UNIVERSITE SAAD DAHLEB DE BLIDA

Faculté des sciences

Département d'informatique

MEMOIRE DE MASTER

En Informatique

Option : Ingénierie du Logiciel

THÈME :

Détection automatique des sentiments dans les réseaux sociaux

Réalisé par
Debbih Mohamed Islem

Encadré par
Mme. Bacha Siham

Devant membres de jury

Président :	Mr. Bala. M	MCB	Blida 1
Examinatrice :	Mme. Berramdane D	MAA	Blida 1

Septembre 2021

Je remercie Allah...

Je remercie mes parents

Résumé

Ce mémoire porte sur l'analyse des sentiments des commentraies sur les réseaux sociaux en particulier sur la classification supervisée binaire de données issues des pages de marques algériennes sur Facebook. Une des difficultés majeures lors de l'exploration de telles données par des méthodes d'apprentissage supervisées est de posséder un jeu de données suffisant en nombre d'exemples pour l'entraînement des modèles notamment dialecte. En effet, il est généralement nécessaire de catégoriser les données manuellement avant de réaliser l'étape d'apprentissage. La taille importante des jeux de données rend cette tâche de catégorisation très coûteuse. Ce travail présente un outil d'analyse de sentiments des commentraie écrits en dialecte algérien et anglais. Cet outil est fondé sur une approche de deep learning. Les résultats obtenus pour le diaelcte sont encourageants.

Mots clés : apprentissage profond, analyse des sentiments, dialecte algérien, word embedding

Abstract

This thesis focuses on the sentiment analysis of comments on social networks, in particular on the supervised binary classification of data from Algerian brand pages on Facebook.One of the major difficulties when exploring such data by supervised learning methods is to have a sufficient number of data sets for models training, especially dialect. Indeed, it is generally necessary to label the data manually before performing the learning step. The large size of the datasets makes this labellisation task very expensive. This work presents a tool for sentiment analysis of comments written in Algerian dialect and English . This tool is based on a deep learning approach. The results obtained for Algerian dialect are encouraging.

Keywords :
deep learning, sentiment analysis, algerian dialect, word embedding

Table des matières

Table des figures

Liste des tableaux

Introduction

Contexte

Au cours des deux dernières décennies, les plates-formes de commerce électronique américaines et européennes se sont doucement glissées dans la vie quotidienne de beaucoup d'individus à travers le monde. De nos jours, les grands acteurs du e-commerce comme Amazon, eBay et le géant chinois AliExpress proposent des milliers de produits de tous genres aux consommateurs presque partout dans le monde, tout en créant une expérience d'achat personnalisée et engageante pour chaque client en lui permettant de faire ses achats dans le confort de son foyer sans la pression d'un vendeur. Tous ces sites de e-commerce ont changé la façon dont on met les consommateurs en contact avec les magasins de leur produits préférés - cela implique une plus grande accessibilité, un confort accru et une gamme de produits diversifiée parmi lesquels choisir. C'est ainsi que l'achat et la vente en ligne ont pris une partie importante de la vie quotidienne de nombreuses personnes. En 2017, 68% des internautes dans l'Union européenne ont effectué des achats en ligne [1]. Aux États-Unis, 46% des consommateurs préfèrent acheter par internet, selon The Future of Retail Report 2018 de Walker Sands Communications [2], et d'après U.S. Department of Commerce quarterly ecommerce figures, les consommateurs ont dépensé 601,75 milliards de dollars en ligne auprès des marchands américains en 2019, une hausse de 14,9% par rapport aux 523,64 milliards de dollars en 2018 [3].

En 2020, avec la pandémie de Covid-19, les consommateurs se tournaient de plus en plus vers le numérique, vendant et achetant davantage de biens et de services en ligne. Selon les estimations de Digital Commerce 360, les consommateurs américains ont dépensé 861,12 milliards de dollars en 2020, en hausse incroyable de 44% par rapport à 2019 [4]. Il s'agit de la plus forte croissance annuelle et la plus rapide du commerce électronique aux États-Unis depuis des années. Et d'après les statistiques de la société d'études eMarketer, le commerce électronique a fortement progressé dans toutes les régions du monde par au moins deux chiffres en 2020, par exemple, L'Amérique latine a connu une croissance remarquable de 36,8%. Il s'agit d'une

1. https ://ec.europa.eu/eurostat/statistics-explained/index.php ?title=E-commerce_statistics_for_individuals
2. https ://www.walkersands.com/resources/the-future-of-retail-2018/
3. https ://www.digitalcommerce360.com/2020/02/19/us-ecommerce-sales-grow-14-9-in-2019/
4. https ://www.digitalcommerce360.com/article/us-ecommerce-sales/

augmentation par rapport au taux de croissance de 23,2% réalisé en 2019 dans la même région. Le Moyen-Orient et l'Afrique ont connu une croissance moindre, mais également importante (19.8%) [5]*. La plateforme de commerce électronique africaine Jumia a réalisé un bond de 50% de ses transactions au premier semestre 2020 [6].

En Algérie, comme beaucoup de pays en développement, la crise du Covid lui a fait gagner plusieurs années de croissance dans le secteur du commerce électronique. Car en dépit de cette situation sanitaire instable, et au regard du maintien des mesures préventives pour des périodes indéterminées, beaucoup de secteurs et des magasins ont opté pour la vente électronique via les différentes plateformes et applications comme Jumia, Facebook (Market Place, pages, groupes...etc.) et Instagram. Impactés par la fermeture de leurs magasins en raison de la pandémie, les commerçants et même des grandes marques ont dû trouver une alternative pour écouler leur marchandise et assurer un service minimum à leur clientèle habituelle en se tournant vers le commerce électronique.

En absence de réglementation sur le E-commerce il n'existe pas de statistiques officielles sur cette croissance du commerce en ligne en Algérie, mais selon certains chiffres communiqués par l'ancien ministre de la Poste et des Télécommunications, Brahim Boumzar, le paiement en ligne, via la carte Edhahabia, a enregistré près de 4 millions d'opérations en 2020, comparativement à 2019 où il était d'environ 670.000, soit une augmentation de 487% d'opération en une année. De son côté, le DG du GIE Monétique, Madjid Messaoudène, a annoncé une hausse de près de 400% des opérations en ligne a été enregistrée depuis le début de la pandémie du Coronavirus. Et selon le rapport annuel de la Conférence des Nations unies sur le commerce et le développement (CNUCED), l'Algérie a enregistré une avancée notable, en passant de la 109e place à la 80e place au niveau mondial, figurant ainsi parmi les 4 pays ayant réalisé la plus grande progression au niveau mondial, aux côtés du Brésil, Ghana, et la République du Laos [7].

Cette croissance du E-commerce à travers les différentes plateformes coïncide avec la prolifération du web 2.0 et la révolution des platefromes sociales et communautaires telles que les forums de discussion, les blogs et les réseaux sociaux. Ces plateformes représentent des espaces interactifs permettent aux utilisateurs de partager leurs idées, leurs opinions, et d'exprimer leurs points de vue et leurs sentiments sur un sujet, un produit, une personne, etc.
Pendant les dernière années, avec l'apparition des médias sociaux tels que Facebook, Twitter et Instagram, l'internet a connu encore un plus grand élan. Basés sur des techniques de communication faciles et accessibles pour tous, ces médias sociaux représente une phase transitoire dans le cycle de vie de l'internet. Aoujourd'hui, chaque information peut être diffusée sur l'ensemble des médias sociaux. Ces médias, dans lesquels les internautes peuvent s'inscrire, créer un réseau virtuel en ajoutant des contacts et échanger avec eux des messages et des opinions en temps

5. https ://www.emarketer.com/content/global-ecommerce-update-2021

6. https ://unctad.org/fr/news/suite-la-covid-19-le-numerique-et-le-commerce-electronique-arrivent-un-tournant-de-leur

7. https ://unctad.org/system/files/official-document/tn_unctad_ict4d17_en.pdf

réel, font désormais partie intégrante du quotidien des gens. Grâce à leur facilité d'utilisation et leur libre accès, ils connaissent un succès grandissant jour après jour et gagnent la confiance du public. Ils rassemblent des individus, des entreprises et des organisations avec la possibilité de les utilisés à diverse fins, notamment la publicité, la diffusion d'opinions politiques, l'obtention des commentaires des utilisateurs sur les produits et la diffusion d'informations.
L'ecommerce a bénéficier d'un considérable essor grace aux réseaux sociaux tels que Facebook, Pintrest et Instagram. L'usage de ces platefromes par les professionnels et les internautes a changé : La communication autours des marques et des produits se fait désormais sur ces réseaux sociaux qui représentent aujourd'huit le refet de l'évolution de l'opinion public sur internet. Ces discussions peuvent être une source de notoriété inopinée pour les entreprises ou faire perdre tout confiance et décrédibiliser leur réputation.

Facebook et Instagram qui sont aujourd'hui utilisé pour vendre toutes sortes de produits est accompagnée d'une forte augmentation de données sous forme d'avis, commentaires, recommandations, notes et retours générées par des consommateurs. Ces avis et commentaires laissés par les clients qui peuvent concerner pratiquement tout, y compris la qualité du produit, le prix, la qualité du service...etc ont de l'importance, qu'ils soient issus des réseaux sociaux, des forums, des sites d'avis en ligne ou des blogs, car ils ont désormais de la valeur aux yeux des clients potentiels (prospects). Selon Trustpilot, près de 89% des consommateurs dans le monde entier font l'effort de lire des avis des internautes avant d'acheter des produits en ligne [10]. C'est pourquoi il est devenu essentiel pour toute entreprise souhaitant se développer sur Internet d'être capable de contrôler, analyser et mesurer ces opinions et avis en ligne. C'est cette nécessité qui a donné naissance au métier de Community manager.

Problématique et motivation

Aujourd'hui, les marques et les commerçants ne sont pas étrangères aux réseaux sociaux. En Algérie comme partout dans le monde, les entreprises commencent doucement à se rendre compte de la valeur de se connecter véritablement avec les consommateurs et de traiter les réseaux sociaux comme plus qu'un canal de promotion. Mais gérer les comptes de médias sociaux d'une marque nécessite pas mal de travail et d'investissement. Investir dans les relations avec les consommateurs n'a jamais été aussi essentiel à la croissance des entreprises de commerce électronique. Et devant telle évolution, pour développer une e-notoriété de marque, améliorer la visibilité d'un ou plusieurs produits ou services, créer un lien avec les clients et prospects, un Community manager est désormais indispensable au sein des entreprises pour développer leur communication digitale et leurs ventes.

Le Community manager peut se voir assigner plusieurs missions en fonction de la taille et du secteur d'activité et du statut sous lequel il travaille. En temps normal, sa mission consiste à interagir avec la communauté et stimuler les discussions au nom de l'entreprise sur les différents réseaux sociaux comme Facebook, Twitter, Instagram, etc. Il est chargé de l'animer et

d'interagir avec elle pour la fidéliser sur le long terme. Pour cette raison, une bonne analyse des retours, commentaires et des discussions des utilisateurs sur les produits est indissociables pour suivre les performances de l'activité d'une entreprise sur les réseaux sociaux. Ces données sont partagées avec les dirigeants et les décideurs comme un outil d'aide à la décision pour qu'ils adaptent leurs stratégies.

Cependant, l'analyse de tous ces avis et commentaires sur les différents réseaux sociaux manuellement demande énormément de temps. Parcourir et lire chaque avis ou commentaire sur chaque publication un par un sur plusieurs plateformes par le Community manager est quasiment impossible. Dans ce cas, un outil d'analyse des sentiments qui traite les avis et interprète d'une manière automatique leurs polarités s'ils sont positifs ou négatifs peut faire gagner un temps précieux au Community manager qui lui permet de concentrer sur des tâches plus importantes, la création de contenu par exemple. C'est ce qui nous intéresse dans ce travail.

Objectifs et défis

L'objectif de notre travail est de réaliser un système d'analyse de sentiments multilingual en exploitant différentes techniques d'apprentissage profond dans le but de détecter la polarité (distinguer le positif du négatif) des avis et commentaires liés aux publications des pages de marques algériennes sur Facebook. Cela mis notre étude sur le terrain du domaine du marketing digital dans le but d'aider les entreprises à mieux comprendre ce que leurs clients pensent d'eux et de leurs produits et services. Cela leur permet de bâtir une relation positive, forte, et durable avec leurs consommateurs et mieux répondre à leurs attentes avec des produits bien conçu.

Pour atteindre cet objectif, nous sommes confrontés à un ensemble de défis et de problématiques, car la plupart des ressources et des systèmes qui existent sont destinés à l'Anglais. Cependant, dans un contexte multilingue comme le nôtre, la présence du dialecte algérien représente un grand défi. Le défi consiste à gérer sa complexité morphologique et développer les ressources nécessaires en réalisant notre propre corpus.

Contribution

Comme mentionné précédemment, dans ce travail nous nous intéressons spécifiquement à l'analyse des sentiments des avis des consommateurs algériens sur Facebook, pour cela, dans un premier temps et pour répondre au manque de ressources, nous avons commencé par construire un corpus spécifique au domaine pour le Dialecte algérien que nous appelons DzReveiws qui permette de répondre à notre problématique. Dans un deuxième temps, nous exploitons différentes architectures d'apprentissage profond avec différentes méthodes et techniques de word embedding pour l'analyse des sentiments. Nous pouvons résumer notre contribution dans le cadre de ce travail comme suit :

— Concevoir un corpus d'avis consommateurs en dialecte algérien spécifique à la tâche d'analyse des sentiments ;

— Exploiter les différentes techniques de word embedding pour la représentation des mots telque Word2vec, GloVe, Embedding layer et FastText adapté à notre cas de dialec ;

— Proposer des méthodes appliquées pour la première fois sur un corpus de dialecte algérien ;

— Proposer un système d'analyse de sentiments adatpté au contexte algérien et traitant plusieurs langues (français, anglais et dialecte algérien).

— Développer une approche générique qui permet de traiter des données diversifiées liées aux produits commerciales à promouvoir.

Organisme d'accueil

Ce travail a été ralisé dans le cadre d'un stage effectué au sein de l'agence de communication Innocom. Il s'agit d'une agence de conseil en communication 360° crée en 2012, spécialisée dans le marketing digital.

L'agence apporte des solutions et stratégies à tous les type d'entreprises ou institutions. Leur savoir-faire réside dans l'accompagnement de leurs clients jusqu'a la réalisation leurs projets, sachant que chaque projet est unique et que les attentes sont différentes, leurs réponses sont personnalisées avec un champ d'action diversifier et une équipe qualifier et polyvalente.

L'agence croit que le discours commercial classique seule ne suffis pas et trouve de moins en moins d'échos chez des consommateurs enclins avant tout à recevoir de l'information rapidement et facilement. Donc, se déployer sur le web via le site web, les réseaux sociaux et les applications mobile est impérative dans notre temps modern. Ces derniers permettent d'augmenter la visibilité et de favoriser un sentiment d'adhésion aux marques. C'est dans ce but que l'agence propose des solutions web et e-marketing à ses clients.

Chapitre 1

Traitement automatique du langage naturel

1.1 Introduction

Le traitement automatique du langage naturel ou *Natural Langage Processing* en anglais (NLP), est un domaine interdisciplinaire représente une branche importante de l'intelligence artificielle. Il s'intéresse essentiellement à la conception des méthodes et d'outils informatiques qui permettent à la machine de comprendre, manipuler, générer et analyser le langage naturel parlé par les humains. Cela couvre un grand nombre de tâches et d'applications, comme la traduction automatique, résumé automatique de texte, l'étiquetage morpho-syntaxique, l'extraction d'informations à partir de documents ou encore « comprendre » ce que veut dire un texte. Dans ce dernier cas, nous nous trouvons généralement avec des tâches comme la fouille de textes, et l'analyse des sentiments. Durant ces dernières décennies, et depuis le début des travaux dans ce domaine, diverses directions de recherches ont été poursuivies. En effet, le NLP a fait des progrès considérables au cours des dernières années. De ce fait, beaucoup de techniques, de méthodes et d'outils ont été développés et utilisés dans ce sens. L'ensemble de ces techniques est souvent appelé *pipeline*. Il s'agit généralement de plusieurs étapes de traitement (Figure 1.1). Dans ce chapitre, nous présentons les techniques les plus importantes du traitement

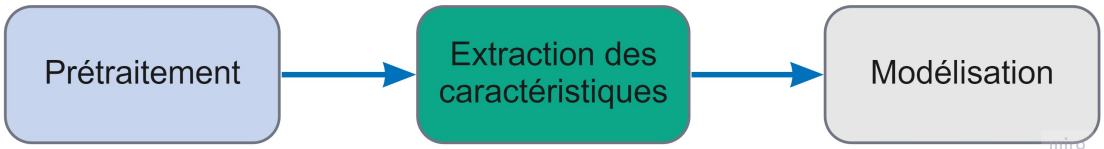

FIGURE 1.1 – Le pipeline de modélisation NLP [35]

automatique du langage naturel et les plus utilisées quand il s'agit de traiter des énoncés textuels. Ces techniques sont souvent appliquées sur l'une des trois phases qui forment le pipeline : la

phase de prétraitement (*Text preprocessing*, en anglais), la phase d'extraction de caractéristiques (*Feature Extraction*, en anglais), et la phase modélisation (*Modeling*, en anglais).

1.2 Prétraitement

La tâche de prétraitement des données textuelles désigne le processus global qui vise à nettoyer et normaliser ces données qui viennent souvent dans un état brut et non structuré. Par exemple, si les données sont des tweets qui ont été collectées directement en utilisant l'API Twitter, alors généralement, ils peuvent contenir beaucoup de bruit : des adresse HTML, des hashtags, des noms d'utilisateurs (précédés par @) et des mots qui n'ont pas d'impact sur l'orientation générale de ceux-ci. Par conséquent, ces textes bruts se prête rarement à l'analyse directement. Ceci est dû au format de ces derniers qui peut influencer de manière non négligeable les résultats. C'est pourquoi il est indispensable de transformer ces données brutes en données exploitables. Dans ce but, il existe différentes techniques qui peuvent être utilisées dépendant du matériau textuel et le problème en question. La Figure 1.2 illustre l'ensemble de ces technique. Des techniques et des opérations comme, la suppression des ponctuations, l'élimination des nombres et des dates, l'élimination des mots vides et/ou les mots qui ne fournissent aucune information ou valeur sur le problème étudié, la lemmatisation, la racinisation, la tokenisation, etc [81].

