelle, ce qui signifie que le sens d'un mot est pris en compte en fonction de son contexte dans le texte. Par exemple, le mot "port" aura des représentations différentes dans des phrases telles que « Il est allé au port pour embarquer sur le bateau » et « Le port de l'uniforme est obligatoire à l'école ». BERT utilise une approche bidirectionnelle, en prenant en compte à la fois les mots précédents et les mots suivants dans une phrase pour générer ses représentations.

4.3.4.1 Architecture de BERT

L'architecture de BERT repose sur un encodeur Transformer bidirectionnel avec plusieurs couches, largement inspiré de l'implémentation de (Vaswani et al. 2017) [62]. Deux configurations principales, BERTBASE et BERTLARGE, sont définies par le nombre de couches (L), la taille cachée (H), et le nombre de têtes d'auto-attention (A).

Chaque encodeur transforme une séquence de 512 jetons en une représentation vectorielle. La sortie de chaque encodeur peut être considérée comme une représentation contextuelle de la séquence d'entrée, obtenue grâce au mécanisme d'attention.

Le tableau 4.4 présente les différentes spécifications pour chaque configuration :

Configurations / Paramètres	BERTBASE	BERTLARGE
Nombre de couches (L)	12	24
Taille cachée (H)	768	1024
Nombre de têtes d'auto-attention (A)	12	16
Total des paramètres	110 millions.	340 millions.

Tableau 4.4 : Les caractéristiques distinctes de BERTBASE et BERTLARGE.

La figure 4.8 représente les deux architectures : BERTBASE et BERTLARGE :

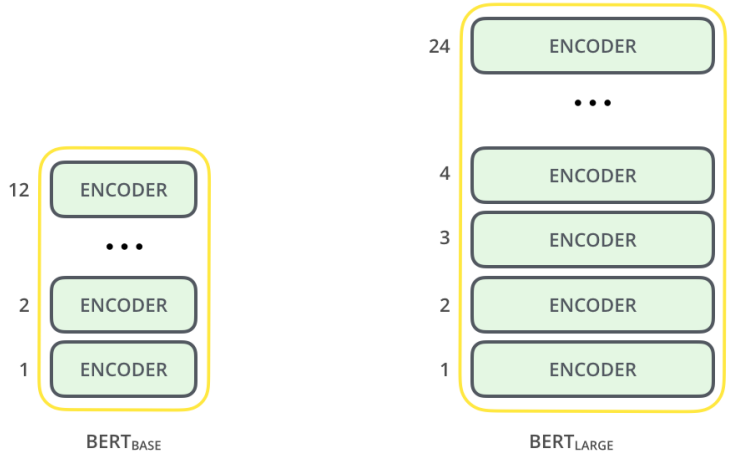


Figure 4.8 : Architecture de BERTBASE et BERTLARGE.

4.3.4.2 Entraînement de BERT

BERT se démarque des modèles de NLP pré-entraînés précédents par son pré-entraînement non supervisé, qui ne nécessite pas de données étiquetées. L'apprentissage de BERT s'effectue en deux étapes : Pré-entrainement (Pre-training) et le Réglage fin (Fine-tuning).

La figure 4.9 représente ces deux étapes :

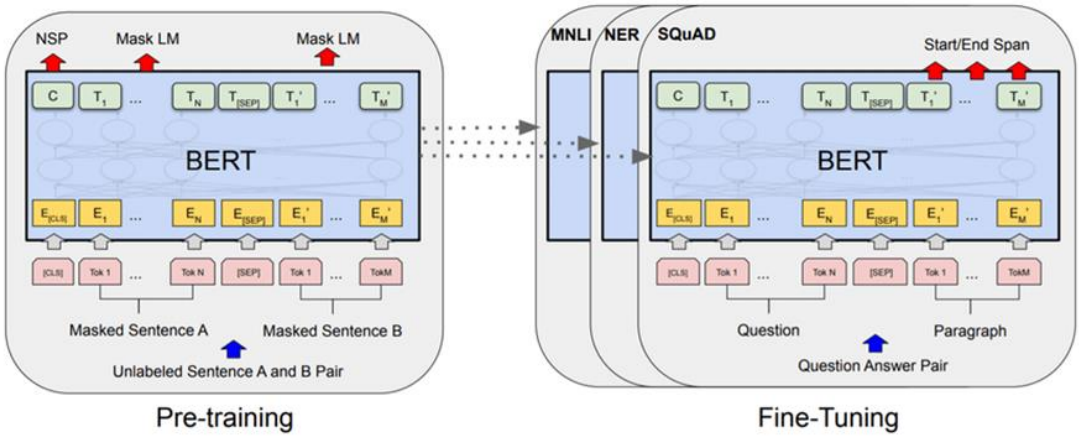


Figure 4.9 : Pré-entrainement de BERT vs. Réglage fin. [61]

[CLS] représente un symbole particulier ajouté devant chaque exemple d'entrée, tandis que [SEP] fonctionne en tant que jeton de séparation spécial, par exemple, pour distinguer les questions des réponses.

4.3.4.3 Pré-entraînement de BERT

Il utilise deux tâches distinctes lors du pré-entraînement :

a) Masked Language Modeling (MLM)

Dans cette tâche, BERT doit prédire les mots masqués dans une séquence donnée. Par exemple, il peut prédire "cafard" dans la phrase "Le lion ne s'associe pas avec le [MASK]". Environ 15% des mots sont masqués aléatoirement et le modèle est entraîné à les reconstruire.

La figure 4.10 représente le processus de modélisation du langage masqué par BERT.

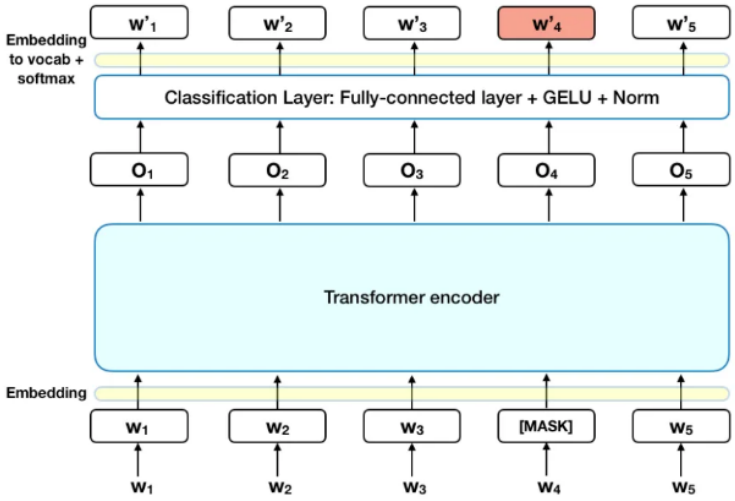


Figure 4.10 : Modélisation de Langage Masquée.

b) Prédiction de la Phrase Suivante (NSP)

Dans cette tâche, BERT doit déterminer si une séquence A est suivie ou non d'une séquence B. Par exemple, il doit identifier si l'affirmation "Il va pleuvoir" est logiquement suivie de "Je prends mon parapluie".

Pour optimiser cette tâche pendant l'entraînement, une approche méthodique est adoptée dans le traitement de l'entrée. Une séquence A est créée avec le jeton [CLS] au début de la première phrase, suivi de la première phrase et du jeton [SEP] à la fin. Une séquence B est construite en ajoutant la deuxième phrase après le jeton [SEP] de la première séquence. Chaque jeton des deux séquences est associé à une embedding de phrase distincte, indiquant clairement son appartenance à la séquence A ou B. Des embeddings positionnels sont ajoutés à chaque jeton pour refléter leur position relative dans chaque séquence. Cette méthode assure que le modèle BERT, lors de la NSP, prédit correctement si la séquence A est suivie de la séquence B, renforçant ainsi la compréhension des relations entre les phrases successives.

La figure 4.11 montre la représentation d'entrée de BERT, obtenue par la somme des embeddings de jetons, de segmentation et de position :

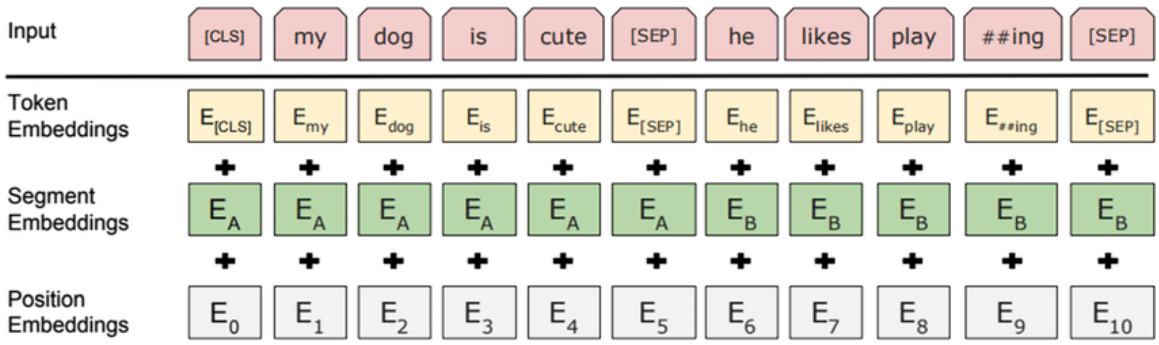


Figure 4.11 : Représentation d'entrée de BERT - Somme des embeddings de jetons, de segmentation et de position. [61]

Grâce à ces deux tâches, BERT parvient à capturer les relations contextuelles et les dépendances entre les mots et les phrases. Son pré-entraînement sur des données massives, telles que des textes Wikipédia et des livres, lui permet d'obtenir des représentations de texte riches et adaptées à diverses tâches de traitement du langage naturel [61].

