REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

Université Saad Dahleb blida 1 Faculté des Sciences
Département d'Informatique

M E M O I R E

Pour obtention du diplôme de

Master

Spécialité « Ingénierie du Logiciel »

Thème :

Fouille D'opinions par Deep Learning

Réalisé et présenté

Par

ANIS HENTIT et **HACHMI HADJALA**

Soutenu le 14 juillet 2021 devant le jury composé de :

M. A. Zahra Fatma Zohra	Université de Blida 1	Présidente
M. B. Kameche Abdallah Hicham	Université de Blida 1	Examinateur
M. A. Chikhi Nacim	Université de Blida 1	Promoteur

Table des matières

Table des Figures

Remerciements

Nous commencerons par remercier nos familles et amis pour nous avoir soutenu et encouragé durant la réalisation de ce projet.

Nous adressons notre reconnaissance tout particulièrement à Mr CHIKHI Nassim pour avoir été notre promoteur de projet.

Nous le remercions ; pour nous avoir guidé sans nous diriger, pour sa pertinence, l'apport de ses compétences, son enthousiasme et sa disponibilité.

Nous tenons tout aussi à remercier notre camarade KERBOUCHE Mahfoudh qui nous a fait part de son savoir et de son expérience dans le domaine de l'apprentissage automatique.

Nous tenons à remercier les membres du jury pour avoir pris le temps d'examiner et évaluer notre travail.

Nous n'oublierons pas les professeurs qui nous ont suivi tout au long de notre parcours universitaire, nous leur sommes reconnaissants pour leurs efforts et le savoir qu'ils nous ont transmis.

Résumé

Les avis des clients jouent un rôle très important dans la vie quotidienne. Lorsque nous devons prendre une décision, l'opinion d'autres personnes est également prise en compte. Aujourd'hui, de nombreux internautes publient leurs avis sur de nombreux produits par le biais de blogs, de sites d'évaluation et des réseaux sociaux.

Les organisations commerciales et les entreprises sont toujours désireuses de connaître l'avis des consommateurs ou des particuliers sur leurs produits, leur assistance et leurs services.

Dans le domaine du commerce électronique, des achats en ligne et du tourisme en ligne, il est essentiel d'analyser automatiquement la grande quantité de données sociales présentes sur le Web de manière automatique, il est donc très important de créer des méthodes qui les classent automatiquement. L'extraction d'opinions, parfois appelée classification des sentiments, est définie comme l'extraction et l'analyse de critiques, de points de vue, d'émotions et d'opinions automatiquement à partir de textes, de données volumineuses et de discours au moyen de diverses méthodes.

Dans ce mémoire, nous allons proposer une approche originale de fouille d'opinions basée sur le Deep Learning qui sera évaluée et comparée à d'autres approches en utilisant des jeux de données réels.

À cet effet, nous proposons d'utiliser un modèle basé sur une architecture a couches neurale récurrente pour la détection du sentiment d'un texte. Les résultats obtenus après l'évaluation de notre modèle se sont avérés prometteurs.

Mots-clés : Fouille d'opinions, Deep Learning, Apprentissage Automatique, Intelligence Artificielle

Abstract

Customer opinions play a very important role in our daily lives. When we have to make a decision, other people's opinions are also taken into account. Nowadays, many people post their opinions on many products through blogs, review sites and social networks.

Business organizations and companies are always eager to get feedback from consumers or individuals on their products, support and services.

In the field of e-commerce, online shopping and online tourism, it is essential to analyze the large amount of social data present on the web automatically, so it is very important to create methods that automatically rank them. Opinion mining, sometimes called sentiment classification, is defined as the extraction and analysis of reviews, views, emotions, and opinions automatically from text, big data, and speech using various methods.

In this thesis, we will propose a novel opinion mining approach based on Deep Learning that will be evaluated and compared to other approaches using real datasets.

To this end, we propose to use a model based on a recurrent neural layer architecture for text sentiment detection. The results obtained after the evaluation of our model were promising.

Keywords: Opinion mining, Deep Learning, Machine Learning, Artificial Intelligence

ملخص

تلعب آراء العملاء دورًا مهمًا للغاية في حياتنا اليومية. عندما يتعين علينا اتخاذ قرار ، يتم أيضًا أخذ آراء الآخرين في الاعتبار. في الوقت الحاضر ، ينشر العديد من الأشخاص آرائهم على العديد من المنتجات من خلال المدونات ومواقع المراجعة والشبكات الاجتماعية .

تحرص المؤسسات والشركات التجارية دائمًا على الحصول على تعليقات من المستهلكين أو الأفراد حول منتجاتهم ودعمهم وخدماتهم .

في مجال التجارة الإلكترونية والتسوق عبر الإنترنت والسياحة عبر الإنترنت ، من الضروري تحليل الكمية الكبيرة من البيانات الاجتماعية الموجودة على الويب تلقائيًا ، لذلك من المهم جدًا إنشاء طرق تقوم بترتيبها تلقائيًا. يُعرّف التنقيب عن الآراء ، والذي يُطلق عليه أحيانًا تصنيف المشاعر ، بأنه استخراج وتحليل المراجعات والآراء والعواطف والآراء تلقائيًا من النص والبيانات الضخمة والكلام باستخدام طرق مختلفة .

في هذه الأطروحة ، سنقترح نهجًا جديدًا للتنقيب عن الرأي يعتمد على التعلم العميق الذي سيتم تقييمه ومقارنته بالنُهج الأخرى باستخدام مجموعات بيانات حقيقية.

تحقيقًا لهذه الغاية ، نقترح استخدام نموذج يعتمد على بنية عصبية متكررة لاكتشاف المشاعر النصية كانت النتائج التي تم الحصول عليها بعد تقييم نموذجنا واعدة.

الكلمات المفتاحية: **التنقيب عن الرأي ، التعلم العميق ، التعلم الآلي ، الذكاء الاصطناعي**

Introduction

L'industrie ou les entreprises manufacturières qui produisent de nouveaux produits veulent savoir ce que leurs clients pensent de ces produits et cette information peut être obtenue en étudiant les avis qui sont postés sur des sites d'évaluation de leurs produits et services [1].

Dans le même temps, les utilisateurs ou les consommateurs veulent aussi savoir quel produit acheter, alors ils lisent également les avis concernant les produits qui les intéressent et essaient de prendre leurs décisions en se basant sur ces avis.

Il est évident que les opinions en ligne gagnent en popularité de jour en jour. Ces opinions représentent une mine d'informations qui peuvent être bénéfiques pour l'industrie ainsi que pour les consommateurs [2].

Cependant, faire ce travail manuellement est une tâche ennuyeuse et fastidieuse. De ce fait, les entreprises préfèrent l'information dans un format plus facile à utiliser, de sorte que l'automatisation de ce processus est très utile [3]. Sur le web, les opinions peuvent être exprimées sous forme de texte, d'image, de données audio ou vidéo. Dans ce mémoire, nous nous intéressons à l'exploration de texte car il s'agit d'un domaine largement étudié.

L'extraction d'opinions peut être définie comme une sous-discipline de la linguistique informatique qui s'intéresse à l'opinion exprimée dans un document [4] [5].

La classification des sentiments consiste à déterminer la subjectivité, la polarité (positive/négative) et la force de la polarité (faiblement positive, légèrement positive ou fortement positive) d'un texte d'opinion [2] [6]. Plusieurs termes ont été utilisés par les chercheurs pour définir la classification des sentiments : l'exploration d'opinion, l'analyse des sentiments, l'extraction des sentiments ou la notation affective [6]. Pour des raisons de simplicité dans ce mémoire, notre utilisation du terme "Opinion Mining" englobe toutes ces significations.

Le travail présenté dans ce mémoire vise à répondre à la problématique « Fouille d'opinions par Deep Learning » par l'élaboration d'un modèle basé sur une architecture a couches neurales récurrentes qui pourra classer un texte selon sa polarité.

Le mémoire est organisé en quatre principaux chapitres.

Dans le **Chapitre 1**, nous présentons la ***problématique « Fouille d'opinions »***. Nous commençons par présenter les différents domaines d'application de cette technologie dans une première phase, qui sera suivie par un état de l'art général sur les différentes techniques existantes.

Dans le **Chapitre 2**, nous présentons la ***problématique « Fouille d'opinions par Deep Learning »***.

Dans un premier temps, nous expliquons le concept d'apprentissage automatique et quelques notions de base. Ensuite, nous faisons le tour des techniques basées sur le Deep Learning pour la fouille d'opinions.

Dans le **Chapitre 3**, nous présentons notre approche de l'exploration d'opinions à partir de commentaires de clients et expliquons les choix pour lesquels nous avons opté.

Dans le **Chapitre 4,** nous donnons les résultats obtenus pour l'ensemble des données contenant les avis des clients qui sont publiés sur des sites d'avis en ligne.

Chapitre 1 Fouille d'opinions

1.1 Domaines étroitement liés à la fouille d'opinions

Extraction d'informations (IE) : Il s'agit de la transformation d'informations textuelles non structurées en un format structuré qui est généralement stocké dans des bases de données et peut être utilisé à des fins d'exploration de données (**Data mining**) basées sur l'apprentissage machine. Des données spécifiques sont extraites des corpus et s'insèrent dans un modèle existant. Cela améliore la précision des informations extraites et peut être utilisé comme base pour catégoriser les données extraites. La reconnaissance d'entités nommées (NER) est une méthode par laquelle les entités nommées sont identifiées dans le texte. Il peut être utilisé comme condition préalable à l'extraction d'informations et améliore la récupération d'informations en indexant et en interrogeant les bases de données [7].

Traitement du langage naturel (NLP) : il s'agit des processus que les ordinateurs utilisent pour convertir le langage humain (écrit / parlé) en connaissances utiles ou pratiques qu'un ordinateur peut comprendre et utiliser tout en interagissant avec d'autres ordinateurs. La NLP consiste à traiter du texte en utilisant des connaissances lexicales, syntaxiques et sémantiques. La NLP est également une sous-discipline de l'intelligence artificielle [7] [8].

Apprentissage machine (ML) : L'apprentissage machine ou simplement l'apprentissage automatique (ML), fait référence à des processus qui impliquent la construction et l'évolution de dictionnaires machine qui modélisent le comportement, les pensées et les réponses humains [8]. Il existe diverses tâches d'exploration de données pouvant être effectuées : l'apprentissage supervisé (classification), l'apprentissage non supervisé (clustering), l'exploration de règles d'association et l'exploration séquentielle de modèles [7].

Exploration de données Web : nous introduisons l'exploration de données car elle a été largement appliquée dans les tâches d'exploration d'opinion. L'exploration de données est également appelée découverte de données ou de connaissances dans les bases de données. C'est la découverte de connaissances utiles à partir de sources de données telles que les bases de données et le Web [7]. Cela implique divers domaines tels que l'apprentissage automatique, les statistiques, les bases de données, l'intelligence artificielle, la récupération d'informations et la visualisation de données [7].

1.2 Vue d'ensemble de la notion de fouille d'opinions

L'objectif principal de l'exploration d'opinions est d'extraire des opinions à partir de textes non structurés à l'aide d'algorithmes, statistiques ou un mélange des deux techniques [7]. L'analyse des sentiments est essentiellement concernée par les éléments fondamentaux suivants :

- L'entité ou la cible en cours d'évaluation (par exemple, un restaurant)
- L'attribut de la cible auquel l'opinion est dirigée (par exemple, la qualité du service ou la qualité de la nourriture)
- La polarité de l'opinion (par exemple positif, négatif ou neutre)
- Le porte-parole (par exemple le consommateur individuel) et la date à laquelle l'avis a été émis

Formellement, une opinion est définie comme un tuple ($\mathbf{e_i;a_{ij};h_k;s_{ijkl};t_l}$) où $\mathbf{e_i}$ est la ième entité, $\mathbf{a_{ij}}$ désigne le jème aspect de l'entité $\mathbf{e_i}$ vers lequel l'opinion est dirigée, $\mathbf{h_k}$ est le kème détenteur d'opinion, $\mathbf{s_{ijkl}}$ est la polarité (ou sentiment) de l'opinion de $\mathbf{h_k}$ envers l'aspect $\mathbf{a_{ij}}$ de l'entité $\mathbf{e_i}$ au temps $\mathbf{t_l}$ [7]. Un document d'opinion **D** contient les opinions d'un ensemble de détenteurs d'opinion sur un certain nombre d'entités. Par conséquent, le principal objectif de l'analyse des sentiments est de trouver tous les tuples d'opinions ($\mathbf{e_i;a_{ij};h_k;s_{ijkl};t_l}$) dans un document donné ou sur un ensemble de documents. Comme indiqué précédemment, la polarité d'opinion $\mathbf{s_{ijkl}}$ peut être généralement définie en termes de trois niveaux : positif, négatif ou neutre [8].

La Figure 1 montre un aperçu des processus de l'extraction d'opinions. Il est logiquement dérivé d'une analyse critique de la recherche existante dans le domaine de la fouille d'opinions. Nous notons qu'il existe deux méthodes principales par lesquelles une opinion peut être extraite. Naturellement, la première étape consiste à connaître l'élément pour lequel un avis est requis grâce à l'extraction de l'élément. Cela ne donne qu'une opinion globale positive / négative sur l'élément, sans aucun détail spécifique sur ce qui est réellement considéré. On ne sait rien des avantages et des inconvénients des fonctionnalités [9].

Cependant, pour connaître le sentiment de l'élément, en fonction des sentiments de ses fonctionnalité, une extraction de fonctionnalité doit être effectuée. Cela permet de faire la différence entre les bonnes et les mauvaises fonctionnalités [9].

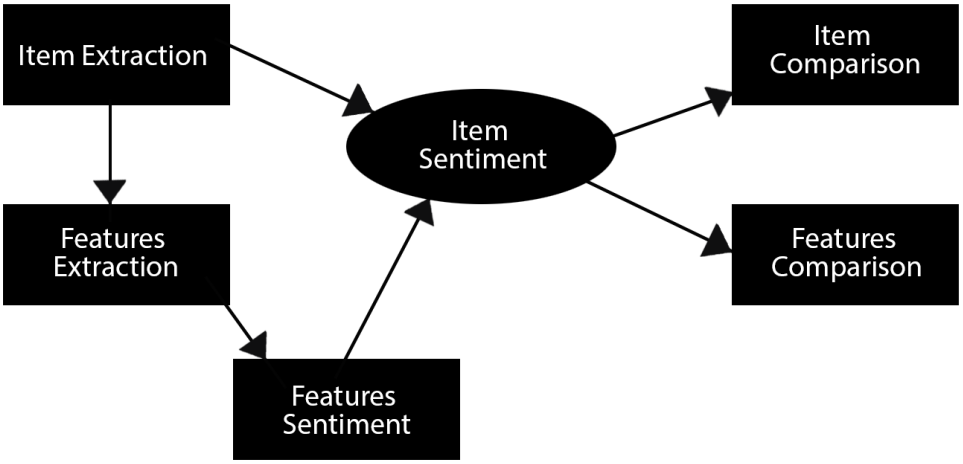

Figure 1 Opinion mining Framework

A l'étape 1, bien que l'on puisse savoir que l'élément A est bon/mauvais, il n'y a aucune justification apparente à cela. Cependant, à l'étape 2, on peut savoir pourquoi l'élément A est bon/mauvais. Prenons l'exemple de la caméra A, alors que l'étape 1 peut nous informer que la caméra est bonne/mauvaise, l'étape 2 peut nous informer sur la fonctionnalité de zoom, l'autonomie de la batterie, le poids et la taille de la caméra. Sur la base de ces détails supplémentaires, une opinion plus subjective peut être déduite [9].

Il est intéressant de noter que les chercheurs ont largement concentré leurs efforts sur le sentiment des éléments, mais que des recherches minimes ont été menées sur l'extraction des éléments [8].

En outre, comme le montre la Figure 1, la sortie de l ’ « item sentiment » peut être utilisée comme entrée pour la comparaison d'éléments et la comparaison de caractéristiques. La Figure 1, nous montre qu'un évaluateur peut avoir la possibilité de comparer l'item A avec l'item B ou les caractéristiques de l'item A avec celles de l'item B. Dans le reste de cette section, nous présentons une étude approfondie des 6 domaines de ce Framework [9].

1.2.1 Extraction d'éléments (Item extraction)

Tels qu'utilisés dans le contexte de ce document, les termes : entité et objet ont la même connotation qu’item. Le nom de l'élément est essentiel pour connaître le domaine d'application d'un terme de recherche [9]. L'extraction d'article fait référence à l'objet mentionné dans la revue, par ex. Appareil photo, téléphone portable, lecteur MP3, ordinateur portable ou film. Nous considérons l'extraction d'items comme critique pour extraire des opinions car elle nous aide à connaître l'item dont l'opinion est extraite. Malheureusement, peu de documentation est

disponible dans ce domaine. Cependant, [10] fourni une littérature complète sur l'extraction non supervisée d'entités Web [9].

1.2.2 Extraction de caractéristiques (Feature extraction)

Cette approche va plus loin que l'approche précédente en termes de recherche d'une granularité plus fine de l'opinion. La motivation de base de l'approche basée sur les caractéristiques est qu'une opinion négative d'un client sur un produit ne signifie pas nécessairement que ce client n'aime pas tous les aspects de ce produit, et vice versa [9].

Cette approche modélise un produit constitué d'un certain nombre de sous-composants, dont chacun est associé à un ensemble d'attributs qui peuvent être évalués au moyen d'expressions d'opinion. Considérons un appareil photo numérique, qui a plusieurs caractéristiques comme illustré dans [2]: qualité d'image, autonomie de la batterie, zoom, taille, poids. Par exemple, un appareil photo avec une qualité d'image inférieure peut avoir une très longue durée de vie de la batterie et un poids léger. Une étape très importante de l'approche basée sur les fonctionnalités consiste à extraire des fonctionnalités pour différents produits et services [2].

L'extraction de caractéristiques est une activité qui peut être effectuée après l'extraction de l'élément afin de savoir pourquoi un élément a été catégorisé en tant que positif ou négatif. Plutôt que de simplement établir une orientation sémantique des termes utilisés pour exprimer une opinion et en extraire une opinion basée sur cela, l'extraction de caractéristiques passe à un niveau inférieur à l'extraction d'éléments en recherchant les caractéristiques d'éléments dont l'opinion peut être utilisée pour extraire le sentiment de caractéristique [2]. Il intéressant de noter que bien que plusieurs travaux aient été effectués dans ce domaine, ils ne sont pas aussi complets que ceux de l'Item sentiment, mais certainement plus complets que ce qui a été fait pour l'extraction des items. Un travail notable a été effectué par [3] qui a formellement défini et modélisé le processus d'extraction de caractéristiques pour l'exploration d'opinion [9].

1.2.3 Sentiment de caractéristique (Feature sentiment)

Le sentiment de fonctionnalité (caractéristique) est l'opinion exprimée pour un élément en fonction de ses fonctionnalités. Une fois les fonctionnalités identifiées, un sentiment de fonctionnalité peut être exprimé pour chaque fonctionnalité, ce qui nous dit quelque chose sur les points les plus faibles et les plus forts des fonctionnalités d'un élément. Par exemple, longue durée de vie de la batterie, taille portable, belles couleurs, excellente distribution, mauvais acteurs [9]. Un point d'intérêt ici est le cas de la critique de film. Les recherches montrent qu'il est difficile pour les algorithmes actuels de faire la distinction entre les opinions du

film en général et celles des acteurs [9]. Dans certains cas, un film peut recevoir une critique positive parce qu'un acteur populaire est impliqué, mais le film lui-même peut être inintéressant. Avec des scénarios de cette nature, il devient nécessaire de différencier les critiques du film de celles des acteurs. [11] Ont réussi à développer un algorithme qui peut être utilisé pour extraire les sentiments des caractéristiques pour les critiques en japonais, agissant ainsi comme un précédent pour de futures recherches sur les algorithmes d'extraction de sentiments indépendants de la langue [9].

1.2.4 Sentiment d'élément (Item sentiment)

Cela fait référence au sentiment général exprimé sur l'objet, par exemple, recommandé / non recommandé, bon / mauvais, acheter / ne pas acheter, excellent film / film ennuyeux [9]. Ceci est très utile si une opinion dominante doit être rapidement connue. Une grande partie de la littérature se concentre sur la découverte du sentiment de l'item. Les chercheurs se sont montrés très intéressés par ce domaine. Notamment, [12] a présenté un paradigme couramment utilisé qui fournit une base pour extraire l'opinion d'un élément. Les termes utilisés dans les revues sont supposés être subjectifs et peuvent être divisés en trois groupes, positifs, négatifs ou neutres. La plupart des recherches se sont concentrées sur la pondération de la subjectivité des termes positifs et négatifs pour extraire les sentiments de la critique tout en ignorant les sentiments neutres. Cependant, d'autres chercheurs pensent que les termes neutres (objectifs) doivent être pris en considération et peuvent améliorer l'exactitude des résultats [9].

1.2.5 Comparaison d'élément (Item comparison)

La comparaison d'éléments représente une granularité plus fine d'une opinion entre l'entité A et d'autres entités, par exemple, la caméra AB par rapport à la caméra XY. Les caméras AB ont tendance à avoir des critiques plus positives que les caméras XY. Ceci est important pour les acheteurs qui n'ont peut-être pas le temps de magasiner pour le meilleur prix et le meilleur produit à acheter. Les acheteurs peuvent gagner beaucoup de temps dans les processus de prise de décision pour les articles banals, réalisant ainsi une double économie en termes de temps et d'argent. [11] ont développé une nouvelle façon de représenter graphiquement les comparaisons.

1.2.6 Comparaison de fonctionnalités (Feature comparison)

La comparaison des fonctionnalités sert à la comparaison entre l'entité A et d'autres entités. À titre d'exemple, on peut vouloir savoir comment les fonctionnalités d'une caméra PQ se comparent à celles d'une caméra XY. Voici un exemple de comparaison de fonctionnalités : la caméra PQ a tendance à avoir une durée de vie de la batterie plus longue et une fonctionnalité de zoom supérieure à celle de la caméra XY. Certaines innovations qui ont eu lieu dans ce domaine consistent à afficher graphiquement toutes les fonctionnalités d'un produit [9]. Sans aucun

doute, cela ajoute à l'expérience d'achat en affichant tous les avantages et inconvénients entre les caractéristiques des articles. De plus, les fabricants peuvent facilement identifier les fonctionnalités qui ne répondent pas aux attentes et aux besoins des consommateurs [9].

1.3 Domaines d'application de la fouille d'opinions

L'exploration d'opinions peut être utilisée dans divers domaines pour répondre à des objectifs variés. Dans cette section, nous profitons de l'occasion pour présenter quelques-unes des plus courantes. Les exemples présentés dans cette section ne sont pas exhaustifs mais simplement un aperçu des possibilités.

1.3.1 Shopping

L'utilisation la plus populaire de l'exploration d'opinion est peut-être l'aide à la décision pour les consommateurs. Les consommateurs participent activement aux achats comparatifs sur Internet [9]. Les sites Web populaires comme amazon.com permettent aux clients d'exprimer leurs opinions sur leurs sites Web. Les clients peuvent facilement consulter les avis sur les produits et identifier comment les fonctionnalités entre les produits se comparent les unes aux autres. Dans certains cas, après qu'une opinion ait été extraite et traitée, les connaissances sont présentées à l'utilisateur dans un format graphique pour une comparaison facile des caractéristiques du produit. Nous examinerons les commentaires sur un produit électronique d'une boutique en ligne [9].

« J'avais besoin d'un ordinateur portable de haute puissance pour répondre aux besoins de mon entreprise. Dell a offert une variété de produits qui répondaient à mes exigences. Leurs informations sur les produits ont été expliquées de manière concise et complète, ce qui a rendu mon processus de sélection très facile. Leur site Web était facile à utiliser et présenté de manière attrayante. La marchandise a été livrée à temps, conformément à leur promesse. Les frais d'expédition étaient à un prix très raisonnable. J'ai pu répondre à tous mes besoins en ligne, évitant ainsi d'avoir recours à leur service d'assistance téléphonique. Leur aide et leur support en ligne étaient excellents. Je recommanderais Dell Small Business à tout le monde. » [9].

Nous notons que l'opinion porte sur Dell Small Business et que les différentes fonctionnalités dont il est question sont les délais de livraison, les frais d'expédition, l'assistance et la navigation sur le site Web.

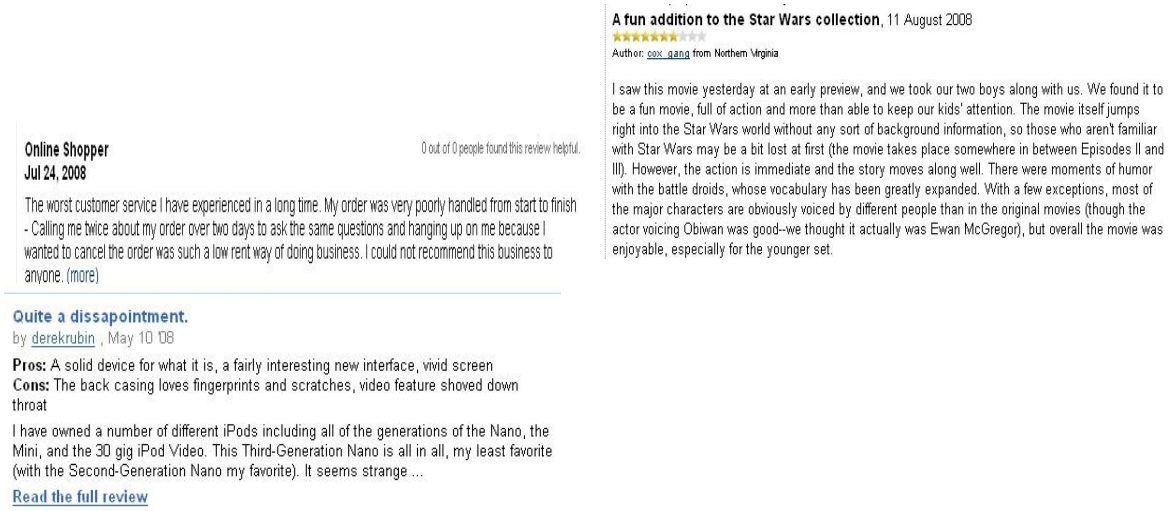

Figure 2 extraits d'avis sur le web

1.3.2 Divertissement

Les cinéphiles et les téléspectateurs à domicile peuvent accéder rapidement à l'opinion sur les sorties récentes et les films et programmes populaires. Actuellement, il existe la base de données de films sur Internet (IMDB) qui fournit des critiques en ligne de films ainsi que de programmes télévisés. Cela sert de guide aux personnes qui ne savent pas quels films regarder. Nous présentons un extrait de l'IMDB. “Christopher Nolan's second bundle of joy "The Dark Knight" EXCEEDED all of my expectations!!! I can HONESTLY tell you that: as good as Jack Nicholson was in Batman'89 he is CHILD'S PLAY compared to this Joker. He is sadistic, psychotic, and downright SCARIER and PSYCHOLOGICALLY disturbing than the previous incarnation of The Clown Prince of Crime and Ledger gives it his all to do him justice. The action is great, and the plot is deeper and engrossing. [13]” [9].

1.3.3 Gouvernement

Les gouvernements peuvent exploiter les opinions dominantes sur les politiques publiques. Les candidats aux élections peuvent devenir plus informés sur les spécificités du sondage d'opinion. Ces connaissances peuvent aider les politiciens à identifier leurs forces et leurs faiblesses en fonction de leur électorat. Considérez les opinions politiques suivantes qui ont été exprimées. « Attendez-vous à plus d'inflation. Plus de chômage. Vraiment, nous avons besoin d'un meilleur processus de sélection. Qui choisit ces personnes ? Ils entrent dans l'histoire en augmentant

les taux pour la première fois avant une élection. Je n'ai aucune confiance dans les chiffres publiés. » [14].