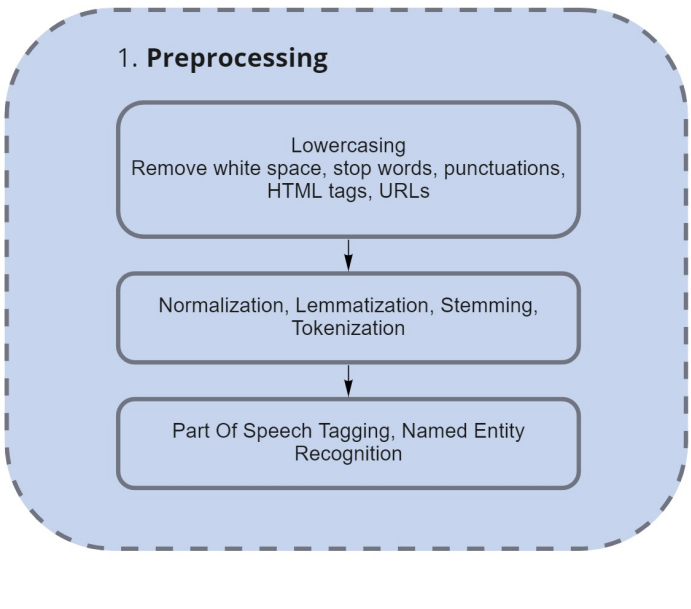

FIGURE 1.2 – Les étapes de la phase de prétraitement d'une donnée textuelle

1.2.1 Casse

Cette opération consiste à modifier la mise en majuscules/minuscules, ou casse, d'un texte. Il s'agit de mettre tout l'énoncé textuel à la même casse, souvent tout en minuscule. Cela rend tout le texte sur un pied d'égalité, ce qui permet d'éliminer la confusion des majuscules. Par exemple, le mot « Produit » et le même que « produit », donc une fois que nous appliquons la conversion en minuscule le mot résultant est le même. Cela permet notamment de réduire la taille du vocabulaire dans les approches de la deuxième étape du pipeline, ce qui contribue considérablement à la cohérence des sorties attendue par le modèle.

1.2.2 Suppression des balises HTML

Les données textuelles peuvent être sous différents formats, des tweets, des posts de blog, des statuts Facebook, des commentaires dans des forums, des réponses à des questionnaires, un catalogue de produits sur un site e-commerce (avec des attributs comme la marque, la référence, la catégorie, la description...etc.), extraits de sites web ou d'articles scientifiques. Collecter ces données directement sur des sites en utilisant les différentes techniques d'extraction (web scraping) comme les API par exemple pour construire un corpus nécessite souvent une étape de nettoyage. Etant donné que ces sites web contiennent de nombreux éléments qui ne contiennent pas d'informations utiles en matière de l'NLP. Des éléments tel que les balise HTML et les code du style et de Javascript qui forment les squelettes des pages web.

1.2.3 Suppression des mots vides

Cette opération consiste à supprimer ce qu'on appelle en anglais les *stopwords* ou les mots vides en français. Il s'agit d'un ensemble de mots « non pertinents » qui apparaissent très fréquemment dans les textes de la langue étudiée mais qui n'apportent que peu ou pas de valeur à la signification et la compréhension du texte. En français, des mots vides évidents pourraient être « le », « la », « de », « du », « ce », etc. Toutefois, il vaut mieux ne pas supprimer les mots vides car ils sont une partie cruciale de la solution de certains problèmes, ce qui peut impacter négativement les résultats du modèle final.

1.2.4 Racinisation

La racinisation *Stemming* en anglais, consiste à ramener un mot à sa frome canonique (dite aussi racine) ou *stem* en anglais. Ces stemmes généralement sont des mots que nous pouvons utiliser pour construire de nouveaux mots en leur ajoutant des affixes. Cependant, le stemming désigne le processus qui vise à regrouper les nombreuses variantes d'un mot ayant la même racine comme un seul et même mot. Cette racine correspond à la partie du mot restante une fois la suppression des préfixe et suffixe est faite. Par exemple, le stemme de « terrain » est « terr » et regroupe également : Terrasse, territoire, territorial. En utilisant cette opération, il faut prendre en

considération le fait qu'elle ne se soucie pas des règles et contraintes linguistiques. En effet, elle peut extraire des racines de mots inconnus qui n'existent pas dans le vocabulaire linguistique. Elle va cependant générer pas mal d'erreurs en regroupant des mots qui ne devraient pas l'être (par exemple, des mots de différentes significations).

1.2.5 Lemmatisation

La lemmatisation est très similaire à la racinisation. Mais contrairement à cette dernière, elle consiste à supprimer les affixes de mots afin de retrouver pour chaque flexion le terme primaire que l'on désigne sous le terme « forme canonique » ou *lemme* en se basant sur une analyse des règles grammaticales, de ce fait, le lemme correspond toujours à un terme issu de l'usage ordinaire des locuteurs de la langue (présent dans les dictionnaires de la langue). Le but est de réduire les variantes morphologiques à une forme commune. Par exemple, les deux mots « inclinaison » et « inclination » deviennent le mot « incliner ».

1.2.6 Tokenisation

La tokenisation ou segmentation en français, est l'opération de découpage du texte en plusieurs petites unités qui peuvent être des phrases ou des mots appelés *tokens* (Figure 1.3). Identifier ces tokens semble être une tâche relativement simple en utilisant la façon la plus courante qui est basée sur l'espace comme premier séparateur (dite aussi délimiteur). Ce "blanc" dans le texte est le point de départ de toute tokenisation. Pourtant, ce n'est pas toujours aussi simple, car chaque langue a ses propres constructions grammaticales - le japonais [78] et le chinois [83], par exemple, utilisent l'espace blanc et les délimiteurs pas de la même façon que l'anglais. En outre, il peut y avoir des cas où d'autres caractères que des espaces blancs peuvent être utilisés pour la segmentation. D'où la nécessité parfois d'un bon tokenizer avec des règles précises et une liste de délimiteurs personnalisés qui va bien au-delà de la simple segmentation par des espaces.

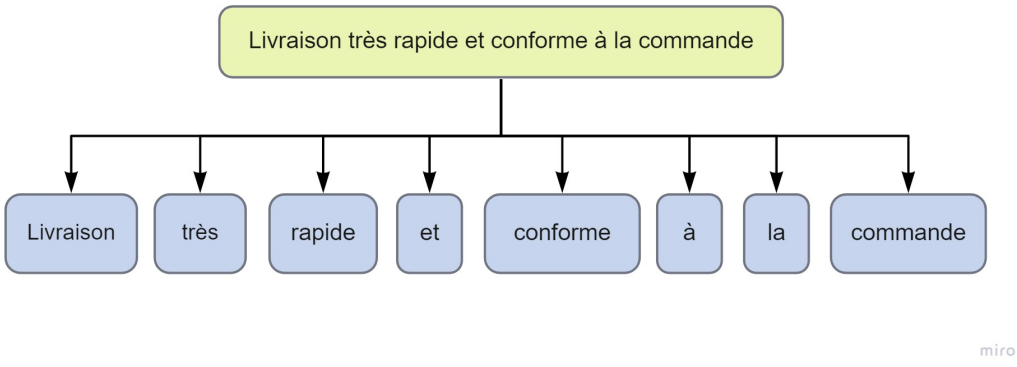

FIGURE 1.3 – Exemple illustre l'opération de tokenisation [20]

1.2.7 Étiquetage morpho-syntaxique

L'analyse morphosyntaxique ou l'étiquetage morpho-syntaxique *Part-of-Speech tagging* en anglais (POS tagging) est l'un des mécanismes fondamentaux de l'analyse du langage [10]. Elle constitue une étape essentielle et importante pour résoudre pas mal de problèmes du traitement automatique de la langue. Le but est d'identifier la catégorie grammaticale à laquelle appartiennent les mots d'un énoncé textuel donné. Il s'agit d'associer chacun de ces mots avec une étiquette mentionnant sa fonctionnalité grammaticale notamment nom, verbe, adverbe, adjectif, etc (Figure 1.4). Ce processus est basé généralement sur la définition du mot et l'hypothèse que la catégorie d'un mot dépend de son contexte local.

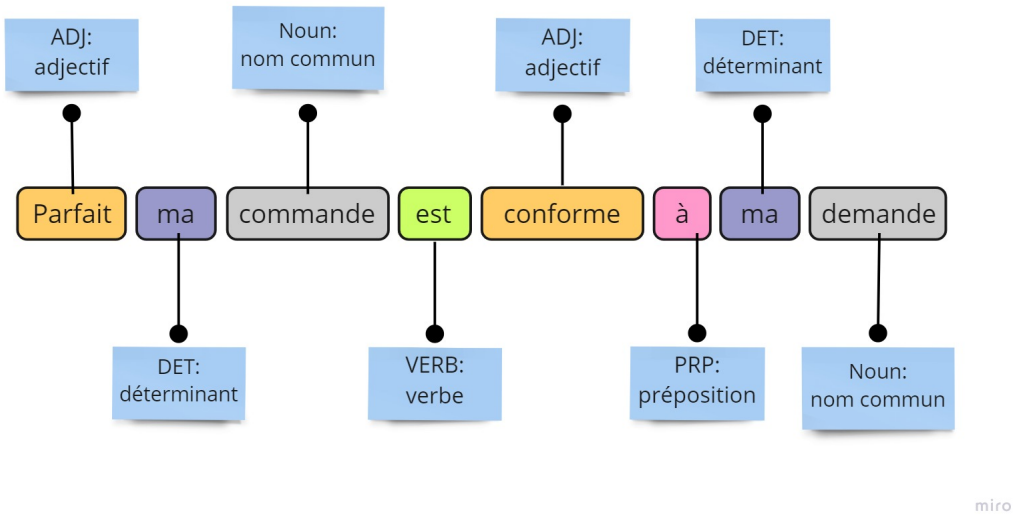

FIGURE 1.4 – Exemple d'analyse morphosyntaxique d'une phrase

1.2.8 Reconnaissance d'entités nommées

Ou *Named entity recognition* en anglais (NER), est une tâche importante dans de nombreuses problèmes de traitement automatique du langage quand il s'agit de l'extraction de l'information dans des énoncés textuels. Le terme « entité nommée » (EN) est apparu pour la première fois en 1996 lors de la sixième conférence MUC (Message Understanding Conference) [26] afin de faire référence aux noms de personnes, d'organisations, de lieux, des dates, des valeurs monétaires, etc., présentes dans un texte donné. Le REN consiste à reconnaitre et identifier ces entités dans le texte d'une façon automatique et leur attribuer une étiquette telle que : « personne », « organisation », « date », « lieu », etc. (Figure 1.5).

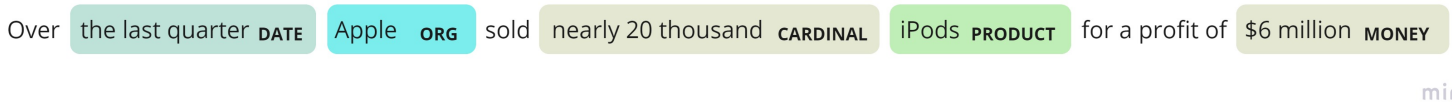

FIGURE 1.5 – Exemple de la econnaissance d'entités nommées dans une phrase [1]

1.3 Extraction de caractéristiques

L'extraction des caractéristiques ou *Features extraction* en anglais, représente généralement la deuxième phase dans le pipeline. Il s'agit d'une étape très importante dans le processus du TAL. Elle consiste à extraire l'information du texte et la transformer sous une forme de caractéristiques numérique sous forme de vecteurs exploitables par la machine afin de les utiliser dans la prochaine étape qui est la modélisation. Ce processus est appelé aussi vectorisation ou représentation de texte. Dans cette section, nous présentons quelques différentes méthodes de représentation qui existent, comme l'encodage one-hot (*One hot encoding*, en anglais), la représentation en sac de mots (*Bag of Word*, en anglais), le Tf-idf et le word embedding ou plongement de mots en français [32].

1.3.1 L'encodage one-hot

L'encodage one-hot ou encodage à chaud (One hot encoding, en anglais), est l'une des méthodes de représentation vectorielle des données textuelles les plus classiques et les plus simples. Elle consiste à représenter chaque document ou phrase par une matrice 2D de taille n x m , où n étant le nombre de mots (tokens) dans le document ou la phrase, et m la taille du vocabulaire. Chaque mot du vocabulaire est représenté par un vecteur binaire avec toutes ses valeurs nulles à l'exception de l'index du mot où la valeur peut être soit 1 pour indiquer la présence du mot dans le texte ou bien 0 pour indiquer son absence (Figure 1.6).

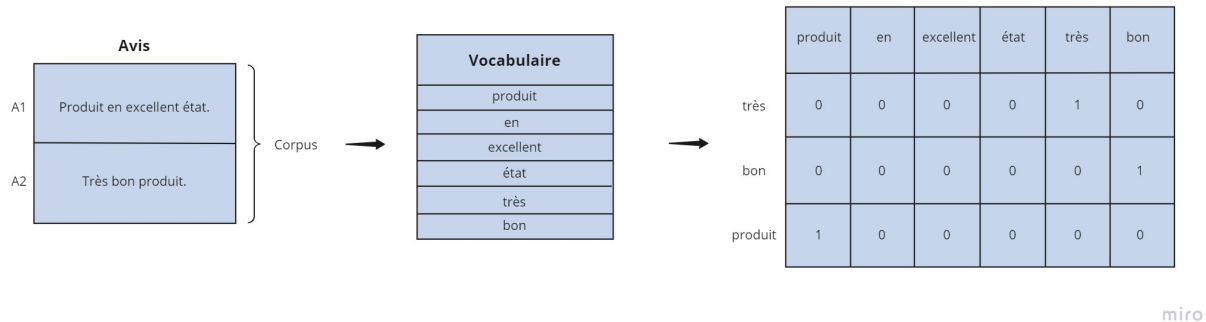

FIGURE 1.6 – Exemple d'un encodage one-hot

1.3.2 Sac de mots

Le sac de mots (*Bag of words* - BoW, en anglais), est une autre méthode classique pour représenter un texte sous un format vectorielle [22]. Cette méthode consiste à compter le nombre de fois qu'un mot donné est apparu dans le texte (tel qu'une phrase ou un document) sans tenir compte de l'ordre d'apparition ni du rôle grammatical, mais en conservant la multiplicité (la fréquence de chaque occurrence) (Figure 1.7). D'ailleurs c'est de ce fait qu'elle tire son nom « sac » de mots, car il s'agit d'une représentation où toute information sur l'ordre ou l'occurrence contextuelle des mots dans le texte est ignorée.

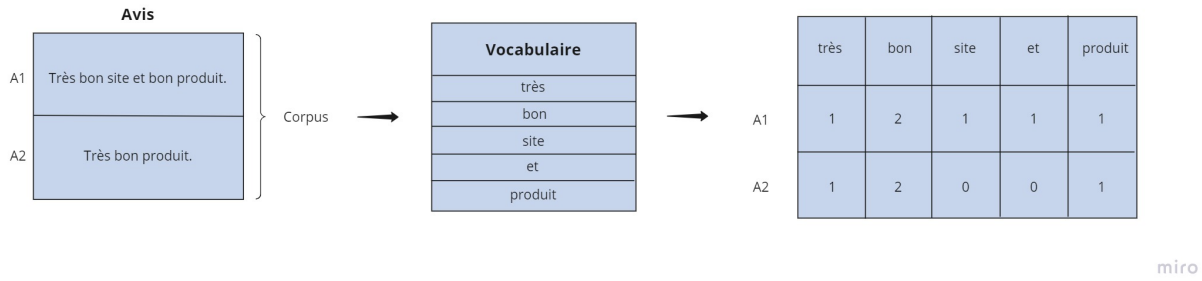

FIGURE 1.7 – Exemple de la représentaion Sac de mots

1.3.3 TF-idf

Term Frequency-Inverse Document Frequency (TF-IDF), est une méthode qui combine deux mesures de calcul de pondération de fréquence de mot : Le « Terme Frequency » et « Inverse Document Frequency ». Cette méthode statistique permet de mesurer l'importance d'un mot en fonction de sa fréquence dans le document qui le contient (TF) [1] pondérée par sa fréquence d'apparition dans tout le corpus ou la collection (IDF). Autrement dit, l'importance d'un terme dans un document ne dépend pas de sa fréquence dans un document particulier seulement, mais aussi en fonction de sa distribution et de son utilisation dans l'ensemble des documents. L'idée derrière cette technique est la suivante : un mot qui apparaît fréquemment dans un document spécifique, est probablement plus important que les mots qui apparaissent dans presque tous les documents. Ces derniers ne sont probablement pas discriminatoires. Ainsi, un mot apparaissant dans un grand nombre de documents dans le corpus ou la collection aura un poids faible, en revanche, un poids plus important est donné à un mot discriminant. La mesure TF-IDF est calculée à l'aide de la formule décrite ci-dessous :

$$Tf.idf(t_k, Dj) = TF(t_k, Dj)IDF(t_k) \tag{1.1}$$

1. http ://orion.lcg.ufrj.br/Dr.Dobbs/books/book5/chap14.htm

où $TF(t_k, D_j)$ est le poids ou la fréquence du terme t_k dans le document D_j, et $IDF(t_k)$ est la fréquence du document inverse (Inverse Document Frequency) du même terme. Cette dernière est calculée par la formule suivante :

$$IDF(t_k) = \log \frac{|N|}{DF(t_k)} \tag{1.2}$$

où :

— N est le nombre total de document dans le corpus

— $DF(t_k)$ est le nombre document contenant le term t_k

Afin de bien comprendre le fonctionnement de TF-IDF, nous allons voir un petit exemple concret. Supposons que nous avons un corpus contient trois avis (documents) suivant :

Avis 1 :

Produits au top, envois rapide.

Avis 2 :

Livraison rapide et produit de qualité.

Avis 3 :

Top. Produits de qualité. Livraison rapide et soignée. Produits bien emballés.