4.3.4.4 Réglage fin de BERT

Suite à la phase de pré-formation, BERT subit un affinement pour des tâches spécifiques par l'addition d'une architecture dédiée à la tâche au-dessus du modèle pré-entraîné. Ce processus s'effectue en utilisant des données étiquetées spécifiques à la tâche en question. À l'exception des couches de sortie, les architectures demeurent constantes aussi bien pendant la phase de pré-formation que durant le processus de réglage fin. Les mêmes paramètres du modèle pré-entraîné sont exploités pour initialiser les modèles dédiés à diverses tâches en aval. Au cours du réglage fin, une adaptation est appliquée sur l'ensemble des paramètres.

4.4 Conclusion

Ce chapitre a été consacré à la présentation des diverses méthodes de vectorisation employées dans le cadre de notre projet de recherche, notamment Word2Vec, FastText, GloVe et BERT. Chacun de ces modèles a fait l'objet d'une définition détaillée, décrivant de manière exhaustive son mécanisme, son architecture, ainsi que les spécificités de son entraînement. L'objectif était de fournir une bonne compréhension des choix méthodologiques sous-jacents à notre recherche.

CHAPITRE 5: MÉTHODOLOGIE

CHAPITRE 5: MÉTHODOLOGIE

5.1 Introduction

Dans ce chapitre, nous commencerons par explorer l'architecture globale de notre système en introduisant l'organigramme de flux. Ensuite, nous passerons en revue les différentes étapes que nous avons appliqué tout au long du processus du développement du système en expliquant en détaille ses objectifs spécifiques.

5.2 Organigramme du flux de travail :

Pour la réalisation de notre projet de recherche, nous avons entrepris plusieurs étapes clés, telles que celles illustrées dans la Figure 5.1.

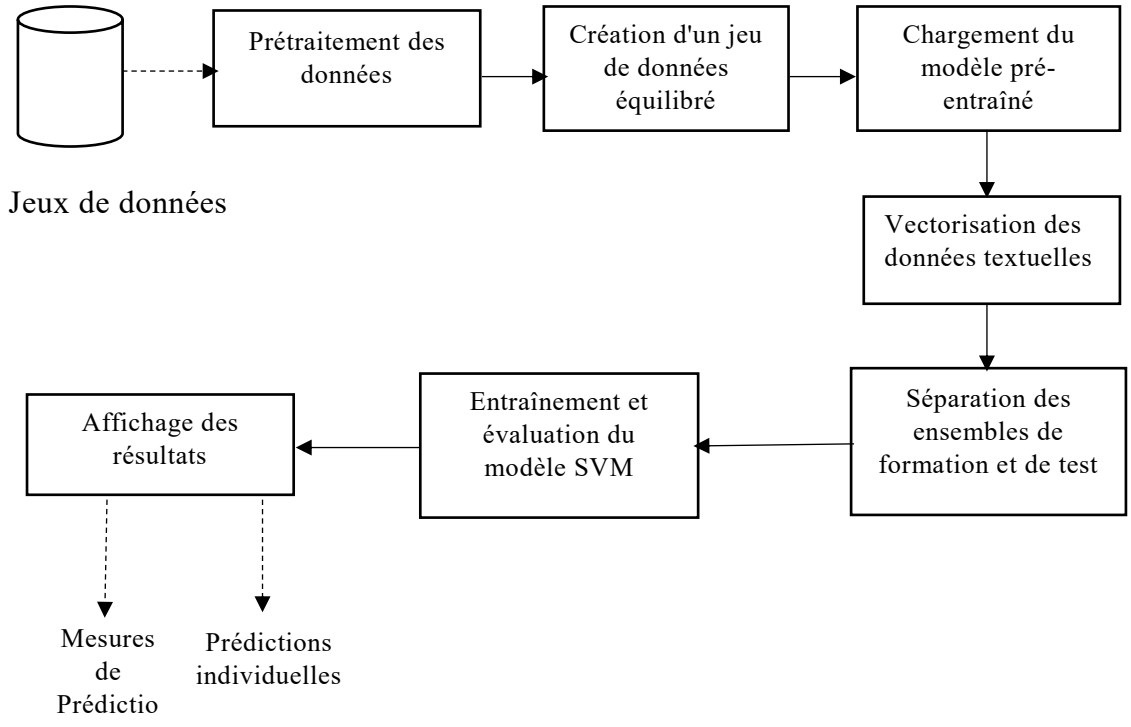


Figure 5.1 : Organigramme du processus de recherche.

5.3 Jeu de données

Nous avons utilisé le jeu de données SMS Spam Collection qui comprend 5574 messages SMS en anglais, étiquetés comme "ham" (légitime) ou "spam". Les messages sont organisés en deux colonnes : ‘Category’ pour l'étiquette et ‘Message’ pour le texte brut. Les messages ont été collectés à partir de différentes sources internet, notamment le site Web Grumbletext, le corpus NUS SMS (NSC) et la thèse de doctorat de Caroline. Ce corpus, décrit par Almeida, Hidalgo et Yamakami (2012) [63], a été largement utilisé dans des recherches sur la détection des spams SMS.

La figure 5.2 donne une capture d'écran affichant les dix premières lignes de cette base de données :

index	Category	Message
0	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry que
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives around here though
5	spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? T
6	ham	Even my brother is not like to speak with me. They treat me like aids patent.
7	ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune
8	spam	WINNER!! As a valued network customer you have been selected to receivea Â£900 prize reward! To clai
9	spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for F

Figure 5.2 : Affichage des dix premières lignes de la base de données.

5.3.1 Prétraitement des données

Le prétraitement des données constitue une étape cruciale dans l'analyse de données, englobant diverses méthodes pour nettoyer, transformer et normaliser les données brutes en vue de les rendre exploitables.

Dans le cadre de notre recherche, nous avons appliqué plusieurs techniques de prétraitement, chacune illustrée avec un exemple dans le tableau 5.1.

Étape de Prétraitement	Exemple d'Entrée	Résultat de Prétraitement
Conversion en minuscules	"Bonjour à tous"	"bonjour à tous"
Remplacement des adresses e-mail	"Contactez-moi à john.doe@email.com"	"Contactez-moi à emailaddr"
Remplacement des URL	"Visitez notre site web à www.example.com"	"Visitez notre site web à httpaddr"
Remplacement des numéros de téléphone	"Mon numéro de téléphone est +1 123 456 7890"	"Mon numéro de téléphone est phonenumbr"
Remplacement des symboles monétaires	"Le prix est de 20$"	"Le prix est de moneysymb"
Remplacement des nombres	"Il y a 10 pommes dans le panier"	"Il y a numbr pommes dans le panier"
Suppression des caractères spéciaux	"J'aime les frites !"	"J’aime les frites"

Tableau 5.1 : Techniques de prétraitement des données appliquées

5.3.2 Création d'un jeu de données équilibré

Dans certaines tâches de classification, les jeux de données peuvent être déséquilibrés, ce qui peut entraîner des problèmes de biais et de performance [64].

La création d'un jeu de données équilibré est importante pour garantir des résultats précis et fiables dans les tâches de classification. La technique que nous avons utilisée pour équilibrer notre jeu de données est appelée Oversampling (ou suréchantillonnage). Elle consiste à augmenter la taille de l'échantillon de la classe minoritaire. Cela peut être fait en répétant des échantillons existants de la classe minoritaire ou en générant de nouveaux échantillons synthétiques pour équilibrer les classes [65].

La figure 5.3 présente les deux classes de notre base de données, équilibrées :

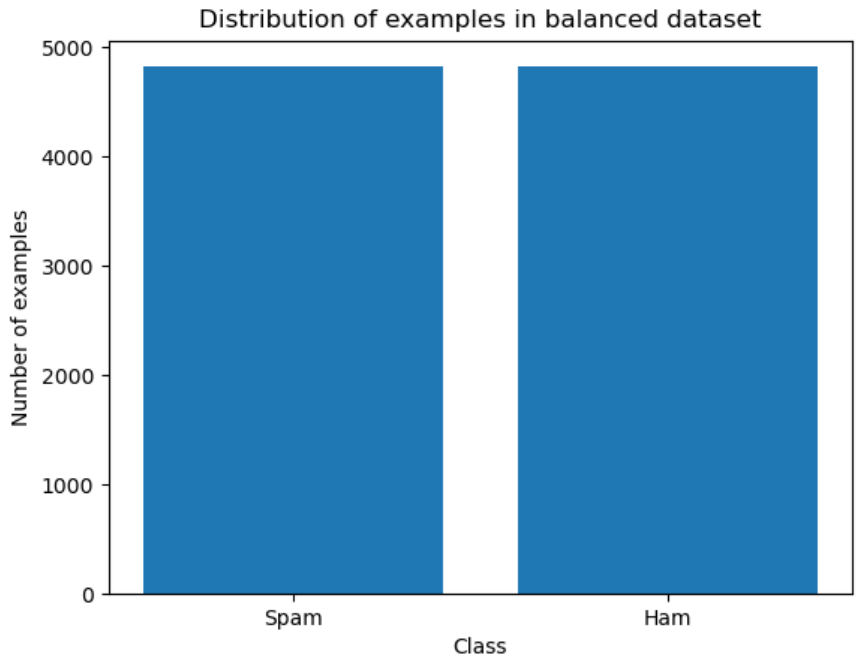


Figure 5.3 : Distribution équilibrée des classes dans la base de données.

5.4 Chargement du modèles pré-entraînés

Nous avons délibérément opté pour l'utilisation de modèles de vectorisation pré-entraînés, formés sur de vastes corpus textuels, tels que les pages de Wikipédia. Cette décision repose sur leur facilité d'utilisation, éliminant ainsi la complexité associée à la création exhaustive de modèles originaux, laquelle nécessiterait des ressources informatiques considérables. De plus, ces modèles préalablement établis peuvent être facilement adaptés à des tâches spécifiques, réduisant ainsi la nécessité de volumineuses quantités de données d'entraînement et de durées prolongées. Finalement, leur capacité à atteindre rapidement des performances optimales s'explique par leur acquisition préalable étendue de connaissances linguistiques [66].