Un rapide coup d'œil à ces termes indique un sentiment d'insatisfaction parmi l'électorat. En outre, les principaux sujets de préoccupation sont abordés en fonction de ce qui manque et des attentes.
Les questions de politique publique classent normalement les électeurs dans l'un des trois groupes suivants : pour, contre ou neutre. Un bon exemple est la déclaration : « Je pense que tout cela semble extrêmement dur. L'ennui, au contraire, est un signe d'intelligence. [14] Une déclaration de ce genre indique clairement que l'opinion est pour la requête. L'avantage de l'exploration d'opinions par rapport aux sondages d'opinion traditionnels comme les sondages téléphoniques est qu'on peut déterminer pourquoi les électeurs sont pour ou contre une proposition [9]. La plupart des sites Web, en particulier ceux dont l'objectif fondamental est de fournir des informations, permettent aux internautes d'exprimer leurs opinions sur leurs sites Web.

1.3.4 Recherche et développement

Les revues de produits peuvent être utilisées par les entreprises de fabrication pour améliorer les fonctionnalités et fournir une plate-forme pour l'innovation. Les applications basées sur le Web pourraient offrir des plates-formes aux clients pour concevoir des produits et soumettre les conceptions aux entreprises de fabrication. Une approche de cette nature pourrait considérablement aider à établir des fonctionnalités appréciées par les clients. Considérez la critique suivante pour un produit électronique, "La molette cliquable est horrible et manque complètement de réponse et de sensibilité." [15] C'est une opinion négative exprimée à propos de la molette cliquable. L'utilisation de lettres majuscules signifie au lecteur l'étendue de la déception. Si l'exploration d'opinions est capable de détecter des émotions de ce type exprimées dans un texte évaluatif, elle s'avérera très bénéfique. Cela servira d'indicateur sur la façon dont le produit a été reçu par un consommateur. Cependant, après avoir exprimé des opinions négatives sur les caractéristiques du produit, une déclaration telle que « bien que je sois vraiment déçu, c'est probablement toujours le meilleur lecteur de musique haute capacité du marché [15] ». Cette affirmation positive indique aux services marketing que le lecteur de musique est toujours le meilleur du marché et que c'est la haute capacité qui est privilégiée par les clients [9].

1.3.5 Commercialisation (Marketing)

Les entreprises peuvent désormais faire des économies sur les dépenses de marketing en demandant des avis sur leurs sites Web et leurs sites Web d'évaluation [16]. Cela élimine la nécessité pour les consultants commerciaux de mener des

enquêtes, car les entreprises peuvent désormais disposer de toutes les données dont elles ont besoin en ligne. L'avènement d'Internet a apporté avec lui de nouvelles méthodes de marketing. Nous présentons le marketing viral à titre d'exemple. Le marketing viral consiste à utiliser les réseaux sociaux pour diffuser des informations sur les produits et services [16]. Avec l'avènement d'Internet, les réseaux sociaux tels que **MySpace**, **Facebook** et **Hi5** proposent une nouvelle plateforme d'échange d'informations. La famille et les amis peuvent désormais se recommander des produits / services ou rechercher davantage de connaissances sur un produit ou un service avant de s'engager.
Pour encourager les publications et les recommandations entre pairs, les spécialistes du marketing offrent normalement des incitations comme des remises pour les recommandations qui se transforment en achats [16].

1.4 Niveaux d'analyse des sentiments

L'analyse des sentiments a été étudiée principalement à trois niveaux :

Au niveau des documents : L'analyse des sentiments au niveau des documents classe l'opinion des documents dans son ensemble en différents sentiments, pour un produit ou un service. Ce niveau classe les documents d'opinion en sentiment positif, négatif ou neutre [9].

Au niveau des phrases : L'analyse des sentiments au niveau des phrases détermine si chaque phrase exprime une opinion positive, négative ou neutre pour un produit ou un service. Ce type est utilisé pour les avis et commentaires contenant une phrase et rédigés par l'utilisateur [14] [9].

Au niveau d'entité et d'aspect : L'analyse des sentiments au niveau d'aspect est l'extraction d'opinions et la synthèse basées sur la fonctionnalité. La classification concerne l'identification et l'extraction des caractéristiques du produit à partir des données sources. Ce type est utilisé lorsque nous avons besoin de sentiments sur l'aspect / la fonctionnalité souhaitée dans un avis [9].

1.5 Processus d'analyse des sentiments

Le processus de base de l'analyse des sentiments consiste en une série d'étapes préliminaires comprenant l'acquisition de données, le prétraitement du texte et la sélection des fonctionnalités. Ces étapes initiales sont suivies par un processus de

classification des sentiments. Le flux de processus global d'une exploration de sentiments ou d'opinions typique est décrit dans la Figure 3 [9].

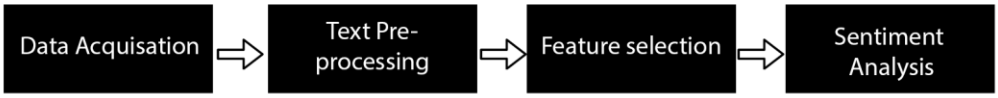

Figure 3 Flux du processus de base de l'extraction de sentiments

1.6 Classification des solutions existantes

Les travaux existants sur l'analyse des sentiments peuvent être classés à partir de différents points de vue : techniques utilisées, vue du texte, niveau de détail de l'analyse du texte, niveau de lecture, etc. [17].

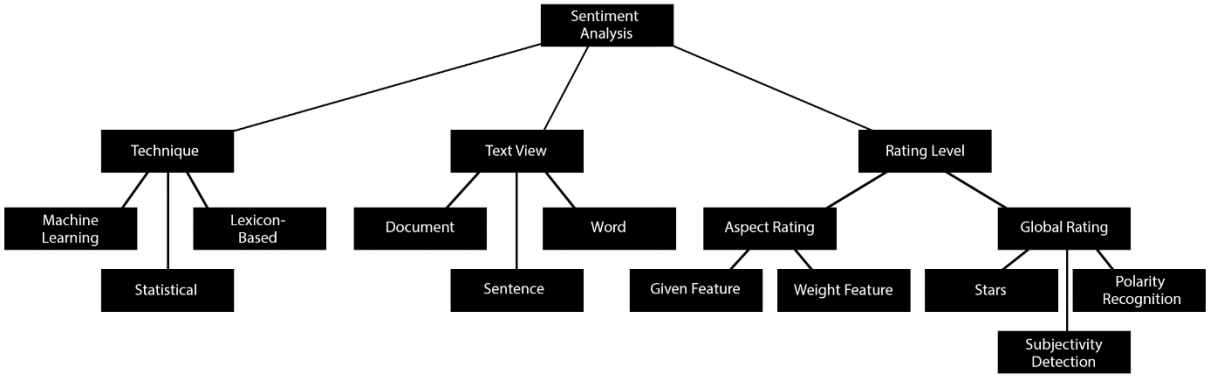

Figure 4 Classification des analyses des sentiments

- La méthode d'apprentissage automatique utilise plusieurs algorithmes d'apprentissage pour déterminer le sentiment en s'entraînant sur un ensemble de données connu [17].
- L'approche basée sur le lexique consiste à calculer la polarité des sentiments pour une critique en utilisant l'orientation sémantique des mots ou des phrases dans la critique. L ‘ « orientation sémantique » est une mesure de la subjectivité et de l'opinion dans le texte [18].
- L'approche basée sur des règles recherche les mots d'opinion dans un texte, puis le classe en fonction du nombre de mots positifs et négatifs. Elle considère différentes règles de classification telles que la polarité du dictionnaire, les mots de négation, les mots d'appoint, les expressions idiomatiques, les émoticônes, les opinions mitigées, etc. [18].
- Les modèles statistiques représentent chaque texte exprimant une opinion comme un mélange d'aspects latents et de notes. On suppose que les aspects et leurs notes peuvent être représentés par des distributions multinomiales

et essayer de regrouper les termes principaux en aspects et les sentiments en évaluations [17].

Il existe une autre classification qui est d'avantage orientée sur la structure du texte : niveau document, classification au niveau de la phrase ou au niveau des mots / caractéristiques. Cette classification a le même principe que celui expliqué dans **1.4**. La Figure 4 présente une classification détaillée des méthodes existantes. Cette classification n'est pas exclusive. Une solution peut entrer dans plus d'une catégorie.

1.7 Approches de la Fouille d'opinions

Les approches de la Fouille d'opinions peuvent être décomposées en deux catégories majeures : ***l'approche basée sur le Machine Learning et l'approche basée sur le lexique*** [18] , la Figure 5 met en évidence ces deux approches en détail :

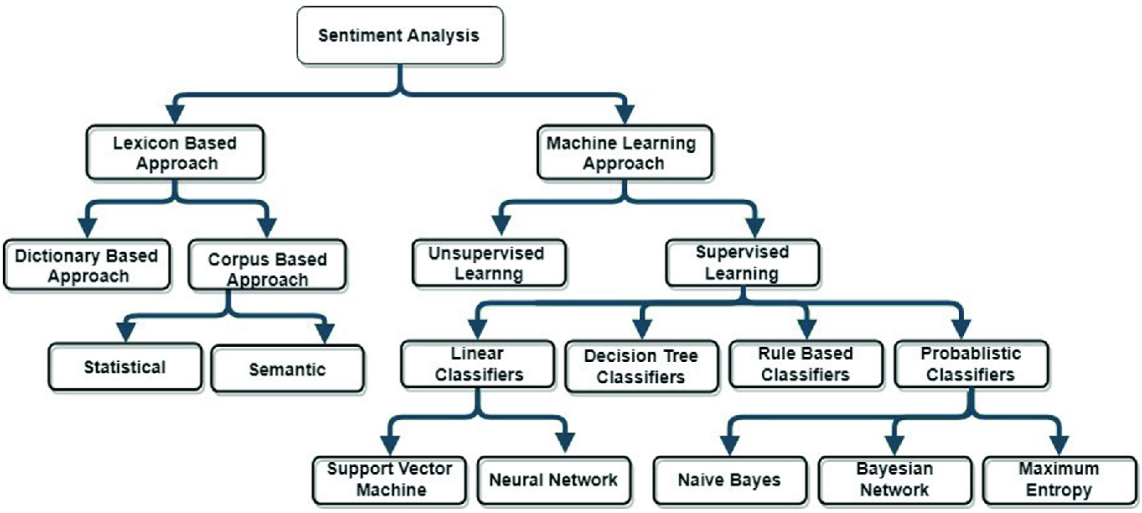

Figure 5 Approches de la fouille d'opinions [18]

1.7.1 Approche basée sur le lexique (Lexicon based approach)

Cette technique est régie par l'utilisation d'un dictionnaire composé de lexiques préétiquetés. Le texte d'entrée est converti en jetons par le Tokenizer. Chaque nouveau jeton rencontré est comparé ensuite au lexique du dictionnaire. S'il y a une correspondance positive, le score est ajouté au pool total de scores pour le texte d'entrée. Par exemple, si « dramatique » est une correspondance positive dans le dictionnaire alors le score total du texte est incrémenté. Autrement dans le cas contraire, le score est décrémenté ou le mot est marqué comme négatif. Bien que cette technique semble être de nature amateur, ses variantes se sont avérées dignes ([19], [20]). La Figure 6 montre le fonctionnement de la technique lexicale.

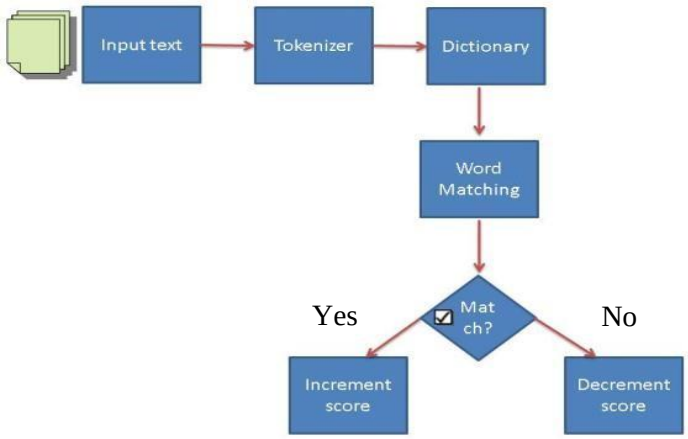

Figure 6 Travail d'une technique lexicale [21]

La classification d'un texte dépend de la note totale qu'il obtient. Un travail considérable a été consacré à la mesure des meilleures informations lexicales qui fonctionnent. Une précision d'environ 80% sur des phrases simples peut être obtenue par l'utilisation de lexiques marqués à la main composés uniquement d'adjectifs, qui sont cruciaux pour décider de la subjectivité d'un texte évaluatif comme le démontre [21]. L'auteur de [22] a étendu ce travail en utilisant la même méthodologie et a testé une base de données de critiques de films, a rapporté une précision de seulement 62%. Autre que l'approche lexique étiqueté à la main, [12] est venu avec une variante en utilisant le moteur de recherche Internet pour marquer la polarité des mots inclus dans le travail de [22]. Ils ont utilisé deux requêtes du moteur de recherche AltaVista: mot cible + «bon» et autre mot cible + «mauvais». Le score a été évalué par la recherche qui a produit le nombre maximal de résultats, ce qui a rapporté une amélioration de la précision antérieure de 62% à 65%. Dans des recherches ultérieures ([20], [23]), la notation des mots a été réalisée en utilisant la base de données WordNet. Ils ont comparé le mot cible avec deux mots pivots («bon» et «mauvais») et ont trouvé la distance minimale de chemin (MPD) entre les mots de la pyramide WordNet. Le MPD est converti en un score incrémentiel, qui est stocké dans le dictionnaire de mots. Cette variante a donné une précision [23] de 64%.

L'auteur de [19] propose une autre méthode qui présente une alternative à ([20], [23]), en prenant la motivation de [21], en évaluant l'écart sémantique entre les mots en soustrayant simplement l'ensemble des positifs des négatifs donnant une précision de 82%. L'analyse lexicale a une limite : ses performances (en termes de complexité temporelle et de précision) se dégradent drastiquement avec la croissance exponentielle de la taille du dictionnaire (nombre de mots).

1.7.2 Analyse basée sur l'apprentissage automatique

L'apprentissage automatique est l'une des techniques les plus importantes qui suscite l'intérêt des chercheurs en raison de son adaptabilité et de sa précision. Dans l'analyse des sentiments, on utilise principalement les variantes d'apprentissage supervisé de cette technique. Il comprend trois étapes : la collecte des données, le prétraitement, les données de formation, la classification et le traçage des résultats. Dans les données d'apprentissage, une collection de corpus étiquetés est fournie. Le classificateur est présenté une série de vecteurs de caractéristiques à partir des données précédentes. Un modèle est créé sur la base de l'ensemble de données d'apprentissage qui est utilisé sur le texte nouveau / invisible à des fins de classification. Dans la technique d'apprentissage automatique, la clé de la précision d'un classificateur est la sélection des fonctionnalités appropriées. En général, les uni-grammes (phrases de mot unique), les bi-grammes (deux phrases consécutives), les trigrammes (trois phrases consécutives) sont sélectionnés comme vecteurs de caractéristiques. Il existe une variété de fonctionnalités proposées à savoir le nombre de mots positifs, le nombre de mots négatifs, la longueur du document, les machines à vecteurs de support (SVM) ([24], [25]) et l'algorithme Naïve Bayes (NB) [26]. La précision varie de 63% à 80% en fonction de la combinaison de diverses caractéristiques sélectionnées.

Le fonctionnement de cette technique peut être expliqué comme suit :

Première étape : Collecte de données

A ce stade, les données à analyser sont explorées à partir de diverses sources telles que les blogs, les réseaux sociaux (Twitter, MySpace, etc.) en fonction du domaine d'application [27].

Deuxième étape : Prétraitement

À cette étape, les données acquises sont nettoyées et préparées pour être introduites dans le classificateur. Le nettoyage comprend l'extraction des mots-clés et des symboles. Par exemple, les émoticônes sont les smileys utilisés sous forme textuelle pour représenter les émotions, par ex. ":-)", ":)", "=)", ": D", ":-(", ":(", "= (", "; (", etc.) Correction de toutes les majuscules et tout en minuscules dans une casse commune, en supprimant les textes non anglais (ou en langue proposée), en supprimant les espaces blancs et les tabulations inutiles, etc. [27].

Troisième étape : Données d'entrainement

Une collection de données étiquetées à la main est préparée par la méthode de crowdsourcing la plus couramment utilisée. Ces données sont le carburant du classificateur ; elles alimenteront l'algorithme à des fins d'apprentissage [27].

Quatrième étape : Classification

C'est le cœur de toute la technique. En fonction des besoins de l'application, SVM ou Naïve bayes est déployé pour l'analyse. Le classificateur (après avoir terminé la

formation) est prêt à être déployé sur les tweets / texte en temps réel à des fins d'extraction de sentiments [27].

Cinquième étape : Résultats

Les résultats sont tracés en fonction du type de représentation sélectionné, c'est-à-dire des tableaux, des graphiques, etc. Le réglage des performances est effectué avant la publication de l'algorithme [27].

La technique d'apprentissage automatique est confrontée à des défis dans : la conception d'un classificateur, la disponibilité des données d'entraînement, l'interprétation correcte d'une phrase imprévue. Il surmonte la limitation de l'approche lexicale de la dégradation des performances et fonctionne bien même lorsque la taille du dictionnaire augmente de manière exponentielle [27].

1.7.3 Analyse hybride

Les progrès de l'analyse des sentiments ont incité les chercheurs à explorer la possibilité d'une approche hybride qui pourrait collectivement montrer la précision d'une approche d'apprentissage automatique et la vitesse de l'approche lexicale. Dans [28], les auteurs utilisent des lexiques de deux mots et des données non étiquetées, divisant ces lexiques de deux mots en deux classes discrètes négatives et positives. Des pseudo documents englobant tous les mots de l'ensemble de lexiques choisis sont créés. Puis on calcule la similitude cosinus entre les pseudos documents et les documents non étiquetés. Selon la mesure de la similitude, les documents se sont vu attribuer un sentiment positif ou négatif. Cet ensemble de données de formation a ensuite été transmis à un classificateur bayes naïf à des fins de formation.
Une autre approche présentée par [29], a dérivé un « cadre unifié » utilisant des informations lexicales d'arrière-plan comme associations de classes de mots. Les auteurs ont renouvelé les informations pour des domaines particuliers en utilisant les ensembles de données disponibles ou des exemples de formation et ont proposé un classificateur appelé Polling Multinomial Classifier (PMC) (également connu sous le nom de bayes naïves multinomiales). Des données étiquetées manuellement ont été incorporées à des fins de formation. Ils ont affirmé que l'utilisation des connaissances lexicales améliorait les performances. Une autre variante de cette approche a été présentée par [30]. Mais jusqu'à présent, seuls [29] ont pu revendiquer de bons résultats.

Chapitre 2 Fouille d'opinions et Deep Learning

Le Deep Learning est devenu une technique d'apprentissage automatique puissante qui apprend plusieurs couches de représentations ou de caractéristiques des données et produit des résultats de prédiction de pointe. Parallèlement au succès du Deep Learning dans de nombreux autres domaines d'application, ce dernier est également couramment utilisé dans l'analyse des sentiments ces dernières années. Cette partie donne d'abord un aperçu de l'apprentissage en profondeur, puis fournit une étude complète de ses applications actuelles dans l'analyse des sentiments.

2.1 Vue d'ensemble de la notion d'apprentissage automatique

L'apprentissage en tant que processus générique consiste à acquérir de nouveaux comportements, valeurs, connaissances, compétences ou préférences ou à les modifier. Le comportementalisme, le cognitivisme, le constructivisme, l'expérience et l'apprentissage social définissent la théorie de l'apprentissage personnel, c'est-à-dire comment les humains apprennent.

Contrairement à ce qui vient naturellement aux humains, les machines s'appuient sur des données : l'apprentissage par expérience. Au niveau très fondamental, l'apprentissage automatique (ML) est une catégorie d'intelligence artificielle qui permet aux ordinateurs de penser et d'apprendre par eux-mêmes. Il s'agit de faire en sorte que les ordinateurs modifient leurs actions afin d'améliorer les actions pour atteindre plus de précision, où la précision est mesurée en termes de nombre de fois que les actions choisies se transforment en actions correctes [31].

Le domaine de l'apprentissage automatique a pour objectif de savoir comment construire des programmes informatiques qui s'améliorent automatiquement avec l'expérience. Ces dernières années, de nombreuses applications d'apprentissage automatique à succès ont été développées, allant des programmes d'exploration de données qui apprennent à détecter les transactions frauduleuses par carte de crédit, aux systèmes de filtrage d'informations qui apprennent les préférences de lecture des utilisateurs, aux véhicules autonomes qui apprennent à conduire sur les routes publiques. Dans le même temps, il y a eu des avancées importantes dans la théorie et les algorithmes qui constituent les fondements de ce domaine [31].

2.2 Apprentissage automatique

L'apprentissage automatique est un domaine multidisciplinaire ayant un large éventail de domaines de recherche renforçant son existence. Celles-ci sont illustrées dans la Figure 7. La simulation des modèles ML est significativement liée aux statistiques computationnelles dont l'objectif principal est de se concentrer sur la réalisation de prédictions via des ordinateurs. Il est également co-lié à l'optimisation mathématique qui relie les modèles, les applications et les cadres au domaine des statistiques. Les problèmes du monde réel ont une complexité élevée, ce qui en fait d'excellents candidats pour l'application du ML [31].

L'apprentissage automatique peut être appliqué à divers domaines de l'informatique pour concevoir et programmer des algorithmes explicites avec une sortie haute performance, par exemple, le filtrage des courriers indésirables, la détection de fraude sur les réseaux sociaux, négociation d'actions en ligne, détection de visage et de forme, diagnostic médical, prévision du trafic, reconnaissance de caractères et recommandation de produits [31]. Les voitures autonomes de Google, Netflix présentant les films et les émissions qu'une personne pourrait aimer, les moteurs de recommandation en ligne - comme les suggestions d'amis sur Facebook, « plus d'articles à considérer » et « obtenez-vous un petit quelque chose » sur Amazon, et la détection des fraudes par carte de crédit, sont tous des exemples concrets d'application de l'apprentissage automatique [31].

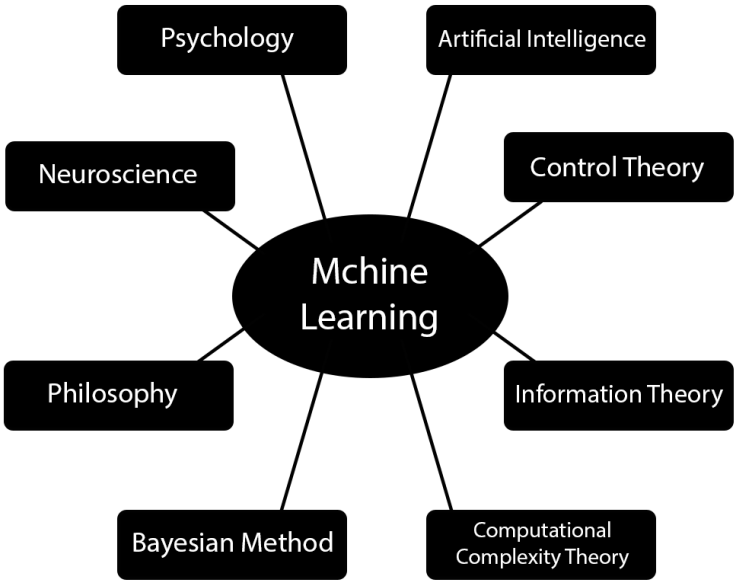

Figure 7 Disciplines multiples du ML

2.3 Méthodes de l'apprentissage automatique

Ces jours-ci, depuis le développement de nouvelles technologies informatiques dans le domaine du Big Data, l'apprentissage automatique n'est pas comme il l'était dans le passé. Aujourd'hui, de nombreux algorithmes d'apprentissage automatique ont été développés [32], mis à jour et améliorés. Le développement récent de l'apprentissage automatique octroie la capacité d'appliquer automatiquement une variété de calculs mathématiques complexes à un big data, qui calcule les résultats beaucoup plus rapidement. La programmation adaptative est très populaire. Elle est utilisée dans l'apprentissage automatique où les applications sont capables de reconnaître des modèles, d'apprendre par expérience, d'extraire de nouvelles informations à partir de données ou d'optimiser la précision et l'efficacité de leur traitement et de leur sortie. En outre, les techniques d'apprentissage automatique [33] sont utilisées pour travailler avec des données multidimensionnelles qui sont présentes dans divers domaines d'application. Ainsi, en fonction du résultat souhaité de l'algorithme, les algorithmes d'apprentissage automatique sont organisés dans les groupes suivants (voir Figure 8) :

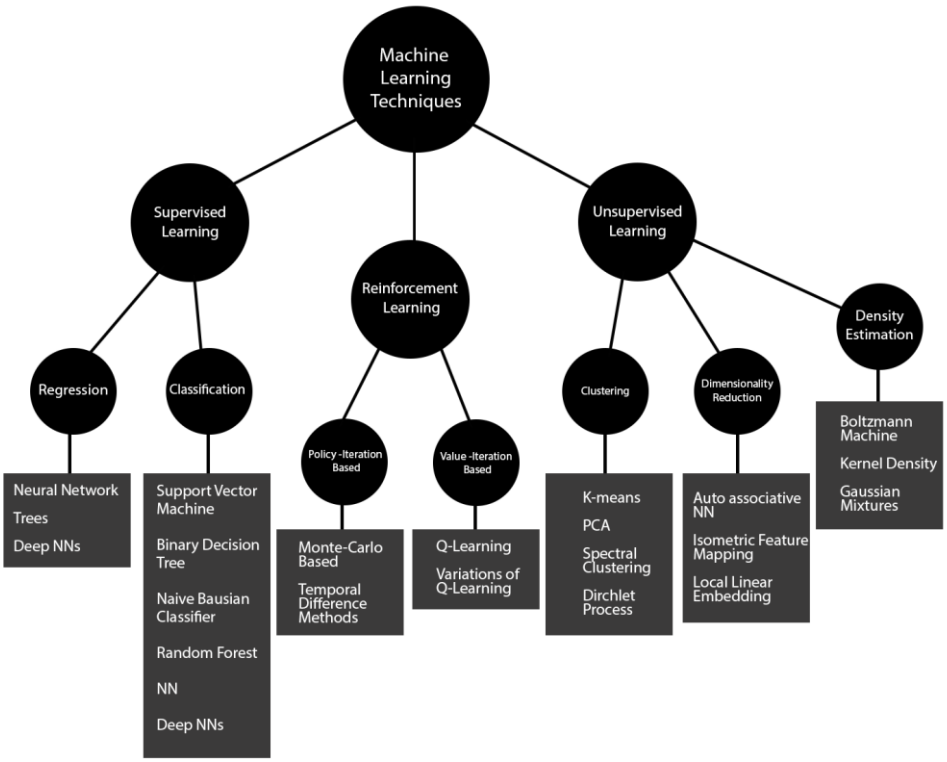

Figure 8 techniques de l'apprentissage automatique

Apprentissage supervisé : les différents algorithmes génèrent une fonction qui mappe les entrées aux sorties souhaitées. Une formulation standard de la tâche d'apprentissage supervisé est le problème de classification : l'apprenant doit apprendre (pour approximer le comportement de) une fonction qui mappe un vecteur dans l'une de plusieurs classes en examinant plusieurs exemples d'entrée-sortie de la fonction [33].