La première étape consiste à supprimer les mots qui portent peu d'information à notre exemple d'analyse. Le résultat de cette étape est dans la Figure 1.8.

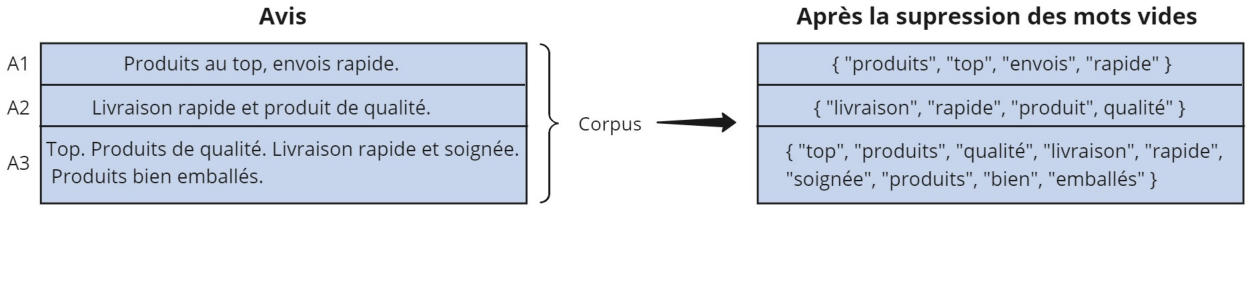

FIGURE 1.8 – La suppression des mots vides

Maintenant nous allons calculer le TF. Pour cela, nous allons d'abord créer une matrice term-document (mot-document), ensuite nous allons utiliser la normalisation en utilisant la formule de calcul ci-dessous. La Figure 1.9 illustre les résultats de cette étape.

$$TF(t) = \frac{f_{t_i}}{n_i} \tag{1.3}$$

où :

— f_{t_i} est la fréquence du mot t dans le document i

— n est le nombre total de termes dans le document i

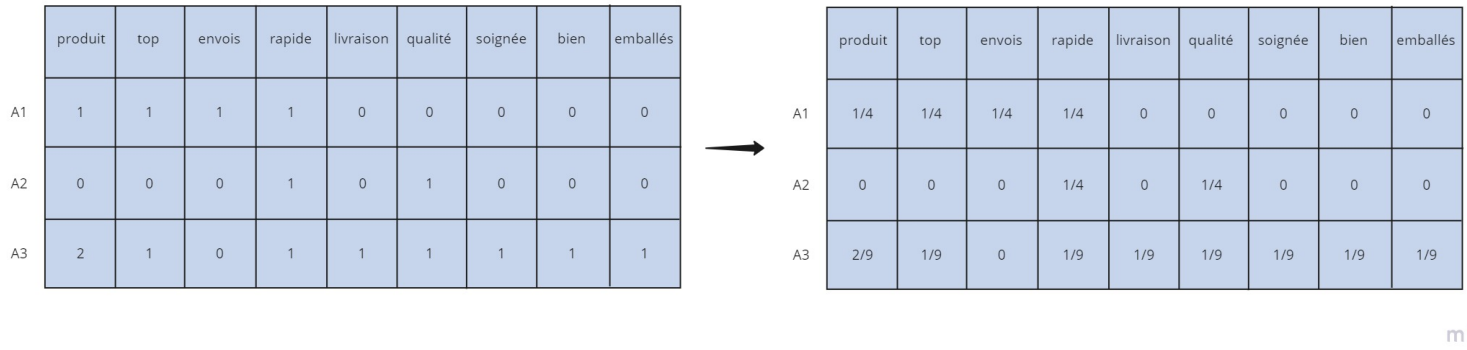

FIGURE 1.9 – Calcul de TF

Dans cette troisième étape, nous allons calculer l'IDF. Mais avant, nous devons trouver le nombre de documents dans lesquels chaque mot apparaît ($DF(t)$) (Figure 1.10) :

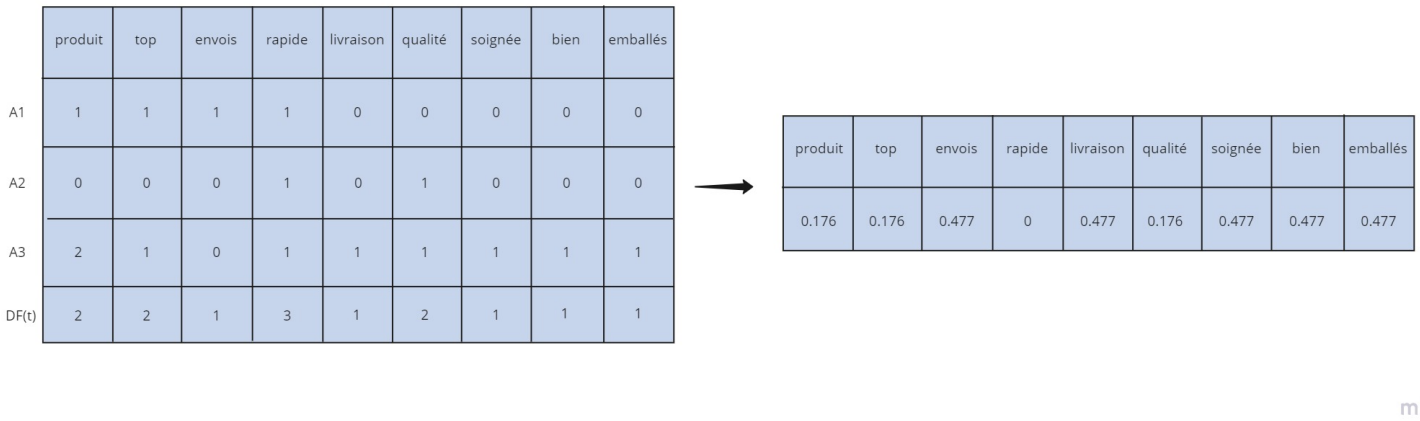

FIGURE 1.10 – Calcul de IDF

Enfin, nous calculons le Tf-idf en multipliant le Tf et le Idf comme le montre la Figure 1.11 :

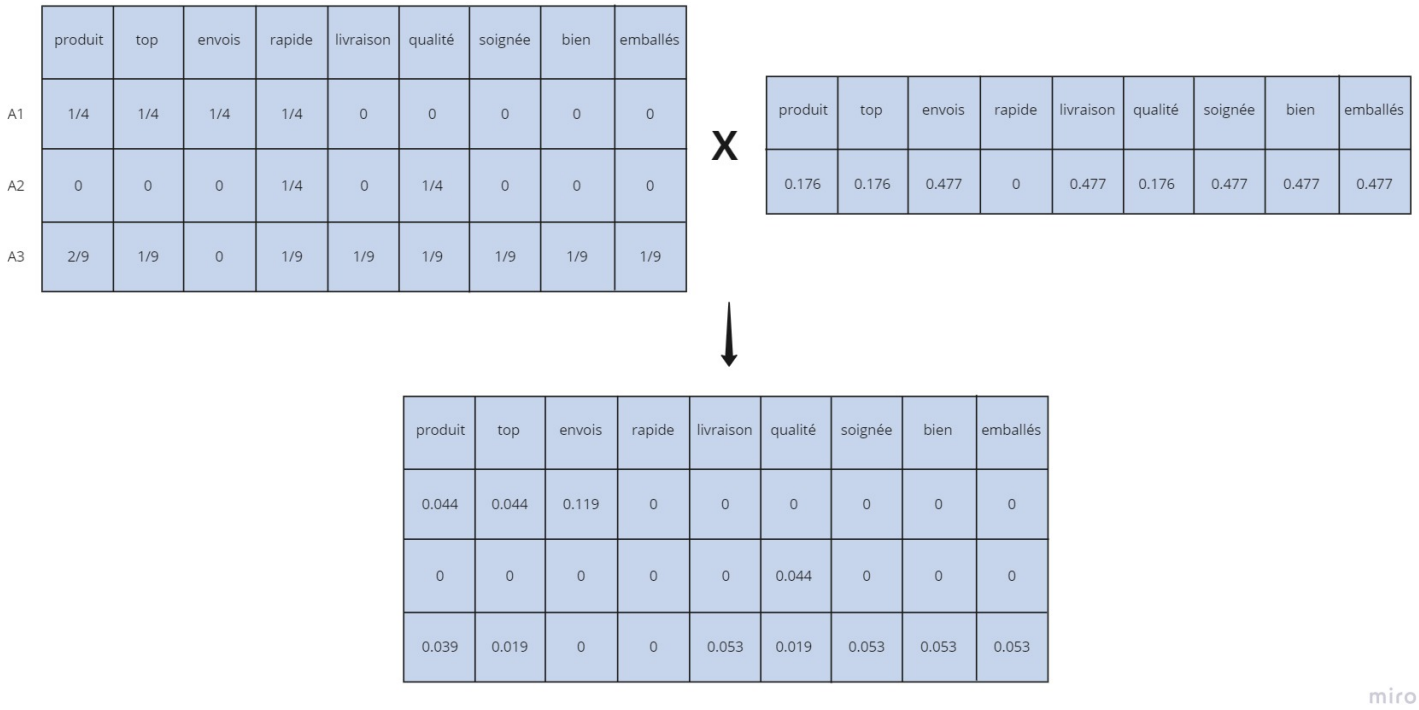

FIGURE 1.11 – Calcul de Tf-idf

1.3.4 Word Embedding

Le *word embedding* ou plongement de mots en français, représente un ensemble de méthodes d'apprentissage automatique pour la représentation des mots par des vecteurs de nombre réels, décrits dans un modèle vectoriel (*Vector Space Model*, en anglais) (VSM). Il s'agit de plongements de mots du vocabulaire sous forme de vecteurs denses à valeur réelles et de dimension faible. Ils constituent une projection des mots dans un espace vectoriel de faible dimension de façon à préserver le contexte, la similarité sémantiques et syntaxiqus. Cette technique repose sur l'hypothèse de Harris et connue sous le nom de *Distributional Semantics* [27]. Cette hypothèse de distribution indique qu'un mot est caractérisé par son contexte, c'est à dire par les mots qui l'entourent. Donc, les mots qui apparaissent dans des contextes similaires ont également des significations semblables. Par conséquent, les mots proches sémantiquement ou syntaxiquement possèdent des vecteurs correspondants qui sont relativement proches dans l'espace d'embeddings. En se basant sur la sémantique distributionnelle, plusieurs méthodes ont été proposées pour construire des représentations vectorielles de mots capturant des liens entre les mots via une prise en compte des contextes de mots comme Word2Vec [48], Glove [60] et FastText [9]. Ces techniques de représentation de données textuelles ont permis d'améliorer les performances des méthodes de traitement automatique des langues d'une façon importante [40]. Nous allons voir ces méthodes en détails dans le prochain chapitre.

1.4 Modélisation

Après les deux premières étapes du pipeline, le prétraitement et l'extraction des caractéristiques, où les données textuelles inexploitables passent à des données textuelles préparées et structurées, vient la prochaine et la dernière étape : La modélisation. Malgré le fait que les données doivent toujours être prétraitées, et l'importance de comprendre la syntaxe, la sémantique et la structure du langage, cela n'est pas suffisant à lui seul pour être en mesure de découvrir des motifs (patterns en anglais) et des insights dans des ensembles de données textuelles volumineux. Pour cette raison, dans cette étape, il s'agit de construire un modèle de *machine learning* (apprentissage automatique en français) ou de *deep learning* (apprentissage profond ne anglais) qui peut exploiter ces données afin d'effectuer des tâches spécifique selon le besoin et la nature du problème traité. Pour cela, il existe différentes approches qui varient selon le type et le volume des données. Dans cette section, nous discutons des catégories de l'apprentissage automatique.

1.4.1 Apprentissage automatique

L'apprentissage automatique ou l'apprentissage machine *Machine Learning* en anglais, est un sous-domaine de l'intelligence artificielle, qui vise à travers la conception et le développement d'algorithmes et de techniques basé sur des approches mathématiques et statistiques, à donner aux ordinateurs la capacité d'«apprendre» des informations pertinente à partir d'un ensemble de données sans être explicitement programmés. Le but est de concevoir un modèle capable d'effectuer des tâches telles que la prédiction et la classification sur de nouvelles données. Cependant, Le défi consiste à trouver un équilibre entre être capable d'apprendre avec précision l'ensemble de relation entre les critères caractérisant les données d'entrées (d'entraînement), et être capable de se comporter correctement face à de nouvelles situations (données) jamais vu auparavant. On parle de capacité du modèle à généraliser. Aujourd'hui, deux principaux types d'algorithmes de machine learning sont utilisés : l'apprentissage supervisé et l'apprentissage non supervisé.

1.4.1.1 Apprentissage supervisé

L'apprentissage supervisé (supervised learning en anglais) est l'une des tâches d'apprentissage automatique les plus couramment utilisés. Ce type d'apprentissage consiste à apprendre à partir d'un ensemble de données étiqueté de paires (entré-sortie) en entrés de l'algorithme de machine learning afin de trouver une fonction de prédiction qui associe l'entrée à la sortie. Cette phase est appelée phase d'apprentissage (learning) ou d'entraînement (training). Le but est d'apprendre à prédire pour toute nouvelle entrée X non présente dans l'ensemble d'apprentissage (appelé données de test), la sortie Y avec la plus grande précision possible. Dans cette approche, on distingue deux grands types de problèmes : La classification où Y désigne une valeur discrète (classe) et la régression où Y est une valeur continue.

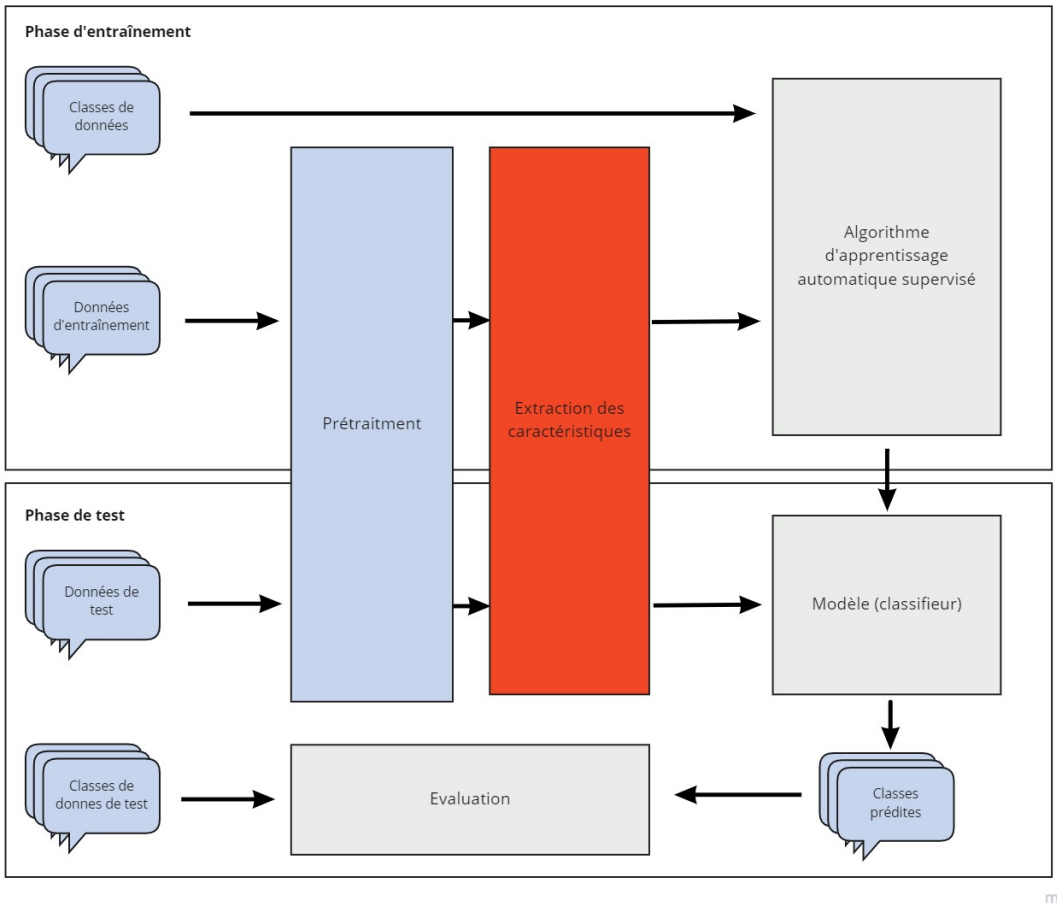

FIGURE 1.12 – Le processus de la classification supervisée [70]

Par exemple, le problème d'analyse des sentiments correspond à un problème de classification dans lequel plusieurs algorithme et techniques d'apprentissage supervisé ont été utilisés. Naeem S et al. [54], dans leurs travaille qui s'est intéressé à classification des avis de clients sous deux classes (positive et négative), ont utilisé des algorithmes d'apprentissage automatique classique avec l'approche supervisée. Pour faire leurs comparaisons, ils ont basé leurs expérimentations sur quatre classifieurs : Multinomial naïve bayes, k-plus proche voisin, forêt d'arbres décisionnels et régression logistique avec lequel ils ont atteint la meilleure précision.

1.4.1.2 Apprentissage non supervisé

Contrairement à l'apprentissage supervisé, le non supervisé (*unsupervisd learning* en anglais) consiste à traiter des données non-étiquetées. Il s'agit de la situation où on dispose seulement des entrées X sans avoir au préalable les sorties Y. Par conséquent, la phase d'entraînement du modèle dans cette approche consiste à traiter l'ensemble de données d'entraînement pour identifier des caractéristiques en communes aux entrées. Le but est de trouver une structure qui permet de regrouper les données de façon à ce qu'elles soient les plus similaires au sein

d'un même groupe (cluster). Cette tâche est appelée « clustering ». Parmi les algorithmes de clustering utilisés dans cette approche, nous distinguons l'algorithme de k-means [73] et le DBSCAN [77] par exemple. Fuente-Tomas et al. [16] ont proposé une solution basé sur la méthode de clustering K-means pour une classification de gravité de la maladie bipolaire ou le trouble bipolaire (Bipolar disorder en anglais). Le but est d'aider les cliniciens dans le processus de la médecine personnalisée et la prise de décision partagée. Dans cette étude, 20 sujets avec un diagnostic de de trouble bipolaire sous traitement ambulatoire ont été classés en cinq groupe (clusters) différents en se basant sur 12 variables de cinq domaines.