Le tableau 5.2 présente les modèles utilisés :

Modèle	Données	Taille du vecteur	Vocabulaire
Word2vec	Google News	300 dimensions	3 Millions
FastText	Wikipédia + Common Crawl	300 dimensions	2 Milliards
GloVe	Articles de presse, Livres, Pages Web	300 dimensions	400 000
BERT	Livre + Wikipédia	300 dimensions	30 000

Tableau 5.2 : Caractéristiques des modèles de vectorisation utilisés

- L'algorithme Word2Vec se développe à partir d'une portion du corpus Google News, un ensemble massif de 100 milliards de mots, générant des vecteurs pré-entraînés en 300 dimensions qui couvrent environ 3 millions de mots et expressions [67].
- FastText, un modèle conçu dans le cadre étendu des textes anglais, se meut avec agilité entre les pages Web et les extraits de Wikipédia, accumulant 2 milliards de mots. Chaque terme s'anime dans un vecteur tridimensionnel de 300 dimensions, ajoutant une couche de sophistication et de nuance [68].
- Le modèle GloVe, extrait du fichier glove.6B.300d.txt, exploite des vecteurs pré-entraînés provenant de sources variées telles que des articles, des livres et des fragments du web, s'alimentant de 6 milliards de mots anglais. Ces vecteurs, en 300 dimensions, capturent avec finesse la richesse sémantique [69].
- BERT, élaboré à partir d'un vaste corpus de textes anglais comprenant le web et Wikipédia, totalisant 2 milliards de mots, se distingue par chaque mot de sa représentation, résonnant en un vecteur de 300 dimensions [70].

5.5 Vectorisation des données textuelles

Comme abordé dans le chapitre antérieur, intitulé « Méthodes de vectorisation », cette étape constitue la conversion du texte en données numériques, une étape cruciale pour les SVM qui opèrent sur des données de nature numérique. Ainsi, la vectorisation préalable des données textuelles s'avère impérative dans le contexte de notre recherche.

Nous avons employé les techniques de vectorisation : Word2Vec, FastText, BERT et GloVe pour accomplir cette transformation cruciale de nos données.

Le schéma de ce processus est représenté de manière visuelle dans la figure 5.4.

Figure 5.4 : Processus de vectorisation des données textuelles

Chaque technique de vectorisation génère un vecteur de mot différent. Pour illustrer, prenons le modèle pré-entraîné Word2Vec en exemple. Le mot 'spam' est associé au vecteur de la figure 5.5 dans un espace vectoriel de 300 dimensions :

```
[ 0.05957031 -0.10839844  0.01660156  0.46875    -0.15527344  0.62890625 0.02685547  0.20019531
  0.48242188  0.12695312  0.24609375 -0.3125      0.43945312  0.21972656 -0.11669922 -0.01623535
  0.05322266  0.2421875  -0.1640625  -0.24804688  0.19042969  0.46875     0.48632812 -0.08349609
 -0.31835938 -0.28125     0.46289062 -0.05834961  0.63671875  0.30859375 -0.12207031 -0.54296875
  0.3515625  -0.0390625   0.01879883 -0.35546875 -0.16210938 -0.0055542   0.60546875  0.23144531
  0.02722168 -0.21289062  -0.03515625  0.08544922 -0.40039062 -0.24511719 0.43359375 -0.09716797
 -0.26367188 -0.27734375 -0.12695312 -0.17480469 -0.16210938 -0.01867676 -0.00349426  0.2109375
 -0.02026367 -0.50390625 -0.11425781  0.01806641  0.10253906 -0.01342773 -0.5078125   0.17089844
 -0.07373047 -0.07714844 -0.5390625  -0.03466797 -0.18066406  0.77734375 -0.54296875  0.34960938
  0.05883789 -0.14355469 -0.2421875  -0.2734375   0.27148438 -0.00741577  0.07763672  0.0291748
 -0.10839844 -0.19335938  0.07275391  0.10742188 -0.00976562 -0.13964844 -0.0078125   0.06787109
 -0.28710938 -0.05737305  0.37304688 -0.26367188 -0.16113281 -0.23730469  0.37109375 -0.17578125
 -0.20898438 -0.21191406  0.00286865  0.44140625  0.41601562 -0.15429688  0.12158203 -0.25195312
  0.01794434  0.10302734 -0.52734375  0.47265625 -0.22460938  0.00285339 -0.02429199  0.14160156
 -0.5        -0.20996094  0.36914062  0.39453125  0.16601562 -0.20800781  0.27148438  0.33984375
 -0.29882812  0.33984375 -0.28710938  0.24609375  0.11767578 -0.42773438  0.15234375  0.19628906
  0.09423828 -0.4296875  -0.43554688 -0.0625     -0.02978516  0.03686523 -0.06884766  0.03369141
  0.17675781  0.17089844  0.14550781 -0.07080078  0.11669922 -0.30859375  0.35546875 -0.0189209
  0.05688477 -0.15722656  0.359375   -0.12695312 -0.24023438 -0.09960938  0.68359375 -0.01031494
  0.12060547  0.11279297 -0.27734375 -0.05786133 -0.45898438  0.07373047  0.02148438 -0.34570312
 -0.63671875  0.20996094 -0.33007812 -0.15820312  0.08398438 -0.7890625  -0.39453125 -0.19238281
 -0.05786133  0.6796875  -0.20703125  0.18457031 -0.28515625  0.296875    -0.13085938 -0.23339844
  0.296875    0.28515625 -0.01708984 -0.14941406  0.41210938 -0.03588867 -0.51171875  0.18359375
  0.11621094 -0.28710938 -0.03759766  0.20410156 -0.25976562 -0.06738281 -0.3046875  -0.03588867
 -0.16992188  0.00418091 -0.09765625  0.5546875   0.10644531 -0.43164062  0.00543213 -0.359375
  0.26171875  0.27539062  0.3125      0.41796875 -0.09667969  0.50390625  0.01037598 -0.23632812
 -0.19628906 -0.18554688 -0.06835938  0.08105469 -0.33984375  0.08251953 -0.49414062  0.58203125
 -0.05688477  0.31835938 -0.2109375  -0.109375   -0.26953125 -0.12353516  0.06079102 -0.5859375
  0.00805664 -0.03759766  0.17675781 -0.125       0.19238281  0.38085938 -0.27148438  0.00799561
 -0.55859375 -0.10839844 -0.15917969 -0.19628906  0.07275391 -0.08935547  0.41015625  0.15234375
  0.28515625  0.17382812 -0.16601562 -0.33984375 -0.16796875 -0.11474609  0.19238281  0.02099609
 -0.15332031 -0.24609375 -0.36328125 -0.39648438 -0.43359375  0.65234375  0.12792969 -0.06103516
  0.12792969 -0.265625   -0.3828125  -0.19335938  0.08886719  0.05224609  0.19433594 -0.01757812
 -0.04931641 -0.40039062 -0.28125     0.2578125  -0.2734375  -0.20605469  0.34179688  0.26757812
 -0.17089844 -0.04882812  0.10400391 -0.29101562 -0.39648438 -0.01879883  0.26757812  0.14355469
 -0.25390625 -0.04858398  0.38671875  0.00302124  0.21679688 -0.15332031  0.62890625 -0.20019531
  0.00775146  0.2109375   0.12158203 -0.04321289  0.078125   -0.1484375   0.08300781 -0.10546875
  0.17773438 -0.08056641  0.06103516 -0.17773438]
```

Figure 5.5 : Représentation vectorielle du mot 'spam' avec Word2Vec

Chaque nombre dans ce vecteur représente la position du mot dans un espace de dimension 300, capturant ainsi ses relations sémantiques avec d'autres mots.

- Algorithme pour la vectorisation des textes est:

```
Début
Variables : texte, model_type, representation_semantique
  - Si model_type == "Word2Vec" alors
        word2vec_model = load_word2vec_model().
        Tokenisez(texte), éliminez les mots d'arrêt(texte), lematisez(texte).
            representation_semantique = MoyenneDesVecteurs(word2vec_model, mots).
  - Sinon, si model_type == "GloVe" alors
        embeddings_index = load_glove_model().
        Tokenisez(texte), éliminez les mots d'arrêt(texte), lematisez(texte).
        representation_semantique = MoyenneDesVecteurs(embeddings_index, mots).
  - Sinon, si model_type == "FastText" alors
        ft_model = load_fasttext_model().
        Tokenisez(texte), éliminez les mots d'arrêt(texte), lematisez(texte).
        representation_semantique = VecteurRepresentatif(ft_model, mots).
  - Sinon, si model_type == "BERT" alors
        tokenizer, model = load_bert_model().
        Tokenisez en sous-mots (tokens).
        representation_semantique = MoyenneDesEmbeddings(model, tokens).
  - Fin si
    Retournez representation_semantique
Fin de l'algorithme
```

5.6 Séparation des données en ensembles de formation et de test

La séparation des données en ensembles de formation et de test est importante en apprentissage automatique. Elle divise les données en deux parties distinctes : formation et test. Nous avons utilisé 75% de notre jeu de données pour l'ensemble de formation pour entrainer les modèles SVM et 25% pour l'ensemble de test pour évaluer la performance des modelés (voir figure 5.6).