Apprentissage non supervisé : modélise un ensemble d'entrées : les exemples étiquetés ne sont pas disponibles [33].

Apprentissage semi-supervisé : combine des exemples étiquetés et non étiquetés pour générer une fonction ou un classificateur approprié [33].

Apprentissage par renforcement : l'algorithme apprend une politique sur la façon d'agir compte tenu d'une observation du monde. Chaque action a un impact sur l'environnement, et l'environnement fournit des commentaires qui guident l'algorithme d'apprentissage [33].

Outre ces groupes, Les techniques d'apprentissage automatique sont essentiellement divisées en deux groupes généraux, l'apprentissage supervisé et non supervisé.

2.4 Apprentissage supervisé

L'apprentissage supervisé est un paradigme d'apprentissage automatique pour acquérir les informations de relation entrée-sortie d'un système [34]. L'apprentissage supervisé implique l'apprentissage d'un mappage entre un ensemble de variables d'entrée X et une variable de sortie Y et appliquer ce mappage pour prédire les sorties de données nouvelles. L'apprentissage supervisé est la méthodologie la plus importante de l'apprentissage automatique et il a également une importance centrale dans le traitement des données multimédias [35].

Les techniques d'apprentissage automatique supervisé tentent de découvrir la relation entre les attributs d'entrée (variables indépendantes) et un attribut cible (variable dépendante). Les techniques supervisées peuvent en outre être classées en deux catégories principales : classification et régression [36].

2.4.1 Régression

Il est très important de collecter, d'analyser et de modéliser une grande quantité de données afin d'effectuer des prédictions précieuses dans divers domaines. L'apprentissage par régression est utilisé comme modèle de prédiction [37]. Les valeurs de la variable dépendante sont prédites par le modèle de régression basé sur les valeurs des variables indépendantes. Par régression, si après l'expérience E, le programme améliore ses performances P, on dit alors que le programme effectue un apprentissage par régression [37].

2.4.2 Classification

La classification est une approche d'exploration de données (apprentissage automatique) utilisée pour prévoir l'appartenance à un groupe pour les instances de données [38]. Bien qu'il existe une variété de techniques disponibles pour l'apprentissage automatique, la classification est la technique la plus largement utilisée [39]. La classification est une tâche admirée dans l'apprentissage automatique, en particulier dans les plans futurs et la découverte de connaissances.

Bien que la classification soit une technique bien connue dans l'apprentissage automatique, elle souffre de problèmes tels que la gestion des données manquantes. Des valeurs manquantes dans l'ensemble de données peuvent poser problème pendant les phases d'apprentissage et de classification. Certaines des raisons potentielles de données manquantes sont présentées dans [40] comprennent : Non saisie du dossier en raison d'une idée fausse, données reconnues non pertinentes au moment de la saisie, suppression des données en raison d'un écart avec d'autres données documentées et d'un dysfonctionnement de l'équipement.
La classification est classée comme l'un des problèmes les plus étudiés par les chercheurs des domaines de l'apprentissage automatique et de l'exploration de données [8]. Un modèle général d'apprentissage supervisé (techniques de classification) est présenté dans la Figure 9 ci-dessous.

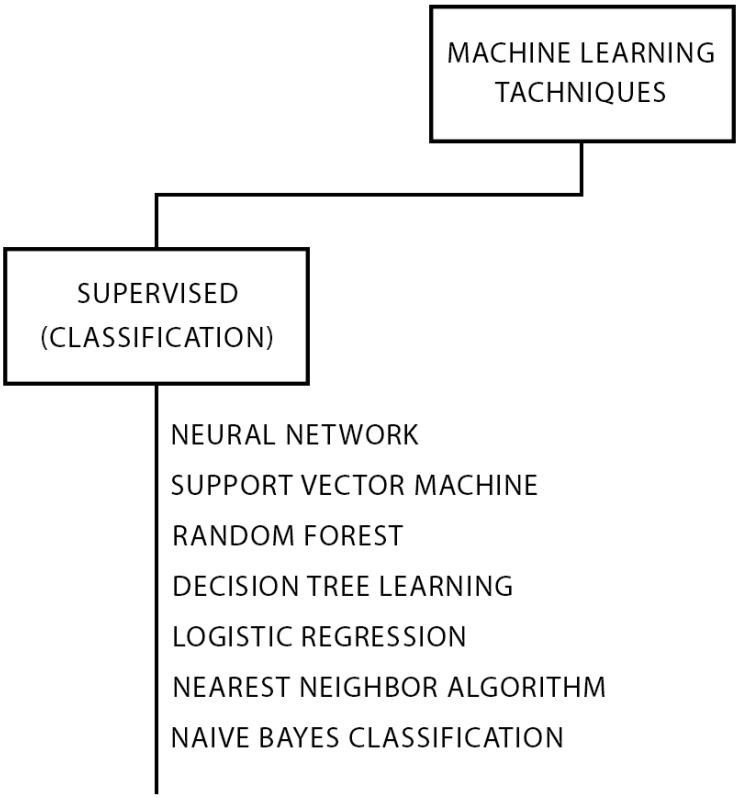

Figure 9 Modèle générale de classification

Les techniques de classification peuvent être catégorisées en deux catégories (paramétrique et non-paramétrique) comme présenté dans la Figure 10 ci-dessous.

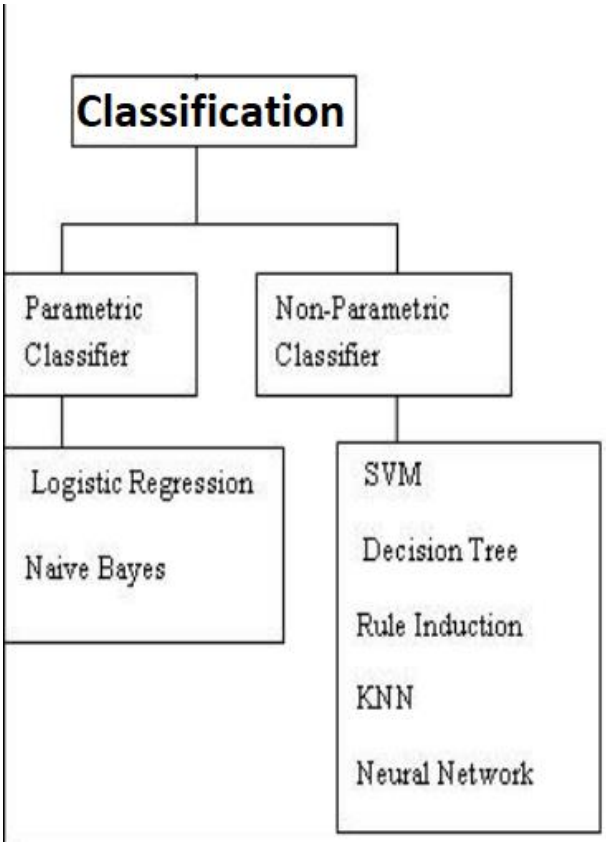

Figure 10 catégories des techniques de classification [46]

Classificateur paramétrique :

Les hypothèses peuvent grandement simplifier le processus d'apprentissage, mais peuvent également limiter ce qui peut être appris. Les algorithmes qui simplifient la fonction sous une forme connue sont appelés algorithmes d'apprentissage automatique paramétriques [46]. Un modèle d'apprentissage qui résume les données avec un ensemble de paramètres de taille fixe (indépendamment du nombre d'exemples d'apprentissage) est appelé un modèle paramétrique. Quelle que soit la quantité de données que vous fournissez à un modèle paramétrique, il ne changera pas d'avis sur le nombre de paramètres dont il a besoin [41].

Les algorithmes impliquent deux étapes [46] :

1. Sélectionner un formulaire pour la fonction.
2. Apprendre les coefficients de la fonction à partir des données d'entraînement.

Une forme fonctionnelle facile à comprendre pour la fonction de mappage est une ligne, comme elle est utilisée dans la régression linéaire : **b0** + **b1*****x1** + **b2*****x2** = 0

Où **b0**, **b1** et **b2** sont les coefficients de la ligne qui contrôlent l'intersection et la pente, et **x1** et **x2** sont deux variables d'entrée.

Prendre la forme fonctionnelle d'une ligne simplifie grandement le processus d'apprentissage. Il ne nous reste plus qu'à estimer les coefficients de l'équation linéaire et nous avons un modèle prédictif du problème.

Avantages des algorithmes d'apprentissage automatique paramétriques [46] :

- Plus simple : ces méthodes sont plus faciles à comprendre et à interpréter les résultats.
- Vitesse : les modèles paramétriques sont très rapides à apprendre à partir des données.
- Moins de données : ils ne nécessitent pas autant de données d'entraînement et peuvent bien fonctionner même si l'ajustement aux données n'est pas parfait.

Limitations des algorithmes d'apprentissage automatique paramétriques [46] :

- Contrainte : en choisissant une forme fonctionnelle, ces méthodes sont fortement contraintes à la forme spécifiée.
- Complexité limitée : les méthodes sont plus adaptées à des problèmes plus simples.
- Mauvais ajustement : en pratique, il est peu probable que les méthodes correspondent à la fonction de mappage sous-jacente.

Classificateur non-paramétrique :

Les algorithmes qui ne font pas d'hypothèses solides sur la forme de la fonction de mappage sont appelés algorithmes d'apprentissage automatique non paramétriques. En ne faisant pas d'hypothèses, ils sont libres d'apprendre n'importe quelle forme fonctionnelle à partir des données d'entraînement. Les méthodes non paramétriques sont utiles lorsque l'on a beaucoup de données et aucune connaissance préalable, et que l'on ne voudrait pas trop se soucier de choisir les bonnes fonctionnalités [41].

Les méthodes non paramétriques cherchent à mieux adapter les données d'apprentissage lors de la construction de la fonction de mappage, tout en conservant une certaine capacité à généraliser aux données invisibles. En tant que tels, ils sont capables de s'adapter à un grand nombre de formes fonctionnelles [46].

Un modèle non paramétrique facile à comprendre est l'algorithme des k plus proches voisins qui effectue des prédictions basées sur les k modèles d'apprentissage les plus similaires pour une nouvelle instance de données. La méthode ne suppose rien sur la forme de la fonction de mappage autre que les modèles qui sont proches sont susceptibles d'avoir une variable de sortie similaire [46].
Voici quelques autres exemples d'algorithmes d'apprentissage automatique non paramétriques populaires [46] :

- K-Voisins les plus proches
- Arbres de décision comme CART et C4.5

- SVM

Avantages des algorithmes d'apprentissage automatique non paramétriques [46] :

- Flexibilité : Capable de s'adapter à un grand nombre de formes fonctionnelles.
- Puissance : Aucune hypothèse (ou hypothèses faibles) sur la fonction sous-jacente.
- Performances : peut entraîner des modèles de prédiction plus performants.

Limitations des algorithmes d'apprentissage automatique non paramétriques [46] :

- Plus de données : nécessite beaucoup plus de données d'entraînement pour estimer la fonction de cartographie.
- Plus lent : beaucoup plus lent à s'entraîner car ils ont souvent beaucoup plus de paramètres à entraîner.
- Sur-ajustement : il y a plus de risque de sur-ajustement des données d'entraînement et il est plus difficile d'expliquer pourquoi des prédictions spécifiques sont faites.

2.5 Machine à vecteurs de support

Les machines à vecteur de support (SVM) (Figure 11) sont un ensemble de méthodes d'apprentissage supervisé connexes utilisées pour la classification et la régression. Ils appartiennent à une famille de classification linéaire généralisée. Une propriété spéciale des SVM est que les SVM minimisent simultanément l'erreur de classification empirique et maximisent la marge géométrique [42]. Les SVM sont donc appelés classificateurs à marge maximale. Les SVM sont basés sur la minimisation du risque structurel (SRM). Les SVM cartographient le vecteur d'entrée dans un espace de dimension supérieure où un hyperplan de séparation maximale est construit. Deux hyperplans parallèles sont construits de chaque côté de l'hyperplan qui sépare les données [43]. L'hyperplan de séparation est l'hyperplan qui maximise la distance entre les deux hyperplans parallèles [43]. On suppose que plus la marge ou la distance entre ces hyperplans parallèles est grande, meilleure sera l'erreur de généralisation du classificateur. Le SVM est une technique discriminante et, comme il résout le problème d'optimisation convexe de manière analytique, il renvoie toujours le même paramètre d'hyperplan optimal, contrairement aux algorithmes génétiques (AG) ou aux perceptrons, tous deux largement utilisés pour la classification dans l'apprentissage automatique. Pour les perceptrons, les solutions dépendent fortement des critères d'initialisation et de terminaison [43] [42].

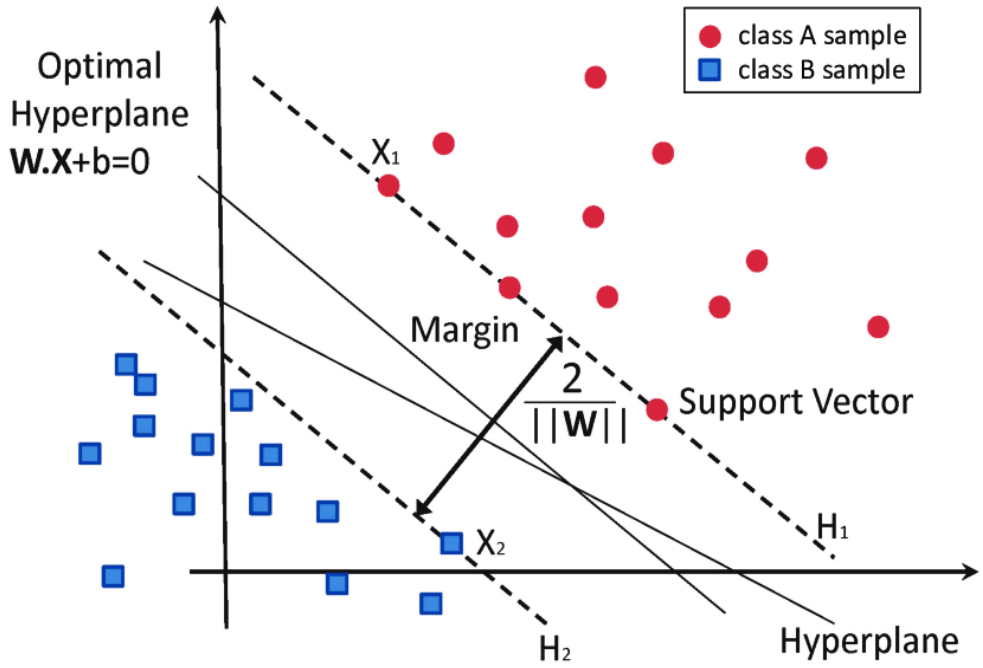

Figure 11 Classification des données par une machine à vecteurs de support (SVM) [51].

2.6 Forêt aléatoire (Random forest)

Les forêts aléatoires ont été introduites par Leo Breiman [44], inspiré par les travaux antérieurs d'Amit et Geman [45]. Bien que cela ne soit pas évident d'après la description de [44], les forêts aléatoires sont une extension de l'idée de mise en sac de Breiman [46] et ont été développées pour concurrencer le boosting. Les forêts aléatoires peuvent être utilisées soit pour une variable de réponse catégorielle, appelée dans [44] "classification", soit pour une réponse continue, appelée à nouveau "régression". De même, les variables prédicatrices peuvent être soit catégoriques, soit continues. D'un point de vue informatique, les forêts aléatoires sont attrayantes car elles :

- Traitent naturellement à la fois la régression et la classification (multi classe) ;
- Sont relativement rapides à former et à prédire ;
- Ne dépendent que d'un ou deux paramètres de réglage ;
- Ont une estimation intégrée de l'erreur de généralisation ;
- Peuvent être utilisés directement pour les problèmes à haute dimension ;
- Peuvent facilement être mis en œuvre en parallèle.

Statistiquement, les forêts aléatoires sont attrayantes en raison des fonctions supplémentaires qu'elles supplémentaires qu'ils offrent [47], telles que :

- Mesures de l'importance des variables ;
- Pondération différentielle des classes ;
- Imputation des valeurs manquantes ;
- Visualisation ;
- Détection des valeurs aberrantes ;
- Apprentissage non supervisé.

Comme son nom l'indique, une forêt aléatoire est un ensemble basé sur des arbres, chaque arbre dépendant d'une collection de variables aléatoires. Plus formellement, pour un vecteur aléatoire p-dimensionnel X = (X1..., Xp), T représentant les variables d'entrée ou prédicteurs à valeur réelle et une variable aléatoire Y représentant la réponse à valeur réelle, nous supposons une distribution conjointe inconnue P_{XY} (X, Y). L'objectif est de trouver une fonction de prédiction f(X) pour prédire Y. La fonction de prédiction est déterminée par une fonction de perte L (Y, f(X)) et définie pour minimiser la valeur attendue de la perte.

2.7 Apprentissage profond

L'apprentissage en profondeur est un type d'apprentissage automatique qui fonctionne mieux sur les données non structurées. Les techniques d'apprentissage en profondeur sont plus efficaces que les techniques actuelles d'apprentissage automatique [48]. Elles permettent aux modèles informatiques d'apprendre progressivement des caractéristiques à partir de plusieurs niveaux de données. Avec l'augmentation de la quantité de données disponibles et le développement du matériel qui alimente les ordinateurs puissants, la popularité de l'apprentissage en profondeur augmente également [48].

L'apprentissage en profondeur (comme le montre la Figure 12) est une forme d'apprentissage automatique qui peut utiliser des algorithmes supervisés ou non supervisés, ou les deux. Bien qu'il ne soit pas nécessairement nouveau, l'apprentissage en profondeur est récemment redevenu populaire comme moyen d'accélérer la résolution de certains types de problèmes informatiques difficiles, en particulier dans les domaines de la vision par ordinateur et du traitement du langage naturel (NLP) [48].

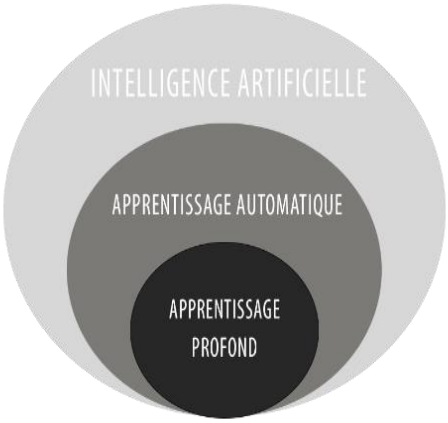

Figure 12 Apprentissage profond

2.7.1 Apprentissage profond vs méthodes classiques

L'apprentissage profond implique l'application de réseaux de neurones profonds avec plus de couches et plus de données que les algorithmes d'apprentissage automatique traditionnels, ce qui nécessite des modèles plus grands et plus de calculs [49]. Ceci est également utile, car même si la quantité de données augmente, les performances des algorithmes d'apprentissage automatique traditionnels ne peuvent pas être améliorées après un certain temps, mais les performances des algorithmes d'apprentissage en profondeur sont proportionnelles à la quantité de données et à la quantité de données, et la diversité des données [49]. Comme le montre la Figure 13. Un réseau de neurones artificiels est un système qui apprend à agir sur la base d'exemples, sans avoir besoin de procédures spécifiques explicites.

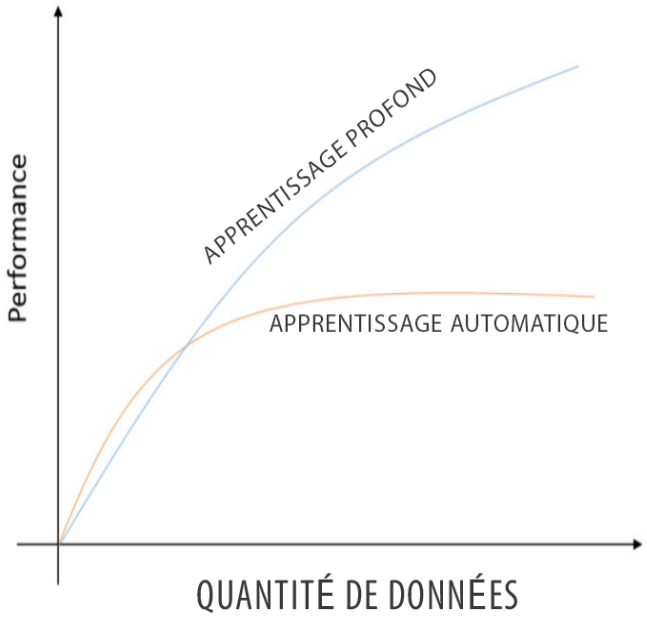

Figure 13 Performance de l'apprentissage profond [49]

2.7.2 Réseau de neurones

Pour comprendre comment fonctionnent les réseaux neuronaux et d'où ils viennent, nous devons mentionner les neurones. Nous pouvons nous poser certaines questions, comme par exemple : dans le cerveau humain, comment les données partent-elles de nos yeux pour que nous les reconnaissions et prononcions ensuite quelques mots qui semblent avoir un sens ? Un neurone est l'unité de base de notre cerveau qui contient toutes ces informations (Figure 14) [50]. Le neurone est également l'unité de base du calcul dans un réseau neuronal, souvent appelé nœud ou unité [50]. Le cerveau humain est constitué d'environ 10^{11} unités de calcul, les "neurones", qui travaillent en parallèle et échangent des informations par l'intermédiaire de leurs connecteurs, les "synapses". Ces neurones additionnent toutes les informations qui leur parviennent et, si le résultat est supérieur au potentiel donné, appelé potentiel d'action, ils envoient une impulsion par l'axone à l'étape suivante [50]. L'anatomie des neurones humains est présentée dans la Figure 14.

De la même manière, le réseau neuronal artificiel se compose d'unités de traitement organisées en couches d'entrée, cachées et de sortie. Les nœuds ou unités de chaque couche sont connectés aux nœuds des couches adjacentes. Chaque connexion a une valeur de poids. Les entrées sont multipliées par les poids respectifs et additionnées à chaque unité [51]. La somme subit ensuite une transformation basée sur la fonction d'activation, qui est dans la plupart des cas une fonction sigmoïde, tan hyperbolique ou unité linéaire rectifiée (**ReLU**). Ces fonctions sont utilisées car

elles ont une dérivée mathématiquement favorable, ce qui facilite le calcul des dérivées partielles du delta d'erreur par rapport aux poids individuels [51]. Les fonctions sigmoïde et tan hyperbolique écrasent également l'entrée dans une plage de sortie étroite ou une option, c'est-à-dire **0/1** et **-1/+1** respectivement. Ils mettent en œuvre une non-linéarité saturée car les sorties plafonnent ou saturent avant/après les seuils respectifs. **ReLu**, quant à lui, présente des comportements à la fois saturés et non saturés avec **f(x) = max(0,x).** La sortie de la fonction est ensuite utilisée comme entrée pour l'unité suivante dans la couche suivante. Le résultat de la couche de sortie finale est utilisé comme solution au problème [51].

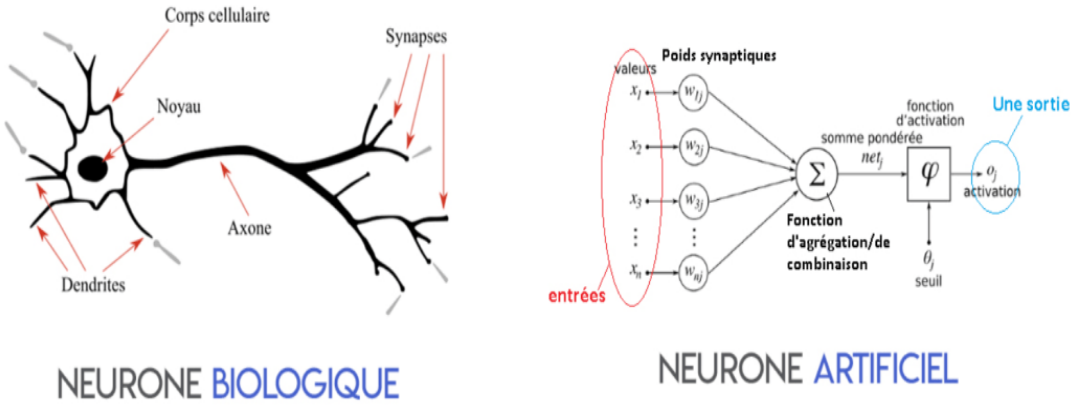

Figure 14 Neurone biologique et artificiel [52]

2.7.2.1 Réseau de neurones convolutive (CNN)

Un réseau de neurones convolutifs est typiquement composé d'une collection de couches qui seront triées par leurs fonctionnalités. Le CNN est particulièrement galvanisé par la structure de la région corticale. Le modèle structurel a supporté la région corticale d'un mammifère. Le cortex comporte de petites zones de cellules qui sont sensibles à des zones particulières du champ de vision [53].

Un réseau neuronal convolutif (CNN) se compose d'une couche d'entrée et d'une couche de sortie, ainsi que de plusieurs couches cachées. Ces couches sont généralement divisées en trois types : CONV, POOL et FC (abréviation de fully-connected). Nous allons en outre écrire expressément la fonction d'activation de non-linéarité comme une couche qui applique une non-linéarité par élément. Dans cette section, nous avons tendance à discuter de la façon dont ces couches sont généralement empilées pour créer des ConvNets entiers.

Les CNN se concentrent principalement sur l'idée que les données contiennent des images, ce qui permet de construire l'architecture de la manière la plus adaptée à la nécessité de traiter cette forme particulière d'information [53]. Cependant, une des variations significatives est que les couches au milieu du CNN sont composées de

neurones organisés en trois dimensions connues sous le nom de dimensionnalité spatiale de l'entrée (la hauteur, la largeur et la profondeur). L'architecture d'un CNN est modélisée de manière à tirer parti de la structure 2D d'une image d'entrée (ou de diverses entrées 2D similaires à un signal vocal), ce qui est souvent réalisé à l'aide de connexions voisines et de poids liés, suivis d'une mise en commun qui donne des caractéristiques invariantes en termes de traduction [53] [54]. Un CNN se compose d'une série de couches de convolution et de sous-échantillonnage accompagnées éventuellement de couches entièrement connectées, c'est-à-dire qu'une ou plusieurs couches entièrement connectées sont présentes après de nombreuses couches de convolution et de mise en commun. L'entrée et la sortie de chaque étape sont des ensembles de tableaux appelés cartes de caractéristiques [54]. Dans le cas d'une image colorée, chaque carte de caractéristiques est un tableau 2D contenant un canal de couleur de l'image d'entrée, un tableau 3D pour une vidéo et un tableau 1D pour une entrée audio [53]. L'étage de sortie représente les caractéristiques extraites de tous les emplacements sur les données.