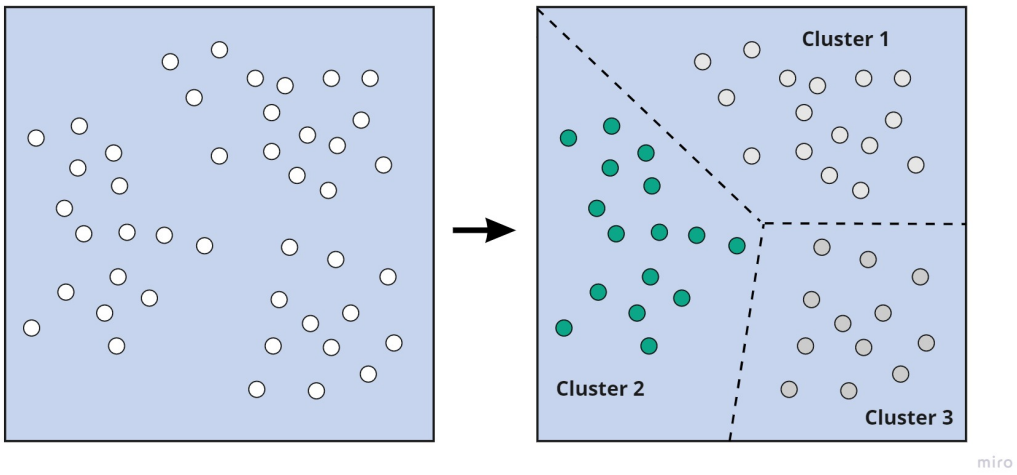

FIGURE 1.13 – Exemple de clustering réalisé sur un nuage de points. Le clustering a constitué 3 groupes [66]

1.5 Conclusion

Dans ce chapitre, nous avons commencé par présenter la notion du traitement automatique du langage naturel. Ensuite nous avons fait un tour sur les différentes techniques utilisées dans l'étapes de prétraitement de données textuelles dans le domaine de l'NLP telles que la racinisation, la lemmatisation, la tokenisation, etc., Nous avons vu aussi les différentes méthodes d'extraction de caractéristique comme le One-hot encoding, le sac de mots et le Tf-idf, et leur importance dans la dernière étape du processus de traitement du texte. Et pour terminer nous avons vu les différentes approches d'apprentissage automatique qui peuvent être utilisées dans la dernière étape pour créer un modèle capable de résoudre un problème réel de TAL.

Chapitre 2

Analyse des sentiments

2.1 Introdcution

Avec le développement rapide et la popularisation de la technologie du commerce électronique en Algérie, les consommateurs utilisent de plus en plus les réseaux sociaux et spécialement Facebook pour s'informer au sujet d'une marque, mais aussi pour exprimer leurs sentiments et avis sur ses produits et services. Le comportement humain est influencé par les opinions des autres. Ces opinions peuvent jouer un rôle crucial lors du processus de prise de décision liée à l'achat des clients potentiels. Par conséquent, ces données réprésentent une source d'informations importantes pour les entreprises, car l'analyse de ces données va leur permettre d'évaluer leurs images de marque, identifier les besoins de leurs clients ou utilisateurs du produit et comprendre leurs attentes. De ce fait, le domaine d'analyse des sentiments a connu dernièrement un intérêt croissant. Il est devenu une pièce importante à prendre en compte lors de l'élaboration de plan marketing et dans le processus de développement et d'amélioration de la qualité produit [57].

Dans ce premier chapitre, notre objectif est d'évoquer les principales notions liées à l'analyse des sentiments. Pour cela, nous introduisons d'abord la notion d'analyse des sentiments. Ensuite, nous présentons les différentes tâches et niveaux d'analyse. Nous exposons également les différentes approches existantes pour l'analyse des sentiments. Enfin, nous soulignons les différents domaines d'applications de l'analyse des sentiments.

2.2 Notions d'analyse des sentiments

Dans la littérature, les deux termes *analyse de sentiments* et *analyse d'opinion (opinion mining - en anglais)* sont souvent utilisés de manière interchangeable. Il s'agit de l'une des thématiques de recherche les plus étudiée dans le domaine de traitement automatique des langues (TAL). Il s'intéresse aux outils de traitement automatique qui permettent d'identifier et d'extraire les opinions, les sentiments et les informations subjectives exprimés dans des textes.

Les sentiments peuvent concerner différents entités, telles que des produits, des services, des organisations, des événements, des thématiques, des personnes publiques,...etc. Les textes peuvent être disponible sous différentes formats (types) : un article dans un journal, commentaire/critique dans un site internet, post/commentaire dans des réseaux sociaux (Facebook, twitter, reddit, etc.). Dans ce travail, nous s'intéressons aux textes de type commentaire/critique sur le réseau social Facebook.

2.2.1 Tâches d'analyse des sentiments

L'analyse des sentiments peut s'avérer difficile et compliqué comme problème. Il s'agit de résoudre un certain nombre de tâches qui peuvent être de natures différentes. Cela implique différentes stratégies et techniques d'analyse pour identifier les opinions contenues dans un texte dans un contexte particulier. Nous distinguons les 6 tâches principales suivantes :

— Détection de la polarité

— Détection de l'intensité de la polarité

— Détection de la subjectivité

— Indentification de l'entité

— Identification de la polarité par aspect

— Identification du détenteur de l'opinion

Cependant, il ne faut pas confondre ces tâches. Pour cela nous allons d'abord expliquer chacune d'entre elles, en considérons les exemples (Commentaire A), (Commentaire B), (Commentaire C) et (Commentaire D) suivants :

Commentaire A :

La première bêta de Windows 11 est désormais disponible

Commentaire B :

Excellent site. Choix des produits, prix réduits, service sérieux, emballage soigné, livraison rapide et sûre. Je recommande vivement.

Commentaire C :

J'adore ce site, il y a beaucoup de choix et des prix très intéressant, de plus l'envoi est très rapide. Seul bémol, un seul échantillon dans le colis c'est peu.

Commentaire D :

Mon papa a été très content de son cadeau. Et je suis d'autant plus satisfaite que le colis a été livré rapidement ... il l'a donc reçu le jour de son anniversaire ! Merci à vous.

L'analyse ou la détection de polarité consiste à déterminer et classer la polarité des énoncés en différents sentiments : positive ou négative (ou éventuellement neutre). En lisant nos exemples, nous remarquons que les polarités des commentaires B et D sont toutes les deux positives.
Pour aller plus loin, une analyse de polarité à grain fin peut être utilisée. Dans ce cas, une analyse à granularité fine allant du niveau d'analyse le plus global (classification binaire, classes génériques) au plus fin (excellent, bien, moyen, mauvais, très mauvais, etc.).
L'analyse ou la détection de la subjectivité est généralement un problème de classification binaire. Elle consiste à distinguer les faits des opinions, ce qui revient à faire la distinction entre les phrases objectives et les phrases subjectives. Le commentaire A par exemple, est considéré un énoncé objectif (fait). Par contre, le reste des commentaire (B, C et D) sont considéré comme des énoncés subjectifs (opinions).
La tâche d'extraction d'entité consiste à déterminer l'objet cible ou l'entité qui a été évaluée dans un énoncé. Pour les commentaires B et C, l'entité est le site internet de la marque. Une entité peut avoir un ensemble de composants ou d'aspects, et la tâche d'analyse des sentiments par aspect est un type d'analyse d'opinion qui vise à déterminer et identifier la polarité de chaque aspect de l'entité citée (évoquée) dans un énoncé d'opinion [58]. Par exemple, dans le commentaire B, la polarité positive pour l'aspect prix portée par le mot *réduits* et la polarité positive pour l'aspect service portée par le mot *sérieux*. Il est possible de vouloir connaitre et préciser le détenteur de l'opinion. Le commentaire D par exemple, contient les opinions de deux personnes : l'auteur du commentaire et son père.
Dans un énoncé, il est possible aussi de trouver certains aspects avec des polarités positives et d'autres aspects avec des polarités négatives. Il important de savoir qu'il arrive d'attribuer une polarité positive à un avis, toutefois, cela ne conduit pas à la conclusion que l'auteur avait des opinions positives sur chaque aspect de l'entité en question. Par exemple, le commentaire C. La même chose pour le cas des opinions négatives.

Avec un problème aussi difficile et complexe que l'analyse d'opinions, il est important que toutes ces tâches soient prises en considération afin de mieux comprendre et cerner le problème.

2.2.2 Type d'opinions

Une autre distinction importante qui doit être faite lors de l'analyse des sentiments est celle qui existe entre une opinion régulière et une opinion comparative. En fait, une opinion peut être classée en différents type selon deux traits : l'explicité et la régularité [63]. En effet, l'explicité permet de faire la distinction entre l'opinion explicite et implicite. Alors que la régularité permet de faire la distinction entre l'opinion régulière et comparative.

De ce fait, et selon le trait d'explicité, l'opinion est exprimée directement sur l'entité et/ou ses aspects évoqués dans l'énoncé. Par exemple, le commentaire B représente une opinion

directe puisqu'il fait référence à l'entité site et il là décrit d'une manière explicite et lui associe une polarité positive. En revanche, et contrairement à l'opinion explicite, l'opinion implicite est exprimée d'une manière indirecte. Dans le commentaire E, l'auteur exprime sa déception concernant la qualité du service de livraison d'une manière implicite.

Commentaire E :

Cadeau promis pour la fête des pères. Arrive après.

Commentaire F :

Prix hallucinant, j'ai offert à ma fille le même parfum que j'ai mais le prix est divisé par 2 de celui que je me suis offert il y a quelques mois sur Sephora ! ! ! Emballage et délais de livraison parfait.

D'autre part, selon le trait de régularité, il y a deux façons dont une opinion peut être analysée et évaluée : régulière et comparative. Par exemple, les commentaires B, C (page 16) représentent des opinions régulières puisqu'elles sont exprimées d'une façon simple et directe, et elles portent sur une seule et unique entité dans l'énoncé textuel sans mentionner d'autres entités similaires. En opposition à l'opinion régulière, l'opinion comparative porte sur la comparaison entre plusieurs entités similaires. Si on regarde le commentaire F, on trouve que l'internaute a exprimé une comparaison entre deux entités : le site sur lequel elle a récemment fait un achat (MyOrigines) et Sephora. L'analyse des expressions d'opinions comparatives est plus difficile par rapport aux autres types. La difficulté réside dans le fait de déterminer les entités préférées par le détenteur de l'opinion parmi les différentes entités comparées. Par exemple, dans le commentaire F, l'entité préférée du détenteur est le site MyOrigines.
Les opinions explicites et régulières sont plus faciles à analyser que les opinions implicites et comparatives qui demandent beaucoup plus travail et d'effort d'analyse.

2.2.3 Niveaux d'analyse des sentiments

L'analyse d'opinions dans les énoncés textuels peut se produire à différents niveaux. Cela dépend de la source et la longueur des énoncés, et aussi du comportement des détenteurs. Par exemple, sur le réseau social de microblogage Twitter, les tweets ne doivent pas dépasser 140 caractères, alors que sur d'autres plateformes, comme par exemple un site d'avis de consommateurs ou un site de marque ou même un réseau social comme Facebook, la longueur du commentaire n'est pas limitée. On trouve aussi que certains consommateurs ont tendance à écrire de longues critiques expliquant tout en détail, par rapport à d'autres qui préfèrent faire court et aller droit au but.

En général, une opinion peu t être évaluée et analysée selon quatre différents niveaux de granularité : au niveau du document, au niveau de la phrase, au niveau de mot et au niveau de l'aspect. Ces quatre niveaux d'analyse sont détaillés dans la suite.

2.2.3.1 Niveau document

La tâche d'analyse d'opinion à ce niveau consiste à associer une polarité positive ou négative (ou neutre) à l'ensemble de document [56]. Ce niveau d'analyse suppose que les opinions exprimées dans un document donné concernent une seule entité (par exemple, un seul produit ou un seul aspect du produit). Ceci signifie que cette analyse ne convient pas dans les cas où un document évalue ou compare plusieurs entités en même temps. Cela peut entraîner une perte d'informations.

2.2.3.2 Niveau phrase

Ce niveau d'analyse est étroitement lié à deux tâches qu'on a vu précédemment : La classification de subjectivité et la détection de la polarité. Ce qui signifie que le processus d'analyse d'opinion dans ce cas comporte généralement deux étapes. La première étape consiste à distinguer les phrases objectives des phrases subjectives, c'est-à-dire différencier entre les phrases de faits et les phrases d'opinions. Dans la deuxième étape, il s'agit d'associer aux phrases subjectives une polarité : positive, négative ou neutre [41].

2.2.3.3 Niveau aspect

L'analyse au niveau aspect va plus loin que les deux analyses au niveau document et phrase. Cette analyse vise un niveau de granularité plus fin. En effet, l'objectif est d'identifier et déterminer la polarité de chaque aspect évoqué dans le document par le détenteur de l'opinion [62]. Ceci nécessite un double traitement, d'abord identifier l'entité et les aspects de l'entité en question, ensuite évaluer la polarité sur chaque aspect. Par exemple, le commentaire C évalue quatre aspects : choix de produits, prix, livraison, l'échantillonnage. L'analyse de ce commentaire permet d'identifier la polarité pour chaque aspect : (choix de produits, positive), (prix, positive), (livraison, positive) et (l'échantillonnage, négative).

2.2.3.4 Niveau mot

L'analyse d'opinion au niveau mot représente le niveau le plus bas. Il s'agit de déterminer la polarité d'un mot, ce qui n'est pas toujours facile, car un mot peut avoir des polarités différentes selon le contexte ou domaine dans lequel il est employé. Par exemple, la polarité du mot « longtemps » dans le commentaire G est négative. Alors que dans le commentaire H est positive. Ce niveau d'analyse est important car il permet notamment la construction de lexique de mots polarisés. Une sorte de dictionnaire composé d'un ensemble de couple (ω, *pol*), où *pol* est une valeur réelle reflète la polarité du mot ω.
Les mots polarisés sont généralement des adjectifs et des adverbes. Ces deux traits grammaticaux sont les plus utilisés pour exprimer des opinions, même si dans certains cas on peut trouver des verbes et des noms. Au final, ces mots porteurs d'opinion peuvent être classés en deux classes

reflétant deux états émotionnels différents : désirable ou indésirable. Par exemple, dans le cas de l'état désirable on trouve des mots ont une polarité positive, des mots tels que : « satisfait », « heureux », « magnifique », « content », « beau », etc., tandis que dans l'état indésirable on trouve des mots qui ont une polarité négative, des mots tels que : « ennuyant », « horrible », « médiocre », « décevant » et « mauvais ». Ces mots polarisés sont généralement utilisés dans une analyse d'opinion de haut niveau (document, phrase ou aspect).

Commentaire G :

Je n'ai jamais reçu ma commande, je ne veux pas avoir à faire à vous et ne plus recevoir sms. Je prie de ne plus m'envoyer quoi que ce soit, Merci. Ça a duré assez longtemps cette histoire de commande.

Commentaire H :

Téléphone très agréable à l'utilisation en raison de son ergonomie et son design. Bonne prise en main, mais attention, il glisse ! ! La batterie dure longtemps près de deux jours en utilisation. Je suis satisfaite de mon achat.

2.3 Approches d'analyse d'opinions

Il existe énormément de recherches et travaux qui ont été effectuées sur l'analyse d'opinions. Par conséquence, de nombreuses et différentes méthodes et techniques ont été proposées pour résoudre ce problème. Aujourd'hui dans la littérature, trois approches d'analyse peuvent être distinguées : l'approche linguistique ou symbolique, l'approche statistique ou numérique et l'approche hybride. L'approche linguistique repose sur des lexiques et des règles linguistiques, tandis que l'approche statistique utilise des méthodes d'apprentissage automatique. Et ce qui concerne l'approche hybride, elle combine les deux approches précédentes en utilisant à la fois les lexiques et les algorithmes d'apprentissage automatique.

2.3.1 Approche linguistique

L'approche linguistique (dite aussi symbolique) est une approche qui se base sur le lexique et les règles linguistiques pour déterminer la polarité d'un énoncé textuel. Elle consiste à calculer la polarité d'un texte en utilisant l'orientation sémantique des mots qui composent ce texte et se trouvent dans le lexique. Au fait, cette approche dépend principalement de ce dernier. Il s'agit d'un ensemble de mots généralement des adverbes et des adjectifs (parfois même des noms et des verbes) polarisés soit d'une façon qualitative (positive ou négative, ou parfois neutre) ou quantitatif où les mots négatifs ont des valeurs inférieures à zéro et les mots positifs ont des valeurs supérieures à zéro. Cependant, la principale tâche dans cette approche est la conception et la construction de lexiques. Pour cela, deux façons différentes sont possibles : basé sur le corpus ou basé sur un dictionnaire.

2.3.1.1 Approche basé sur un dictionnaire

Les méthodes et les techniques dites à base de dictionnaires repose sur un ensemble initial de mots appelé jeu initial ou graine. Elles consistes à élargir cet ensemble en utilisant des dictionnaires de synonymes et antonymes, des base de données lexicale (*WordNet* par exemple) ou un thésaurus. Il s'agit d'enrichir le jeu initial en ajoutant les synonymes de chaque mot polarisé ω dans l'ensemble de départ en leurs attribuant la même polarité que ω, et en ajoutant les antonymes de ω en leur associant l'opposée de la polarité de ω.

2.3.1.2 Approche basé sur un corpus

Les techniques de cette approche consistent à trouver de nouveaux mots polarisés à extraire à partir d'un corpus annoté donné et un jeu initial, en utilisant deux approches différentes : approche statistique et approche sémantique. Dans l'approche statistique, les nouveaux mots polarisés sont extraits à partir du corpus par calcul de fréquence d'occurrence et la co-occurrence de mots. Par exemple, Abdulla et al [3] ont utilisé la pondération de fréquence de mots (TF) (*term frequency*) d'un corpus de commentaires équilibré pour extraire une liste de mots polarisés. Alors que Turney et Littman [79] ont utilisé l'information de co-occurrences de mots pour étendre leur jeu initial de mots polarisés. Ils ont proposé deux méthodes. La première est basée sur le calcul d'information mutuelle *Pointwise Mutual Information* (PMI), et la deuxième est L'analyse sémantique latente *Latent semantic analysis* (LSA). Ces méthodes sont basées sur l'hypothèse que les mots proches dans un document ont probablement la même polarité.