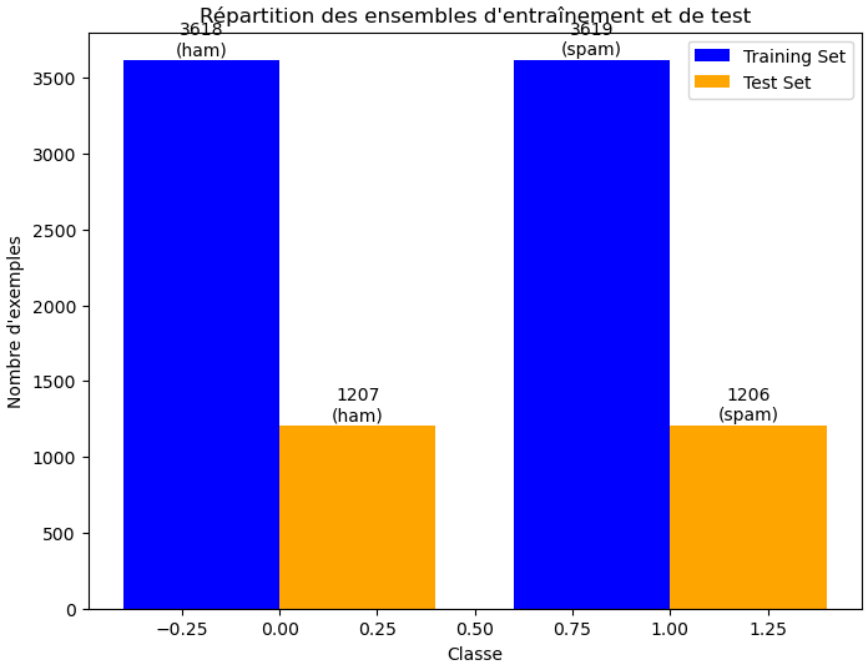


Figure 5.6 : Répartition des exemples entre les ensembles d'entraînement et de test

5.7 Entraînement et évaluation du modèle SVM

Après la préparation minutieuse des données et la création des ensembles de formation et de test, nous entamons la phase cruciale de l'entraînement et de l'évaluation de notre modèle SVM en utilisant la classe SVC de la bibliothèque scikit-learn, une référence fiable pour les algorithmes complexes. Une recherche exhaustive d'hyperparamètres est menée avec 'GridSearchCV', agissant comme un enquêteur systématique pour identifier les valeurs optimales influençant les performances du modèle [71].

Les hyperparamètres, essentiels au modèle, sont choisis avec précision. Le dilemme stratégique du choix entre les noyaux linéaire ('linear') et gaussien ('rbf') est résolu méthodiquement par notre algorithme de recherche ‘GridSearchCV’.

Le paramètre 'C', influençant la régularisation, est exploré à travers des tests approfondis avec quatre valeurs différentes (1, 10, 100 et 1000). Chacune de ces valeurs est comme une clé pour mieux comprendre l'impact sur les caractéristiques du modèle.

Avec les hyperparamètres optimaux déterminés, notre modèle SVM est ajusté aux données d'entraînement via la méthode fit. Dans une analyse approfondie, la performance du modèle sur l'ensemble de test est évaluée à l'aide des métriques d'exactitude, de précision, de rappel et de mesure F1, fournissant une compréhension approfondie de la puissance intrinsèque de notre création.

5.8 Conclusion

Dans ce chapitre, nous avons exposé notre méthodologie pour développer le modèle de classification de spam en utilisant les différentes méthodes de vectorisation combinée avec SVM. Dans le chapitre suivant, nous passerons à l’évaluation de modèles utilisés pour l'implémentation informatique et présenterons les diverses expérimentations que nous avons menées.

CHAPITRE 6 : IMPLÉMENTATION ET RÉSULTATS

CHAPITRE 6 : IMPLÉMENTATION ET RÉSULTATS

6.1 Introduction :

Dans ce chapitre, nous décrirons le langage de programmation utilisée pour le développement de modèles, en mentionnant quelques bibliothèques importantes qui ont aidé dans le processus du développement. Ensuite, nous mettrons l'accent sur les différentes métriques utilisées dans notre projet de recherche pour évaluer les performances de chaque modèle de classification. Nous présenterons également en détail les résultats de nos expérimentations afin d'analyser l'efficacité des modèles considérés.

6.2 Implémentation informatique

6.2.1 Langage de programmation

Notre système a été conçu en utilisant Python 3, un langage de programmation reconnu pour sa puissance, sa facilité d'apprentissage, sa syntaxe élégante et son typage dynamique. Il est couramment utilisé pour le développement rapide d'applications, la création de scripts, et bénéficie d'une vaste bibliothèque standard [72].

6.2.2 Bibliothèques informatiques

Pour la réalisation de ce projet, nous avons utilisé plusieurs bibliothèques logicielles populaires. Nous citerons quelques bibliothèques importantes :

- **Nltk :** Bibliothèque pour le traitement du langage naturel [73].
- **Numpy :** une bibliothèque pour le calcul numérique avec des tableaux multidimensionnels [74].
- **Pandas :** une bibliothèque pour la manipulation et l'analyse de données [75].

- **Scikit-learn** : une bibliothèque pour l'apprentissage automatique (machine learning) [76].
- **TensorFlow** : TensorFlow est une plate-forme complète en code ouvert conçue pour l'apprentissage machine [77].
- **Transformers** : une bibliothèque pour les modèles de langage pré-entraînés et le traitement du langage naturel [78].
- **Fasttext** : une bibliothèque pour le traitement de texte et la classification de texte basée sur des modèles de vecteurs de mots [79].
- **Torch** : une bibliothèque pour le calcul tensoriel utilisée dans PyTorch, un framework d'apprentissage automatique [80].
- **Joblib** : une bibliothèque pour la sérialisation d'objets Python, utile pour sauvegarder et charger des modèles entraînés [81].
- **PyQt5** : une bibliothèque pour la création d'interfaces graphiques en utilisant le framework Qt. Elle permet de développer des applications avec une interface utilisateur conviviale et réactive [82].

6.2.3 Interface utilisateur

Nous avons employé la bibliothèque PyQt5 pour concevoir une interface graphique (GUI) de bureau pour notre détecteur de spam. L'interface comporte une zone de texte pour saisir le message, une liste déroulante permettant de choisir le modèle de prédiction (Word2Vec + SVM, FastText + SVM, GloVe + SVM, BERT + SVM), un bouton pour effectuer la prédiction, une étiquette pour afficher le résultat de la prédiction, et une image centrée.

Lorsque l'utilisateur entre un message et sélectionne un modèle, en cliquant sur le bouton "Predict", le script utilise la fonction de prédiction 'predict_spam' pour effectuer la prédiction et affiche le résultat sur l'étiquette prévue à cet effet.

La figure 6.1 illustre la fenêtre principale de notre application :

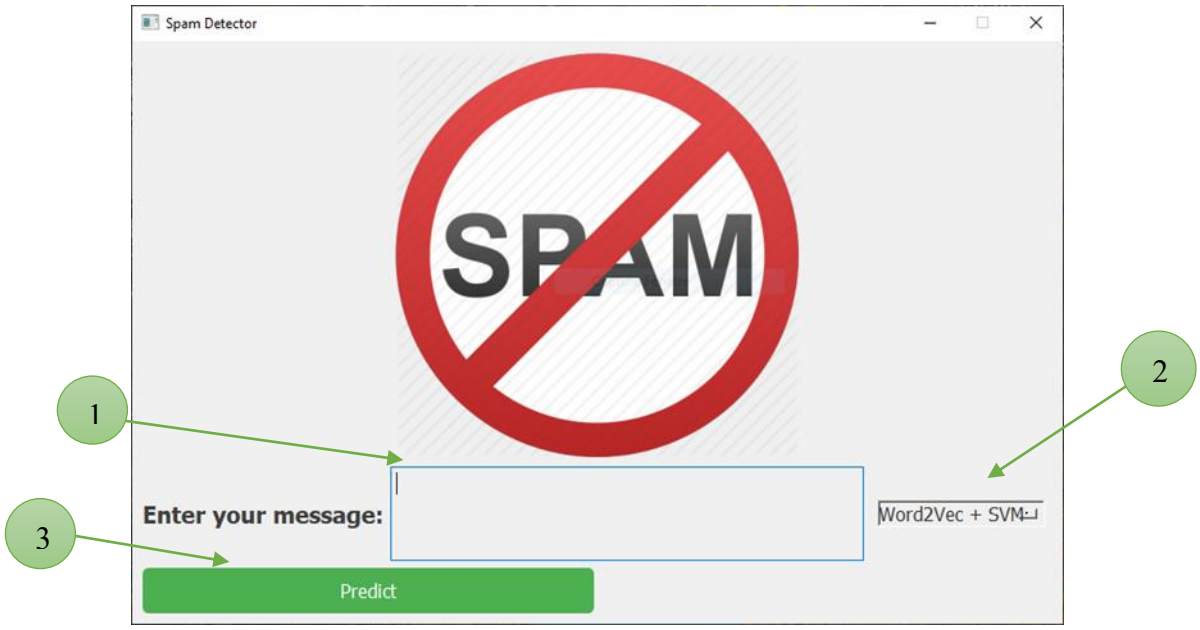


Figure 6.1 : Fenêtre principale de l'application

La figure 6.2 illustre le résultat du programme de détection de spam, qui a correctement classifié le message comme étant du spam.

Figure 6.2 : Classification du message en tant que spam

6.3 Métriques d’évaluation

Dans cette étude, nous avons décidé d'utiliser plusieurs métriques pour évaluer les performances du modèle. Les mesures sélectionnées incluent, l’exactitude de prédiction, le score de précision, le score de rappel et le score F1 ainsi que la matrice de confusion.

6.3.1 Matrice de confusion

La matrice de confusion constitue un tableau répertoriant le nombre d'occurrences associées à deux évaluateurs, à savoir la classification réelle et la classification prédite [83].

	Prédite	
Classes	Positif	Négatif
Positif	Vrais Positif (VP)	Faux Négatif (FN)
Négatif	Faux Positif (FP)	Vrai Négatif (VN)

Tableau 6.1 : Matrice de confusion

Voici les explications pour les différentes cases de la matrice de confusion (CM) :

- VP (Vrai Positif) : L'observation réellement de la classe "Positif" est correctement prédite comme "Positif".
- FN (Faux Négatif) : L'observation réellement de la classe "Positif" est incorrectement prédite comme "Négatif".
- FP (Faux Positif) : L'observation réellement de la classe "Négatif" est incorrectement prédite comme "Positif".
- VN (Vrai Négatif) : L'observation réellement de la classe "Négatif" est correctement prédite comme "Négatif".