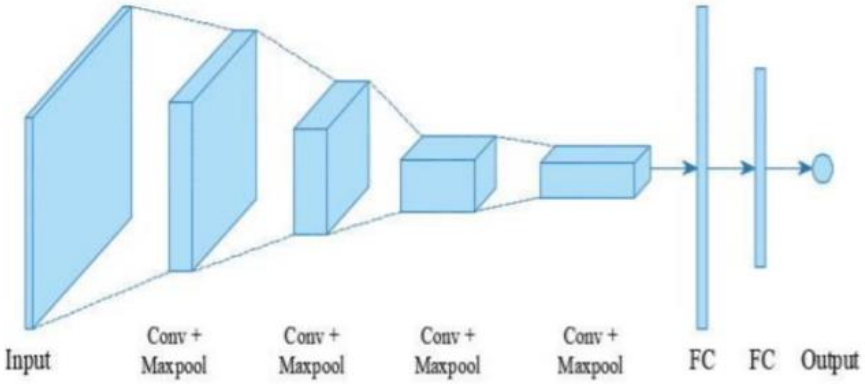

Figure 15 Architecture CNN [54]

La fonctionnalité de base de l'exemple CNN ci-dessus peut être décomposée en quatre domaines clés [54]:

- Comme on le trouve dans d'autres formes de CNN, la couche d'entrée contient les valeurs des pixels de l'image.
- La couche convolutionnelle déterminera la sortie des neurones qui sont connectés à des régions locales de l'entrée par le calcul du produit scalaire entre leurs poids et la région connectée au volume d'entrée. L'unité linéaire rectifiée (communément appelée ReLu) a pour but d'appliquer une fonction d'activation "par éléments" telle que la fonction sigmoïde à la sortie de l'activation produite par la couche précédente.

- La couche de mise en commun effectue alors simplement un sous-échantillonnage le long de la dimension spatiale de l'entrée donnée, ce qui réduit encore le nombre de paramètres dans cette activation.
- Les couches entièrement connectées effectueront ensuite les mêmes tâches que les ANN (Réseau neuronal artificiel) standards et tenteront de produire des scores de classe à partir des activations à utiliser pour la classification. Il est également suggéré d'utiliser ReLu entre ces couches, afin d'améliorer les performances.

Grâce à cette simple méthode de transformation, les CNN sont capables de transformer l'entrée originale couche par couche en utilisant des techniques de convolution et de sous-échantillonnage pour produire des scores de classe à des fins de classification et de régression [54].

2.7.2.2 Modèles de séquence

La modélisation séquentielle, en termes simples, est le processus qui consiste à générer une séquence de valeurs en analysant une série de valeurs d'entrée. Ces valeurs d'entrée peuvent être des données de séries chronologiques où une variable spécifique, par exemple la demande d'un produit particulier, varie sur une période donnée [55]. La sortie pourrait être la prédiction de la demande pour les périodes suivantes. Un autre exemple pourrait être la prédiction de texte, où, sur la base de la séquence de la dernière phrase et d'un ensemble de conditions et de règles préchargées, l'algorithme de modélisation de séquence prédit ce que pourrait être le mot suivant. En utilisant la modélisation des séquences, les entreprises peuvent obtenir plus que la simple génération de modèles et la prédiction [55].

Pour la génération de phrases ou la traduction de textes, tous les mots générés dépendent des mots générés apriori (parfois, ils dépendent aussi des mots qui suivent) [56]. Par conséquent, nous devons avoir un certain biais, basé sur nos résultats précédents. C'est là que les modèles de séquence (comme leur nom l'indique) entrent en jeu. Les modèles de séquence ont le sens de la mémorisation de ce qui s'est passé précédemment dans la séquence de données [56]. Cela aide le système à gagner en contexte.

Les modèles séquentiels exploitent les informations séquentielles. Dans le modèle de réseaux neuronaux précédent, toutes les entrées (et sorties) sont indépendantes les unes des autres [55] [56]. Toutefois, pour traiter certains problèmes de la vie réelle, cette approche peut ne pas être appropriée. Si nous voulons prédire le prochain mot d'une phrase, il est préférable de connaître la séquence de mots précédente. Les modèles de séquence se souviennent de toutes ces relations lors de leur apprentissage. Dans ces architectures de réseaux neuronaux, la sortie d'une unité particulière dépend des calculs précédents [56].

2.7.2.3 Réseau de neurones récurrents (RNN)

Un réseau neuronal récurrent (RNN) est un type particulier de réseau neuronal artificiel qui permet de poursuivre des informations liées à des connaissances antérieures en utilisant un type particulier d'architecture en boucle. Ils sont employés dans de nombreux domaines concernant des données avec des séquences, comme la prédiction du prochain mot d'une phrase. Ces réseaux en boucle sont dits récurrents car ils effectuent les mêmes opérations et calculs pour chaque élément d'une séquence de données d'entrée [57]. Les RNN possèdent une mémoire, qui les aide à prendre des informations à partir de séquences passées. Un réseau neuronal récurrent (RNN) est l'une des architectures de base. Nombre des architectures avancées actuelles s'inspirent des RNN [58]. La principale caractéristique d'un RNN est que le réseau possède des connexions de rétroaction, contrairement à un réseau neuronal à anticipation traditionnel. Cette boucle de rétroaction permet au RNN de modéliser les effets des premières parties de la séquence sur les dernières parties de la séquence, ce qui est une caractéristique très importante lorsqu'il s'agit de modéliser des séquences [58].

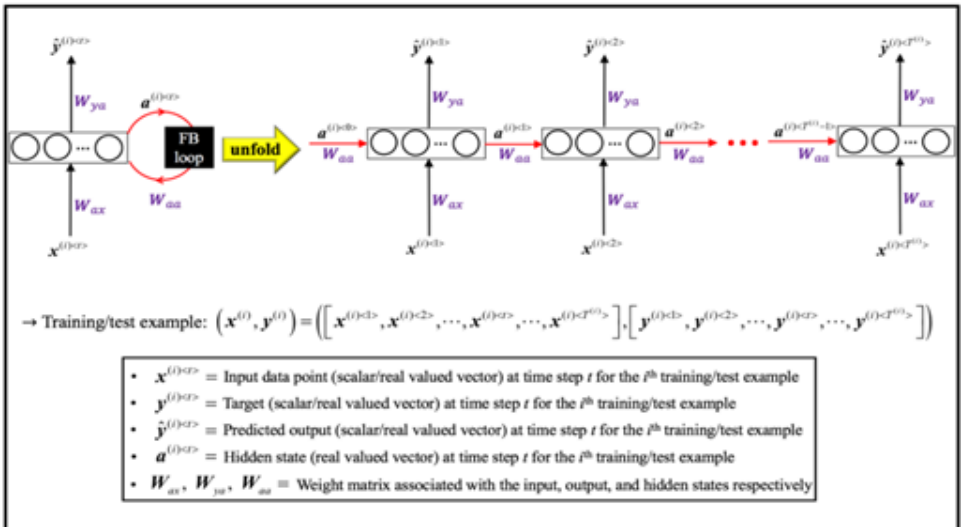

Figure 16 RNN a une seule couche [59]

La Figure 16 montre un réseau neuronal récurrent (RNN) à une seule couche cachée dans sa forme de base [59] [57]. Il se compose d'un seul ensemble d'unités d'entrée, cachées et de sortie, les unités cachées s'alimentant elles-mêmes par le biais de boucles de rétroaction. En termes simples, comme le montre clairement sa version dépliée, un RNN peut être considéré comme de multiples copies (dans le temps) du même réseau, chacune transmettant un message à son successeur. Les paramètres entraînables pour un RNN comprennent les poids et les biais qui sont partagés entre tous les pas de temps. Les équations 1 et 2 spécifient tous les calculs à chaque pas de temps pendant le passage vers l'avant d'un RNN [59]. La

Figure 17 ci-dessous montre les deux équations qui spécifient tous les calculs à chaque pas de temps pendant le passage vers l'avant d'un RNN.

$$a^{<t>} = \psi_1\left(W_{ax}x^{<t>} + W_{aa}a^{<t-1>} + b_a\right)$$

$$= \psi_1\left(\begin{bmatrix} W_{ax} & W_{aa} \end{bmatrix}\begin{bmatrix} x^{<t>} \\ a^{<t-1>} \end{bmatrix} + b_a\right)$$

$$\equiv \psi_1\left(W_a\left[x^{<t>}; a^{<t-1>}\right] + b_a\right)$$

$$\hat{y}^{<t>} = \psi_2\left(W_{ya}a^{<t)} + b_y\right) \equiv \psi_2\left(W_y a^{<t)} + b_y\right)$$

Figure 17 Equations RNN

2.7.2.4 Types de réseaux de neurones récurrents (RNN)

Il existe quatre types de réseaux neuronaux récurrents :

One to One RNN :

Ce type de réseau neuronal est connu sous le nom de réseau neuronal « Vanille » (standard). Il est utilisé pour les problèmes généraux d'apprentissage automatique, avec une seule entrée et une seule sortie (Figure 18) [60].

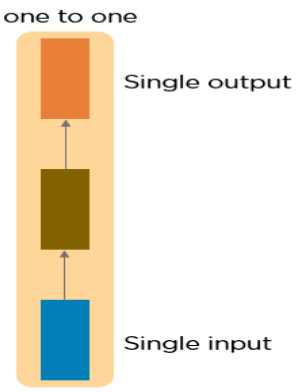

Figure 18 one to one RNN [60]

One to Many RNN :

Ce type de réseau neuronal possède une entrée unique et des sorties multiples (Figure 19) [60].

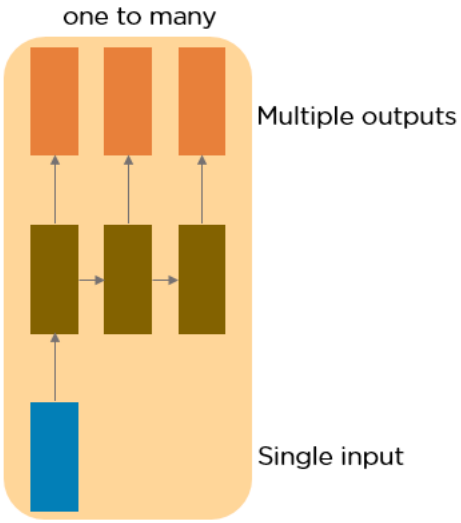

Figure 19 One to Many RNN [60]

Many to One RNN :

Ce RNN prend une séquence d'entrées et génère une sortie unique. L'analyse des sentiments est un bon exemple de ce type de réseau où une phrase donnée peut être classée comme exprimant des sentiments positifs ou négatifs (Figure 20) [60].

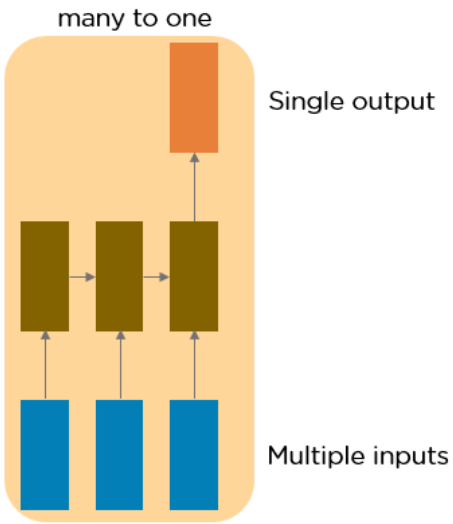

Figure 20 Many to One RNN [60]

Many to Many RNN :

Ce RNN prend une séquence d'entrées et génère une séquence de sorties. La traduction automatique est un des exemples ou ce type de RNN est utilisé (Figure 21) [60].

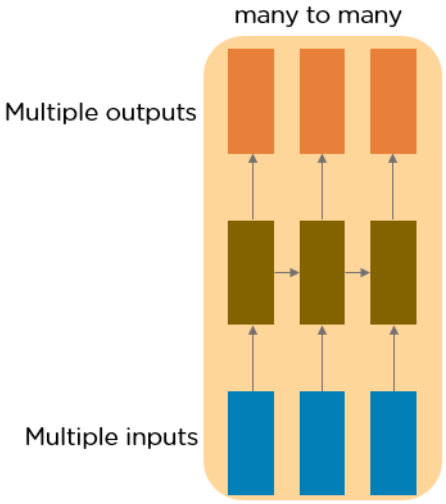

Figure 21 Many to Many RNN [60]

2.8 Vanishing Gradient

Alors que l'explosion des gradients peut conduire à l'effondrement du modèle, une conséquence directe de la disparition des gradients « Vanishing Gradients » (Figure 22) est que les informations des étapes temporelles antérieures ne peuvent pas être utilisées efficacement pour faire des prédictions aux étapes ultérieures, ou en d'autres termes, les informations des étapes temporelles antérieures ne peuvent pas être retenues efficacement à long terme [59]. Cela ne signifie pas qu'il est impossible d'apprendre, mais qu'il peut être très long d'apprendre les dépendances à long terme, car le gradient d'une interaction à long terme a une amplitude exponentiellement plus faible que le gradient d'une interaction à court terme, et en tant que tel, le signal concernant ces dépendances aura tendance à être caché par les plus petites fluctuations provenant des dépendances à court terme [59] [61].

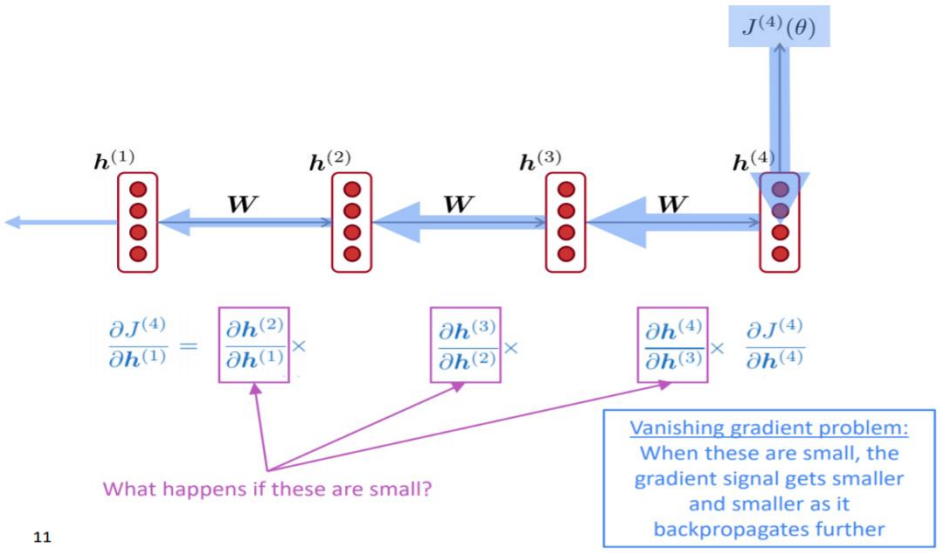

Figure 22 Problème du gradient évanescent [61]

2.8.1 Long Short-Term Memory (LSTM)

Le réseau de mémoire à long terme (LSTM) (Figure 23) est un type spécial de RNN qui peut apprendre des dépendances à long terme. Tous les RNN ont la forme d'une chaîne de modules répétitifs. Dans un RNN standard, ce module répétitif a généralement une structure simple [62]. Cependant, le module répétitif des LSTM est plus compliqué. Il n'y a pas qu'une seule couche de réseau de neurones, mais quatre couches qui interagissent d'une manière spéciale. De plus, il a deux états : l'état caché et l'état cellulaire [62].

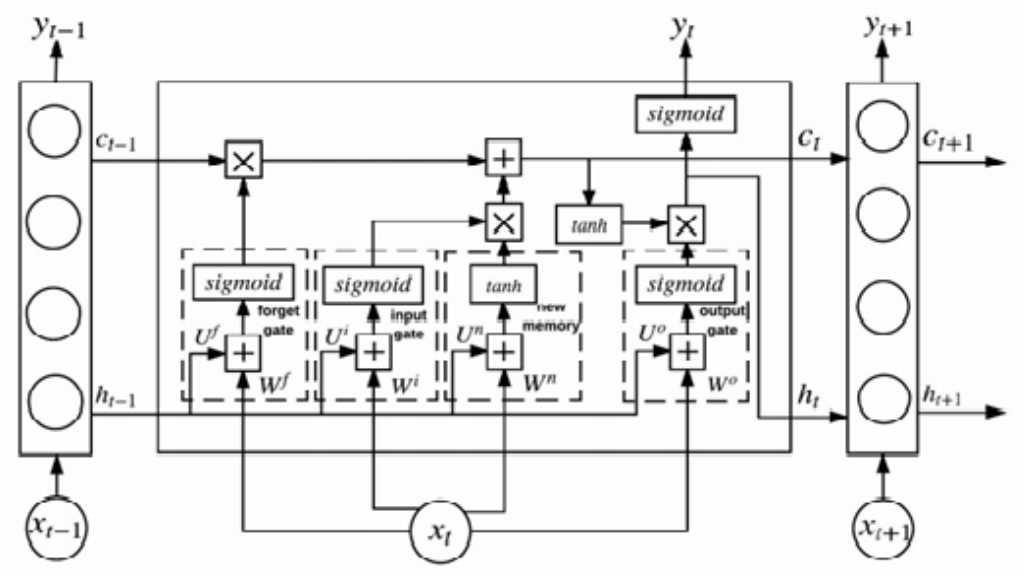

Figure 23 Architecture LSTM [62]

2.8.2 Gated Recurrent Unit (GRU)

GRU est une variante plus simple de LSTM, introduite en 2014 par Cho et al. [63] et Chung et al. [64]. Par rapport à LSTM, la structure interne de GRU est plus simple et plus facile à former car elle implique moins de calculs [62]. Ces simplifications sont principalement réalisées de deux manières :

1. Les portes d'entrée et d'oubli sont combinées en une seule porte appelée porte de mise à jour.
2. L'état/unité interne et l'état caché fusionnent. La Figure 24 montre les cellules de l'unité GRU et les calculs impliqués [62].

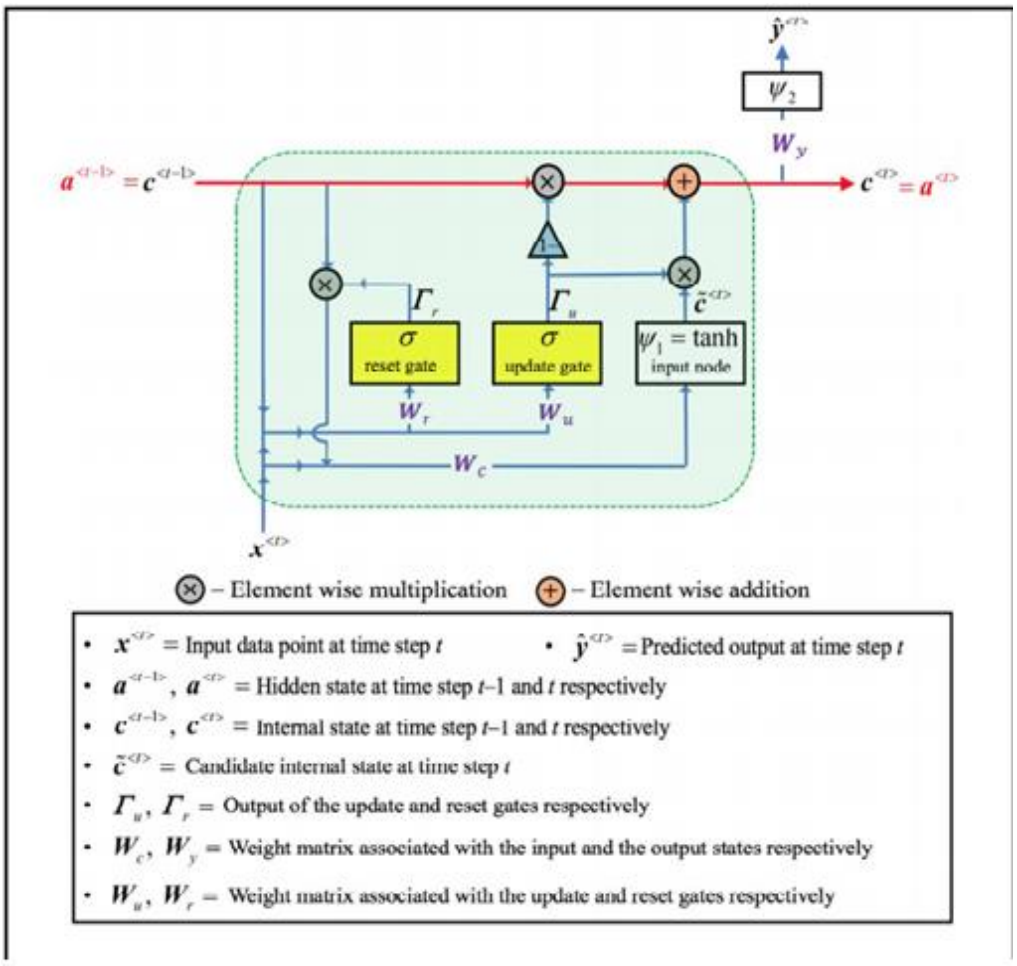

Figure 24 Architecture GRU [62]

2.9 Apprentissage profond pour l'analyse des sentiments

Dans les modèles d'apprentissage profond existants, la classification du sentiment des phrases est généralement formulée comme un problème de classification conjointe à trois voies, à savoir la prédiction d'une phrase comme positive, neutre et négative. Comme pour la classification des sentiments au niveau des documents, la représentation des phrases produite par les réseaux neuronaux est également importante pour la classification des sentiments au niveau des phrases [9] [62]. En outre, comme une phrase est généralement courte par rapport à un document, certaines informations syntaxiques et sémantiques (par exemple, les arbres d'analyse, les lexiques d'opinion et les balises de partie de discours) peuvent être utilisées pour aider [62]. Des informations supplémentaires, telles que les évaluations des critiques, les relations sociales et les informations inter-domaines, peuvent également être prises en compte. Par exemple, les relations sociales ont été exploitées pour découvrir les sentiments dans les données des médias sociaux tels que les tweets.

Dans les premières recherches, les arbres d'analyse (fournissant des informations sémantiques et syntaxiques) ont été employés pour utiliser des mots originaux comme entrées dans les modèles neuronaux afin de mieux déduire la composition des émotions [62]. Mais récemment CNN et RNN deviennent de plus en plus populaires, ils n'ont pas besoin d'analyser l'arbre pour extraire les caractéristiques de la phrase. En revanche, CNN et RNN utilisent des mots intégrés en entrée, qui ont déjà codé certaines informations sémantiques et syntaxiques [22] [62]. De plus, l'architecture du modèle CNN ou RNN peut également aider à apprendre la relation interne entre les mots d'une phrase. Les travaux connexes sont décrits en détail ci-dessous :

Dans le traitement du langage naturel, les CNN ont montré leur efficacité, atteignant de bons résultats dans l'analyse sémantique, la modélisation de phrases ou même la récupération de requêtes de recherche. Les CNN ont également été largement utilisés pour l'analyse des sentiments (Kim [65]; Kalchbrenner et al [66]).

Kim [65] a introduit un principe basé sur l'entraînement d'un CNN simple avec une couche de convolution sur des vecteurs de mots obtenus à partir d'un modèle de langage neuronal non supervisé. En ce qui concerne les vecteurs de mots, l'auteur a utilisé les vecteurs word2vec disponibles publiquement qui ont été entraînés sur 100 milliards de mots provenant de Google News.

Initialement, Kim [65] garde les vecteurs de mots statiques et n'apprend que les autres paramètres du modèle, obtenant ainsi déjà d'excellents résultats. Cependant, comme l'apprentissage de vecteurs spécifiques à la tâche par le biais d'un réglage fin entraîne des améliorations supplémentaires, l'auteur décrit une modification simple de l'architecture, pour permettre l'utilisation de vecteurs pré-entraînés et spécifiques à la tâche, en ayant plusieurs canaux. Les différents canaux sont

initialisés avec word2vec et chaque filtre du CNN leur est appliqué. Cependant, les gradients ne sont rétro-propagés qu'à travers l'un des canaux, ce qui permet au modèle de n'ajuster qu'un seul ensemble de vecteurs tout en gardant l'autre statique. L'évaluation de cette approche montre que le pré-entraînement non supervisé des vecteurs de mots est un ingrédient important de l'apprentissage profond pour le traitement du langage naturel.

Socher et al. [67] Ont d'abord proposé un réseau d'auto encodeurs récursifs (RAE) semi-supervisé (Figure 25) pour la classification du sentiment au niveau de la phrase, qui obtient une représentation vectorielle de dimension réduite pour une phrase.

Le but des auto encodeurs est d'apprendre une représentation de leurs entrées. Dans le passé, les auto encodeurs n'ont été utilisés que dans un cadre où la structure de l'arbre était donnée a priori. Socher et al. [67] ont examiné ce contexte avant de poursuivre avec leur modèle qui ne nécessite pas une structure d'arbre donnée.

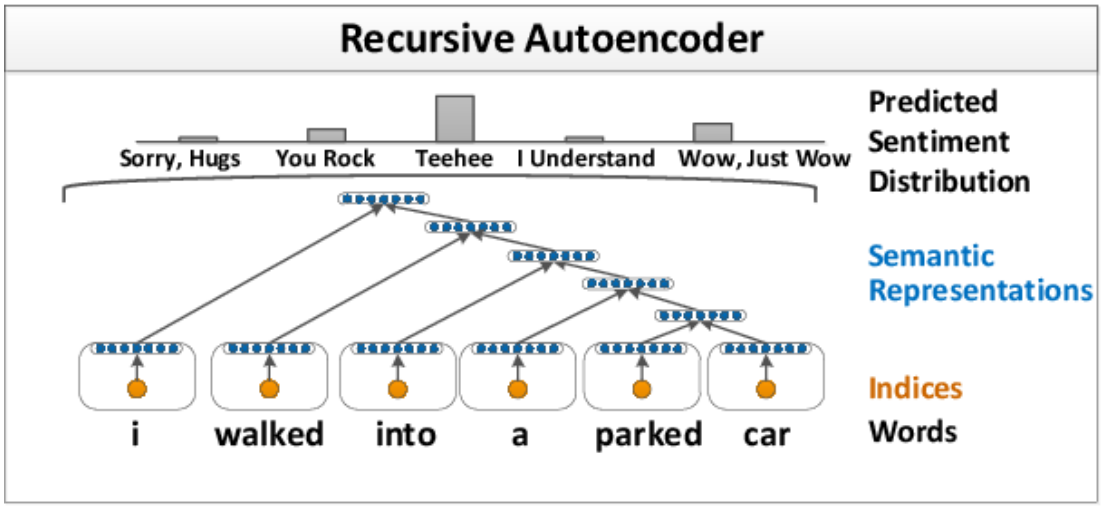

Figure 25 Auto encodeur récursif [67]

Plus tard, Socher et al. [68] ont proposé un réseau de neurones récurrents matrice-vecteur (MVRNN) (Figure 26), dans lequel chaque mot est associé à la représentation matricielle (sauf la représentation vectorielle) dans la structure arborescente. L'arborescence est obtenue à partir d'un analyseur externe.