2.3.2 Approche numérique

L'approche numérique (dite aussi statistique) repose sur des techniques d'apprentissage automatique *Machine Learning* et/ou profond *Deep Learning*. Dans cette approche, le problème d'analyse d'opinion qui consiste à déterminer la polarité d'un énoncé donné, est considéré comme un problème de classification basé sur la polarité. Le processus commence par une *phase d'apprentissage* et il se termine par une *prédiction*. Dans la phase d'apprentissage, l'algorithme ou le modèle est entraîné sur un corpus d'apprentissage (un ensemble d'exemples annotés en polarité). Alors que dans la dernière étape de prédiction le modèle entraîné devrait pouvoir prédire la polarité d'un nouveau commentaire non étiqueté. Il faut noter aussi que contrairement à l'approche symbolique qui analyse le commentaire mot par mot et agrège la polarité globale du commentaire à la fin, l'approche numérique traite le commentaire en entier et attribue une polarité globale à la fin.

2.3.2.1 Apprentissage automatique

L'apprentissage automatique consiste à entraîner un modèle (dite aussi classifieur) d'analyse prédictive. Pour y parvenir, cette approche requiert d'exploiter pleinement les données

d'apprentissage disponible. D'une part, par la quantité et la qualité de données d'apprentissage, et d'autre part en utilisant le « Feature Engineering » ou ingénierie des caractéristiques. Il s'agit d'extraire des caractéristiques (*features* en anglais) qui représente l'information contenue dans le texte. Parmi les caractéristiques qui peuvent être utilisées sont : les mots, n-grams, part of speech, la pondération de fréquence de mots, sac de mots, etc [4].
Parmi les algorithmes d'apprentissage automatique (classifieurs) qui ont été largement utilisés dans le domaine de l'analyse d'opinion on trouve : l'algorithme naïf de Bayes (NB) [39], les machines à vecteurs de support (SVM) [87], les k plus proches voisins (KNN) [37], la régression logistique (LR) [6] et l'entropie maximale (ME) [85].

Machine à vecteurs de support Les machines à vecteurs de support ou séparateurs à vaste marge, dite aussi *Support Vector Machine* en anglais (SVM), représentent un modèle d'apprentissage automatique supervisé découle directement des travaux de Valdimir Vapnik [13, 59, 80]. Ce modèle linéaire est destiné à résoudre plusieurs problèmes notamment la classification binaire. Il fonctionne dans le but de trouver un hyperplan qui va séparer les données en deux classes positive et négative en recourant à la marge maximale entre ces deux groupes de données.

Naive Bayes Naive Byses, est une méthode d'apprentissage supervisé couramment utilisée pour la classification de texte. Elle repose sur une méthode probabiliste basé sur le théorème de Bayes. Ce dernier est fondé sur l'hypothèse que les prédicteurs sont indépendants les uns des autres (d'où le terme *nïve*), que chaque caractéristique classée est indépendante de la valeur de toute autre caractéristique [68].

Arbre de décision L'arbre de décision est une méthode de classification supervisé basé sur l'utilisation d'un arbre de décision comme modèle prédictif. Elle consiste à partitionner le grand ensemble de données en sous-ensemble plus petits. Un arbre de décision est une structure semblable à un organigramme et peut être converti en un ensemble de règles de contrôle conditionnelles (si-alors).

Régression logistique La régression logistique est une méthode de classification qui étudie le rapport entre une variable principale et des variables explicatives. Il s'agit de prédire et/ou expliquer une variable catégorielle Y à partir d'un ensemble de descripteurs X. Cette méthode peut prendre différentes formes : binaire ou multinomiale,

K plus proche voisins La méthode des k plus proches voisins *K-Nearest Neighbors* en anglais (KNN) fait partie des algorithmes d'apprentissage automatique supervisé les plus simples qui peuvent être utilisés pour la classification. L'algorithme KNN suppose que les données similaires existent à proximité. En d'autres termes, des choses similaires sont proches les unes

des autres. En effet, le principe de ce modèle consiste à choisir les k données les plus proches du point étudié afin d'en prédire sa classe.

2.3.2.2 Apprentissage profond

L'apprentissage profond, ou *Deep Learning*, est un sous-ensemble des méthodes d'apprentissage automatique basé sur des réseaux de neurones artificiels à plusieurs couches de traitement d'information non linéaires [17]. Un réseau de neurones est en effet capable d'apprendre différentes caractéristiques pertinentes et de plus en plus représentatives des données par lui-même grâce à ces couches.
Plusieurs architectures neuronales existantes en fonction du champ d'application. Parmi les plus utilisées pour la tâche d'analyse d'opinions on trouve : les réseaux de neurones récurrents *recurrent neural network* (RNN), des RNN simples [12] aux réseaux récurrent à mémoire court (Long short-term memory, en anglais) (LSTM) [82, 86], et les réseaux de neurones convolutifs (CNN) [15, 38].

2.3.3 Approche hybride

L'approche hybride est une méthode qui consiste à combiner les deux approches précédentes, l'approche symbolique et l'approche numérique. Cette méthode a été proposées pour améliorer les performances et l'efficacité du modèle construit pour l'analyse d'opinions en utilisant à la fois les méthodes d'apprentissage automatique ou profond et des lexiques [64].

2.4 Domaines d'application de l'analyse d'opinions

Le domaine d'analyse des sentiments s'est considérablement développé au cours des dernières années. Aujourd'hui, l'analyse d'opinions et des sentiments a des applications utiles dans divers domaines de différentes industries, telles que : La politique, la santé, l'éducation, l'e-commerce et le marketing.

La politique Au cours de la dernière décennie, l'utilisation de comptes Twitter politiques a explosé. De nos jours, il semble que chaque leader politique et toute sa famille ont un compte Twitter, et ils ressentent tous le besoin de partager leurs opinions avec leurs électorats sur la plateforme. En fait, de nombreux dirigeants utilisent Twitter comme principal canal de communication avec le public en ligne. Cependant, cela a causé des problèmes intéressants. Souvent, les tweets ne sont pas contrôlés, et lorsqu'ils proviennent des comptes « personnels » plutôt que de « communications officielles gérées par le gouvernement », ils peuvent être assez polarisants et/ou problématiques. D'où l'importance de l'analyse des sentiments dans ce domaine. Par exemple, Caetano et al. [11] ont proposé une analyse de l'homophilie politique parmi les utilisateurs de Twitter lors de l'élection présidentielle américaine de 2016 .

La santé L'analyse d'opinions est également utilisée dans le domaine de la santé. Aujourd'hui, il existe plusieurs plateformes dédiées à la santé avec des fonctionnalités sociales permettant aux patients de partager leurs sentiments à propos de leurs maladies, traitements, la qualité des soins de santé qu'ils reçoivent, etc. Pratiquement chaque interaction avec un prestataire ou un hôpital déclenchera une réaction, positive ou négative. Par conséquent, l'analyse des sentiments dans le domaine de santé est très précieuse et offre de nombreux avantages, tels que l'utilisation de ces informations pour peut améliorer l'expérience du patient et la qualité des soins. Par exemple, l'analyse de contenus relatifs à la santé permet de mesurer l'impact d'une maladie sur la personne touchée et son expérience avec cette maladie. Des études dans ce sens ont été menées pour des maladies comme le cancer par exemple [14].

L'éducation L'apprentissage en ligne est une excellente alternative moderne à l'apprentissage traditionnel et il est de plus en plus utilisé. En effet, plusieurs sites web et plateformes de cours en ligne ouverts et massifs *Massive Open Online Courses* en anglais (MOOC) proposent des cours en ligne avec un espace d'interaction (généralement un forum) qui permet aux participants d'interagir et de socialiser entre eux et avec les animateurs du cours. L'exploration de toutes ces données des interactions, des réponses à des questions, des avis, etc., et l'application de l'analyse d'opinions sur elles est considéré une étape importante dans le développement des MOOC, car elle permet de mieux comprendre l'attitude des apprenants envers les cours, évaluer leur degré de satisfaction et leur attente et les éventuels prédicteurs du sentiment d'abandon. Par conséquent, les enseignants peuvent planifier leurs cours en temps opportun pour garder plus d'élèves sur la bonne voie. De plus, la comparaison des niveaux de sentiment de différents cours peut révéler des modèles d'émotions et d'attitudes communs entre les apprenants, ce qui peut être d'une importance pratique pour les concepteurs pédagogiques. Cela peut aussi donner des idées précieuses à la direction du développement de la plateforme [44, 65].

L'e-marketing L'analyse de sentiments est aussi appliquée dans le domaine de commerce et de l'e-marketing (ou marketing digital). Il est évident aujourd'hui que le secteur du commerce électronique génère de grandes quantités de données sur les différentes plateformes (sites web, réseaux sociaux, etc.). Au fait, les entreprise et les marques analysent de plus en plus les commentaires de ses clients et leurs retours afin de découvrir ce qu'ils aiment et/ou n'aiment pas à propos de leurs produits et services. Cette analyse fournit des informations inestimables aux spécialistes du marketing et aux décideurs. Par exemple, analyser les avis des clients sur un produit peut aider la marque dans l'orientation des futures conceptions pour améliorer ce produit. Elle s'avère utile aussi sur le plan marketing, car les professionnels du marketing donnent le meilleur d'eux-mêmes lorsqu'ils parviennent à comprendre leur public cible. Les métriques traditionnelles qui concentrent sur le nombre de vues, de clics, de commentaires, de partages, etc., ne suffit pas. Cela signifie que les entreprises doivent comprendre l'opinion des consommateurs et du public vis-à-vis leur marque, de leurs publications sur les réseaux

sociaux et de leurs compagnes de publicité aussi pour adopter les bonnes stratégies en créant des compagnes marketing avec des messages personnalisés pour répondre aux besoins du public [5, 12, 36, 42].

2.5 Challenges de l'analyse d'opinions

L'analyse d'opinion peut être une tâche difficile. Cette difficulté provient du fait que le langage humain est complexe. Entraîner un modèle capable d'analyser les sentiments dans un texte des réseaux sociaux avec toutes les diverses nuances grammaticales, les variations culturelles, l'argot et les fautes d'orthographe, etc, représente un vrai défi dans le processus d'analyse. Par conséquent, les challenges liés à cette tâche sont liés fortement à la qualité et la quantité des ressources disponible. Nous résumons dans la suite certains de ces challenges et difficultés.

2.5.1 Ironie et sarcasme

Dans un énoncé sarcastique, les gens expriment leurs sentiments négatifs en utilisant des mots positifs. En d'autres termes, le sarcasme est les mots utilisés pour exprimer autre chose que le sens littéral d'une phrase et en particulier le contraire. Ce fait permet aux énoncés sarcastiques de tromper facilement les modèles d'analyse d'opinions, à moins qu'ils ne soient spécifiquement conçus pour prendre en compte ce type de phrases.
On trouve souvent des expressions sarcastiques dans le contenu généré par les internautes sur les différents réseaux sociaux, tels que les commentaires Facebook, les tweets, etc. Par exemple, le sarcasme dans une critique indique souvent l'insatisfaction du client comme le montre les commentaires I et J.

Commentaire I :

Je rêve. C'est formidable. J'ai pas ma commande mais j'ai 6€

Commentaire J :

Pour se déguiser en fantôme elle est parfaite.

Différents consommateurs peuvent utiliser le sarcasme dans leurs commentaires de manières différente. Cela peut rendre la tâche du modèle d'analyse difficile. Si on regarde les exemples précédents, on remarque qu'ils contiennent des mots positifs comme « formidable », « parfaite » qui indiquent une polarité positive alors que les deux énoncés se réfèrent en fait à la polarité négative..

2.5.2 Phrase comparatives

Les phrases comparatives peuvent être difficile à analyser car elle n'exprime pas toujours un sentiment ou une opinion. Par exemple les commentaires K et L ne mentionnent

aucune émotion, qu'elle soit positive ou négative. Dans ce cas pour qu'on puisse déterminer la polarité du commentaire on a besoin de plus d'informations ou d'indices concernant son contexte.

Commentaire K :

Le design de l'iPhone 12 est un peu différent des précédents modèles.

Commentaire L :

En taille, les deux téléphones sont très similaires.

2.5.3 Phrase conditionnelle

Les phrases conditionnelles sont l'une des structures linguistiques couramment utilisées dans les textes. Et quand il s'agit de l'analyse d'opinions dans des énoncés textuels, ce type de phrase peut être difficile à analyser à cause de la subordonnée circonstancielle de condition. Dans le commentaire M par exemple, nous peuvons avoir l'impression que le détenteur de l'opinion il a été déçu. Cependant, nous ne pouvons pas être sûr de la polarité de ce commentaire, si elle est positive ou négative [25].

Commentaire M :

Le film aurait été parfait sans la partie musicale.

Commentaire N :

J'aurais donné 5 étoiles si la tasse avait pu contenir un plus grand volume de thé.

2.5.4 Opinion spam

Dans un domaine comme le e-commerce par exemple où la concurrence est de plus en plus féroce, la réputation numérique est devenue un actif valorisé pour les entreprises. Les internautes jouent évidemment un rôle déterminant pour l'e-réputation d'une entreprise. Que ce soit par le biais des échanges sur les réseaux sociaux, des avis sur Google ou les sites de critiques, du bouche-à-oreille numérique, du partage d'informations, etc. Avec un smartphone dans la poche et quelques tweets ou commentaire, l'internaute peut créer un engouement ou une polémique en quelques secondes. D'autre côté, la concurrence constitue, elle aussi, un facteur pouvant nuire à la e-réputation et la popularité d'une marque, en publiant de faux avis sur les sites de critique, des commentaires mesquins sur Facebook, elle lance des remeurs outrageuses et infondées, ou pire encore, elle s'acharne sur l'un des meilleurs produits de la marque avec un ton diffamatoire et des informations erronées, calomnieuses, voire fausses pour que plus personne n'ait envie de l'acheter. D'où la grande importance d'effectuer une veille constante auprès de ces opinions spams qui tentent délibérément d'induire en erreur les nouveaux prospects ou le système d'analyse d'opinion.

2.5.5 Négation

Dans l'analyse des sentiments, il est très important de traiter la négation, car généralement elle agit directement sur l'orientation de la polarité. Donc des mots de négation comme « ne » et « pas » sont souvent pris en compte dans les travaux avancés. Par exemple, les deux phrases " La qualité du produit est bonne" et " La qualité du produit n'est pas bonne" peuvent être considérées comme étant très similaires par le système d'analyse simple [43, 75].

2.5.6 Focalisation sur l'anglais et Pénurie de ressources

Appliquer la tâche d'analyse d'opinion sur d'autres langues que l'anglais peut représenter un vrai défi. La majorité des recherches qui ont été menés dans ce domaine se sont focalisés sur la langue anglaise. ont été faites en utilisant des outils, des ressources données et corpus sont en anglais. En revanche, peu de travaux ont été réalisés en d'autres langues, par exemple,la langue arabe ou les diffrents dialèctes. Ceci s'explique par l'indisponibilté des corpus et données et le faible nombre de ressources développées pour ces langues. Par exemple, un lexique d'opinions polarisées est souvent utilisé pour créer un système d'analyse d'opinions. Cependant, ce type de ressources manque cruellement en langue comme le français et celles qui existent sont souvent des traductions des lexiques anglais qui restent trop généraux pour être adaptés à un domaine spécifique comme le marketing ou la médecine par exemple.

2.5.7 Complexité linguistique

Le problème d'analyse d'opinion est très difficile à cause des caractéristiques grammaticales et morphologiques complexes de certaines langues qui rendent la tâche du traitement automatique plus difficile encore. Par exemple, la langue arabe par ces trois catégories : l'arabe classique, l'arabe standard moderne et l'arabe dialecte, représente l'une des langues les plus répandues dans le monde, les plus riche, et les plus complexes. Sa complexité réside dans ses caractéristiques telle que l'agglutination et la non voyellation et sa richesse morphologique qui conduit à un vocabulaire épars. Toutes ces caractéristiques génèrent des cas d'ambiguïté et rendent l'analyse syntaxique de l'arabe plus ardue.

2.6 Conclusion

Nous avons présenté, dans ce premier chapitre, la notion de l'analyse des sentiments, avec toutes ces tâches. Nous avons vu aussi les différents types ainsi que les niveaux d'analyse. Par la suite, nous avons abordé les différentes approches existantes dans la littérature pour ce genre de problème. Nous avons également présenté les différents domaines d'applications avec quelques difficultés qui peuvent être rencontrées durant le processus de création d'un modèle d'analyse.

Chapitre 3

Analyse des sentiments basé sur l'apprentissage profond : Etat de l'art

3.1 Introdcution

L'apprentissage profond (DL : Deep Learning en anglais) est un ensemble de méthodes dérivé de l'apprentissage automatique (Machine Learning) basé sur les réseaux de neurones artificiels inspirés du cerveau humain. Ces méthodes on permis des progrès importants et rapides dans les tâches du NLP telles que la traduction automatique et l'analyse des sentiments. En effet, les algorithmes d'apprentissage profond traitent des informations plus complexes que le Machine Learning. Le processus d'apprentissage dans ce cas est qualifié de profond parce que la structure des réseaux de neurones se compose de plusieurs couches d'entrée, de sortie et masquées qui sont toutes reliées entre elles. Chaque couche contient des unités qui transforment les données d'entrée en informations que la couche suivante peut utiliser. Grace à cette architecture les algorithmes de Deep Learning peuvent modéliser les données avec un haut niveau d'abstraction. Les premiers réseaux ont éclos sous le nom de perceptron. Ce dernier représente la forme la plus simple d'un réseau de neurones, il est composé d'un seul neurone (Figure 3.1), et ne permet de résoudre que des problèmes linéairement séparables. Plus tard, de nouveaux types de réseaux ont vu le jour, et ils ont permet de résoudre des problèmes de classification complexes.