6.3.2 L’exactitude de prédiction

Avec la mise en équilibre de nos données, cette métrique demeure un indicateur approprié pour évaluer l'efficacité de nos modèles. Elle offre une

évaluation globale de la capacité de notre modèle à classifier correctement les spams, tout en prenant en considération l'équilibre entre les différentes classes [84].

L'exactitude ACC se calcule comme le rapport du nombre de prédictions correctes sur le nombre total de prédictions. Si y_{true} représente les étiquettes réelles et y_{pred} représente les étiquettes prédites, l'exactitude peut être exprimé comme suit :

$$ACC = \frac{\sum_{i=1}^{N} 1\left(y_{\text{true},i} = y_{\text{pred},i}\right)}{N}$$

Où 1 est une fonction indicatrice qui vaut 1 si la condition est vraie, et 0 sinon, et N est le nombre total de prédictions.

Dans un contexte de classification binaire, cela peut être défini plus précisément comme :

$$ACC = \frac{VP + VN}{VP + VN + FP + FN}$$

6.3.3 La précision, le rappel et F1

La précision évalue la qualité des prédictions correctes effectuées par le modèle lors de la classification [84]. Elle peut être calculée à l'aide de la formule suivante :

$$Précision = \frac{VP}{VP + FP}$$

Le rappel mesure l'exactitude de la prédiction des classes positives parmi toutes les prédictions pertinentes. Contrairement à la précision, il tient compte des faux négatifs. Sa formule est la suivante :

$$Rappel = \frac{VP}{VP + FN}$$

La mesure F1 représente la moyenne des deux métriques précédentes. Elle s'avère particulièrement appropriée dans les cas de données non équilibrées. Le calcul de F1 peut être effectué de la manière suivante :

$$F1 = \frac{2 \times Précision \times Rappel}{Précision + Rappel}$$

6.3.4 La surface sous la courbe (AUC)

L'évaluation de La surface sous la courbe (AUC) implique l'analyse de la portion sous la Courbe ROC (Receiver Operating Characteristic) [85]. La Courbe ROC offre une représentation visuelle permettant d'équilibrer les erreurs de type FP (Faux Positifs) et TP (Vrais Positifs) en affichant les taux de FP en fonction des taux de VP sur les axes des abscisses et des ordonnées, respectivement [86]. Chaque classificateur génère un point (taux FP, taux TP) sur la courbe (voir figure 6.3).

- Le Taux de Vrais Positifs (TPR) est donné par:

$$TPR_{spam} = \frac{Nombre\ de\ messages\ spam\ correctement\ classés\ comme\ spam}{Nombre\ total\ de\ messages\ spam}$$

- Quant au Taux de Faux Positifs (FPR), il est calculé comme suit :

$$FPR_{spam} = \frac{Nombre\ de\ messages\ non-spam\ incorrectement\ classés\ comme\ spam}{Nombre\ total\ de\ messages\ non-spam}$$

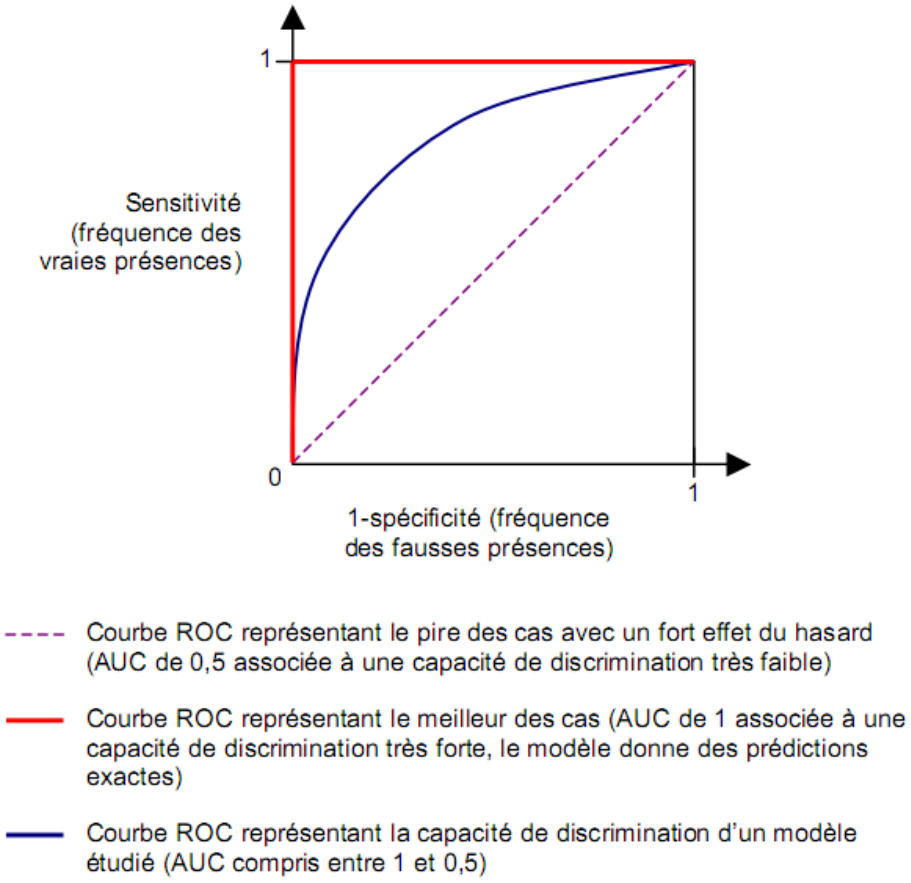


Figure 6.3 : Interprétation de la courbe ROC [88].

6.4 Résultats des expérimentations

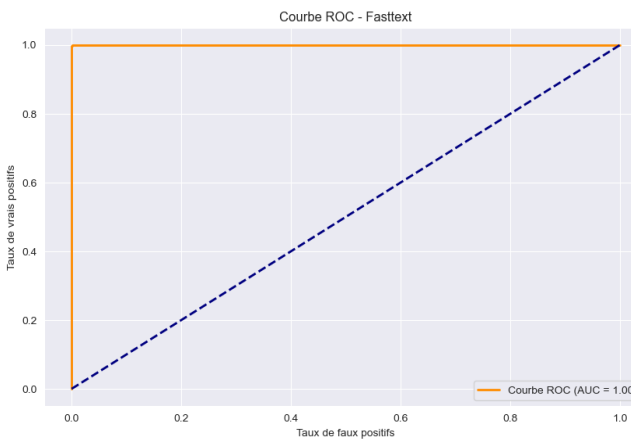
En analysant ces indicateurs de performance, nous obtenons une compréhension approfondie des rendements de nos modèles de classification, permettant ainsi une évaluation minutieuse de leurs forces et faiblesses. Initialement, nous les avons évalués avec les données d'entraînement, comme présenté dans le tableau 6.1. Cependant, notre approche vise à les soumettre à un test plus rigoureux en les confrontant à de nouvelles données, dont les résultats détaillés sont consignés dans le tableau 6.2.

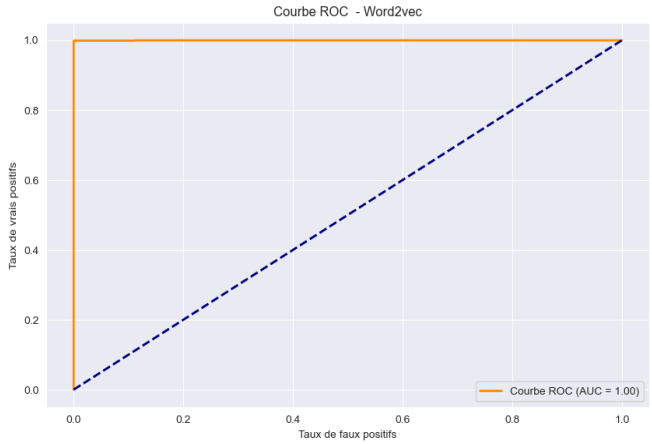
6.4.1 Évaluation sur la base de données d'entraînement

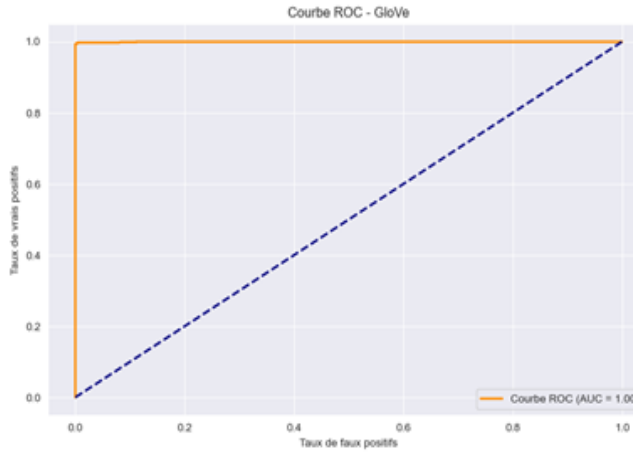
La répartition des données testées est donnée à la figure 5.6.

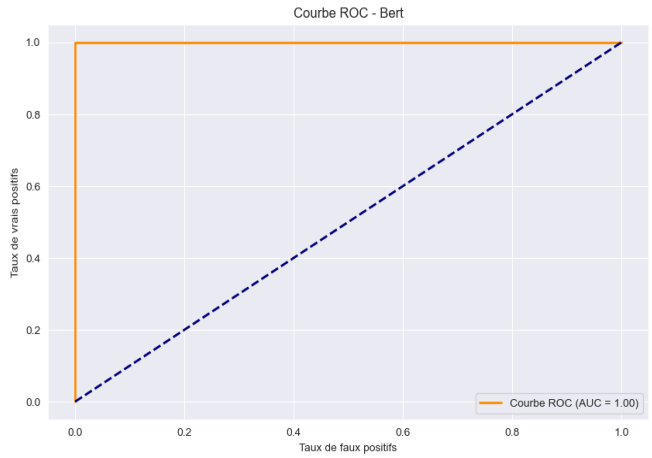
- Les figures suivantes illustrent la courbe ROC pour les quatre modèles (SVM + Bert, SVM + GloVe, SVM + Word2vec, SVM + FastText).