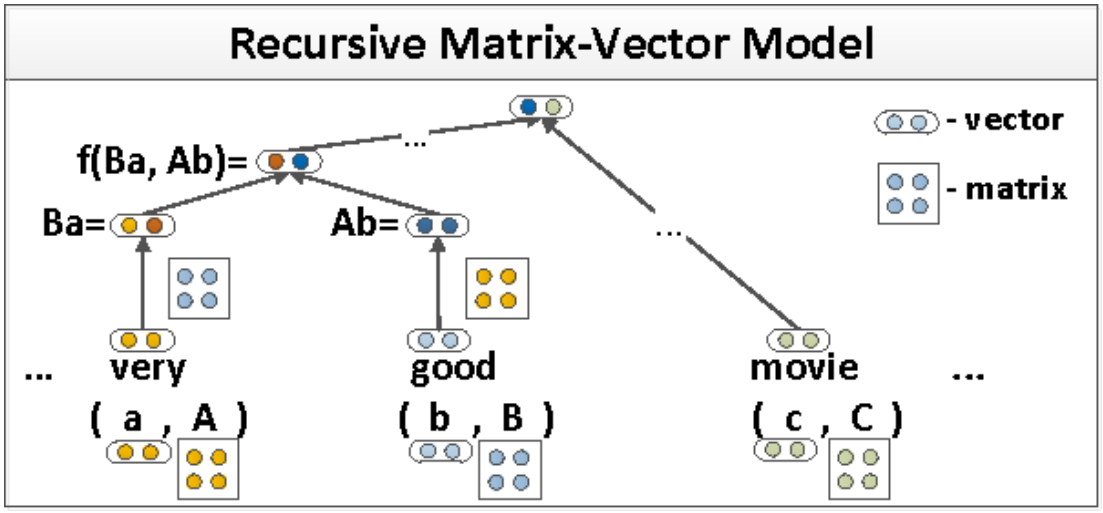

Figure 26 MVRNN [68]

Dans l'analyse des sentiments, la capture du sens des phrases plus longues a également fait l'objet d'une grande attention. Cependant, pour être en mesure d'extraire cette information, il faut remédier au manque de ressources de compositionnalité importantes et étiquetées. Socher et al. [69] ont créé le corpus Stanford Sentiment Treebank, c'est-à-dire le premier corpus permettant de capturer les effets de composition du sentiment dans la langue, en fournissant des arbres d'analyse entièrement étiquetés. Le type d'informations contenues dans ce jeu de données permet à la communauté d'entraîner et de développer de nouveaux modèles compositionnels.

L'exploitation de cette nouvelle ressource d'information a permis à Socher et al. [69] de proposer des réseaux de neurones récurrents tenseurs (RNTN), qui utilisent des fonctions de combinaison de tenseurs pour mieux capturer les interactions entre les éléments.

Le RNTN aborde plusieurs problèmes liés aux RNN standard (Socher et al. [67]) et à l'architecture MV-RNN (Matrix-Vector Recursive Neural Network) proposée précédemment (Socher et al., [68]). Dans le modèle MV-RNN, les paramètres sont associés aux mots et chaque fonction de composition qui calcule les vecteurs de phrases plus longues dépend des mots réels combinés. Cependant, le nombre de paramètres peut devenir très important et dépend de la taille du vocabulaire. Considérant ce problème, Socher et al. [69] Ont suggéré qu'il était plus plausible qu'il y ait une seule fonction de composition avec un nombre fixe de paramètres.

En bref, l'idée principale de la RNTN est d'utiliser la même fonction de composition basée sur un tenseur pour tous les nœuds de l'arbre de compositionnalité.
Les expériences ont montré que le modèle RNTN améliore la détection des sentiments au niveau des phrases, obtenant de meilleurs résultats que le modèle

MV-RNN. Un autre aspect pertinent, est que ce nouveau modèle capture la négation de différents sentiments.

Qian et al. [70] ont proposé deux modèles plus avancés, Tag-guided Recurrent Neural Network (TG-RNN) (Figure 27) [70], qui sélectionne une fonction de combinaison basée sur les balises de partie du discours dans une phrase, et un réseau de neurones récurrent. Réseau de tenseurs neuronaux récurrents (TE-RNN/RNTN), qui apprend l'intégration des balises, puis combine l'intégration des balises et des mots.

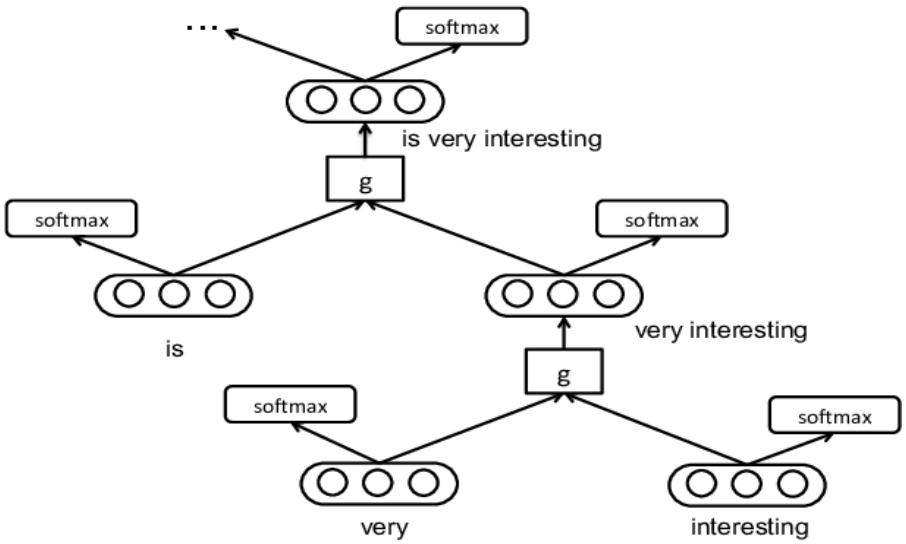

Figure 27 Tag-guided Recurrent Neural Network (TG-RNN) [69]

Kalchbrenner et al. [66] ont proposé un CNN dynamique (appelé DCNN) (Figure 28) pour la modélisation sémantique des phrases. DCNN utilise l'opérateur de pooling dynamique K-Max comme fonction de sous-échantillonnage non linéaire. La carte des caractéristiques induite par le réseau peut capturer la relation entre les mots. Kim [65] a également proposé l'utilisation de CNN pour la classification des sentiments au niveau de la phrase et a essayé plusieurs variantes, à savoir CNN-rand (initialisation aléatoire des mots intégrés), CNN-static (les mots intégrés sont pré-entraînés et fixes), CNN-non-statique (pré-entraîné et fixe), et CNN-non-statique. Former et ajuster les mots intégrés et CNN-multicanal (utiliser plusieurs ensembles de mots intégrés).

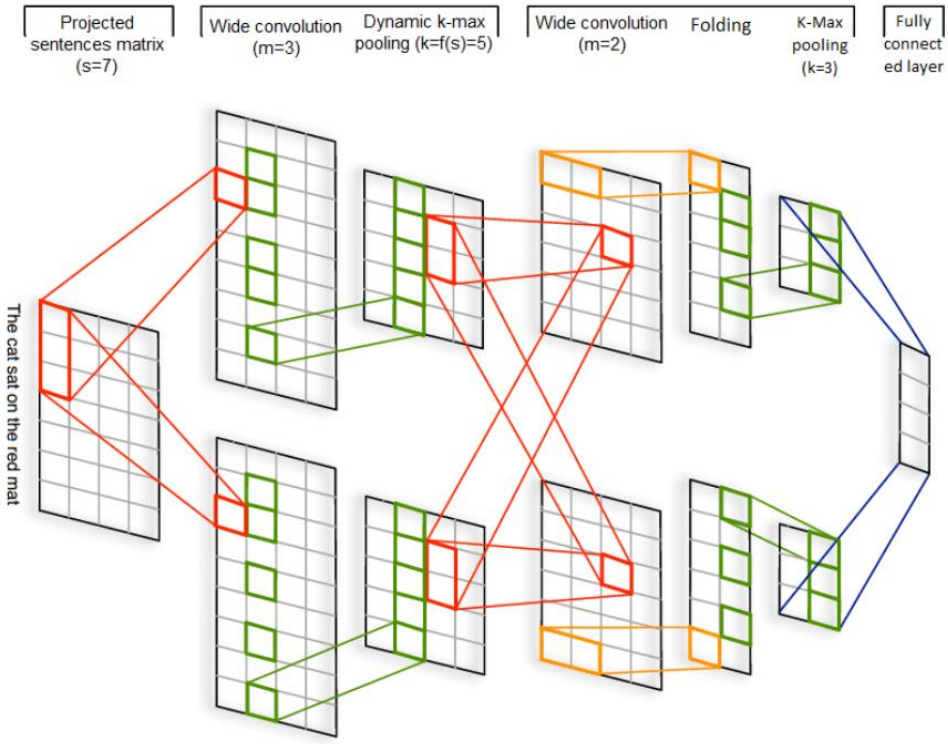

Figure 28 CNN dynamique DCNN [66]

Wang et al. [71] Ont utilisé le LSTM pour la classification des sentiments sur Twitter en simulant les interactions des mots pendant le processus de composition. Les opérations multiplicatives entre les mots intégrés par le biais de structures de porte sont utilisées pour fournir plus de flexibilité et produire de meilleurs résultats de composition par rapport aux opérations additives dans les réseaux neuronaux récurrents simples. Comme pour le RNN bidirectionnel, le LSTM unidirectionnel peut être étendu à un LSTM bidirectionnel [72] en permettant des connexions bidirectionnelles dans la couche cachée.

Wang et al. [73] Ont proposé un modèle CNN-LSTM régional (Figure 29) [73], qui se compose de deux parties : un CNN régional et un LSTM, pour prédire les notes d'éveil de valence d'un texte.

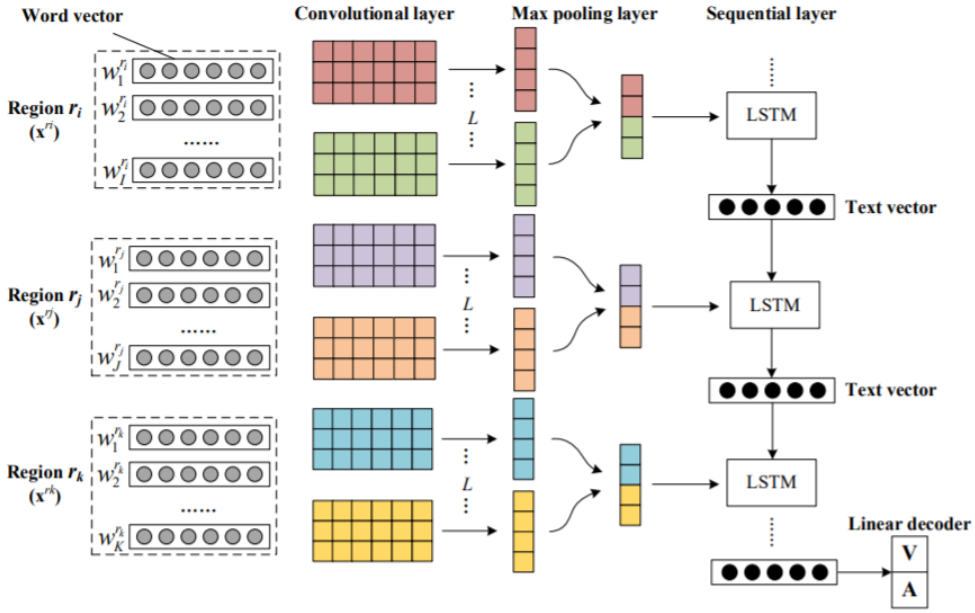

Figure 1: System architecture of the proposed regional CNN-LSTM model.

Figure 29 Modèle régional CNN-LSTM [73]

2.10 Conclusion

L'application de l'apprentissage profond à l'analyse des sentiments est devenue un sujet de recherche populaire d'actualité.

Dans cette partie, nous avons présenté diverses architectures d'apprentissage profond et leurs applications dans l'analyse des sentiments. Plusieurs de ces techniques d'apprentissage profond ont montré des résultats de pointe pour diverses tâches de fouille d'opinions.

Avec les progrès de la recherche et des applications de l'apprentissage profond, nous pensons qu'il y aura d'autres études plus avancées dans ce domaine pour l'analyse des sentiments dans un avenir proche plus particulièrement dans la fouille d'opinions basé sur les aspects .

Chapitre 3 Approche proposée

Après avoir présenté une vue globale sur l'existant dans le domaine de la fouille d'opinions et de l'apprentissage en profondeur, nous allons dans ce qui suit, d'abord présenter notre méthodologie pour répondre à la problématique citée dans notre introduction, à savoir « ***Fouille d'opinions par Deep Learning »*** et expliquerons pourquoi nous avons fait ces choix**.**

3.1 Traitement des données

Avant de relayer des données d'entrainement à notre modèle ces dernières doivent être prétraitées pour réduire le taux de bruit présent dans le corpus et de ce fait, fournir une base d'apprentissage solide pour notre modèle.

Les étapes de prétraitement et de traitement sont détaillées dans la section suivante.

3.1.1 Suppression des balises HTML

Le web génère des tonnes de données textuelles et ce texte peut contenir des balises HTML. Ces balises HTML n'ajoutent aucune valeur aux données textuelles et permettent uniquement un rendu correct par le navigateur. C'est pour cela qu'il est nécessaire de nettoyer le corpus en éliminant ces balises qui n'ont aucun impact sur la polarité sentimentale du texte.

Nous allons utiliser le texte non traité qui apparaît ci-dessous pour couvrir tous les cas d'utilisation que nous voulons examiner dans cet section (Le texte est inspiré d'une des critiques du corpus de critiques alimentaires d'Amazon. Nous avons modifié le texte pour couvrir tous les cas d'utilisation).

« **Why is this $12 when the same product is available for $10 here?
http://www.domain.com/product/dp/B00004RBDY
I don't understand reason behind two different prices.
The Victor M380 and M502 traps are unreal, /of course, — total fly genocide. Pretty stinky, but only right nearby.** »

Après cette étape le texte devient :

« **Why is this $12 when the same product is available for $10 here? http://www.domain.com/product/dp/B00004RBDY I don’t understand reason behind two different prices. The Victor M380 and M502 traps are unreal, /of course, — total fly genocide. Pretty stinky, but only right nearby.** »

3.1.2 Suppression des URL

Les URL (ou Uniform Resource Locators) dans un texte sont des références à un emplacement sur le web, mais ne fournissent aucune information supplémentaire. Nous les supprimons donc également en prenant notre échantillon de texte et en analysant chaque mot, en supprimant les mots ou les chaînes de caractères commençant par « http ».

Le texte devient :

« **Why is this $12 when the same product is available for $10 here? I don’t understand reason behind two different prices. The Victor M380 and M502 traps are unreal, /of course, — total fly genocide. Pretty stinky, but only right nearby.** »

3.1.3 Développer les mots contractés

Dans les communications verbales et écrites quotidiennes en anglais, beaucoup de personnes ont tendance à contracter des mots courants comme "you are" qui devient "you’re". Convertir les contractions dans leur forme naturelle apportera plus de compréhension au modèle.

Après cette opération le texte donné en exemple devient :

« **Why is this $12 when the same product is available for $10 here? I do not understand reason behind two different prices. The Victor M380 and M502 traps are unreal, /of course, — total fly genocide. Pretty stinky, but only right nearby.** »

3.1.4 Suppression des caractères spéciaux

Les caractères spéciaux tels que - (trait d'union) ou / (slash) n'apportent aucune valeur ajoutée, c'est pourquoi nous les supprimons généralement. Les caractères sont supprimés en fonction du cas d'utilisation. Si nous effectuons une tâche où la

devise ne joue pas de rôle (par exemple dans l'analyse des sentiments), nous supprimons le $ ou tout signe monétaire.

Ici, nous allons également supprimer les mots qui contiennent un chiffre, comme M380 dans notre exemple.

Après cette opération le texte donné en exemple devient :

« **Why is this when the same product is available for here I do not understand reason behind two different prices The Victor and traps are unreal of course total fly genocide Pretty stinky but only right nearby** »

3.1.5 Suppression des mots vides (Stopwords)

En dehors des URL, des balises HTML et des caractères spéciaux, il existe des mots qui ne sont pas nécessaires pour des tâches telles que l'analyse des sentiments ou la classification de textes. Des mots comme "je", "moi", "vous", "il" et d'autres augmentent la taille des données textuelles mais n'améliorent pas les résultats de façon notable ; il est donc judicieux de les supprimer. Nous allons également convertir tout le texte en minuscules car le texte en python est sensible à la casse.

Après cette opération le texte donné en exemple devient :

« **product available not understand reason behind two different prices victor traps unreal course total fly genocide pretty stinky right nearby** »

3.1.6 Racinisation (Stemming)

Pour des raisons grammaticales, les documents vont utiliser différentes formes d'un même mot, comme organiser, organise, et organiser. En outre, il existe des familles de mots apparentés par dérivation ayant des significations similaires, comme démocratie, démocratique et démocratisation. Dans de nombreuses situations, il semble qu'il serait utile qu'une recherche sur l'un de ces mots renvoie des documents contenant un autre mot de l'ensemble.

L'objectif de l'étymologie et de la lemmatisation est de réduire les formes flexionnelles et parfois les formes dérivées d'un mot à une forme de base commune.

Après cette opération finale de prétraitement le texte donné en exemple devient :

« **product available not understand reason behind two different price victor trap unreal course total fly genocide pretty stinky right nearby** »

3.1.7 Tokenisation

La tokenisation est le processus qui consiste à convertir le texte en tokens avant de le transformer en vecteurs. Il est également plus facile de filtrer les tokens inutiles. Par exemple, un document en paragraphes ou des phrases en mots. Dans notre cas, nous tokenisons les critiques en mots.

Pour gagner en performance la tokenisation est faite sur les **n** mots les plus fréquents dans le corpus. Ceci est effectué en créant un vocabulaire qui associe chaque mot du corpus à son nombre d'occurrences. Ainsi, le dictionnaire final ne contiendra que les tokens ayant un nombre d'occurrences supérieur au seuil **n**.

Après cette opération de traitement le texte donné en exemple devient :

['product', 'available', 'not', 'understand', 'reason', 'behind', 'two', 'different', 'price', 'victor', 'trap', 'unreal', 'course', 'total', 'fly', 'genocide', 'pretty', 'stinky', 'right', 'nearby']

3.1.8 Padding

Tous les réseaux neuronaux ont besoin d'entrées ayant la même forme et la même taille. Cependant, lorsque nous prétraitons et utilisons les textes comme entrées pour notre modèle, toutes les phrases n'ont pas la même longueur. En d'autres termes, certaines phrases sont naturellement plus longues ou plus courtes. Nous avons besoin d'avoir les entrées de la même taille, et c'est là que le padding intervient.

Considérons les deux phrases suivantes :

« Je suis un étudiant à l'université de blida »

« **Il fera beau demain** »

Si nous fixons la taille de séquence a 10 alors les deux phrases seront converties en vecteurs comme suit :

['Je', 'suis', 'un', 'étudiant', 'à', ' l'université', 'de', 'blida'**, PAD, PAD]**

[‘Il’, ‘fera’, ‘beau’, ‘demain’**, PAD, PAD, PAD, PAD, PAD, PAD]**

De ce fait l’utilisation du padding nous a permis de résoudre le problème des séquences de taille variable.

3.2 Word Embeddings

Les algorithmes de DL reçoivent un vecteur de nombres réels en entrée. Pour effectuer la représentation des mots dans un espace vectoriel (cartographie des mots en nombres), des techniques telles que l'intégration de mots sont utilisées [74].

Le Word Embeddings est une technique qui consiste à traduire les mots en un domaine mathématique où les nombres qui tentent de capturer la sémantique du mot représentent les mots. Ce processus est réalisé automatiquement en utilisant des millions de phrases. L'encodage le plus courant est le « one-hot encoding » [75] mais d'autres formes telles que « Word2Vec » tel que décrit dans [76] ont également été utilisées.

Comment visualiser ce qu'est le Word Embedding ? Eh bien, de manière générale, les mots ayant des significations similaires dans le contexte doivent être intégrés les uns à côté des autres. La Figure 30 montre des exemples de mots dans un espace à deux dimensions :

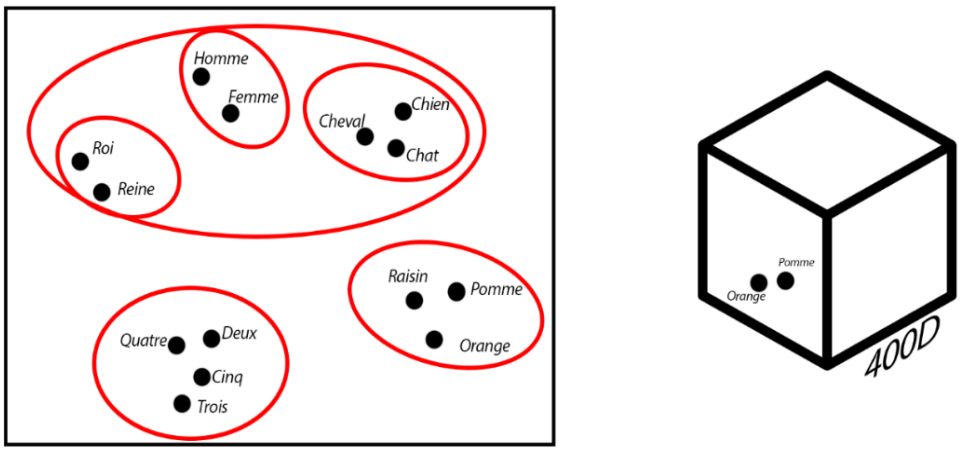

Figure 30 Word Embedding 2D

3.2.1 Word2Vec

Word2Vec est une méthode statistique permettant d'apprendre efficacement un encastrement de mots autonome à partir d'un corpus de textes.

Elle a été développée par Tomas Mikolov et al. Chez Google en 2013 afin de rendre plus efficace l'apprentissage de l'incorporation basé sur un réseau de neurones. Depuis lors, elle est devenue la norme de facto pour le développement d'une incorporation de mots pré-entraînée [76].

En outre, les travaux ont porté sur l'analyse des vecteurs appris et l'exploration des mathématiques vectorielles sur les représentations des mots. Par exemple, si l'on soustrait le "caractère masculin" de "roi" et que l'on ajoute le "caractère féminin", on obtient le mot "reine", ce qui illustre l'analogie suivante : "le roi est à la reine ce que l'homme est à la femme".

Deux modèles d'apprentissage différents ont été présentés, qui peuvent être utilisés dans le cadre de l'approche word2vec pour apprendre l'intégration des mots :

- Le modèle Continuous Bag-of-Words, ou CBOW.
- Le modèle Continuous Skip-Gram.

Le modèle CBOW apprend l'enchâssement en prédisant le mot actuel sur la base de son contexte. Le modèle de saut de mot continu apprend en prédisant les mots environnants à partir d'un mot courant [76].

Le modèle de Skip-Gram continu apprend en prédisant les mots environnants à partir d'un mot courant.

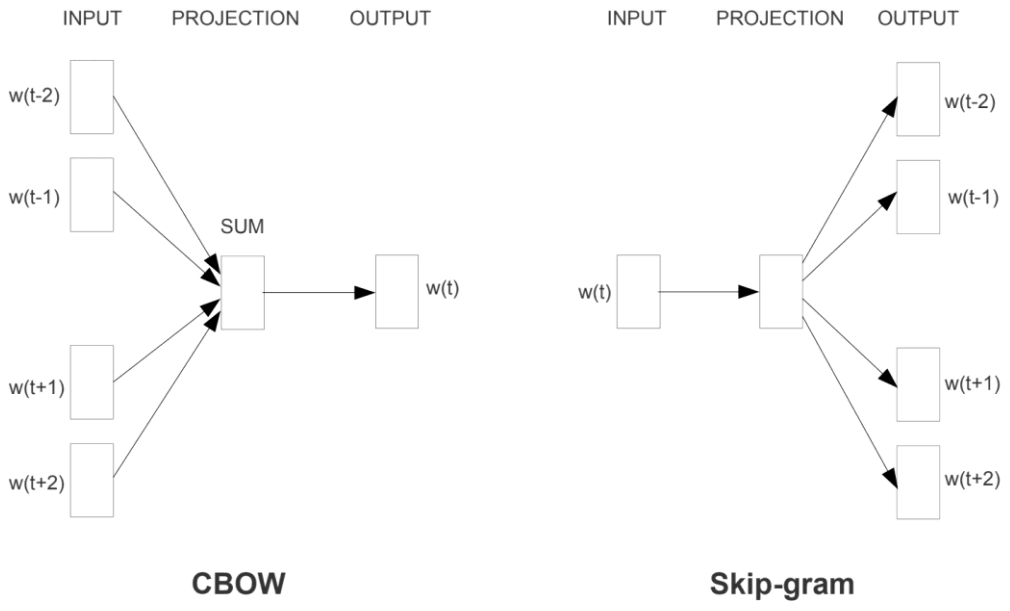

Figure 31 Modèles d'entrainement Word2Vec [76]

3.2.2 Word embeddings avec l'« Embedding layer »

Un Embedding layer capture la signification sémantique des éléments en rapprochant des éléments sémantiquement similaires. Il s'agit de faire correspondre des entrées de grande dimension avec des espaces de faible dimension afin que les entrées similaires soient proches les unes des autres. Ce layer détermine quels mots sont similaires les uns aux autres par des relations géométriques qui capturent des relations sémantiques, telles que la relation entre un pays et sa capitale (Figure 32) [77].

Il n'est qu'une couche cachée spéciale de taille d. Il peut être combiné avec n'importe quelle couche cachée. Cette couche est connectée à une seule couche cachée, qui fait correspondre l'ensemble des index à leur embeddings [77].

L'Embedding layer utilise l'ensemble du vocabulaire de codage. Ces vecteurs sont appris en tant que modèle d'entraînement. Les dimensions résultantes sont : (lot, séquence, fusion). Les poids combinés sont initialisés de manière aléatoire. Au cours du processus de formation, ils sont progressivement ajustés par rétropropagation [77].

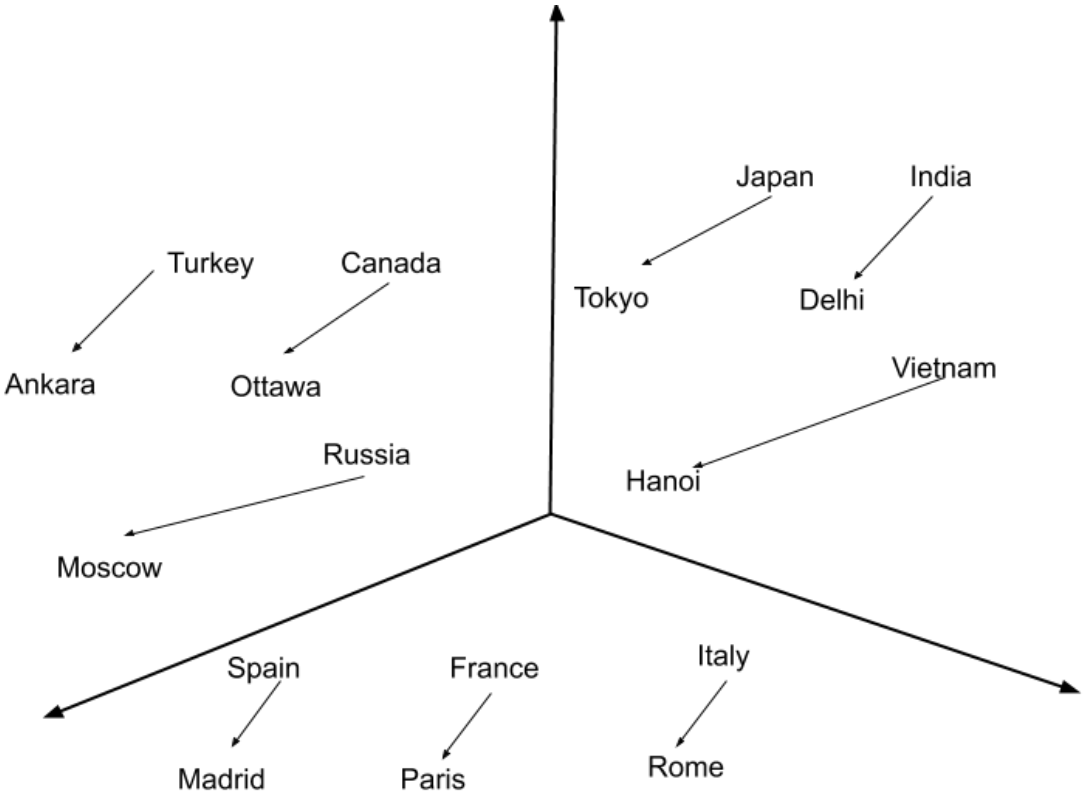

Figure 32 Relations géométriques qui capturent des relations sémantiques [77]

3.3 Architecture du modèle

Comme expliqué précédemment les architectures RNN simples sont confrontées au problème du Vanishing gradient. Une solution à ce problème serait d'utiliser des architectures RNN plus robustes et plus complexes telle que l'architecture LSTM et GRU. Tenant compte de cela, nous avons opté pour l'utilisation de ces deux architectures dans notre modèle, mais au lieu de se contenter d'une couche GRU classique qui apprend seulement de l'entrée (seulement des cellules précédentes) et pas des cellules ultérieures dans la séquence aussi « Obtenir des informations du futur » nous avons estimé qu'une couche bidirectionnelle apporterait un bénéfice notable en performance à notre modèle surtout dans le contexte de l'analyse de la polarité des sentiments de ressources textuelle.