Le perceptron (neurone formel) est formée d'une première couche d'unités (ou neurones) qui prend des entrées $X = (x_1, x_2, ..., x_n)$, où chaque unité correspond à une des variable d'entrée. Ces entrées sont associées à des poids $W = (w_1, w_2, .., w_n)$. Ce neurone prend en entrée également un biais b. Ces unités sont reliées à une seule et unique sortie qui correspond à l'application d'une fonction d'activation échelon (Treshold activation function) (Equation 3.1) sur la somme des unités $X = (x_1, x_2, ..., x_n)$ pondérés par les poids $W = (w_1, w_2, .., w_n)$ et le biais b.

$$f(x) = \ 1 \ \textit{si} \sum_{i=1}^{n} x_i w_i > \textit{threshold} \ \ \textit{sinon} \ \ 0 \tag{3.1}$$

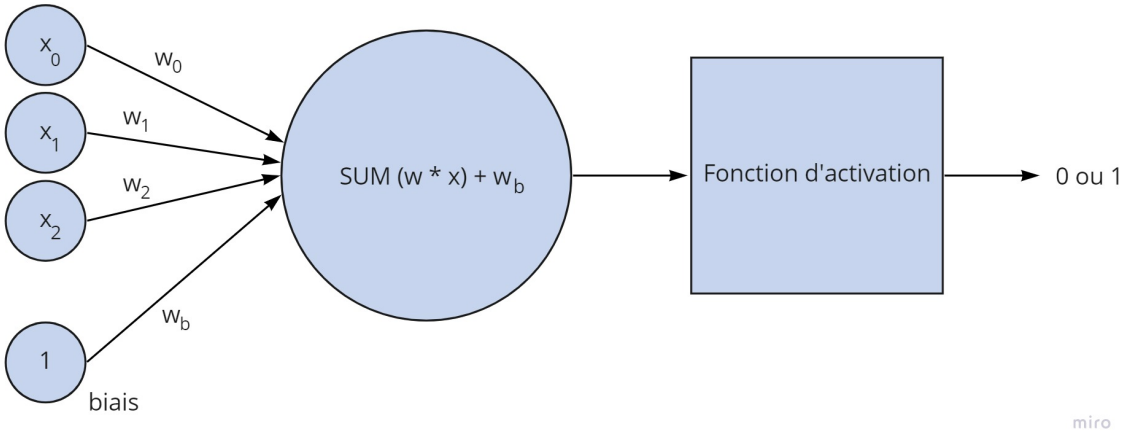

FIGURE 3.1 – Perceptron basic [33]

3.2 Word Embedding

Dans la section 2.3 du chapitre 2 nous avons présenté différentes techniques de représentation de texte telle que One-hot encoding, le TF-IDF et le bag of words. Bien qu'il s'agisse de méthodes efficaces pour extraire des caractéristiques du texte, il existe de graves limitations où nous perdons des informations supplémentaires telles que la sémantique, la structure, l'ordre et le contexte autour des mots voisins dans chaque document texte. En conséquence, nous avons parlé de la notion de word embedding et nous avons donné quelques techniques comme le Word2Vec, Glove et le FastText qui ont été utilisées souvent dans le problème de l'analyse des sentiments par exemple, mais sans donner beaucoup de détails. Dans cette section, nous allons voir ces techniques de près et avec un peu plus de détails.

3.2.1 Word2Vec

Word2Vec est l'un des modèles de word embedding (plongement de mots ou plongement lexical en français) les plus connu. Il a été proposé et développé par une équipe de recherche de Google sous la direction de Thomas Mikolov [48]. Ce dernier, a trouvé un moyen d'encoder le sens des mots dans un espace vectoriel de faible dimension. Il a réussi à entraîner un réseau de neurones pour prédire les occurrences de mots voisins de chaque mot donné. La méthode repose sur des réseaux de neurones à trois couches et cherche à apprendre les représentations vectorielles des mots composant un texte donné, de telle sorte que les mots qui partagent des contextes similaires soient représentés par des vecteurs numériques proches. Cependant, il existe deux variantes de Word2Vec, ainsi, deux architectures ont été initialement proposées : Skip-gram et CBOW. La figure 3.2 illustre les deux variantes.

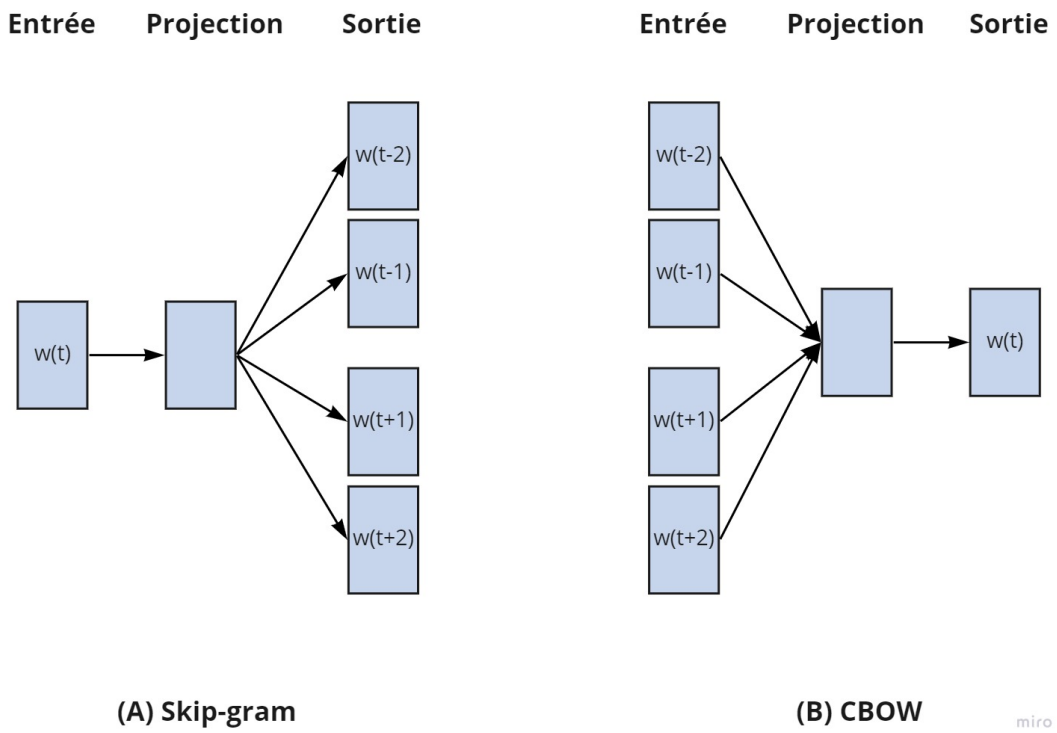

FIGURE 3.2 – Architectures de Skip-gram (A) et CBOW (B) [48]

3.2.1.1 CBOW

La première architecture représente le modèle du sac continu de mots *Continuous Bag-of-Words* en anglais (CBOW). Ce modèle consiste à prédire un mot w_i en fonction de son contexte, c'est à dire les mots qui l'entourent dans une phrase. On appelle ce contexte une fenêtre de n mots à gauche et à droite du w_i.

3.2.1.2 Skip-gram

La seconde architecture est nommée *Skip-gram*. Elle est similaire à CBOW, mais contrairement à cette dernière, cette méthode permet de prédire une fenêtre de contexte c sachant le mot w_i. Autrement dit, il s'agit de prédire, pour un mot donné en entrée, le contexte dont il est issu.

3.2.2 Glove

Comme le Word2Vec, Glove ou *Global Vectors for Word Representation* est une autre méthode populaire qui vise à produire des représentations vectorielles des mots en se basant sur leur contextes. Elle a été développée par Pennington et al. à Stanford [60]. Cependant, la différence par rapport Word2Vec, est que Glove est un modèle avec une approche « count-based

» ou base de comptes en utilisant une matrice de co-occurrenc, contrairement à Word2Vec qui est un modèle prédictif.

3.2.3 FastText

Est un autre modèle de représentation vectorielle des données textuelles proposé par Bojanowslo et al. [8]. Il s'agit d'une extension du modèle Word2Vec, sauf que dans cette méthode, pour apprendre les représentations de mots, on prend en compte l'information de type n-grammes en divisant les mots en sous-mots. FastText considère les mots comme des sacs de n-grammes de caractères, par exemple, si on prend le mot "bonjour" et, n = 2, alors on obtient les sous-mots suivants : <b, bo, on, nj, jo, ou, ur, r>, où "<" et ">" sont des symboles représentent le début et la fin du mot.

3.2.4 Autres méthodes : Elmo et BERT

Toutes les techniques de word embedding vu précédemment ont une chose en commun : chaque mot distinct ne peut avoir qu'une seule représentation vectorielle. Cependant, les dernières années ont été témoins de nouvelles méthodes pour la représentation des mots : les embedding contextuels. Nous citons principalement : Elmo [61] et BERT [18, 25]. Ces modèles sont capables non seulement de gérer les caractéristiques complexes des différentes utilisations des mots telles que la syntaxe et la sémantique, mais aussi les contextes linguistiques selon comment ces utilisations changent dépendamment du contexte.

3.3 Réseaux de neurones pour l'analyse des sentiments

Avant d'aborder quelques différents architectures neuronales utilisées pour le problèmes d'analyse des sentiments, nous présentons d'abord ce qu'on appelle *Perceptron multi-couches*.

3.3.1 Perceptron multi-couches

Nous avons vu dans le début de ce chapitre le premier type de Perceptron, le neurone formel, qui représente le type de réseaux de neurones le plus simple. Ce premier type disposait une couche unique (mono-couche) avec une seule sortie à laquelle toutes les entrées étaient connectées, et n'étaient pas capable de résoudre des problèmes non linéaires. Cependant il existe un deuxième type, le perceptron à couche multiples (PMC) *Multilayer Perceptron (MLP)* [51]. Il s'agit d'un réseau de neurones à propagation avant (*feed-forward* en anglais), dans lequel l'information circule dans un seul sens : de la couche d'entrée vers la couche de sortie en passant

par une ou plusieurs couches cachées. Ces couches constituées d'un nombre variable de neurones successives complètement connectées, c'est-à-dire, les neurones de chaque couche sont reliés en entrée à tous les neurones de la couche précédentes et en sortie à tous les neurones de la couche suivante. La mise en place d'un perceptron multicouche à permet de traiter les problèmes non linéairement séparables, et donc à surmonter la limite du perceptron simple. Cette limitation est surmontée à l'aide d'un algorithme de rétro-propagation ou *Backpropagation* en anglais.

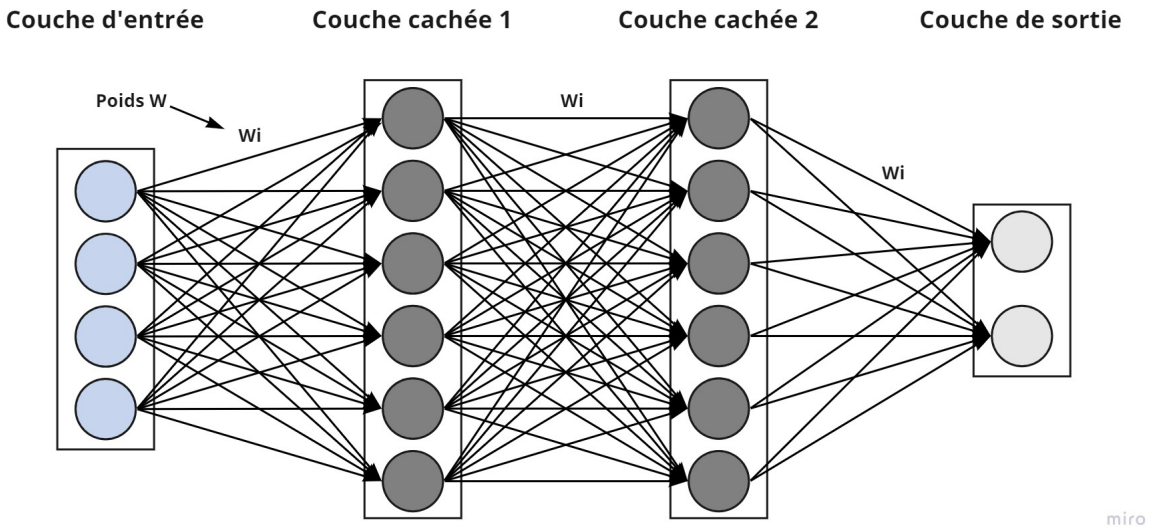

FIGURE 3.3 – Schéma d'un perceptron multi-couches [30]

La rétropropagation du gradient [69] est une méthode qui consiste à calculer le gradient de l'erreur pour chaque neurone du réseau, de la dernière couche vers la première. Il s'agit d'un algorithme d'optimisation des poids w_i. Le but est d'ajuster ces paramètres de ce réseau multi-couches afin de déterminer les meilleurs poids qui permettent de minimiser la différence entre la sortie du réseau et la sortie désirée selon la base d'apprentissage. Cette différence est évaluée par la fonction de perte (dite aussi fonction de coût) *Loss function* en anglais, comme par exemple, l'entropie croisée (*Cross entropy* CE) pour la classification ou l'erreur quadratique moyenne (*Mean Square Error* MSE) pour la régression. En outre, la rétropropagation nécessite d'autres types de fonction d'activation, des fonctions non linéare. Pour un problème de classification par exemple, il existe deux fonctions très utilisées : La fonction Sigmoîde (Equation 3.2) pour une classification binaire, et la fonction Softmax (Equation 3.3) pour une classification en classes multiples.

$$\text{Sigmoïde :} \quad S(x) = \frac{1}{1 + e^{-x}} \tag{3.2}$$

$$\text{Softmax :} \quad \sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} + e^{z_k}} \tag{3.3}$$

En résumé, le but de l'apprentissage supervisé d'un réseau de neurones (un ensemble de perceptrons interconnectés) est de converger vers une configuration optimisée des poids qui

sont au préalable initialisé avec des valeurs aléatoires. En appliquant cette étape plusieurs fois, l'erreur tend à diminuer, et nous finissons par aboutir à une configuration stable qui peut nous offrir une bonne précision de prédiction.

3.3.2 Réseaux de neurones récurrents

Un réseau de neurones récurrent (*Recurrent Neural Network* en anglais) (RNN) fait partie de la famille des MLP, mais contrairement à ces derniers, un réseau RNN comporte des connexion récurrentes (des cycles au sein du réseau de neurones) [23]. Le but principale de cet type d'architectures, présenté dans la Figure 3.4, est de pouvoir manipuler des données de manière séquentielles. Cet aspect de récurrence dans les réseaux RNN consiste à prendre en compte lors d'un instant t, une information extraite lors d'un certain nombre d'instants passés. Autrement dit, à une étape courante, une ou plusieurs informations prédites dans une étape précédente, sont prises en considération. Par conséquent, on dit que la structure des RNN à introduit un mécanisme de mémoire des entrées précédentes, d'où ils peuvent facilement se souvenir de l'état des calculs précédent, ce qui peut impacter les sorties futures. Pour cette raison,

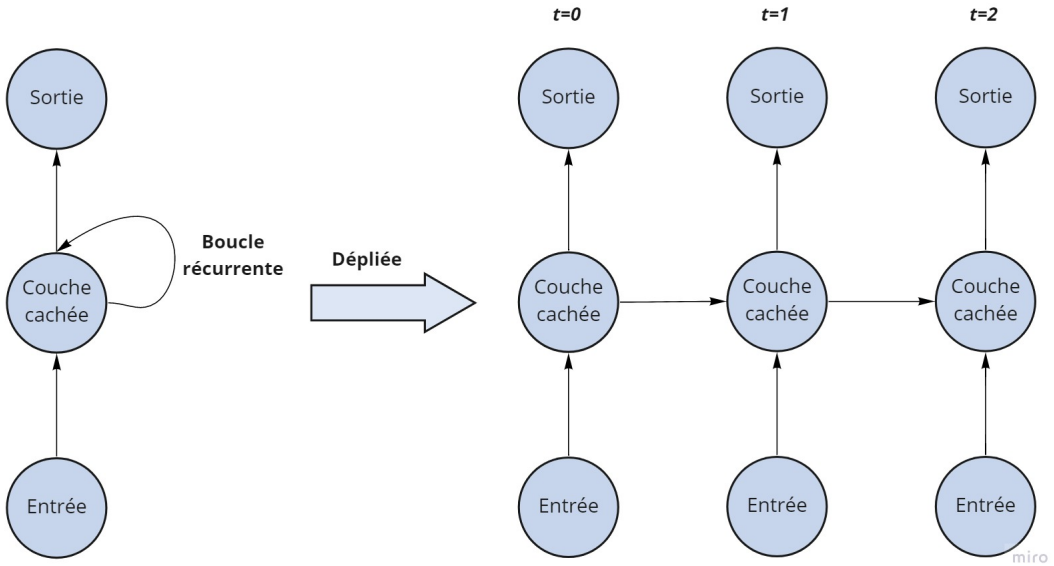

FIGURE 3.4 – Schéma d'un réseau de neurones récurrents [21]

les réseaux de neurones récurrents sont adaptés à plusieurs tâches de traitement automatique du la langue, notamment celles qui consistent à prédire une information séquentielle, c'est à dire quand les données forment une suite et ne sont pas indépendantes les unes des autres, comme la traduction automatique [28] et la reconnaissance vocale [32]. Cependant, ces réseaux peuvent rencontrer des difficultés à apprendre les dépendances quand il s'agit des séquences relativement longues. En effet, au bout d'un certain nombre d'itération, le RNN commence à « oublier » parce qu'il est capable de mémoriser que le passé dit proche. Ceci conduit à deux phénomènes

(problèmes) : dissipation du gradient (vanishing gradient) et l'explosion du gradient (exploding gradient). Pour répondre à ce problème, d'autres architectures ont été proposées, comme les LSTM (*Long Short Term Memory*)

3.3.3 Long short-term memory

Les réseaux de neurones Long short-term memory (LSTM) [29] ont été proposé comme une extension pour les réseaux neurones récurrents. Il s'agit d'une architecture très similaire à l'architecture des RNN mais un peu modifiée, cependant les unités d'un LSTM sont composées d'une mémoire avec des portes logiques (d'entrée, de sortie et d'oubli) qui permettent au modèle de régulariser le flux d'information en apprenant quelles sont les informations pertinentes qui doit conserver ou oublier sur de longue période de temps. L'idée principale des LSTM est de permettre au réseau de décider de stocker ou d'oublier certaines informations passées afin de pouvoir donner du poids aux informations les plus importantes lors d'une étape actuelle. Par conséquent, ce modèle à permet de surmonter les problèmes de gradient évoqués précédemment.