Nous remarquons que la courbe AUC est presque égale à 1, ce qui signifie que les modèles fournissent des prédictions exactes avec une forte capacité de discrimination.



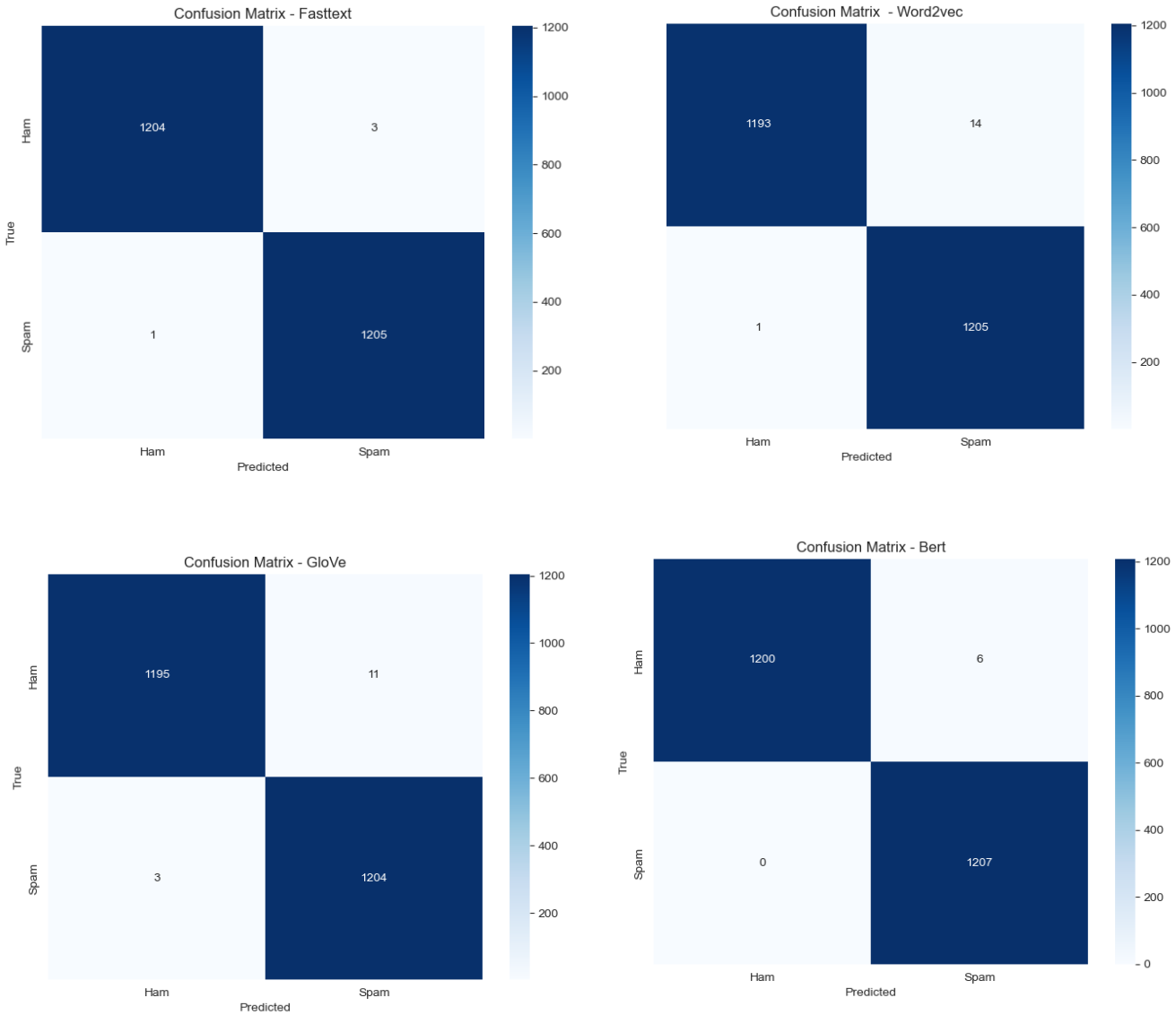






- Les figures suivantes illustrent les matrices de confusion pour chaque modèle.

Nous remarquons que les modèles présentent une forte capacité à prédire les vrais positifs (VP) avec une spécificité élevée indiquant une bonne capacité à éviter les faux positifs (FP).



- Le tableau 6.2 présente les métriques d'évaluation, à savoir l'exactitude, la précision, le rappel et le F1-score, pour chaque modèle lors de l'évaluation sur les données d'entraînement :

Méthode	Exactitude	Précision	Rappel	F1-score	AUC	Temps
Word2vec + SVM	0.9937	0.9938	0.9937	0.9937	0.9999	12m25s
FastText + SVM	0.9970	0.9971	0.9970	0.9970	1.0	1m14s
GloVe + SVM	0.9941	0.9942	0.9941	0.9941	0.9997	39m16s
BERT + SVM	0.9975	0.9975	0.9975	0.9975	1.0	5h47m13s

Tableau 6.2: Performance des modèles sur les données d'entraînement

Les résultats de nos expérimentations révèlent des performances exceptionnelles dans la classification des spams pour toutes les méthodes de classification. Plus précisément, la méthode BERT + SVM a surpassé les attentes en affichant des valeurs d'exactitude, de précision, de rappel et de mesure F1 évaluées à 99,75%. La méthode FastText + SVM a également démontré des performances remarquables, avec des valeurs équivalentes à 99,70%. Pour la méthode Word2Vec + SVM, bien que ses performances aient été légèrement inférieures à celles de FastText + SVM, elle a tout de même atteint des valeurs évaluées à 99,37%. Enfin, la méthode GloVe + SVM a présenté des performances solides, avec des valeurs équivalentes à 99,41%.

Cependant, bien que BERT ait fourni des résultats impressionnants, son temps d'exécution était trop long par rapport aux autres algorithmes, consommant ainsi beaucoup de ressources informatiques. En revanche, le temps d'exécution de FastText était très rapide tout en fournissant d'excellents résultats, ce qui nous amène à le suggérer comme une solution efficace pour ce problème de classification.

6.4.2 Évaluation sur la nouvelle base de données de test

Nous avons constitué une base de données diversifiée comprenant 200 messages distincts dans le but d'évaluer les performances de nos modèles. L'objectif de cette évaluation sur ce nouvel ensemble de données est d'analyser la réaction de nos modèles face à des messages inconnus. Nous cherchons ainsi à évaluer leur aptitude à appliquer leurs connaissances à de nouveaux contenus, une compétence cruciale pour évaluer leur utilité dans des situations réelles où de nouvelles données sont constamment introduites. En d'autres termes, notre objectif est de déterminer la capacité de nos modèles à traiter avec précision et efficacité des messages qu'ils n'ont jamais rencontrés auparavant.

Le tableau 6.3 présente les résultats obtenus pour chaque modèle lors de l'évaluation :

Méthode	Exactitude	Précision	Rappel	F1-score	Temps
Word2vec + SVM	0.7789	0.7789	0.7789	0.7789	1h32m12s
FastText + SVM	0.7788	0.7970	0.7788	0.7409	4h30m26s
GloVe + SVM	0.6884	0.6438	0.6884	0.6438	1h47m24s
BERT + SVM	0.8140	0.8089	0.8140	0.8040	5m50s

Tableau 6.3 : Performance des modèles sur les nouveaux échantillons

Nous constatons une baisse de performance pour tous les modèles, indiquant une diminution de leur capacité à effectuer des prédictions précises face à des messages qu'ils n'ont jamais rencontrés auparavant. Cette baisse peut également être attribuée à la taille réduite des données utilisées pour l'évaluation, ce qui limite la capacité des modèles à généraliser efficacement. Cependant, les résultats demeurent satisfaisants. BERT surpasse toujours les autres modèles avec un scores de précision, de rappel et de F1-score atteignant 81,40 %. Pour le modèle Word2Vec, les performances se sont traduites par des scores identiques de 77,89 % pour toutes les métriques. Le modèle GloVe a présenté des mesures de précision, de rappel et de F1-score équivalentes à 68,38 %, avec une exactitude correspondante de 68,84 %. En ce

qui concerne FastText, le modèle affiche une précision de 79,70 %, un rappel de 77,88% et un F1-score de 74,09 %, avec une exactitude de 77,88 %.

Enfin, ces résultats fournissent des informations essentielles sur la performance de nos modèles face à la diversité des messages. Ils mettent en évidence les forces et les faiblesses de chaque modèle dans des conditions réalistes. Le modèle BERT se démarque clairement comme le meilleur choix en termes de performances et d'efficacité temporelle pour traiter de nouveaux messages. Bien que FastText offre une bonne précision, il est beaucoup plus lent, ce qui le rend moins adapté aux applications nécessitant une réponse rapide. Word2Vec constitue un compromis raisonnable entre performance et temps de traitement, tandis que GloVe montre des performances globalement inférieures.

6.5 Conclusion :

Dans ce chapitre, nous avons exposé les métriques de performance utilisées dans notre projet de recherche. Nous avons ensuite présenté et analysé les résultats obtenus par chaque modèle de classification tant sur la base de données d'entraînement que sur une base de données nouvellement constituée pour évaluer le comportement de ces modèles face à de nouvelles données. Les résultats ont révélé des performances élevées, en particulier pour Bert, qui, malgré son temps de traitement énorme, a obtenu des scores dépassant les 99,70 %. Ces conclusions soulignent l'efficacité des modèles de vectorisation basés sur les réseaux de neurones profonds dans la lutte contre les spams. Ces performances remarquables renforcent la validité et l'applicabilité de notre approche.