Nous proposons un modèle qui consiste en une fusion d':

- Un Embedding layer (pour construire la matrice d'Embedding E)
- Un BGRU layer (un GRU bidirectionnel) avec une fonction d'activation sigmoïde
- Un LSTM layer
- Un Dense layer entièrement connecté (fully connected Dense layer) avec une fonction d'activation sigmoïde.

Nous avons aussi juger nécessaire d'appliquer une régularisation par Dropout aux couches BGRU et LSTM pour échapper au phénomène de sur-apprentissage « Overfitting » que les architectures complexes et profondes telle que là notre ont à faire face. La Figure 33 ci-dessous montre l'architecture que nous proposons.

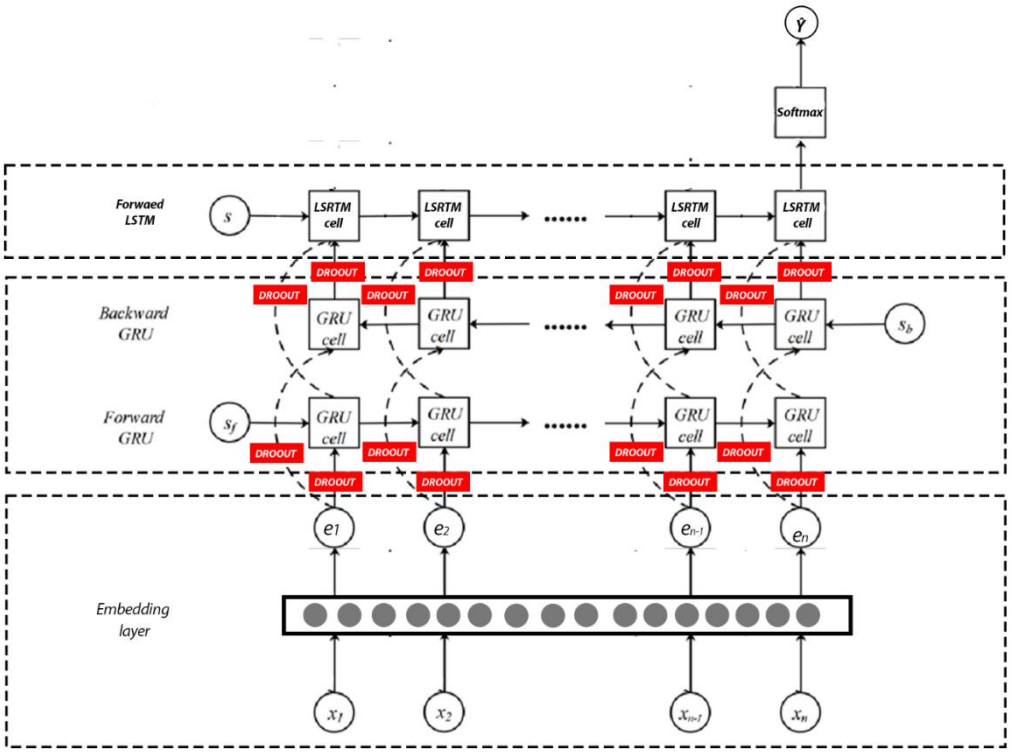

Figure 33 Architecture du model

3.3.1 L'embedding layer

Keras propose un embedding layer qui peut être utilisée pour les réseaux neuronaux sur des données textuelles.

Elle nécessite que les données d'entrée soient codées en entier, afin que chaque mot soit représenté par un entier unique. Cette étape de préparation des données peut être effectuée à l'aide de l'API Tokenizer également fournie avec Keras.

L'embedding layer est initialisé avec des poids aléatoires et apprend une incorporation pour tous les mots de l'ensemble de données d'entraînement.

Il s'agit d'une couche flexible qui peut être utilisée de diverses manières, par exemple :

- Elle peut être utilisée seule pour apprendre un encastrement de mots qui peut être sauvegardé et utilisé dans un autre modèle plus tard.

- Elle peut être utilisée comme partie d'un modèle d'apprentissage profond où l'incorporation est apprise en même temps que le modèle lui-même.
- Elle peut être utilisée pour charger un modèle d'incorporation de mots pré-entraîné, un type d'apprentissage par transfert.

La couche d'embedding est définie comme la première couche de notre modèle. Elle doit spécifier 3 arguments :

- input_dim : C'est la taille du vocabulaire dans les données textuelles. Par exemple, si les données sont codées en entier avec des valeurs comprises entre 0 et 10, la taille du vocabulaire sera de 11 mots.
- output_dim : Il s'agit de la taille de l'espace vectoriel dans lequel les mots seront intégrés. Elle définit la taille des vecteurs de sortie de cette couche pour chaque mot. Par exemple, elle peut être de 32 ou 100 ou même plus.
- input_length : Il s'agit de la longueur des séquences d'entrée, comme nous le définirions pour toute couche d'entrée d'un modèle Keras. Par exemple, si tous nos documents d'entrée sont composés de 1000 mots, cette valeur sera de 1000.

La sortie de l'embedding layer est un vecteur 2D avec un embedding pour chaque mot de la séquence de mots d'entrée (document d'entrée).

Pour notre modèle nous avons choisi d'utiliser un input**_dim** = 200 et un output**_dim** = 128. L'input**_length** n'a pas été spécifié explicitement car la taille des documents en entré utilisés varie largement.

La Figure 34 ci-dessous montre l'architecture d'un Embedding layer.

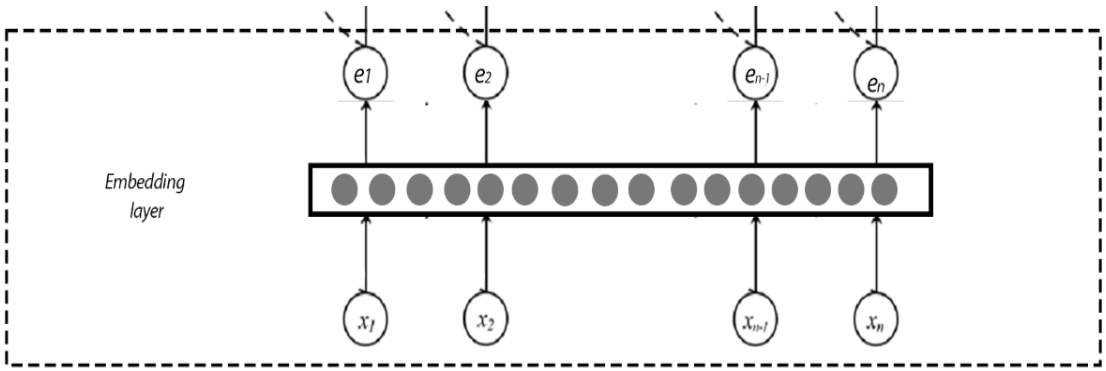

Figure 34 Couche d'Embedding

3.3.2 BGRU layer

Les GRU bidirectionnels sont un type de réseaux neuronaux récurrents bidirectionnels avec uniquement les portes d'entrée et d'oubli. Ils permettent d'utiliser des informations provenant à la fois des étapes temporelles précédentes et des étapes temporelles ultérieures pour faire des prédictions sur l'état actuel, ce qui est La Figure 35 présente la structure typique d'un réseau neuronal récurrent bidirectionnel.

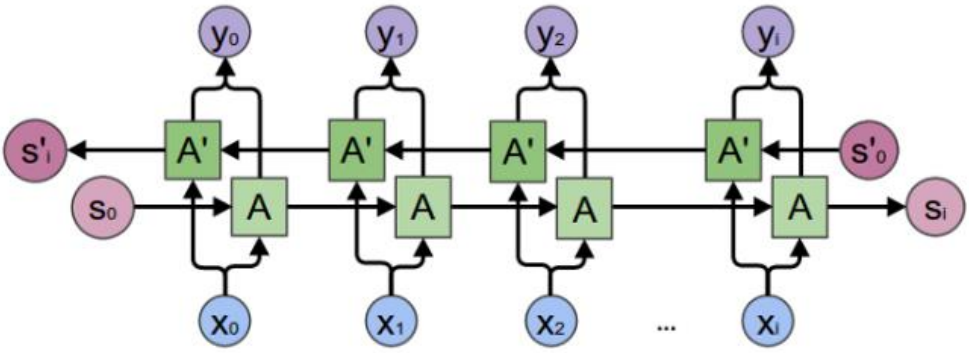

Figure 35 RNN bidirectionnel [78]

En remplaçant chaque A et A' du diagramme par une unité récurrente à porte, on obtient un GRU bidirectionnelle.

Étant donné que les RNN ne peuvent traiter les séquences que de l'avant vers l'arrière et ne peuvent pas obtenir d'informations sur la position, cela entraîne une perte d'informations. Le Bi-RNN ajoute des RNN pour traiter d'autres informations.

La structure de base du Bi-RNN consiste principalement à diviser un RNN ordinaire en deux voies. L'une dans le sens des aiguilles d'une montre et la seconde dans le sens inverse des aiguilles d'une montre, les deux RNN sont connectés à la même couche de sortie. Cette structure fournit l'ensemble des informations contextuelles de la séquence d'entrée de la couche de sortie. La construction d'un modèle d'analyse de sentiments BGRU nécessite que l'historique d'entrée doit être entré dans le GRU avant et le GRU arrière simultanément, capturant ainsi un maximum d'informations contextuelles.

La relation positionnelle entre le mot du sentiment et les autres mots est cruciale puisque nous présageons la polarité d'un document. Cette couche donne en sortie un vecteur qui contient des informations sur l'entrée actuelle et les entrées précédentes ce vecteur passe par la fonction d'activation, et la sortie est le nouvel état caché, ou la mémoire du réseau. Par conséquent, en utilisant BGRU, nous pouvons obtenir de meilleurs résultats.

Pour cette couche nous avons **256** unités de traitement, une fonction d'activation **sigmoïde (**Figure 60**)** et un dropout de **20%** (Figure 36).

3.3.3 LSTM layer

Cette avent dernière couche vas recevoir en entrer les vecteurs produit par la couche BGRU ce qui permettra de sauvegarder le contexte des mots a travers différentes couches et minimiser la perte d'information causée par les reset gates de la couche précédente.

Se layer va quant à lui produire un vecteur final pour chaque entrée qui sera ensuite transmit au layer finale pour une prediction.

Pour cette couche nous avons **128** unités de traitement, une fonction d'activation **sigmoïde (**Figure 60**)** et un dropout de **20%** (Figure 36).

3.3.4 dense layer

Cette dernière couche va quant à elle simplement faire une prédiction sur le vecteur reçu en entrée de la part du layer précèdent. Cette prédiction se fera avec la fonction sigmoïde **(**Figure 60**)**.

Cette couche va produire une sortie représentant la note de polarite prédite de l'entrée (0 ou 1).

3.3.5 Régularisation par « Dropout »

La régularisation consiste à ajouter une pénalité au réseau afin de réduire sa complexité et d'éviter un surapprentissage.

Le phénomène de surapprentissage se produit lorsqu'un modèle s'adapte si étroitement aux données d'apprentissage disponibles qu'il traite le bruit ou les particularités des données comme de véritables signaux de données, ce qui réduit la précision des prédictions sur des instances non vues.

Une préoccupation fréquente au sujet de l'apprentissage profond est que les réseaux profonds doivent être enclins à l'ajustement excessif en raison de leurs énormes degrés de liberté. Cependant, des techniques de régularisation efficaces ont été développées pour pénaliser le surapprentissage « Overfitting ». Depuis les débuts de l'apprentissage automatique, la régularisation a toujours été un élément central et inhérent [79].

Le Deep Learning dispose de plusieurs nouvelles techniques particulièrement utiles, notamment la régularisation par Dropout, sans lequel il n'aurait pas pu faire beaucoup de progrès. La régularisation par Dropout bloque aléatoirement une fraction appelée « dropout rate » (taux d'abandon) des connexions d'un réseau [79].

Cette opération est réalisée grâce à un masque généré aléatoirement, qui a les mêmes dimensions que la connexion entre les deux couches. La Figure 36 montre comment une représentation visuelle de cette opération sur un réseau de neurones simple tandis que la Figure 37 montre son utilisation dans notre modèle.

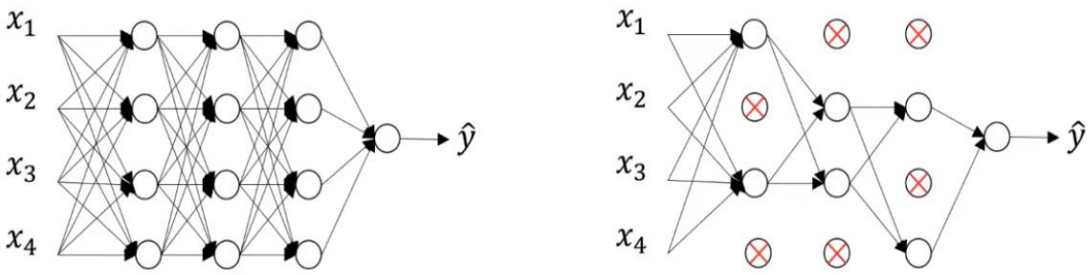

Figure 36 Réseaux sans application du dropout (à gauche) et avec (à droite)

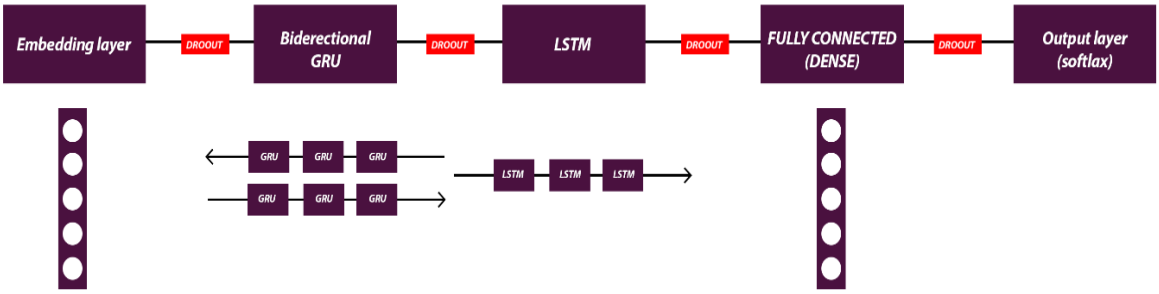

Figure 37 utilisation du dropout dans notre architecture

3.4 Fonction de coût

Pour notre modèle nous avons opté pour l'Entropie croisée binaire comme fonction de coût.

L'Entropie croisée binaire est une fonction de perte utilisée dans les tâches de classification binaire. Il s'agit de tâches qui répondent à une question avec seulement deux choix (oui ou non, A ou B, 0 ou 1, gauche ou droite). Il est possible de répondre à plusieurs questions indépendantes de ce type en même temps [80].

La fonction d'entropie croisée binaire calcule la perte d'un exemple en calculant la moyenne suivante :

$$Loss = -\frac{1}{Output\ size}\sum_{i=1}^{Output\ size} y_i \cdot log\hat{y}_i + (1 - y_i) \cdot \log\,(1 - \hat{y}_i) \qquad (9)$$

Ou $\hat{y}_i$ est la i-ième valeur scalaire dans la sortie du modèle, y_i est la valeur cible correspondante, et $Output\ size$ est le nombre de valeurs scalaires dans la sortie du modèle.

3.5 Conclusion

Dans ce chapitre nous avons d'abord énuméré les différentes étapes de prétraitement et de traitement des données du corpus utilisé puis présenté l'architecture de notre modèle et enfin nous avons exposé en détails le fonctionnement de ce dernier.

Le prochain chapitre sera consacré à l'implémentation et l'évaluation de notre modèle.

Chapitre 4 Implémentation et tests

Dans cette partie, nous présentons le matériel sur lequel nous avons développé notre modèle, les différents outils utilisés ainsi que les composantes applicatives réalisées. Ce modèle sera testé et évalué et par la suite comparé à d'autres architectures qui ont été proposées pour résoudre notre problématique.

4.1 Ressources matérielles

Durant les différentes étapes d'implémentations, d'entrainements et de tests de notre modèle, nous avons exploité les ressources matérielles de notre station personnelle qui possède les spécifications suivantes :

Processeur	Intel i7-8700k @ 3.70GHz
Mémoire vive	16.0 Go
Système d'exploitation	Microsoft Windows 10
Carte graphique	Nvidia Gtx 1080 MHz

4.2 Ressources logicielles

Pour l'implémentation de notre modèle nous avons utilisé de nombreuses ressources logicielles.

Ci-dessous plusieurs titres représentant les différents logiciels utilisés lors du développement de notre application :

Présentation du langage Python :

Python (Figure 38) est devenu l'un des langages de programmation les plus populaires pour la recherche au cours de la dernière décennie. Sa nature gratuite et open-source et sa vaste communauté en ligne sont quelques-unes des raisons de son succès. On trouve d'innombrables exemples d'augmentation de la productivité de la recherche grâce à Python dans une pléthore de domaines en ligne, notamment la science des données, l'intelligence artificielle et la recherche scientifique.

Figure 38 Logo Python

Présentation d'Anaconda :

Anaconda (Figure 39) est une distribution libre et open source des langages de programmation Python et R appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement. Les versions de paquetages sont gérées par le système de gestion de paquets conda. La distribution Anaconda est utilisée par plus de 6 millions d'utilisateurs et comprend plus de 250 paquets populaires en science des données adaptés pour Windows, Linux et MacOS.

Figure 39 Logo Anaconda

Présentation de l'IDE Spyder :

Créé et développé par Pierre Raybaut en 2008, Spyder (Figure 40) est maintenu, depuis 2012, par une communauté de développeurs qui ont pour point commun d'appartenir à la communauté Python scientifique.

En comparaison avec d'autres IDE pour le développement scientifique, Spyder a un ensemble unique de fonctionnalités - multiplateforme, open-source, écrit en Python et disponible sous une licence non-copyleft. Spyder est extensible avec des plugins, comprend le support d'outils interactifs pour l'inspection des données et incorpore des instruments d'assurance de la qualité et d'introspection spécifiques au code Python, tels que Pyflakes, Pylint et Rope.

Figure 40 Logo Spyder

4.3 Frameworks et bibliothèques

Ci-dessous plusieurs titres représentant les différents Frameworks et bibliothèques utilisés lors du développement de notre application :

Présentation du Framework TensorFlow :

TensorFlow (Figure 41) est une bibliothèque logicielle gratuite et open-source pour l'apprentissage automatique. Elle peut être utilisée pour toute une série de tâches, mais elle est particulièrement axée sur la formation et l'inférence de réseaux neuronaux profonds. C'est une bibliothèque de mathématiques symboliques basée sur le flux de données et la programmation différentiable. Elle est utilisée à la fois pour la recherche et la production chez Google.

TensorFlow a été développé par l'équipe Google Brain pour un usage interne à Google. Il a été publié sous la licence Apache 2.0 en 2015.

Figure 41 Logo TensorFlow

Présentation du Framework Keras :

Keras (Figure 42) est une bibliothèque open-source de composants de réseaux neuronaux écrits en Python. Keras est capable de fonctionner au-dessus de TensorFlow, Theano, PlaidML et autres. La bibliothèque a été développée pour être modulaire et conviviale, mais elle a initialement commencé dans le cadre d'un projet de recherche pour le système d'exploitation intelligent neuro-électronique ouvert ou ONEIROS. L'auteur principal de Keras est François Chollet, un ingénieur de Google qui a également écrit XCeption, un modèle de réseau neuronal profond. Bien que Keras ait été officiellement lancé, il n'a été intégré à la bibliothèque centrale TensorFlow de Google qu'en 2017. Un support supplémentaire a également été ajouté pour l'intégration de Keras avec Microsoft Cognitive Toolkit.

Composée d'une bibliothèque de composants d'apprentissage automatique couramment utilisés, notamment des objectifs, des fonctions d'activation et des optimiseurs, la plateforme open-source de Keras offre également un support pour

les réseaux neuronaux récurrents et convolutifs. En outre, Keras propose le développement de plateformes mobiles pour les utilisateurs ayant l'intention de mettre en œuvre des modèles d'apprentissage profond sur les smartphones, tant iOS qu'Android. En 2018, la bibliothèque compte 22 % d'utilisation à travers de ses plus de 200 000 utilisateurs.

Figure 42 Logo Keras

Présentation du module Numpy:

NumPy (Figure 43) est le paquetage fondamental pour le calcul scientifique en Python. Il s'agit d'une bibliothèque Python qui fournit un objet tableau multidimensionnel, divers objets dérivés (tels que les tableaux masqués et les matrices), ainsi qu'un assortiment de routines permettant d'effectuer des opérations rapides sur les tableaux, notamment des opérations mathématiques, logiques, de manipulation de formes, de tri, de sélection, d'E/S, de transformées de Fourier discrètes, d'algèbre linéaire de base, d'opérations statistiques de base, de simulation aléatoire et bien plus encore.

Au cœur du paquetage NumPy se trouve l'objet ndarray. Celui-ci encapsule des tableaux à n dimensions de types de données homogènes, de nombreuses opérations étant effectuées en code compilé pour des raisons de performances.

La vectorisation décrit l'absence de toute boucle explicite, d'indexation, etc., dans le code - ces choses ont lieu, bien sûr, juste "dans les coulisses" du code C optimisé et précompilé. Le code vectorisé présente de nombreux avantages, dont les suivants :

- Le code vectorisé est plus concis et plus facile à lire
- Moins de lignes de code signifie généralement moins de bogues
- Le code ressemble davantage à la notation mathématique standard (ce qui facilite, en général, le codage correct des constructions mathématiques)
- La vectorisation donne lieu à un code plus ‘pythonique’. Sans la vectorisation, notre code serait parsemé de boucles for inefficaces et difficiles à lire.

Figure 43 Logo NumPy

Présentation du module Pandas :

Pandas (Figure 44) est une bibliothèque Python de structures de données riches et d'outils permettant de travailler avec des ensembles de données structurés communs aux statistiques, à la finance, aux sciences sociales et à de nombreux autres domaines. La bibliothèque fournit des routines intégrées et intuitives permettant d'effectuer des manipulations et des analyses courantes sur de tels ensembles de données. Elle a pour but d'être la couche fondamentale pour l'avenir du calcul statistique en Python. Elle sert de complément solide à la pile Python scientifique existante tout en mettant en œuvre et en améliorant les types d'outils de manipulation de données que l'on trouve dans d'autres langages de programmation statistique tels que R.

Figure 44 Logo Pandas

Présentation du module Matplotlib :

Matplotlib (Figure 45) est une bibliothèque de traçage disponible pour le langage de programmation Python en tant que composant de NumPy, une ressource de traitement numérique de données volumineuses. Matplotlib utilise une API orientée objet pour intégrer les tracés dans les applications Python.

Comme Python est largement utilisé dans l'apprentissage automatique, des ressources comme NumPy et matplotlib sont souvent utiles pour modéliser les technologies d'apprentissage automatique. L'idée est que les programmeurs accèdent à ces bibliothèques pour des tâches clés dans un environnement Python plus large, et intègrent les résultats avec tous les autres éléments et fonctionnalités d'un programme d'apprentissage automatique, d'un réseau neuronal ou d'une autre machine avancée. L'utilité de NumPy et de matplotlib est liée aux nombres - l'utilité de matplotlib est spécifiquement liée aux outils de traçage visuel. En un sens, ces ressources sont donc plus analytiques que génératives. Cependant, toute cette infrastructure fonctionne ensemble pour permettre aux programmes d'apprentissage automatique de produire des résultats qui sont utiles aux manipulateurs humains.

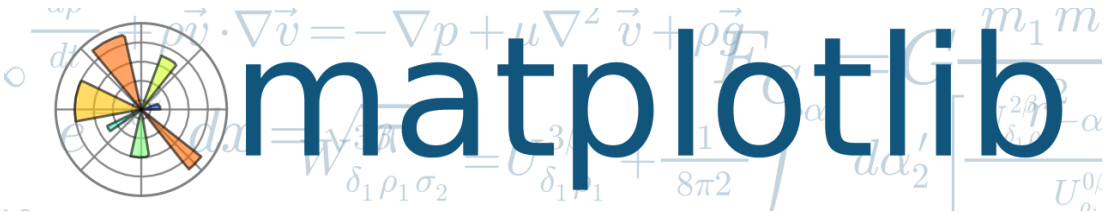

Figure 45 Logo Matplotlib

Présentation du module NLTK :

Natural Language Toolkit est une suite de modules de programmes, de jeux de données, de tutoriels et d'exercices, couvrant le traitement symbolique et statistique du langage naturel. NLTK est écrit en Python et distribué sous la licence open source GPL. Au cours des trois dernières années, NLTK est devenu populaire dans l'enseignement et la recherche.

Natural Language Toolkit (NLTK) a été développé en conjonction avec un cours de linguistique computationnelle à l'Université de Pennsylvanie en 2001 (Loper et Bird, 2002). Il a été conçu avec trois applications pédagogiques en tête : les devoirs, les démonstrations et les projets.

4.4 Acquisition du corpus

Dans l'objectif de mener les observations nécessaires à l'implémentation, l'entrainement et la validation de notre modèle représentant not système, nous avons utilisé deux corpus : le dataset IMDB Movies Reviews [81] et le dataset Amazon Reviews : Unlocked Mobile Phones [82].

Source	Avis positifs	Avis négatifs	Total des avis
IMDB	25000	25000	50000
AMAZON	284954	97061	413840

Le tableau ci-dessus montre une différence d'équilibre flagrante entre les deux classes (positif et négatif) dans le deuxième dataset. Ce déséquilibre peut en effet représenter un problème dans le contexte où notre modèle apprenne majoritairement des revues positives et minoritairement des revues négatives. Cela certes rendra notre modèle très efficace pour classifier correctement les revues positives mais en contrepartie le manque de revues d'entrainement négatives impactera négativement les performances du modèle quand il tentera de classifier des données de test négatives.

Pour remédier à ce problème nous avons jugé nécessaire d'équilibrer les deux classes en fusionnant une partie des revues négatives du dataset d'IMDB avec le dataset d'AMAZONE. Cela nous a effectivement permis d'avoir une base d'entrainement solide avec un équilibre plus ou moins parfait entre les deux classes. La Figure 46 montre la distribution des revues par sentiment dans le dataset fusionné.