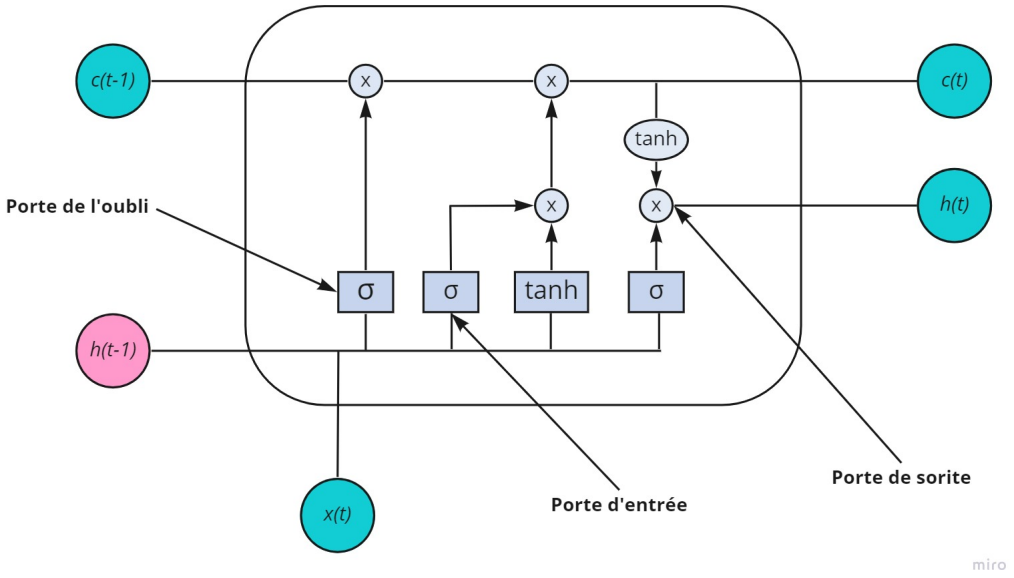

FIGURE 3.5 – Architecture d'une cellule LSTM [55]

3.3.4 Réseaux de neurones convolutifs

Les réseaux de neurones convolutifs ou *Convolutional Neural Network* en anglais (CNN), est un type de réseaux de neurones multi-couches inspirés des travaux de Hubel et Wiesel [31] sur le fonctionnement du cortex visuel des animaux. Ce type de réseau spécial est généralement utilisé quand les données d'entrée sont structurées sous la forme d'une grille,

une image par exemple. L'architecture d'un réseau CNN est construite par la succession de bocs de traitement de convolution et de pooling, d'une ou plusieurs couches cachées, et une couche de sortie. L'opération de convolution consiste à appliquer un filtre ou « kernel » (également appelé matrice de convolution ou noyau). Il s'agit de multiplier chacun des pixels de la matrice de données d'entrée par la matrice de convolution afin de produire une carte de caractéristiques (en anglais features map). Après une convolution, on associe généralement un sous-échantillonnage (pooling) dans le but de réduire quantité d'informations à apprendre . Le pooling ou le sous-échantillonnage en français, est une méthode qui consiste à prendre une large matrice convolée (grille d'informations) et d'en réduire la taille en gardant uniquement les informations importantes.

Aujourd'hui il existe différentes implémentations de réseaux de neurones convolutifs en fonction du problème à résoudre et le domaine d'application notamment le traitement du texte [38].

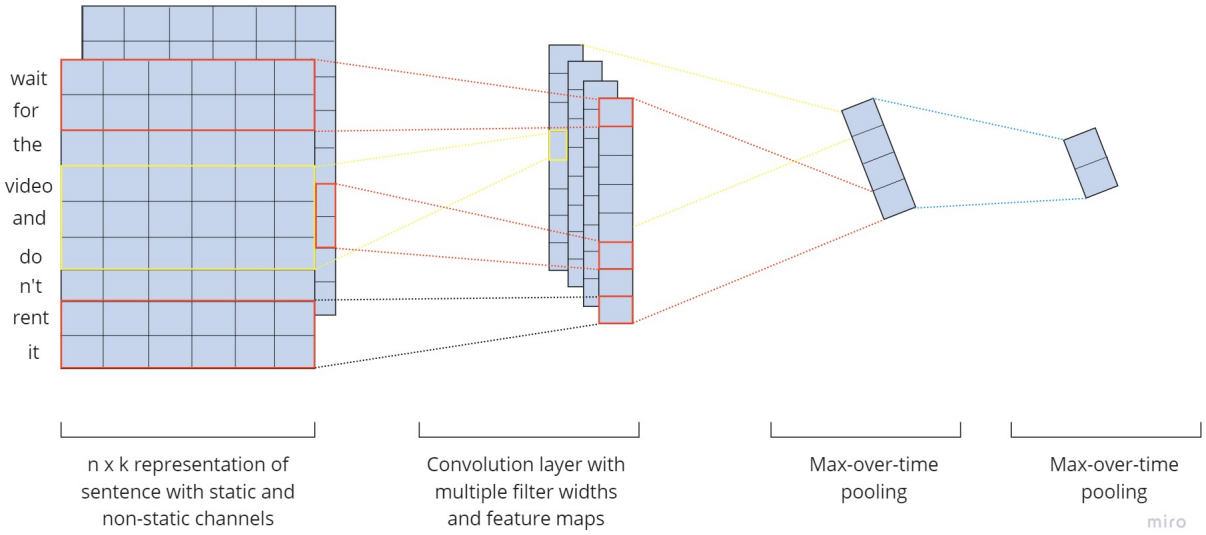

FIGURE 3.6 – Architecture d'un réseau CNN pour une analyse de phrases proposé par Kim Y [38]

3.4 Travaux connexes

Il existe plusieurs travaux qui ont été réalisés dans le domaine de l'analyse des sentiments, mais la majorité ont été fait pour la langue anglaise. Dans cette section, nous présentons, l'état de l'art de quelques travaux réalisés dans le domaine de l'analyse d'opinions en anglais et en d'autre langues comme le français et le dialecte algérien.

3.4.1 Analyse des sentiments en anglais

Dans le but de classer des tweets relatifs aux services de vol de six compagnies aériennes américaines selon la polarité des sentiments, positives, négatives et neutres, Monika et al. [52] ont exploré deux modèles de word ebmedding (Word2Vec et Glove) en utilisant des méthodes d'apprentissage profond. Il s'agit des réseaux de neurones récurrentes (RNN) ainsi que les réseaux de neurones Long short-term memory (LSTM). Pour ce travail, ils ont utilisé 14640 tweets de six compagnies aériennes différentes comme Virgin America, United, US Airways, Delta and Southwest. Où 80% de données ont été utilisées pour l'apprentissage, et 20% pour le test. Les résultats ont été significatifs, ils ont montré une meilleure précision de classification.

Sharma et al. [72] ont fait une analyse des sentiments pour les phrases courtes via le réseau de neurones convolutifs (CNN) en utilisant la méthode de vectorisation Word2Vec après avoir nettoyer leurs données. les vecteurs de mots générés par le modèle Wor2Vec ont était utilisés en entrée de la couche du CNN. Les auteurs ont trouvé que l'extraction des caractéristiques des phrase courtes via Word2Vect et CNN donne de meilleurs résultats. Le modèle proposé dans cette étude a donné une précision de 99.07% pour les données d'apprentissage, et 82.19% pour les données de test.

Sindhu et al. [74] ont proposé un modèle supervisé pour l'identification de la polarité par aspect basé sur un modèle LSTM à deux couches et Word2Vec pour le word embedding. La première couche sert à extraire les aspects décrits dans le texte, alors que la deuxième est faite pour prédire la polarité (positive, négative ou neutre) de ces aspects. Le modèle a été testé sur un ensemble de données étiquetées manuellement construit à partir des commentaires des étudiants à l'université, ainsi que sur un dataset standard SemEval-2014. Cependant, malgré la simplicité de l'architecture du modèle proposé, l'étude a montré que le système a réussi à atteindre de bons résultats : une précision de 91% pour la tâche d'extraction des aspects, et 93% pour la détection de la polarité.

Dans les travaux de Minaee et al [49], un ensemble de méthodes a été proposé, les auteurs ont combiné deux modèles, un LSTM bidirectionnel (Bi-LSTM) et un CNN. L'architecture LSTM est à deux couches, elle obtient en entrée le word embedding généré par le modèle Glove. Les deux modèles ont été entraînés séparément puis le score de probabilité moyen des sorties des deux modèles a été calculé pour avoir le score final de la prédiction. Dans cette étude deux datasets ont été utilisés : IMDB dataset qui contient des critiques de films, et Stanford Sentiment Treebank2 (SST2). Les résultats ont montré que l'ensemble des deux méthodes à attient le meilleur résultat (90% avec le IMDB et 80.5% avec SST2) par rapport à CNN (89.3% avec IMDB et 80.2% avec SST2) et le LSTM (89% avec IMDB et 80% avec SST2).

3.4.2 Analyse des sentiments en français

Kane et al. [34] ont proposé un modèle ABSA (Aspect-based sentiment analysis) pour détecter les aspects abordés dans un texte et le sentiment associé à chacun d'eux. Dans leur

travail, les auteurs ont décidé de combiner trois modèles, CNN (Convolutional Neural Network), Bidirectionnel LSTM (Long Short-Term Memory) et Champ aléatoire conditionnel (Conditional Random Field CRF). Le modèle proposé a montré une grande performance sur un jeu de données en français (SemEval2016 French dataset) au point où il a surpassé les modèles de l'état de l'art pour l'ABSA en français.

Baccouche et al. [7] ont proposé une technique d'étiquetage automatique pour les tweets liés à la santé dans trois langues différentes : anglais, français et arabe. En outre, ils ont appliqué des modèles d'analyse des sentiments tels qu'un modèle CNN, RNN et un modèle LSTM pour classer les tweets en deux et trois classes. L'ensemble de données utilisé contenait à la fois un ensemble de données spécifique au domaine de la santé, et un autre ensemble de domaine non spécifique, il s'agit de datasets de Amazon, IMDB, Yelp and ASTD. La technique d'étiquetage automatique a été utilisé pour prétraiter les tweets liés au domaine de la santé afin de détecter les annotations à l'aide de la connaissance du domaine, du traitement du langage, et des dictionnaires du lexique des sentiments. L'évaluation du modèle a montré que le LSTM-RNN a surpassé les modèles de l'état de l'art pour les données en anglais et en arabe. Le modèle a atteint une précision de 98% sur l'ensemble de données en anglais, une précision de 89% pour les données en français, et une précision de 83% pour les données en arabe.

Afin de faire une analyse de sentiment au niveau document sur des article de journaux français, Rhanoui et al. [67] ont opté pour une solution qui combine un modèle CNN et un Bidirectionnelle LSTM en utilisant Doc2vec comme technique de word embedding, cette dernière est adaptée pour créer une représentation numérique d'un document, quelle que soit sa longueur. Le modèle CNN-LSTM a été comparé ensuite aux modèles CNN, LSTM, Bi-LSTM et CNN-LSTM avec Word2Vec et Doc2Vec. Le modèle CNN-BiLSTM a surpassé les autres modèles avec une précision de 90.66%.

Dans le cadre de la campagne d'évaluation DeFT 2018 portant sur l'analyse d'opinion dans des tweets en français, AL Minard et al. [50] ont participé à 3 des 4 tâches de la campagne : la classification des tweets selon s'ils concernent les transports ou non, la classification des tweets selon leur polarité et l'annotation des marqueurs d'opinion et de l'objet à propos duquel est exprimée l'opinion. Pour La tâche 2, il s'agit de la classification des tweets concernant les transports selon leur polarité (positf, négatif, neutre ou mixposneg), pour cela, ils ont expérimenté deux méthodes d'apprentissage reposant sur des fondements différents, et sur des représentations différentes des données : le boosting d'arbres de décision et les réseaux de neurones récurrents (RNN et Bi-LSTM). Pour la tâche 2, la meilleure performance est obtenue par le modèle BiLSTM avec une F-mesure de 81,31%, 0,98 points de moins que le meilleur modèle de la compétition. La méthode de plongement de mots utilisé pour représenter les mots du texte est FastText.

3.4.3 Analyse des sentiments en dialecte algérien

Mazari et al. [47] ont appliqué différents algorithmes classiques de machine learning (SVM, Naïve Bayes) ainsi que des modèles de deep learning comme le CNN et le RNN sur un corpus contenant des commentaires en arabe dialectal algérien collectés des réseaux sociaux tels que Facebook, Twitter et YouTube durant la période du Hirak en 2019. Le corpus contenait 7800 commentaires sur des sujets politques. L'annotation des commentaires a été faite par un groupe de doctorants qui ont attribué des classes aux commentaires de trois manières : Opinion positive, négative ou neutre. Les résultats obtenus ont été une précision de 63.28% avec le modèle CNN, et 60.97% avec le RNN.

Dans le but d'analyser les sentiments exprimées dans des textes écrits en dialecte algérien, Moudjari et Akli-astouati [53] ont expérimenté plusieurs méthodes bien connues et couramment utilisées pour cette tâche. En effet, ils ont proposé une étude comparative de différents modèles, allant des modèles classiques d'apprentissage automatique tels que les machines à vecteurs de support (SVM), le Naïve Bayes, le Forêt d'arbre décisionnels (Random Forest), en passant par l'approche lexical, en arrivant aux modèles d'apprentissage profond parmi les plus populaires de l'état de l'art (CNN et LSTM). Pour l'apprentissage de représentation de mots, les auteurs ont utilisé plusieurs méthodes également, comme la couche Embedding, le modèle AraVec pré-entraîné, FastText et BERT. Concernant l'ensemble de données, ils ont combiné deux datasets annotés déjà existants. Le premier dataset est constitué de publications et de commentaires collectés depuis des pages Facebook algériennes, alors que le deuxième est un corpus de tweets d'opinion et d'émotions collectés via l'API de Twitter. La meilleure performance de classification a été obtenue par le CNN avec word embedding intégré, tandis que les résultats les moins performant ont été obtenus avec LSTM avec la même méthode de plongement de mots.

Dans le travail de Abdelli et al. [2], deux modèles ont été utilisés pour l'analyse des sentiments des textes écrits en arabe moderne et dialecte algérien, un SVM, ainsi qu'un modèle de deep learning qui est le LSTM. Leur approche a été composé de trois phases principales : (I) phase de collecte des données, où ils ont collecté un grand nombre de commentaires annotés manuellement, (II) phase de prétraitement des données collectées, et (III) phase d'apprentissage et de test du modèle (le modèle SVM et le modèle LSTM). Dans la première phase, les auteurs ont collecté deux corpus, le premier a été utilisé pour entraîner le modèle Word2Vec (le modèle CBOW), tandis que le deuxième a été utilisé comme un jeu de données pour former le modèle pour la tâche d'analyse des sentiments. Pour le word embedding, ils ont utilisé deux méthodes différentes : Tf-idf avec SVM, et Word2Vec avec LSTM. Selon cette étude, le modèle SVM (précision 89%) a surpassé le modèle LSTM (précision 79%).

Soumeur et al [76] ont proposé deux approches pour une analyse de sentiments des commentaires des utilisateurs algériens rédigées en dialecte algérien collectés à partir de 20 pages de marques en Algérie sur Facebook. Tout d'abord, ils ont commencé par la collection

et l'annotation de leur corpus. Ils ont réussi à recueillir plus de 100 000 commentaires, parmi lesquels 25 475 ont été prétraités et annotés comme positifs, négatifs ou neutres. Ensuite, ils ont décidé d'implémenter deux modèles d'apprentissage profond : Un réseau de neurones multicouches (MLP), ainsi qu'un réseau de neurones convolutifs (CNN). Les deux modèles ont obtenu de bons résultats (81.6% et 89,5'% respectivement), avec une précision légèrement meilleure pour le modèle CNN avec une représentation de mots BoW.

3.5 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la notion de l'apprentissage profond dans l'état de l'art de l'analyse des sentiments. Nous avons dans un premier temps présenté la notion de l'apprentissage profond. Nous avons décrit, dans un deuxième temps, les différentes méthodes de construction d'embeddings. Ensuite nous avons abordé la description théorique des réseaux de neurones, où nous avons présenté quelques différentes architectures neuronales telles que les réseaux de neurones récurrents de type RNN et LSTM, ainsi que les réseaux de neurones convolutifs (CNN). À la fin, nous avons recensé et établi différents travaux d'analyse de sentiments qui représentent l'état de l'art des méthodes neuronales pour cette tâche.

Dans le reste de ce mémoire, nous allons présenter notre approche proposée pour l'analyse des sentiments.

Corpus	Auteurs	Word Embedding	Modèle	Dataset	Résultats
Anglais	Monika et al. (2019)	Word2vec, Glove	RNN, LSTM	Twitter US Airline	Train> 80% Validation75%
	Sharma et al. (2020)	Word2vec	CNN	IMDB	99,07% (Train) 82,19% (Test)
	Sindhu et al. (2019)	Word2vec	LSTM	SemEval-2014 DSC*	93%
	Minaee et al. (2019)	Glove	Bi-LSTM + CNN	IMDB, SST2	90% (IMDB) 80,5% (SST2)
	Kane et al. (2021)	-	CNN+BILSTM+CRF	SemEval-2016	77,2%
Français	Baccouche et al. (2018)	Word2vec	CNN, RNN, LSTM	Amazon, IMDB, Yelp, ASTD, DSC	92%
	Rhanaoui et al. (2019)	Doc2vec, Word2vec	CNN+BILSTM	Des journaux français	90,66%
	AL Minard et al. (2018)	FastText	Boosting d'arbre de décision, RNN, BILSTM	-	81,31%
Dialecte	Mazari et al. (2021)	-	CNN, RNN	DSC	63.28% (CNN) 60.97% (RNN)
	Moudjari et Akli-astouati. (2020)	FastText, Aravec, Bert, Couche Embedding	CNN	Mataoui, Twifil, Lema (Mataoui+Twifil)	79%
	Abdeli et al. (2019)	Word2vec, Tf-idf	LSTM, SVM	DSC	79% (LSTM) 89% (SVM)
	Soumeur et al. (2018)	BoW	MLP, CNN	DSC	81,6% (MLP) 89,5% (CNN)

TABLE 3.1 – Tableau récapitulatif des travaux connexes

Chapitre 4

Conception

4.1 Introduction

Dans ce chapitre, nous proposons différentes approches pour l'analyse des sentiments en anglais, français et dialecte algérien. Nous présentons, tout d'abord, le schéma global du notre système qui se compose de plusieurs modules : Prétraitement, extraction de caractéristiques et classification. Ensuite, nous exposons les différents base de données que nous avons utilisées pour chaque langue. Enfin, nous présentons les différentes métriques d'évaluation pour mesurer la performance d'un système d'analyse de sentiments.