CHAPITRE 7 CONCLUSION ET PERSPECTIVES

CHAPITRE 7 CONCLUSION ET PERSPECTIVES

Avec le surgissement de ces avancées technologiques éblouissantes et la prolifération frénétique des gadgets électroniques, les smartphones ont fait irruption, déclenchant une authentique révolution technologique. Cette avancée a prodigieusement métamorphosé notre manière de traiter, d'acquérir et de disséminer des informations entre nous, tout en exécutant avec célérité des tâches quotidiennes telles que les achats en ligne. Cependant, au fil du temps, nos smartphones se métamorphosent en des forteresses numériques, abritant nos données sensibles, y compris les informations bancaires. Cette dualité engendre des interrogations cruciales quant à la sécurité de notre existence numérique, malgré l'indispensabilité de ces dispositifs dans notre vie contemporaine. Ces smartphones, malgré leur propension à faciliter les échanges via l'option SMS, ont regrettablement ouvert la porte à des pratiques frauduleuses. Devant la simplicité d'utilisation de cette fonctionnalité, les individus malintentionnés ont intensifié leurs tentatives de s'approprier des informations, exploitant une brèche béante dans notre expérience numérique.

Face à l'aggravation constante des messages indésirables au quotidien, il devient crucial de mettre en place des classifications dédiées à la lutte contre ce phénomène. C'est dans cette optique que nous avons embrassé une approche novatrice, mobilisant l'apprentissage machine et des techniques de traitement automatique du langage naturel pour ériger des systèmes robustes capables d'endiguer de manière efficace cette menace numérique.

Les résultats issus de cette démarche se révèlent assez captivants. En utilisant les méthodes de vectorisation : Word2Vec, FastText, BERT et GloVe, ces modèles métamorphosent le texte en une représentation numérique, facilitant ainsi son traitement par des algorithmes d'apprentissage automatique, notamment le SVM (Machine à vecteurs de support) dans notre étude de cas.

L'objectif majeur était d'évaluer les performances de ces modèles de vectorisation couplés à l'algorithme SVM pour la détection de spam. Toutes les approches ont exhibé des performances élevées, notamment BERT. Sa capacité à appréhender les nuances sémantiques complexes le distingue, renforçant ainsi son efficacité dans la détection de spam. Cependant, il est crucial de noter que, malgré les performances exceptionnelles de BERT dans la détection de spam, un inconvénient majeur a été observé lors de son utilisation. Le temps d'entraînement significativement plus long par rapport aux autres modèles évalués peut être considéré comme une limitation importante, en particulier dans des scénarios où l'efficacité temporelle est cruciale. Cette considération doit être prise en compte lors du choix d'un modèle pour la détection de spam, en pesant soigneusement les avantages de la précision élevée de BERT par rapport aux contraintes de temps d'entraînement. Par ailleurs, FastText a montré une performance remarquable avec des résultats impressionnants et un temps d'entraînement rapide, ce qui en fait une solution efficace pour la classification de texte, notamment dans des situations où la rapidité est essentielle. Il peut donc être vu comme une alternative pratique à BERT, offrant un juste équilibre entre précision et efficacité temporelle. En ce qui concerne Word2Vec, malgré des performances légèrement inférieures à celles de FastText, elles restent robustes grâce à sa capacité à saisir la sémantique des mots dans un contexte, particulièrement bénéfique pour la détection de spam. Bien que le temps d'entraînement de Word2Vec puisse être plus long que celui de FastText, mais moins que celui de BERT, il représente ainsi une option intermédiaire avec un compromis entre précision et durée d'entraînement. De plus, GloVe a démontré des performances solides avec un taux légèrement inférieur à BERT et FastText, ce qui lui positionne comme une alternative viable qui offre des résultats solides tout en demandant généralement un temps d'entraînement plus court que BERT.

Au-delà de la détection de spams, nos travaux dévoilent des horizons prometteurs pour des applications étendues et des investigations à venir. Les solutions d'intelligence artificielle peuvent contribuer à la détection et à la prévention du vol

d'identité, tant pour les entreprises que pour les consommateurs. En exploitant le traitement du langage naturel (TALN), ces systèmes ont la faculté d'identifier promptement les activités suspectes, engendrant ainsi une réduction des délais d'enquête et des économies de coûts grâce à l'automatisation du processus. Cette approche renforce de manière significative la sécurité en ligne.

Conjointement, les méthodes de prétraitement et de classification peuvent être mobilisées pour filtrer le contenu inapproprié, contribuant ainsi à édifier des espaces numériques plus sûrs. Cette synergie entre les techniques d'intelligence artificielle offre un potentiel considérable pour optimiser la sécurité et l'intégrité des environnements en ligne.

Dans le cadre de l'évolution technique de notre recherche, l'exploration de méthodes de vectorisation émergentes promet une amélioration continue des performances en exploitant des architectures de réseaux neuronaux avancées. Parmi ces approches novatrices, citons des modèles tels que XLNet, une extension de BERT, et T5, adoptant des approches textuelles universelles, démontrant ainsi la diversité et la progression constante dans le domaine de la représentation vectorielle des données. Nous envisageons également d'enrichir notre approche en travaillant sur des corpus de données plus diversifiés et en explorant d'autres classifieurs tels que le naïf de Bayes, le k-plus proches voisins, le Random Forest, etc.

À l'aube des développements futurs, l'adaptation de nos systèmes à des langues plus complexes que l'anglais, comme la langue arabe, représente un défi colossal. Malgré les obstacles liés au déficit de ressources et à la diversité linguistique, la surmonte de ces défis nécessitera des initiatives stratégiques pour élaborer des ressources dédiées et explorer des techniques d'adaptation de modèles.

En conclusion, notre recherche souligne l'importance cruciale de l'apprentissage automatique et du TALN dans le développement de systèmes de filtrage du spam. Les méthodes de vectorisation utilisées fournissent des représentations numériques robustes, améliorant ainsi l'efficacité des algorithmes de

classification. Ces résultats guideront le développement de systèmes de filtrage du spam plus performants, offrant aux utilisateurs une expérience en ligne plus sécurisée et sans encombrement. Les méthodes de vectorisation, adaptables selon des besoins spécifiques, posent les bases pour des applications étendues dans la sécurité en ligne, ouvrant ainsi des perspectives passionnantes pour des recherches futures. La constante capacitée d'adaptation de ces méthodes les positionne comme des outils clés pour relever les défis complexes de la communication numérique.

Bibliographie:

[1] Malwarebytes. (2023, 22 novembre). Spam | What is spam ? | Definition & Types of Spam. https://www.malwarebytes.com/spam

[2] Federal Trade Commission. Tableau des plaintes. Récupéré sur https://public.tableau.com/app/profile/federal.trade.commission/viz/shared/RJB5HMD5Z

[3] RoboKiller. (2023). The state of the robotext in 2023. Récupéré sur https://www.robokiller.com/robokiller-2023-mid-year-phone-scam-report#The-state-of-the-robotext-in-2023

[4] Ousman, Y. Z. P., & Yassine, Z. (2013). Une nouvelle approche pour la détection des spams se basant sur un traitement de données catégorielles. Library and Archives Canada= Bibliothèque et Archives Canada, Ottawa.

[5] Sabri, A. T., Mohammads, A. H., Al-Shargabi, B., & Hamdeh, M. A. (2010). Developing new continuous learning approach for spam detection using artificial neural network (CLA_ANN). *European Journal of Scientific Research*, 42(3), 525-535.

[6] Jaswal, V., & Sood, N. (2013). Spam detection system using hidden markov model. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(7), 304-308.

[7] Puri, S., Gosain, D., Ahuja, M., Kathuria, I., & Jatana, N. (2013). Comparison and analysis of spam detection algorithms. *International Journal of Application or Innovation in Engineering and Management*, 2(4), 255-261.

[8] Sanz, E. P., Hidalgo, J. M. G., & Pérez, J. C. C. (2008). Email spam filtering. *Advances in computers*, 74, 45-114.

[9] Shrivastava, J. N., & Bindu, M. H. (2014). E-mail spam filtering using adaptive genetic algorithm. *International Journal of Intelligent Systems and Applications*, 6(2), 54-60.

[10] Nosseir, A., Nagati, K., & Taj-Eddin, I. (2013). Intelligent word-based spam filter detection using multi-neural networks. *International Journal of Computer Science Issues (IJCSI)*, 10(2 Part 1), 17.

[11] Das, S., Dey, A., Pal, A., & Roy, N. (2015). Applications of artificial intelligence in machine learning: review and prospect. *International Journal of Computer Applications*, 115(9).

[12] Camastra, F., & Vinciarelli, A. (2015). Machine learning for audio, image and video analysis: theory and applications. Springer.

[13] Gonzalez, F. A., & Romero, E. (Eds.). (2009). Biomedical image analysis and machine learning technologies: Applications and techniques: Applications and techniques.

[14] Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys* (CSUR), 34(1), 1-47.

[15] Saidani, N. (2021). A Learning Approach for Spam Detection Using Semantic Representation (Thèse de doctorat, Université du Québec en Outaouais).

[16] Haddam, N. (2023). Apprentissage par renforcement pour le pilotage énergétique de l'éclairage dans un bâtiment connecté (Doctoral dissertation, Université Paris-Saclay).

[17] Juton, A., Noël, V., & Lali, R. (2022, juillet 20). *Introduction à l'apprentissage par renforcement*. Culture Science de l'Ingénieur, École Normale Supérieure Paris-Saclay.

[18] Yvon, F. (2010). Une petite introduction au traitement automatique des langues naturelles. In *Conference on Knowledge discovery and data mining* (pp. 27-36).

[19] Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, 82(3), 3713-3744.

[20] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.

[21] Jalam, R. (2003). Apprentissage automatique et catégorisation de textes multilingues (Doctoral dissertation, Lyon 2).