Après l'équilibrage des deux classes et l'échantillonnage du dataset, le dataset devient :

Source	Avis positifs	Avis négatifs	Total des avis
IMDB+AMAZON	28230	27397	55 627

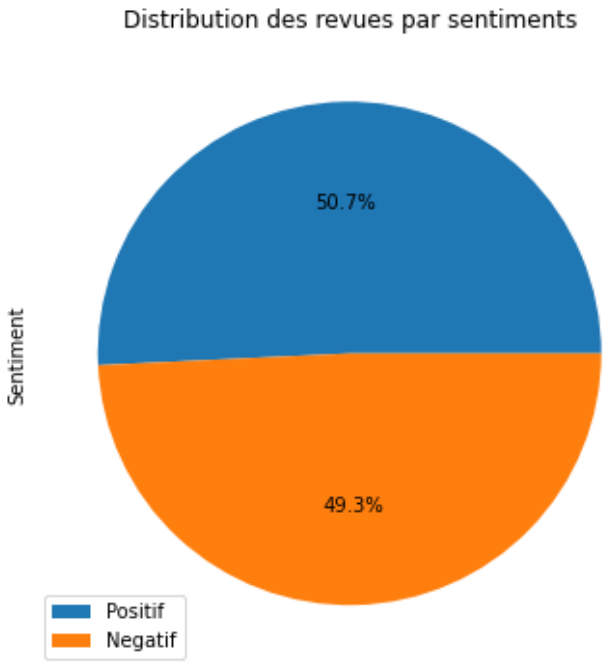

Figure 46 Distribution des revues par sentiments

4.5 Implémentation et fonctionnement

Dans cette section nous présentons la manière dont nous avons implémenté notre modèle et justifions les choix techniques pour lesquels nous avons opté.

4.5.1 Importation des modules et chargement des datasets

La première étape consiste à importer les modules nécessaires et charger les deux datasets (Figure 47).

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt


from wordcloud import WordCloud

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.utils import shuffle

from bs4 import BeautifulSoup
import re

from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import SnowballStemmer, WordNetLemmatizer
from nltk import sent_tokenize, word_tokenize, pos_tag

import tensorflow as tf
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras import utils
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Lambda
from tensorflow.keras.layers import Embedding
from tensorflow.keras.layers import LSTM, SimpleRNN, GRU
from tensorflow.keras.preprocessing.text import Tokenizer
from collections import defaultdict
from tensorflow.keras.layers import Convolution1D
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Embedding



df = pd.read_csv('E:\\Desktop\\pfe\\impl\\amazon_review_full_csv\\phone\\Amazon_Unlocked_Mobile.csv')
df.drop(['Product Name', 'Brand Name', 'Price','Review Votes'],inplace=True, axis=1)

df2 =  pd.read_csv('E:\\Desktop\\pfe\\impl\\IMDBDataset.csv')
```

Figure 47 Chargement des modules et datasets

4.5.2 Prétraitement des données

La phase de prétraitement consiste elle-même de plusieurs étapes présentées ci-dessous :

Fusion des deux datasets :

Dans cette étape nous encodons les sentiments positifs en 1 et les sentiments négatifs en 0 dans les deux datasets puis nous chargeons une partie des revues négatives du dataset d'**IMDB** et les fusionnons avec le dataset d'**Amazon Phones** Figure 48.

```

df = df.sample(frac=0.1, random_state=0) #échantillonnage d'Amazon dataset
df2=df2[df2.sentiment=='negative'].sample(frac=0.7, random_state=0)# échantillonnage des revues negatives de IMDB

# Suppression des valeurs nuls
df.dropna(inplace=True)
df2.dropna(inplace=True)

# Suppression des revues neutres (rating= 3)
df = df[df['Rating'] != 3]

# Encoder 4 etoiles et 5etoiles en 1 (sentiment positif) et 1 etoile et 2 etoiless en 0 (sentiment negatif )
df['Sentiment'] = np.where(df['Rating'] > 3, 1, 0)
df.head()
df2['sentiment'] = np.where(df2['sentiment'] == 'positive', 1, 0)

# fusion des revues negatives d'imdb avec le dataset d'amazon
df2.rename(columns={'review': 'Reviews','sentiment': 'Sentiment'}, errors="raise",inplace = True)
dff =df.append(df2,ignore_index=True)

print(dff.columns)
```

Figure 48 Fusion des deux datasets

Division et nettoyage du dataset:

Dans cette étape nous divisons le dataset en un ratio de 30% de données de tests et 70% de données d'entrainement puis nous nettoyons ces dernières comme expliqué dans le chapitre précédant (Figure 49).

```
X_train, X_test, y_train, y_test = train_test_split(dff['Reviews'], dff['Sentiment'], \
                                                    test_size=0.3, random_state=0)


print('Charger %d exemples entrainement et %d exemples de test. \n' %(X_train.shape[0],X_test.shape[0]))
print('Une revue dans l ensemble d entrainement : \n', X_train.iloc[15])


def cleanText(raw_text, remove_stopwords=False, stemming=False, split_text=False, \
              ):
    '''
    Convert a raw review to a cleaned review
    '''
    text = BeautifulSoup(raw_text, 'lxml').get_text()  #suppression des balises html
    letters_only = re.sub("[^a-zA-Z]", " ", text)  # suppression des non-character
    words = letters_only.lower().split() # convertion en miniscule

    if remove_stopwords: # suppression des stopword
        stops = set(stopwords.words("english"))
        words = [w for w in words if not w in stops]

    if stemming==True: # Racinisation
#         stemmer = PorterStemmer()
        stemmer = SnowballStemmer('english')
        words = [stemmer.stem(w) for w in words]

    if split_text==True:
        return (words)

    return( " ".join(words))
```

Figure 49 Division et nettoyage du dataset

Vectorisation et encodage des données :

Dans cette étape nous vectorisons les données d'entrainement et de tests tandis que nous encodons en encodage « one-hot » leurs labels (Figure 50).

```
top_words = 20000
maxlen = 100
batch_size = 32
nb_classes = 2
nb_epoch = 3


# Vectorisation de X_train et X_test en tensor 2D
tokenizer = Tokenizer(nb_words=top_words) #nous considerons seulement les top 20000 mots dans le corpus
tokenizer.fit_on_texts(X_train)


sequences_train = tokenizer.texts_to_sequences(X_train)
sequences_test = tokenizer.texts_to_sequences(X_test)

X_train_seq = sequence.pad_sequences(sequences_train, maxlen=maxlen)
X_test_seq = sequence.pad_sequences(sequences_test, maxlen=maxlen)


# encodage one-hot de  y_train et y_test
y_train_seq = utils.to_categorical(y_train, nb_classes)
y_test_seq = utils.to_categorical(y_test, nb_classes)

print('X_train shape:', X_train_seq.shape)
print('X_test shape:', X_test_seq.shape)
print('y_train shape:', y_train_seq.shape)
print('y_test shape:', y_test_seq.shape)
```

Figure 50 Vectorisation et encodage

4.5.3 Construction de l'architecture et entrainement du modèle

---

Comme déjà expliqué dans le **chapitre 3** (Figure 33) l'architecture de notre modèle comporte une couche d'embedding, une couche bidirectionnelle GRU,

une couche LSTM et enfin une couche dense à activation sigmoïde. La Figure 51 montre son implémentation.

```
218
219
220  # Construction du modele
221  model1 = Sequential()
222  model1.add(Embedding(top_words, 128))
223  model1.add(
224      tf.keras.layers.Bidirectional(
225          tf.keras.layers.GRU(256, return_sequences=True, activation = 'sigmoid',dropout=0.2, recurrent_dropout=0.2))
226      )
227
228  model1.add(LSTM(128))
229  model1.add(Dense(2, activation='sigmoid'))
230  model1.summary()
231
232  # Compilation du model LSTM
233  model1.compile(loss='binary_crossentropy',
234                 optimizer='adam',
235                 metrics=['accuracy'])
236
237  hist=model1.fit(X_train_seq, y_train_seq, batch_size=128, epochs=6, verbose=1)
238
239
240
```

Figure 51 Architecture du modèle et entrainement

4.5.4 Evaluation du modèle

Dans cette dernière phase le modèle est évalué en utilisant 30% du dataset précédemment échantillonné ceci est illustré à travers le code ci-dessous (Figure 52).

```
269
270
271
272  # Model evluation
273  score = model1.evaluate(X_test_seq, y_test_seq, batch_size=batch_size)
274  print('Test loss : {:.4f}'.format(score[0]))
275  print('Test accuracy : {:.4f}'.format(score[1]))
276
277
278
279
280
281
```

Figure 52 Evaluation du modèle

4.6 Evaluation des résultats et comparaison

Dans cette section nous allons exposer les résultats obtenus par notre modèle et par la suite les comparer aux performances d'autres architecture telle que l'algorithme de foret aléatoire (Random Forest) ainsi que 3 techniques basées sur le Deep Learning.

4.6.1 Résultats de notre modèle

Dans le but d'évaluer notre modèle nous l'avons entrainé et testé avec deux corpus, le corpus d'Amazon ayant un déséquilibre des deux classes (positive **74%** et négative **36%**) et un autre corpus équilibré par la fusion des revues négatives du corpus d'IMDB et du corpus d'Amazon.

Le tableau ci-dessous illustre les résultats obtenus par notre architecture en se basant sur le corpus équilibré :

Ratio classe positive	50.75%
Ratio classe négative	49.25%
Exemple d'entrainement	26688
Exemple de test	11439
Exactitude en entrainement	0.94
Taux de perte en entrainement	0.1324
Exactitude en test	0.95
Taux de perte en test	0.1330

Le tableau ci-dessous représente la matrice de confusion du même cas :

Classes	Précision	Rappel	Score-F1	Effectif
Classe négative	0.95	0.96	0.96	2951
Classe positive	0.96	0.95	0.96	8488

En se basant sur le corpus équilibré notre modèle est tout aussi performant dans la classification des revues négatives que des revues positives. Cela est dû principalement au fait que ce dernier a gagné en expérience avec autant de revues négatives que de revues positives.

Le tableau ci-dessous illustre les résultats obtenus par notre architecture en se basant sur le corpus non équilibré :

Ratio classe positive	74.04%
Ratio classe négative	25.96%
Exemple d'entrainement	26688
Exemple de test	11439
Exactitude en entrainement	0.93
Taux de perte en entrainement	0.1737
Exactitude en test	0.9364
Taux de perte en test	0.2024

Le tableau ci-dessous représente la matrice de confusion du même cas :

Classes	Précision	Rappel	Score-F1	Effectif
Classe négative	0.89	0.86	0.87	2951
Classe positive	0.95	0.96	0.96	8488

Il est clair qu'on se basant sur le corpus non équilibré notre modèle a plus de mal à identifier les revues négatives présentes dans l'ensemble de test (Rappel = **0.86**), cela est vraisemblablement dû au manque d'expérience de ce dernier avec la classe négative qui rappelons-le, a un ratio drastiquement inférieur a la classe positive.

4.6.2 Résultats des architectures similaires

Dans cette section nous allons présenter les résultats obtenus par d'autres architectures similaires basées sur le Deep Learning en se basant sur le corpus équilibré.

CNN + LSTM :

Le tableau ci-dessous illustre les résultats obtenus par l'architecture CNN+LSTM en se basant sur le corpus équilibré :

Ratio classe positive	52%
Ratio classe négative	48%
Exemple d'entrainement	26688
Exemple de test	11439
Exactitude en entrainement	0.9307
Taux de perte en entrainement	0.1751
Exactitude en test	0.9331
Taux de perte en test	0.1981

Le tableau ci-dessous représente la matrice de confusion du même cas :

Classes	Précision	Rappel	Score-F1	Effectif
Classe négative	0.86	0.89	0.87	2951
Classe positive	0.96	0.95	0.95	8488

Il est indéniable que l'architecture CNN+LSTM donne un très bon résultat mais il est tout aussi clair que cette dernière est moins efficace que notre modèle proposé particulièrement dans la classification des éléments de la classe négative (Rappel = 0.89) même en se basant sur un corpus équilibré.

GRU + LSTM :

Le tableau ci-dessous illustre les résultats obtenus par l'architecture GRU+LSTM en se basant sur le corpus équilibré :

Ratio classe positive	50.7%
Ratio classe négative	49.3%
Exemple d'entrainement	26688
Exemple de test	11439
Exactitude en entrainement	0.924
Taux de perte en entrainement	0.1914
Exactitude en test	0.9220
Taux de perte en test	0.2268

Le tableau ci-dessous représente la matrice de confusion du même cas :

Classes	Précision	Rappel	Score-F1	Effectif
Classe négative	0.93	0.76	0.83	2951
Classe positive	0.92	0.98	0.95	8488

La matrice de confusion ci-dessus met en évidence que l'architecture GRU+LSTM est très inefficace par rapport aux autres architectures lorsqu'il s'agit de classifier des revues négatives (Rappel = 0.76) cela montre bien que notre

choix d'utiliser une couche GRU bidirectionnelle a permis à notre modèle de gagner en performance même lorsque qu'il s'agit de classifier des classes non équilibrées.

LSTM :

Le tableau ci-dessous illustre les résultats obtenus par l'architecture ayant une unique couche LSTM en se basant sur le corpus équilibré :

Ratio classe positive	50.7%
Ratio classe négative	49.3%
Exemple d'entrainement	26688
Exemple de test	11439
Exactitude en entrainement	0.924
Taux de perte en entrainement	0.1921
Exactitude en test	0.9306
Taux de perte en test	0.1852

Le tableau ci-dessous représente la matrice de confusion du même cas :

Classes	Précision	Rappel	Score-F1	Effectif
Classe négative	0.88	0.91	0.90	2951
Classe positive	0.96	0.94	0.95	8488

La matrice de confusion ci-dessus met en évidence que l'architecture ayant une couche unique LSTM est performante mais cependant, elle reste toujours moins efficace que celle sur laquelle est basé notre modèle.

4.6.3 Résultats de la méthode Random Forest (Forêt d'arbres décisionnels)

Dans cette section nous allons présenter les résultats obtenus par la méthode Random Forest. Il est aussi nécessaire de préciser que le corpus utilisé pour l'entrainement de cette méthode est de taille inférieure à celui utilisé précédemment pour des raisons de capacités matérielles.

En effet la méthode Random Forest est très gourmande en mémoire et en cycle d'horloge, cela nous a poussé à réduire la taille de l'ensemble d'entrainement. Nous avons aussi eu recours à la technique Grid Search pour optimiser les hyper paramètres et en retenir les plus performants.

Le tableau ci-dessous illustre les résultats obtenus par la méthode Random Forest en se basant sur le corpus équilibré réduit :

Nombre d'estimateurs	100
Profondeur maximale	Illimité
Ratio classe positive	53.1%
Ratio classe négative	46.9%
Exemple d'entrainement	6642
Exemple de validation	3321
Exemple de test	3321
Exactitude en entrainement	0.90
Exactitude en validation	0.899
Exactitude en test	0.898

Le tableau ci-dessous représente la matrice de confusion du même cas pour l'étape de validation et de test :

Etape	Précision	Rappel	Score-F1	Effectif
Validation	0.896	0.918	0.9068	3321
Test	0.888	0.924	0.9056	3321

Il est clair que la méthode Random Forest est une méthode efficace mais elle reste très couteuse en mémoire et en temps de traitement de plus elle ne peut pas bénéficier des avantages de l'architecture de traitement parallèle des données des processeurs graphiques telle que les méthodes présentées dans la section précédente.

4.7 Conclusion

A travers ce chapitre nous avons présenté les outils matériels et logiciels utilisés pour l'implémentation de notre modèle. Nous avons par la suite présenté en détail cette dernière et enfin nous avons exposé les résultats obtenus par notre modèle tout en les évaluant en les comparant avec les performances d'autres architectures.

Conclusion et perspectives

De nos jours, il est observé que dans le processus de prise de décision concernant un produit, un service, un film, des questions sociales, l'analyse des sentiments ou l'extraction d'opinions joue un rôle très important. L'extraction d'opinion n'est pas seulement constituée des concepts de l'extraction de texte, mais aussi des concepts de la recherche d'information. Pour une bonne classification, la pondération des caractéristiques, qui joue un rôle crucial, est l'un des principaux défis de l'extraction d'opinions. Les médias sociaux sont l'une des principales composantes du Web. Selon les calculs, 9 utilisateurs sur 10 utilisent une forme de média social. Aujourd'hui, l'utilisateur d'Internet crée une grande quantité de données. Ainsi, pour le contenu web, les utilisateurs deviennent des co-créateurs. Sur les médias sociaux, la contribution des utilisateurs va du téléchargement de photos et de vidéos aux commentaires, en passant par les articles de blog et les tweets. Sur l'internet, les données disponibles sont des textes non structurés.

Sur les médias sociaux, les points de vue ou les opinions sont exprimés par les commentaires ou les messages des utilisateurs. Avec la croissance de la demande concernant l'accessibilité des ressources d'opinion telles que les critiques de blog, les critiques de films, les tweets de réseaux sociaux, les critiques de produits, il en résulte un nouveau défi : l'extraction d'un grand volume de données/texte et la nécessité d'un algorithme approprié pour l'analyse de l'opinion des autres. C'est important pour les organisations car cela les aide à améliorer leurs services ou leurs produits et à prendre des décisions pour l'avenir.

Nous sommes parvenus, par le biais de ce projet, à proposer une architecture originale de fouille d'opinions en se basant sur l'apprentissage profond qui consiste une couche d'Embedding, une couche bidirectionnelle GRU, une couche LSTM et enfin une couche dense a activation sigmoïde. Notre modèle s'est montré très performant avec le jeu de données utilisé (Amazon+IMDB). Cependant il reste à évaluer ce modelé avec un ensemble de données bien plus volumineux et représentatif des données détenues par les entreprises commerciales et l'intégrer a un logiciel d'analyse de texte.

Ce travail fut une occasion pour nous de compléter de manière transversale nos compétences en informatique et d'élargir et d'approfondir nos connaissances en intelligence artificielle. Cependant des perspectives d'améliorations de notre modèle restent envisageables pour être enrichi par des fonctionnalités avancées telles que la possibilité de noter des aspects individuels des produits examinés qui est un point faible des modèles actuellement proposés qui ont tendance à viser une simple classification globale des avis.

Bibliographie

[1] W. T, «Just how mad are you? Finding strong and weak opinion clauses,» chez *19th National Conference on Artificial Intelligence*, 2004.

[2] M. H. a. B. Liu, "Mining Opinion Features in Customer," in *19th National Conf on Artificial intelligence*, 2004.

[3] L. B. H. M, «Mining and summarizing customer reviews,» chez *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining 22:2525*, 2004.

[4] A. E. a. F. Sebastiani, «Determining Term Subjectivity and Term Orientation for Opinion Mining,» chez *Proceedings the EACL*, 2006.

[5] S. F. E. A, «SENTIWORD NET : A Publicly Available Lexical Resource for Opinion Mining,» chez *Proceedings of LREC*, 2006.

[6] S. F. E. A, «Determining the semantic orientation of terms through gloss classification,» chez *Proceedings of the 14th ACM international conference on Information and knowledge management*, Bremen, Germany, 2005.

[7] B. Liu, «Sentiment analysis and subjectivity,» January 2010.

[8] S. Mukhopadhyay, «Opinion mining in management research: the state of the art and the way forward,» January 2018.

[9] C. w. Haji Binali, «A State Of The Art Opinion Mining And Its Application Domains,» chez *IEEE International Conference on Industrial Technology*, 2009.

[10] M. C. D. D. P. A. T. S. S. D. W. a. A. Y. O. Etzioni, «Unsupervised named-entity extraction from the Web: An experimental study,» 2004.

[11] K. A. a. G. Wang, «An Unsupervised Opinion Mining Approach for Japanese Weblog Reputation Information Using an Improved SO-PMI Algorithm,» April,2008.

[12] T. P, «Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews,» chez *the Meeting of the Association for ComputationalLinguistics*, 2002.

[13] T. D. Knight, «Imdb,» 2008. [En ligne]. Available: https://www.imdb.com/title/tt0468569/.

[14] «The Australian,» 2008. [En ligne]. Available: http://www.theaustralian.news.com.au/politics/opinion/.

[15] «Amazon,» 2008. [En ligne]. Available: https://www.amazon.com/.

[16] A. L. A. B. A. H. J. LESKOVEC, «The Dynamics of Viral Marketing,» chez *ACM Transactions on the web*, 2007.

[17] C. C. J. H. B. Ana¨ıs Collomb, «A Study and Comparison of Sentiment Analysis Methods for Reputation Evaluation».

[18] S. N. D. S. B. Moralwar, Different Approaches of Sentiment Analysis, vol. 3, 2015, p. 6.

[19] P. L. a. M. Turney, «Measuring Praise and Criticism: Inference of Se-mantic Orientation from Association,» chez *ACM Transactions on Information Systems*, 2003.

[20] M. M. M. R. Kamps J., Using WordNet to Measure Semantic Orientation of Adjectives, 2004.

[21] V. W. J. Hatzivassiloglou, «Effects of Adjective Orientation and Gradability on Sentence Subjectivity,» chez *the 18th International Conference on Computational Linguistics*, New Brunswick, NJ, 2000.

[22] A. I. D. Kennedy, «Sentiment Classification of Movie and Product Reviews Using Contextual Valence Shifters,» chez *Computational Intelligence*, 2006, p. 110–125 .

[23] A. B. S. U. M. Andreevskaia, «All Blogs Are Not Made Equal: Exploring Genre Differences in Sentiment Tagging of Blogs,» chez *International Conference on Weblogs and Social Media (ICWSM-2007)*, Boulder, 2007.

[24] J. Akshay, «A Framework for Modeling Influence, Opinions and Structure in Social Media,» chez *the Twenty-Second AAAI Conference on Artificial Intelligence*, Vancouver, July (2007).

[25] K. a. S. M. Durant, «Mining Sentiment Classification from Political Web Logs,» chez *Proceedings of Workshop on Web Mining and Web Usage Analysis of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia , 2006.

[26] S. Prasad, «Micro-blogging sentiment analysis using Bayesian classification methods,» 2010.

[27] H. T. a. D. Patel, «Approaches for Sentiment Analysis on Twitter:A State-of-Art study».

[28] A. P. a. P. Paroubek, «Twitter based system: Using twitter for disambiguating sentiment ambiguous adjectives,» chez *Proceeding SemEval '10 Proceedings of the 5th International Workshop on Semantic Evaluation*, 2010.

[29] X. L. W. L. a. P. Y. B. Liu, «Text classification by labeling words,» chez *Proceedings of the National Conference on Artificial Intelligence*, Menlo Park, CA, 2004.

[30] W. G. a. R. L. P. Melville, «Sentiment analysis of blogs by combining lexical knowledge with text classification,» chez *In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009.

[31] J. Alzubi, «Machine Learning from Theory to Algorithms: An Overview,» *Journal of Physics,* 2018.

[32] A. N.-M. Rich Caruana, «An Empirical Comparison of Supervised Learning Algorithms,» chez *06 Proceedings of the 23rd international conference on Machine learning*, Pittsburgh, Pennsylvania,USA, 2006.

[33] V. Nasteski, «An overview of the supervised machine learning methods,» 2017.

[34] Q. &. W. Y. Liu, «Supervised Learning,» 2012.

[35] P. &. C. M. &. D. S. Cunningham, «Supervised Learning 10.1007/978-3-540-75171-7_2.,» 2008.

[36] A. &. A. A. Soofi, «Classification Techniques in Machine Learning: Applications and Issues,» *Journal of Basic & Applied Sciences,* vol. 13, pp. 459-465, 2017.

[37] Z. B.-w. Shen Rong, «The research of regression model in machine learning field,» chez *MATEC Web of Conferences*, 2018.

[38] S. S. Kesavaraj G, «A study on classification techniques in data mining,» chez *Fourth International Conference on Computing, Communications and Networking Technologies*, 2013.

[39] S. S. K. A. Singh M, «Performance Analysis of Decision Trees,» *International Journal of Computer Applications,* 2013.

[40] D. MH, «Data mining: Introductory and advanced topics,» Pearson Education, India, 2006.

[41] S. J. Peter Norvig, Artificial Intelligence: A Modern Approach, Pearson, 2015.

[42] N. S. D. R.Saji Priya, «A Review on Support Vector Machine: a Classification Method».

[43] G. H. Lewes, «Support Vector Machines for Classification,» chez *Efficient Learning Machines*, 2015, pp. 39-66.

[44] L. Breiman, «Random Forests,» chez *Machine Learning*, 2001, pp. 5-32.

[45] Y. G. D. Amit, «Shape quantization and recognition with randomized trees,» *NeuralComputation,* vol. 9, pp. 1545-1588, 1997.

[46] L. Breiman, «Bagging Predictors,» *Machine Learning,* vol. 24, pp. 123-140, 2001.

[47] D. R. C. a. J. R. S. Adele Cutler, «Random Forests,» chez *Ensemble Machine Learning: Methods and Applications*, springer, january 2011, pp. 157-176.

[48] A. Mathew, «Deep Learning Techniques: An Overview,» chez *Advanced Machine Learning Technologies and Applications* , springer, january 2021, pp. 599-608.

[49] S. Shetty, «Deep Learning Algorithms and Applications in Computer Vision,» *INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING,* pp. 195-201, july 2019.

[50] M. H. Hisham El-Amir, Deep Learning Pipeline, Jizah, Egypt: apress.

[51] A. M. Ajay Shrestha, «Review of Deep Learning Algorithms and Architectures».

[52] B. Maurice, «Fonctionnement du neurone artificiel,» 15 september 2018. [En ligne]. Available: https://deeplylearning.fr/cours-theoriques-deep-learning/fonctionnement-du-neurone-artificiel/. [Accès le 2021].

[53] N. A. J. K. A. Shadman Sakib, «An Overview of Convolutional Neural Network: Its Architecture and Applications,» november 2018.

[54] R. N. Keiron O'Shea, «An Introduction to Convolutional Neural Networks».

[55] N. JOSHI, «Sequence modeling for beginners,» 01 APRIL 2020. [En ligne]. Available: https://www.allerin.com/blog/sequence-modeling-for-beginners.

[56] N. Bostrom, «Machine i ntellige nce i s t he last invention t hat humanity will ever need to make,» TED, march 2015.

[57] A. Subasi, Practical Machine Learning for Data Analysis Using Python, Academic Press, June 21, 2020.

[58] V. S. ,. G. A. K. Siddhartha Bhattacharyya, Hybrid Computational Intelligence, 5th March 2020.

[59] S. A. Zargar, «Introduction to Sequence Learning Models: RNN, LSTM, GRU,» April 2021.

[60] A. Biswal, «Recurrent Neural Network (RNN) Tutorial for Beginners,» 1 june 2021. [En ligne]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn.

[61] A. See, «Natural Language Processing with Deep Learning CS224N/Ling284, Lecture 7: Vanishing Gradients and Fancy RNNs,» stanfordonline, march 2019.

[62] S. W. B. L. Lei Zhang, «Deep Learning for Sentiment Analysis : A Survey».

[63] K. V. M. B. B. D. &. B. Y. Cho, «On the properties of neural machine translation: Encoder-decoder approaches,» 2014.

[64] J. G. C. C. K. &. B. Y. Chung, «Empirical evaluation of gated recurrent neural networks on sequence modeling,» 2014.

[65] K. Y, «Convolutional neural networks for sentence classification,» chez *the Annual Meeting of the Association for Computational Linguistics*, 2014.

[66] N. K. E. C. N. N. M. C. J. G. Shervin Minaee, «Deep Learning Based Text Classification: A Comprehensive Review,» april 2020.

[67] E. H. H. Y. N. Richard Socher, «Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions,» chez *the 2011 Conference on Empirical Methods in Natural Language Processing*, Edinburgh, UK, 2011.

[68] R. S. B. H. C. D. M. A. Y. Ng, «Semantic Compositionality through Recursive Matrix-Vector Spaces,» chez *the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, July 2012.

[69] A. P. W. C. Richard Socher, «Recursive deep models for semantic compositionality over a sentiment treebank,» january 2013.

[70] B. T. M. H. Y. L. X. Z. X. Z. Qiao Qian, «Learning Tag Embeddings and Tag-specific Composition Functions in Recursive Neural Network,» chez *the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, January 2015.

[71] L. Y. S. C. B. a. W. X. Wang X, «Predicting polarities of tweets by composing word embeddings,» chez *the Annual Meeting of the Association for Computational Linguistics*, 2015.

[72] J. S. AlexGraves, «Framewise phoneme classification with bidirectional LSTM and other neural network architectures,» chez *Neural Networks*, august 2005, pp. 602-610.

[73] Y. L.-C. L. R. a. Z. X. Wang J, «Dimensional sentiment analysis using a regional CNN-LSTM model,» chez *the Annual Meeting of the Association for Computational Linguistics*, 2016.

[74] M. L. B. E. O. Ramón Zatarain Cabada, «Mining of educational opinions with deep learning,» pp. 1604-1626, january 2018.

[75] B. B. Jenny Kim, Data Analytics with Hadoop: An Introduction for Data Scientists, O'Reilly Media, Inc, September 9, 2016.

[76] S. I. K. G. J. Mikolov Tomas, «Distributed Representations of Words and Phrases and their Compositionality,» *Advances in Neural Information Processing Systems,* October 2013.

[77] admin, «How Embedding Layer work in Keras?,» 9 December 2019. [En ligne]. Available: https://androidkt.com/how-embedding-layer-work-in-keras/.

[78] C. Olah, «Neural Networks, Types, and Functional Programming,» 03 september 2015. [En ligne]. Available: https://colah.github.io/posts/2015-09-NN-Types-FP/. [Accès le 2021].

[79] C. Shen, «A Transdisciplinary Review of Deep Learning Research and Its Relevance for Water Resources Scientists,» 30 August 2018.

[80] «Binary crossentropy,» [En ligne]. Available: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/binary-crossentropy.

[81] *IMDB Dataset of 50K Movie Reviews.*

[82] *Amazon Reviews: Unlocked Mobile Phones.*

[83] R. Manorathna, «Linear Regression with Gradient Descent,» june 2020.

[84] K. Krzyk, «Coding Deep Learning for Beginners — Linear Regression (Part 2): Cost Function,» [En ligne]. Available: https://towardsdatascience.com/coding-deep-learning-for-beginners-linear-regression-part-2-cost-function-49545303d29f#:~:text=It%20is%20a%20function%20that,Learning%20model%20for%20given%20data.&text=The%20purpose%20of%20Cost%20Function,as%20small%20numbe.

[85] A. Ng, « Linear regression with one variable – Cost Function,» 2014. [En ligne]. Available: https://www.coursera.org/learn/machine-learning.

[86] Y. R. Fetty Fitriyanti Lubis, «Gradient Descent and Normal Equations on Cost Function Minimization for Online Predictive using Linear Regression with Multiple Variables,» Bandung, Indonesia.

[87] T. R. M. D. Randall Wilson, «The need for small learning rates on large problems,» chez *Neural Networks*, 2001.

[88] J. Brownlee, «Logistic Regression for Machine Learning - Machine Learning Mastery,» 1 april 2016. [En ligne]. Available: https://machinelearningmastery.com/logistic-regression-for-machine-learning/. [Accès le 3 june 2021].

[89] D. J. &. J. H. Martin, «Logistic Regression,» chez *Speech and Language Processing*, December 30, 2020.

[90] A. Pant, «Introduction to Logistic Regression,» 22 january 2019. [En ligne]. Available: https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148. [Accès le june 2021].

[91] TRIANGLES, «The cost function in logistic regression,» 29 OCTOBER 2019. [En ligne]. Available: https://www.internalpointers.com/post/cost-function-logistic-

regression#:~:text=A%20better%20cost%20function%20for%20logistic%20regression&text=In%20words%2C%20a%20function%20C,label%20was%20y(i).. [Accès le june 2021].

[92] S. M. F. L. H. Kumar, Twitter Data Analytics, berlin: Springer, 2014.

[93] B. C. E. Swanson, «Native language detection with tree substitution grammars,» chez *the 50th Annual Meeting of the Association for Computational Linguistics*, 2012.

[94] W. Peter, «The Porter stemming algorithm: then and now,» 2006.

[95] M. Porter, Charting a New Course: Natural Language Processing and Information Retrieval., Sunderland, UK: Springer, 2005.

[96] Y. Kim, «Convolutional Neural Networks for Sentence Classification,» chez *Conference on Empirical Methods in Natural Language Processing*, 2014.

[97] A. P. W. C. Richard Socher, «Recursive deep models for semantic compositionality over a sentiment treebank,» chez *the Conference on Empirical Methods in Natural Language Processing*, 2013.

Annexes

Annexe 1 : Régression Linéaire

Dans la régression linéaire, nous ajoutons les entrées multipliées par certaines constantes pour obtenir la sortie [37]. Il crée une corrélation entre Y, une variable dépendante, et X, qui peut être plusieurs variables indépendantes, en utilisant une ligne droite (Ligne de régression). Les objectifs sont de construire le modèle de régression linéaire simple qui résout le problème, de déterminer dans quelle mesure le modèle fournit la solution et de vérifier les hypothèses du modèle [37]. La forme générale d'un modèle de régression linéaire simple (hypothèse) est la suivante (Figure 53) :

$$y = h_\theta(x) = \theta_0 + \theta_1 x_1 + \epsilon$$

Figure 53 Forme générale de régression linéaire simple

Θ0, Θ1 : paramètres du modèle.
y : variable de réponse ou hΘ (x) : fonction d'hypothèse.
x1: prédicteur.
€ (ou b) : terme d'erreur (résiduel).

La Figure 54 [83] ci-dessous présente un exemple d'application de la technique de régression.

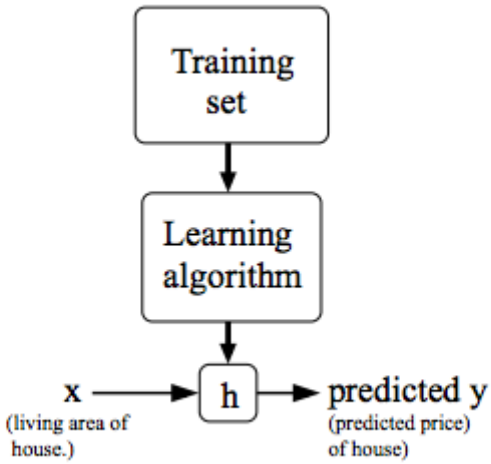

Figure 54 Régression Linéaire

Annexe 2 : La fonction de coût (Régression linéaire)

Il s'agit d'une fonction qui mesure les performances d'un modèle d'apprentissage automatique pour des données. La fonction de coût quantifie l'erreur entre les valeurs prédites et les valeurs attendues et la présente sous la forme d'un seul nombre réel. Selon le problème, la fonction de coût peut être formée de nombreuses manières différentes. Le but de la fonction de coût est d'être soit minimisé, soit maximisé [84].

La fonction d'erreur quadratique est la plus couramment utilisée pour le problème de régression linéaire [85]. La fonction de coût est un choix raisonnable car elle fonctionne bien pour la plupart des problèmes de régression. Il existe deux algorithmes connus de fonction de coût, de descente de gradient et d'équations normales [86].

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- **m:** Number of training examples
- **x(i):** x-value of ith training example
- **y(i):** y-value of ith training example
- **J(θ0 , θ1):** Cost function

Figure 55 Fonction d'erreur quadratique

Où **m** est la quantité de caractéristiques, **h** est la valeur prédictive et **y** est la valeur du monde réel. Pour minimiser la fonction de coût, la différence entre **h** et **y** doit être minimale. Les algorithmes qui minimiseront la fonction de coût sont la descente de gradient et les équations normales [86].

Annexe 3 : Régression linéaire avec descente de gradient

Gradient Descent (Figure 56) est un algorithme d'optimisation. Cet algorithme modifie la valeur **thêta** jusqu'à la convergence pour obtenir la valeur **thêta** avec une fonction de coût minimum [86]. La descente de gradient utilise l'équation suivante, met à jour simultanément la valeur **thêta** jusqu'à la convergence. Cela peut également être appliqué pour des problèmes plus généraux tels que les problèmes de régression linéaire multiple avec 100 ou 1000 prédicteurs [86].

$$\text{repeat until convergence } \{$$
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$
$$(\text{for } j = 1 \text{ and } j = 0)$$
$$\}$$

:= notation: Assignment operator. In programming, it is just = notation.

***α*:** Learning rate. It is a fixed value.

Derivative term: The partial derivative of the cost function J(θ0 , θ1) for j=0 and 1.

After we partially differentiate the cost function J(θ0 , θ1) for j=0 and 1, the algorithm is:

$$\text{repeat until convergence } \{$$
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}$$
$$\}$$

Figure 56 Gradient Descent

Nous suivons les étapes ci-dessous pour implémenter la descente de gradient pour un modèle de régression linéaire simple [86]:

1. Commencez par quelques estimations initiales pour θ0, θ1 (généralement θ0 = 0, θ1 = 0).

2. Choisir une bonne valeur pour α.
3. Mettez à jour simultanément les valeurs de θ0, θ1 pour réduire la fonction de coût J (θ0, θ1) jusqu'à ce que nous nous retrouvions, espérons-le, au minimum.

Annexe 4 : Taux d'apprentissage

Le taux d'apprentissage ou la taille du pas, α, est une variable pour déterminer la taille du pas de la convergence de descente de gradient. Le but de cette partie est de choisir un bon taux d'apprentissage dans la plage de $0{,}001 \leq \alpha \leq 10$. Un bon taux d'apprentissage créera un tracé de la fonction de coût J et du nombre d'itérations d'une courbe qui diminuera et conduira à une ligne droite [86].

Valorisation du taux d'apprentissage :

Un taux d'apprentissage trop élevé se déplace souvent trop loin dans la direction « correcte », ce qui entraîne un dépassement d'une vallée ou un minimum dans la surface d'erreur, ce qui nuit à la précision [87]. En raison de cet effet, un taux d'apprentissage trop élevé prend plus de temps à s'entraîner, car il dépasse continuellement son objectif et « désapprend » ce qu'il a appris, ce qui nécessite un retour en arrière coûteux ou provoque des oscillations improductives [87]. Cette instabilité entraîne souvent également une mauvaise précision de généralisation, car les poids ne peuvent jamais s'installer suffisamment pour se déplacer complètement dans un minimum avant de rebondir à nouveau. Ainsi, si α est trop grand, la descente du gradient peut dépasser le minimum. Dans ce cas, il peut ne pas converger ou même diverger Figure 57 [87]. Si cela se produit lorsque vous exécutez votre code Python, les valeurs NaN sont renvoyées pour θ0, θ1. Dans ce cas, vous devez diminuer la valeur de α et exécuter à nouveau l'algorithme [87].

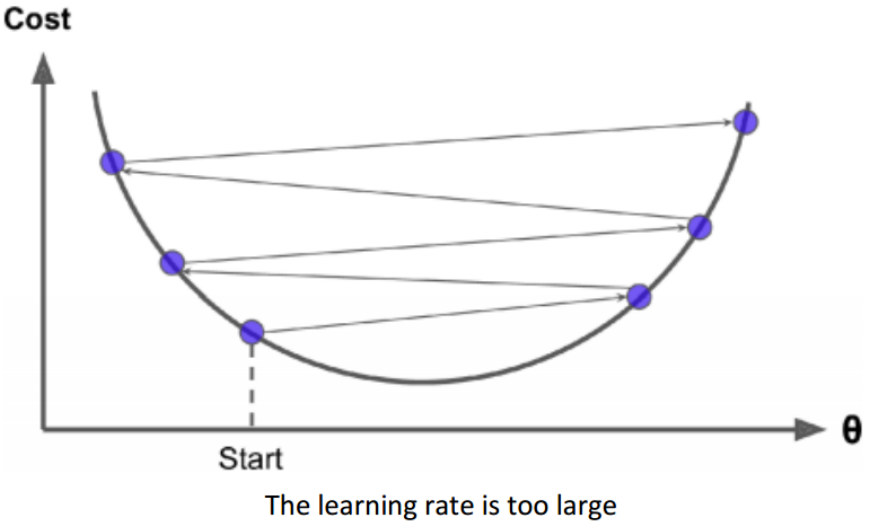

Figure 57 Taux d'apprentissage trop élevé

Une fois que le taux d'apprentissage est suffisamment petit pour éviter de telles sur corrections, il peut procéder de manière relativement fluide à travers le paysage d'erreur, se fixant finalement dans un minimum. Réduire davantage le taux d'apprentissage peut rendre ce chemin plus fluide, ce qui peut améliorer considérablement la précision de la généralisation [87]. Cependant, il arrive un moment où la réduction du taux d'apprentissage devient une perte de temps, entrainant la prise de beaucoup plus de mesures que nécessaire pour emprunter le même chemin au même minimum. Donc, si α est trop petit, la descente du gradient peut être lente. Il faudra plus d'itérations (donc plus de temps) pour atteindre le minimum Figure 58.

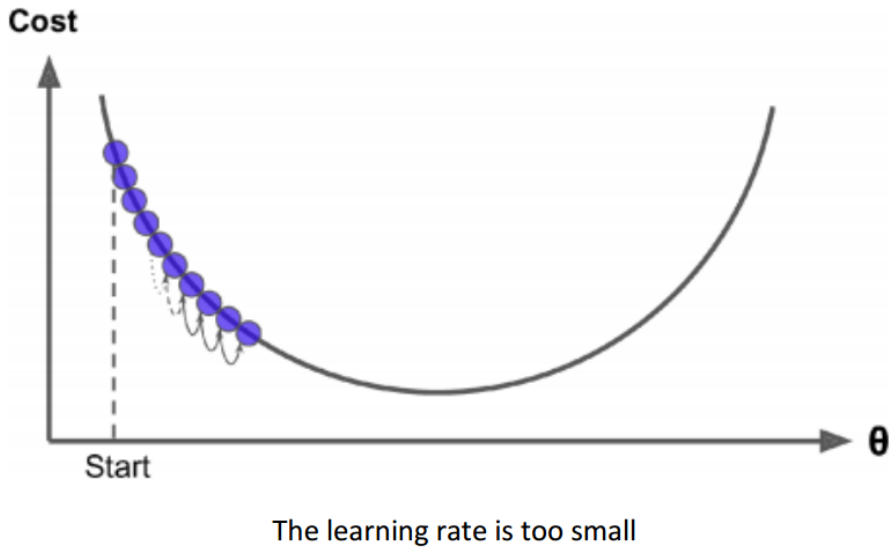

Figure 58 Taux d'apprentissage trop petit

Annexe 5 : Logistic Regression (Régression logistique)

C'est une procédure fiable pour résoudre le problème de classification binaire. La régression logistique est utilisée pour prédire la probabilité d'un résultat qui ne peut avoir que deux valeurs possibles. Le cœur de la régression logistique est la fonction logistique - une courbe en forme de S prenant n'importe quel nombre à valeur réelle et mappant ce nombre a une valeur comprise entre 0 et 1, mais jamais précisément à 0 et 1 [88]. La Figure 59 ci-dessous montre la raideur de la courbe.

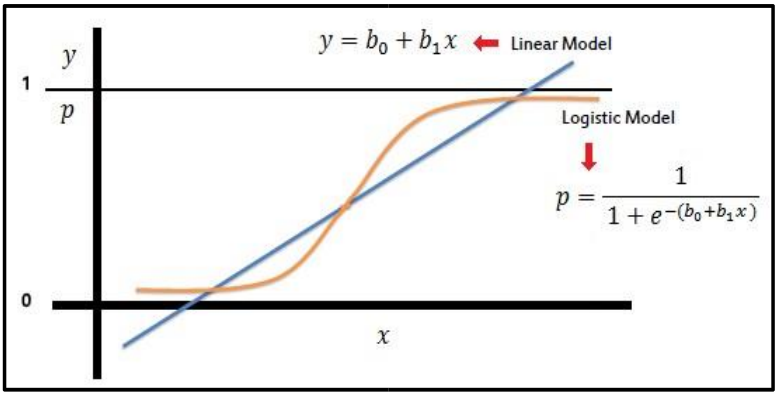

Figure 59 la raideur de la courbe [88]

Annexe 6 : Fonction sigmoïde

L'objectif de la régression logistique binaire est de former un classificateur qui peut prendre une décision binaire sur la classe d'une nouvelle observation d'entrée. Nous présentons ici le classificateur sigmoïde qui nous aide à prendre cette décision.

Considérons une seule observation d'entrée x, que nous représenterons par un vecteur de caractéristiques [x1,x2,...,xn]. La sortie du classificateur y peut être 1 (ce qui signifie que l'observation est membre de la classe) ou 0 (l'observation n'est pas membre de la classe) [88]. Nous voulons connaître la probabilité P (y = 1|x) que cette observation soit membre de la classe. Alors il se peut que la décision est positif « sentiment positif » ou au contraire « sentiment négatif », les caractéristiques représentent le nombre de mots dans un document, P(y = 1|x) est la probabilité que le document ait un sentiment positif et P(y = 0|x) est la probabilité que le document ait sentiment négatif [88] [89].

La régression logistique résout cette tâche en apprenant, à partir d'un ensemble d'apprentissage, un vecteur de poids et un terme de biais. Chaque poids w_i est un nombre réel et est associé à l'une des caractéristiques d'entrée x_i. Le poids w_i représente l'importance de cette caractéristique d'entrée pour la décision de classification et peut être positif (fournissant la preuve que l'instance classée appartient à la classe positive) ou négatif (fournissant la preuve que l'instance classée appartient à la classe négative). Ainsi, nous pourrions nous attendre à ce que, dans une tâche de sentiment, le mot « génial » ait un poids positif élevé et le terme de biais « nul » ait un poids très négatif. Le terme de biais, également appelé l'interception, est un autre nombre réel qui est ajouté aux entrées pondérées. Pour prendre une décision sur une instance de test, après avoir appris les poids lors de l'entraînement, le classificateur multiplie d'abord chaque x_i par son poids w_i,

additionne les caractéristiques pondérées et ajoute le terme de biais ***b***. Le nombre unique résultant ***z*** exprime la somme pondérée des preuves pour la classe [89] [88].

$$Z = \left(\sum_{i=1}^{n} w_i x_i\right) + b \qquad (1)$$

Nous représenterons ces sommes en utilisant la notation du produit scalaire de l'algèbre linéaire. Le produit scalaire de deux vecteurs **a** et **b**, écrit sous la forme ***a·b***, est la somme des produits des éléments correspondants de chaque vecteur [89]. Ainsi, la formule suivante est une formule équivalente à la formule (1).

$$Z = w \cdot x + b \qquad (2)$$

Mais notons que rien dans l'équation (2) n'oblige **Z** à être une probabilité légale, c'est-à-dire à se situer entre 0 et 1. En effet, comme les poids sont à valeur réelle, le résultat pourrait même être négatif ; **Z** va de -∞ à ∞.

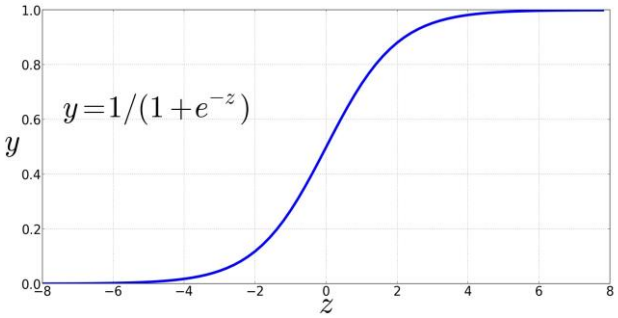

Figure 60 La fonction sigmoïde $y = 1/(1+e^{-z})$ prend une valeur réelle et l'applique à l'intervalle [0,1]. Elle est presque linéaire autour de 0 mais les valeurs aberrantes sont écrasées vers 0 ou 1.

Pour créer une probabilité, nous ferons passer z par la fonction sigmoïde, σ(z). La fonction sigmoïde (ainsi nommée parce qu'elle ressemble à un s) est également appelée fonction logistique et donne son nom à la régression logistique. La fonction sigmoïde a l'équation suivante, représentée graphiquement dans la Figure 60 [89].

$$y = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-z)} \qquad (3)$$

Avantages de la fonction sigmoïde :

La sigmoïde présente un certain nombre d'avantages : elle prend un nombre à valeur réelle et le fait correspondre à l'intervalle [0,1], ce qui est exactement ce que nous voulons pour une probabilité. Comme elle est presque linéaire autour de 0 mais s'aplatit vers les extrémités, elle a tendance à écraser les valeurs aberrantes vers 0 ou 1 [89] [88]. Si nous appliquons la sigmoïde à la somme des caractéristiques

pondérées, nous obtenons un nombre compris entre 0 et 1. Pour en faire une probabilité, nous devons simplement nous assurer que la somme des deux cas, P (y = 1) et P (y = 0), est égale à 1 [89]. Nous pouvons le faire comme suit :

$$P(y = 1) = \sigma(\mathrm{w} \cdot \mathrm{x} + \mathrm{b}) = \frac{1}{1 + \exp\left(-(w \cdot x + b)\right)}$$

$$P(y = 0) = 1 - \sigma(\mathrm{w} \cdot \mathrm{x} + \mathrm{b}) = 1 - \frac{1}{1 + \exp\left(-(w \cdot x + b)\right)} \quad (4)$$

La fonction sigmoïde a la propriété

$$1 - \sigma(x) = \sigma(-\mathrm{x}) \quad (5)$$

Limite de décision :

Nous avons maintenant un algorithme qui, étant donné une instance x, calcule la probabilité **P** (y = 1|x). Comment prendre une décision ? Pour une instance de test x, nous disons oui si la probabilité **P** (y = 1|x) est supérieure à 0,5 et non sinon. Nous appelons 0.5 la limite de décision [89]:

$$\hat{y} = \begin{cases} 1 & si \ \ P(\mathrm{y} = 1|\mathrm{x}) > 0.5 \\ 0 & sinon \end{cases}$$

Annexe 7 : Fonction de coût

La fonction de coût représente l'objectif d'optimisation, c'est-à-dire que nous créons une fonction de coût et la minimisons afin de pouvoir développer un modèle précis avec une erreur minimale [90] [91].

Si nous essayons d'utiliser la fonction de coût de la régression linéaire Figure 55 dans la "régression logistique", elle ne serait d'aucune utilité car elle finirait par être une fonction non convexe avec de nombreux minimums locaux (Figure 61), dans laquelle il serait très difficile de minimiser la valeur du coût et de trouver le minimum global [90].

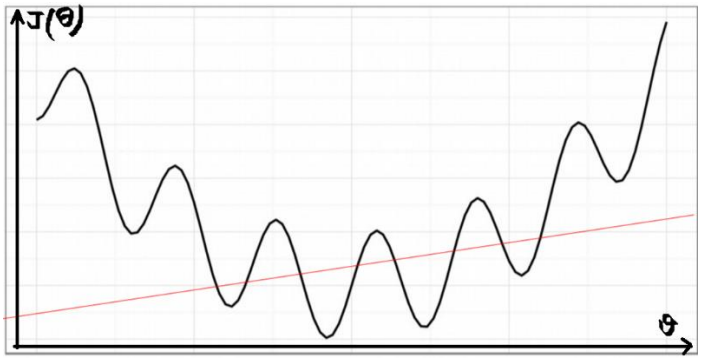

Figure 61 Fonction non convexe

Pour la régression logistique, la fonction de coût est définie comme suit :

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & si \ \ y = 1 \\ -\log(1 - h_\theta(x)) & si \ \ y = 0 \end{cases} \quad (6)$$

Dans le cas où **y**=1, le rendement (c'est-à-dire le coût à payer) se rapproche de 0 lorsque $\mathbf{h_\theta}(x)$ se rapproche de 1. Inversement, le coût à payer croît à l'infini lorsque $\mathbf{h_\theta}(x)$ s'approche de 0. On peut le voir dans la Figure 62. Ci-dessous, en couleur bleue. Il s'agit d'une propriété souhaitable : nous voulons une pénalité plus importante lorsque l'algorithme prédit quelque chose de très éloigné de la valeur réelle. Si le label est **y**=1 mais que l'algorithme prédit $\mathbf{h_\theta}(x)$=0, le résultat est complètement faux [91].

Inversement, la même intuition s'applique lorsque **y**=0, illustrée dans le graphique 2. Ci-dessous, en couleur orange. La pénalité est plus importante lorsque le label est **y**=0 mais que l'algorithme prédit $\mathbf{h_\theta}(x)$=1 [91].

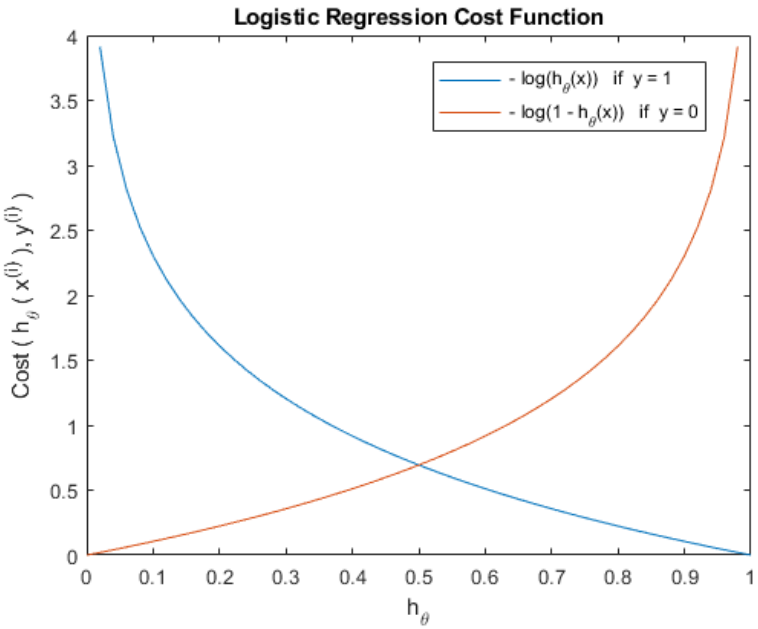

Figure 62 Fonction de cout de la Régression logistique

Ce que nous venons de voir est la version verbeuse de la fonction de coût pour la régression logistique. Nous pouvons la rendre plus compacte en une seule ligne d'expression :

$$Cost(h_\theta(x), y) = -ylog\big(h_\theta(x)\big) - (1-y)log\big(1-h_\theta(x)\big) \qquad (7)$$

Avec l'optimisation en place, la fonction de coût de la régression logistique peut être réécrite comme suit :

$$J(\theta) = -\frac{1}{m}\sum\Big[y^{(i)}\log\Big(h_\theta\big(x(i)\big)\Big) + \big(1-y^{(i)}\big)\log\Big(1-h_\theta\big(x(i)\big)\Big)\Big] \qquad (8)$$

Annexe 8 : Descente de gradient (Régression logistique)

Nous avons la fonction d'hypothèse et la fonction de coût. Il est maintenant temps de trouver les meilleures valeurs pour les paramètres de **θ** dans la fonction de coût, ou en d'autres termes de minimiser la fonction de coût en exécutant l'algorithme de descente de gradient. La procédure est identique à ce qui se fait pour la régression linéaire [91]. Pour minimiser la fonction de coût, nous devons exécuter la fonction de descente du gradient sur chaque paramètre :

$répéter\ jusqu'à\ convergence$

$$\left\{\theta_j \coloneqq \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}\right\} \quad (9)$$