[22] Hayes, P. J. (1992). Intelligent high-volume text processing using shallow, domain-specific techniques. Text-based intelligent systems: Current research and practice in information extraction and retrieval, 227-242.

[23] Mooney, R. J., & Bunescu, R. (2005). Mining knowledge from text using information extraction. *ACM SIGKDD explorations newsletter*, 7(1), 3-10.

[24] Yan, R., Jiang, X., Wang, W., Dang, D., & Su, Y. (2022). Materials information extraction via automatically generated corpus. Scientific Data, 9(1), 401.

[25] Glasgow, B., Mandell, A., Binney, D., Ghemri, L., & Fisher, D. (1998). MITA: An information-extraction approach to the analysis of free-form text in life insurance applications. *AI magazine*, 19(1), 59-59.

[26] Hoy, M. B. (2018). Alexa, Siri, Cortana, and more: an introduction to voice assistants. Medical reference services quarterly, 37(1), 81-88.

[27] Uysal, A. K., Gunal, S., Ergin, S., & Gunal, E. S. (2013). The impact of feature extraction and selection on SMS spam filtering. Elektronika ir Elektrotechnika, 19(5), 67-72.

[28] Foozy, C. F. M., Ahmad, R., & Abdollah, M. F. (2014). A framework for SMS spam and phishing detection in malay language: A case study. Int. Rev. Comput. Softw, 9(7), 1248-1254.

[29] Almeida, T. A., Silva, T. P., Santos, I., & Hidalgo, J. M. G. (2016). Text normalization and semantic indexing to enhance instant messaging and SMS spam filtering. Knowledge-Based Systems, 108, 25-32.

[30] Reaves, B., Blue, L., Tian, D., Traynor, P., & Butler, K. R. (2016, July). Detecting SMS spam in the age of legitimate bulk messaging. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks* (pp. 165-170).

[31] Arifin, D. D., & Bijaksana, M. A. (2016, September). Enhancing spam detection on mobile phone Short Message Service (SMS) performance using FP-growth and Naive Bayes Classifier. In *2016 IEEE Asia Pacific Conference on Wireless and Mobile* (APWiMob) (pp. 80-84). IEEE.

[32] Singh, A., & Batra, S. (2018). Ensemble based spam detection in social IoT using probabilistic data structures. *Future Generation Computer Systems*, 81, 359-371.

[33] Gupta, M., Bakliwal, A., Agarwal, S., & Mehndiratta, P. (2018, August). A comparative study of spam SMS detection using machine learning classifiers. In *2018 eleventh international conference on contemporary computing* (IC3) (pp. 1-7). IEEE.

[34] Sjarif, N. N. A., Azmi, N. F. M., Chuprat, S., Sarkan, H. M., Yahya, Y., & Sam, S. M. (2019). SMS spam message detection using term frequency-inverse document frequency and random forest algorithm. *Procedia Computer Science*, 161, 509-515.

[35] Prasanna Bharathi, P., Pavani, G., Krishna Varshitha, K., & Radhesyam, V. (2021). Spam SMS filtering using support vector machines. In *Intelligent Data Communication Technologies and Internet of Things: Proceedings of ICICI 2020* (pp. 653-661). Springer Singapore.

[36] Guo, Y., Mustafaoglu, Z., & Koundal, D. (2023). Spam detection using bidirectional transformers and machine learning classifier algorithms. *Journal of Computational and Cognitive Engineering*, 2(1), 5-9.

[37] Joseph, P., & Yerima, S. Y. (2022, December). A comparative study of word embedding techniques for SMS spam detection. In *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)* (pp. 149-155). IEEE.

[38] Aliza, H. Y., Nagary, K. A., Ahmed, E., Puspita, K. M., Rimi, K. A., Khater, A., & Faisal, F. (2022, March). A comparative analysis of SMS spam detection employing machine learning methods. In *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)* (pp. 916-922). IEEE.

[39] Sireesha, S. A., Karthik, S. B., Srena, K., Gopal, S. N., & Reddy, S. K. (2023). SMS Spam Detection Using Machine Learning. *Scandinavian Journal of Information Systems*, 35(1), 749-754.

[40] Suparna, K., & Vuddaraju, V. T. S. V. (2023). SMS Spam Detection Using Machine Learning. *Journal of Engineering Sciences*, 14(9).

[41] Hanchate, M. S. (2023). SMS Spam Classification Using Machine Learning. Thèse de master, California State University San Marcos, Département d'informatique.

[42] Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992, July). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144-152).

[43] Guyon, I., Boser, B., & Vapnik, V. (1992). Automatic capacity tuning of very large VC-dimension classifiers. *Advances in neural information processing systems*, 5.

[44] DJEFFAL, A. (2012). Utilisation des méthodes Support Vector Machine (SVM) dans l'analyse des bases de données (Doctoral dissertation, Université Mohamed Khider-Biskra).

[45] Karush, W. (1939). Minima of functions of several variables with inequalities as side constraints. M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago.

[46] Kuhn, H. W., & Tucker, A. W. (1951). Nonlinear programming. *In Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability* (pp. 481-492). Berkeley, CA: University of California Press.

[47] B. Scholkopf and A.J. Smola. (2002). Learning with Kernels Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press.

[48] Mercer, J. (1909). Xvi. functions of positive and negative type, and their connection the theory of integral equations. Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character, 209(441-458), 415-446.

[49] H. König. (1986). Eigenvalue distribution of compact operators. Birkhäuser Verlag BaselBostonStuttgart.

[50] Herbrich, R. (2001). Learning kernel classifiers: theory and algorithms. MIT press.

[51] Soman, K. P., Loganathan, R., & Ajay, V. (2009). Machine learning with SVM and other kernel methods. PHI Learning Pvt. Ltd.

[52] Rakotomalala, R. SVM: Support Vector Machine - Machines à Vecteurs de Support - Séparateurs à Vaste Marge. Université Lumière Lyon 2.

[53] Sutton, A. (2021). Concepts in word embeddings: theory and applications (Doctoral dissertation, University of Bristol).

[54] Desagulier, G. (2018, avril 25). Word embeddings: The (very) basics. Around the word. https://doi.org/10.58079/n4uj

[55] Yun-tao, Z., Ling, G., & Yong-cheng, W. (2005). An improved TF-IDF approach for text classification. *Journal of Zhejiang University-Science A*, 6, 49-55.

[56] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[57] Meyer, D. (2016). How exactly does word2vec work?. Uoregon. Edu, Brocade. Com, 1-18.

[58] Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759.

[59] Cécillon, N., Dufour, R., & Labatut, V. (2021). Approche multimodale par plongements de texte et de graphes pour la détection de messages abusifs. Revue TAL, 62(2), 13-38.

[60] Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).

[61] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 6000–6010.

[63] Almeida,Tiago and Hidalgo,Jos. (2012). SMS Spam Collection. UCI Machine Learning Repository. https://doi.org/10.24432/C5CC84.

[64] Remy, E., Verges, V., Dautreme, E., & Martin-Cabanas, B. (2020, October). Revue des principales approches de résolution du problème de déséquilibre des classes. In Congrès Lambda Mu 22 «*Les risques au cœur des transitions»(e-congrès)-22e Congrès de Maîtrise des Risques et de Sûreté de Fonctionnement, Institut pour la Maîtrise des Risques.*

[65] Brownlee, J. (2020). Random oversampling and undersampling for imbalanced classification. Machine learning mastery.

[66] Budu, E., & Budu, E. (2023, 16 mars). What does pre-training a neural network mean ? | Baeldung on Computer Science. Baeldung on Computer Science. https://www.baeldung.com/cs/neural-network-pre-training

[67] Google Code Archive - Long-term storage for Google Code Project Hosting. https://code.google.com/archive/p/word2vec/

[68] Word vectors for 157 languages · FastText . https://fasttext.cc/docs/en/crawl-vectors.html

[69] Pennington, J. (2014). GLOVE: Global Vectors for Word Representation. https://nlp.stanford.edu/projects/glove/

[70] Bert-base-uncased · Hugging face. https://huggingface.co/bert-base-uncased

[71] Sklearn.model_selection.GridSearchCV. scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[72] Le tutoriel Python. Python documentation. https://docs.python.org/fr/3/tutorial/

[73] Natural Language Toolkit (NLTK). NLTK 3.6.2 documentation. (Consulté en 2023).

[74] NumPy. https://numpy.org/.

[75] Pandas - Python Data Analysis Library. https://pandas.pydata.org/.

[76] SciKit-Learn : Machine Learning in Python — SciKit-Learn 1.4.0 documentation. https://scikit-learn.org/stable/index.html

[77] TensorFlow. https://www.tensorflow.org/?hl=fr

[78] Transformers — Transformers 4.0.0 Documentation. https://huggingface.co/transformers/v4.0.1/index.html

[79] Fasttext. PyPI. https://pypi.org/project/fasttext/

[80] Torch. PyPI. https://pypi.org/project/torch/

[81] Joblib. PyPI. https://pypi.org/project/joblib/

[82] PyQT5. PyPI. https://pypi.org/project/PyQt5/

[83] Kohavi, R., & Provost, F. (1998). Glossary of terms. Machine Learning, 30(2-3), 271-274. https://doi.org/10.1023/A:1017181826899.

[84] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4), 427-437.

[85] Soussia, A. B. (2022). *Analyse prédictive des données d'apprentissage, en situation d'enseignement à distance* (Doctoral dissertation, Université de Lorraine).

[86] Fogarty, J., Baker, R. S., & Hudson, S. E. (2005, May). Case studies in the use of ROC curve analysis for sensor-based estimates in human computer interaction. *In Proceedings of Graphics Interface 2005* (pp. 129-136).

[87] Decout. (2007). Prédiction de la distribution d'alliances de végétation des milieux ouverts d'altitude à l'aide de l'approche dite du maximum d'entropie. Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Interpretation-de-la-courbe-ROC-source-Decout-2007_fig5_235948539 [accessed 30 Jan, 2024]."