4.2 Schéma global du système

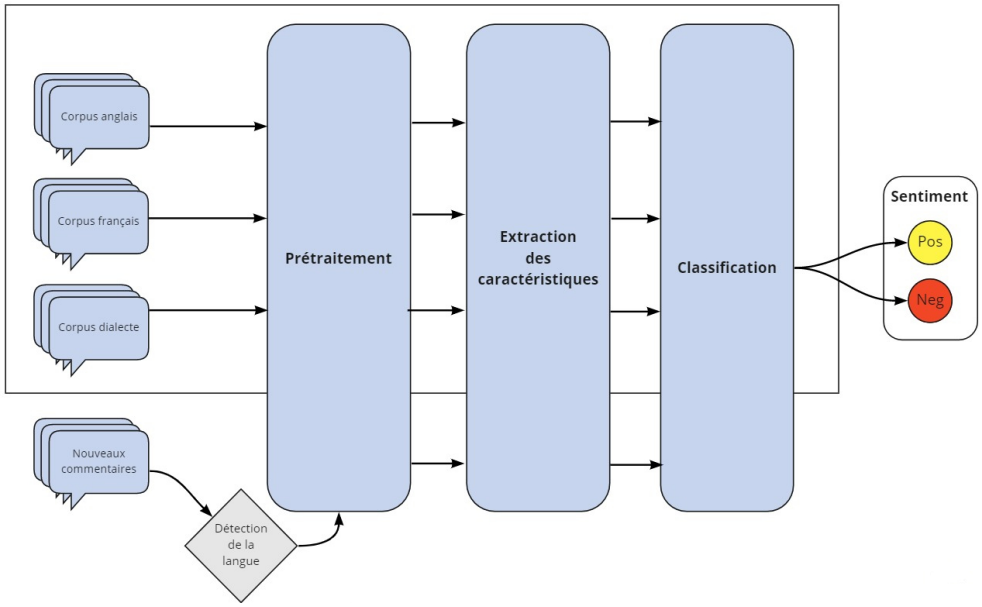

FIGURE 4.1 – Schéma global de conception de notre système d'analyse d'opinion

La Figure 4.1 représente le schéma global de conception de notre système d'analyse d'opinion. La tâche principale à laquelle ce système vise à répondre est de déterminer, pour un commentaire donné, sa classe d'opinion (polarité) : positive ou négative. Pour cela, le système proposé est constitué de trois modules principaux que nous avons vu déjà dans le chapitre 2. Il s'agit d'un module de prétraitement, d'extraction des caractéristiques et des modèles de classification. Cette section présente une description des techniques de chacun des modules réalisés ou intégrés dans ce système.

4.2.1 Module de prétraitement

Pour l'ensemble des techniques utilisées pour le prétraitement des commentaires, nous avons pris en considération la nature du texte de chaque corpus, et la langue cible de chaque modèle, car chaque langue a sa propre structure et ses propres constructions grammaticales. En effet, pour les applications de NLP, la langue arabe, nécessite des étapes de prétraitement et de normalisation différentes que l'anglais ou le français. Par exemple, les notions de lettre majuscule et lettre minuscule n'existent pas dans la langue arabe : l'écriture est donc monocamérale. Pour ces raisons, ces techniques de pré-traitement doivent être adaptés d'une langue à une notre selon le besoin.

Pour le premier corpus dont les textes sont en anglais, nous avons appliqué les diverses techniques de prétraitement classiques, tels que le « lowercasing » pour mettre tout le texte en minuscule, la suppression des tags HTML et les URLs. Nous avons également supprimé les mots et les symboles non alphabétique comme les chiffres, les ponctuations, etc. Certaines de ces opérations ont causées des espaces multiples dans le texte, ce qui nous a poussé à traiter ce problème aussi, donc nous avons fait en sorte d'éliminer ces espaces multiples. Cependant, nous avons décidé de ne pas supprimer les mots vides (stopwords) parce qu'ils peuvent être utiles pour la classification. La Figure 4.2 ci-dessous montre un exemple tiré de corpus anglais, avant et après le prétraitement.

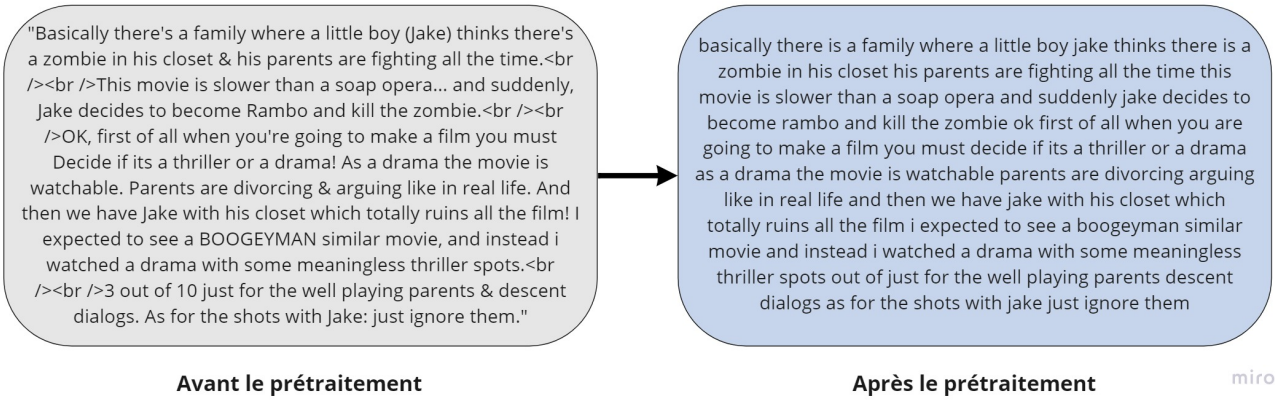

FIGURE 4.2 – Un exemple d'un texte avant et après le prétratiement

Le dernier corpus que nous avons utilisé dans ce travail est un corpus qui contient des énoncés textuels écrits en arabe dialectal algérien. L'arabe dialectal, couramment utilisé sur les plateformes de médias sociaux, fait l'objet d'un vrai challenge pour les outils TALN, car elle partage pratiquement des caractéristiques de l'arabe standard moderne, en plus de ses propres caractéristiques. C'est pourquoi faire face à une telle langue morphologiquement riche, avec son propre lexique, ses règles linguistiques appropriées qui ne suivent aucune règle grammaticale, ses idiomes et ses proverbes particuliers, représente un problème majeur auquel nous sommes confrontés lors du traitement des données des réseaux sociaux. En effet, les utilisateurs des réseaux sociaux tendent à commettre des fautes d'orthographe, utiliser des abréviations, des répétitions, etc. Aussi, chaque utilisateur translittère le mot à sa façon. Par exemple la phrase en dialecte algérien « ما عجبنيش » (en français : il ne m'a pas plu ou je ne l'ai pas aimé) peut être écrite par les utilisateurs des réseaux sociaux de différentes façons : « ما عجبنيش », « ماعجبنيش », « معجبنيش ».

Dans ce cas, et dans le cadre de ce travail, nous avons opté pour un prétraitement pas très compliqué pour le corpus de dialecte, donc nous avons d'abord supprimer les différents types de ponctuations qui peuvent se retrouver dans le texte. Nous avons éliminé également tous les mots non écrits en arabe. Ensuite, nous avons appliqué quelques techniques de normalisation dans le but de gérer les variations de certains mots et aussi éliminer la répétitions (des mots avec une lettre qui se répète comme par exemple le mot « بزاااااف »). La figure 4.3 montre le résultat des techniques utilisées pour le prétraitement sur un exemple tiré de corpus dialecte.

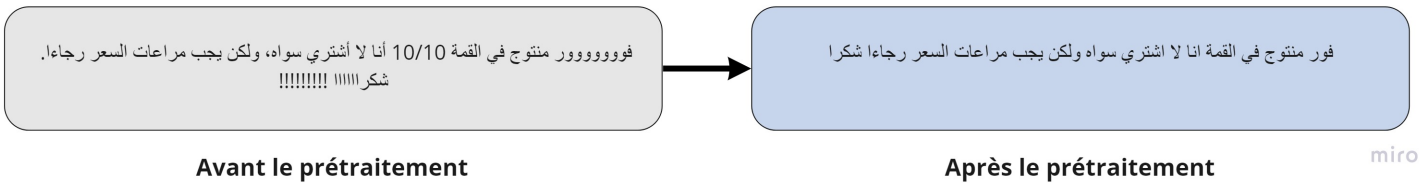

FIGURE 4.3 – Un exemple d'un texte en arabe dialecte avant et après le prétratiement

4.2.2 Module d'extraction de caractéristiques

Dans le chapitre précédent, nous avons présenté différentes méthodes de word embedding tels que Word2Vec, Glove, FastText, BERT, etc. Dans notre travail, nous avons expérimenté quelques-unes, notamment Word2Vec, Glove et FastText. Pour cela, nous avons choisi d'utiliser des modèles pré-entraînés pour deux raisons : La première est le fait que ces modèles pré-entraîner ont été entraînés sur des ensembles de données massifs, des corpus très large avec des milliards de mots différents, car pour apprendre de bonnes représentations de mots, il est nécessaire d'entraîner le modèle avec beaucoup de données [1], ce qui n'est toujours possible,

1. https ://code.google.com/archive/p/word2vec/

comme nous allons le constaté plus loin dans ce travail avec le cas de dialecte algérien. La deuxième raison, tient au fait que nous pouvons entraîner notre propre word embedding, et donc avoir un plongement de mots spécifique à notre corpus, ce qui nous permet normalement d'avoir un modèle plus performent sur nos données de domaine spécifique, mais c'est un peu plus compliqué, puisqu'il y a un nombre important de paramètres à prendre en considération [48], ce qui nécessite beaucoup de ressources matérielles et beaucoup de temps pour s'entraîner.

Word2Vec : La première méthode que nous avons utilisé pour l'anglais est Word2Vec, le modèle pré-entraîné de Google. Il a été entraîné sur une partie de l'ensemble de donnée de Google news, environ 100 milliards de mots, ce qui a donnée des vecteurs de 300 dimensions pour 3 millions de mots et phrases (groupes de mots).

Glove : La deuxième méthode que nous avons testé concerne le modèle pré-entraîné de plongement de mots proposé par l'université de Stanford, Glove. Nous avons choisi le modèle de 300 dimensions comme taille des vecteurs (le dossier téléchargé contenait quatre fichiers pour quatre modèles, 50, 100, 200 et 300 dimensions) pré-entraînés sur 6 milliards de mots provenant d'un corpus composé de Wikipedia 2014 et Gigaword 5 [2].

FastText : Pour traiter le cas de dialecte, nous avons opté pour une solution basé sur le modèle développé par Facebook, FastText. La motivation derrière ce choix réside dans le fait que FastText fonctionne selon le principe de représentation vectorielles de n-grammes de caractères, c'est-à-dire qu'un mot est représenté par la somme des vecteurs de ses n-grammes, ce qui est assez différent de Word2Vec et Glove, qui traitent chaque mot comme la plus petite unité. L'avantage de l'utilisation de FastText dans ce cas, est qu'il est capable de générer de meilleures incorporations de mots pour des mots rare où même des mots qui n'existent pas dans le corpus d'entraînement. En effet, ce principe peut être très utile quand nous avons à faire à une langue très riche et très complexe comme le cas de la langue arabe où un dialecte.
De ce fait, et puisque nous avons supposé que le word embedding entraîné sur nos données du domaine spécifique performent mieux, nous avons d'abord entraîné notre propre modèle FastText en utilisant notre propre corpus. Ensuite nous avons décidé d'expérimenter un modèle de dimensions 300 pré-entraîné sur un ensemble de données composé de Wikipedia et Common Crwal. [3]

4.2.3 Module de Classification

Afin de réaliser notre système d'analyse de sentiments, nous avons expérimenté différentes architectures de réseaux de neurones, à savoir réseaux de neurones convolutifs et des réseaux de neurones Long short-term memory (LSTM). Selon notre étude de l'état de

2. https ://nlp.stanford.edu/projects/glove/
3. https ://fasttext.cc/docs/en/crawl-vectors.html

l'art, ces architectures sont très utilisées et elles ont fait leurs preuves pour l'analyse d'opinion, spécialement en anglais. Nous détaillons dans la suite nos architectures pour chaque modèle.

4.2.3.1 Architecture Bi-LSTM

Nous avons présenté dans le chapitre précédent, dans la section 3.3.3, les réseaux de neurones de type Long short-term memory (LSTM). Nous avons vu que cette famille de modèles est particulièrement adaptée pour analyser des données séquentielles, en raison de leurs mémoire interne qui permet de maintenir l'information pour de longue périodes de temps, ce qui est intéressant, car avoir une mémoire du passé peut aider à prendre de bonnes décisions à un instant t. Cependant, il est parfois intéressant et utile de connaitre le contexte futur et regarder les observations futures par rapport à un instant t. Par conséquent, pour surmonter cette limitation du LSTM qui vient du fait qu'il ne prend en compte que les informations des états précédents, un modèle bidirectionnel a été proposé [71].

Un LSTM bidirectionnel (Bi-LSTM) est un réseau récurrent bidirectionnel de type LSTM. Il consiste à utiliser deux LSTM en parallèle afin de prendre en compte à la fois des informations passées (contexte à gauche) et future (contexte à droite) par rapport à un instant t de tous les éléments de la séquence d'entrée. La Figure 4.4 montre une illustration d'un modèle Bi-LSTM qui est constitué d'une première couche LSTM avant qui parcoure la séquence allant de gauche à droite, et une deuxième couche LSTM arrière qui parcoure la même séquence allant de droite à gauche. Dans un problème comme le nôtre, où nous voulons faire une analyse d'opinions, utiliser l'information après et avant les données étudiées, afin de saisir le contexte des mots de chaque direction est important. Par conséquent, et selon les travaux consultés dans l'état de l'art et les résultats obtenus, ce modèle représente un bon choix à tester.

Nous avons donc décidé de tester une architecture pour l'analyse des sentiments en utilisant deux couches Bi-LSTM empilées l'une sur l'autre. Notre architecture proposée prend en entrée des vecteurs d'embeddings générés par l'une des méthodes de plongement de mots pré-entrainés (et personnalisés) tels que Word2Vec, Glove et FastText. Nous utiliserons cette matrice pour initialiser notre look-up table. Nous avons utilisé également un dropout entre chaque couche qui compose l'architecture, c'est-à dire à chaque itération nous avons des neurones qui sont des désactivés. Enfin, la dernière couche de notre modèle est une couche entièrement connectée et représente sa couche de sortie. Appliquant une fonction d'activation à cette couche de sortie, permet de prédire la polarité du commentaire fournit en entrée. L'architecture complète du notre modèle est présentée par la Figure 4.5.

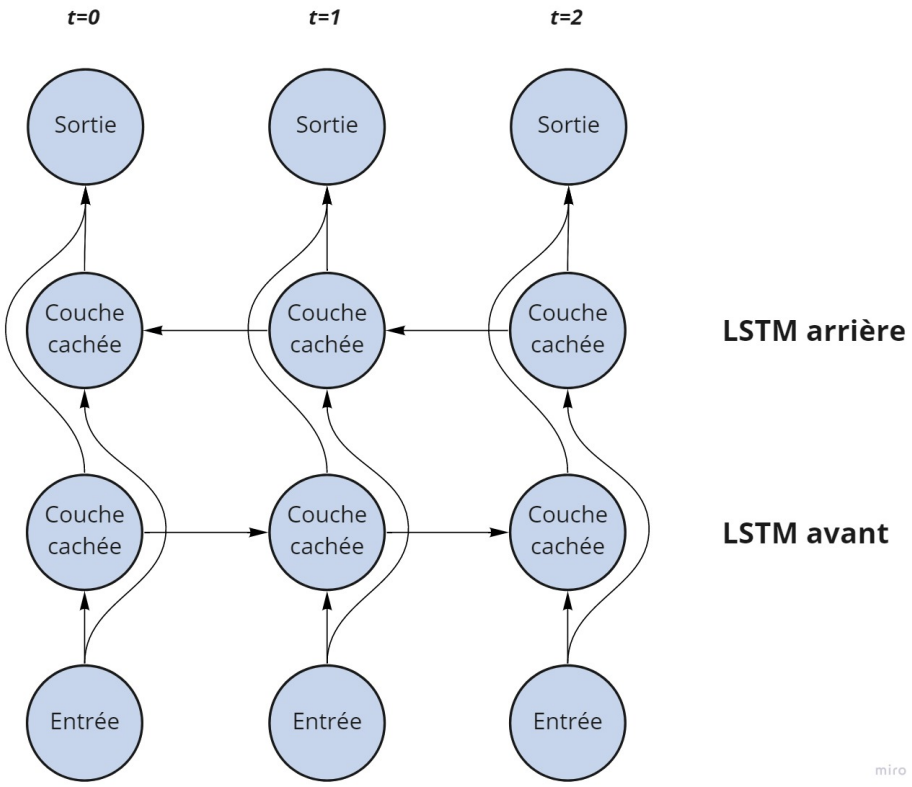

FIGURE 4.4 – Schéma d'un Bi-LSTM [84]

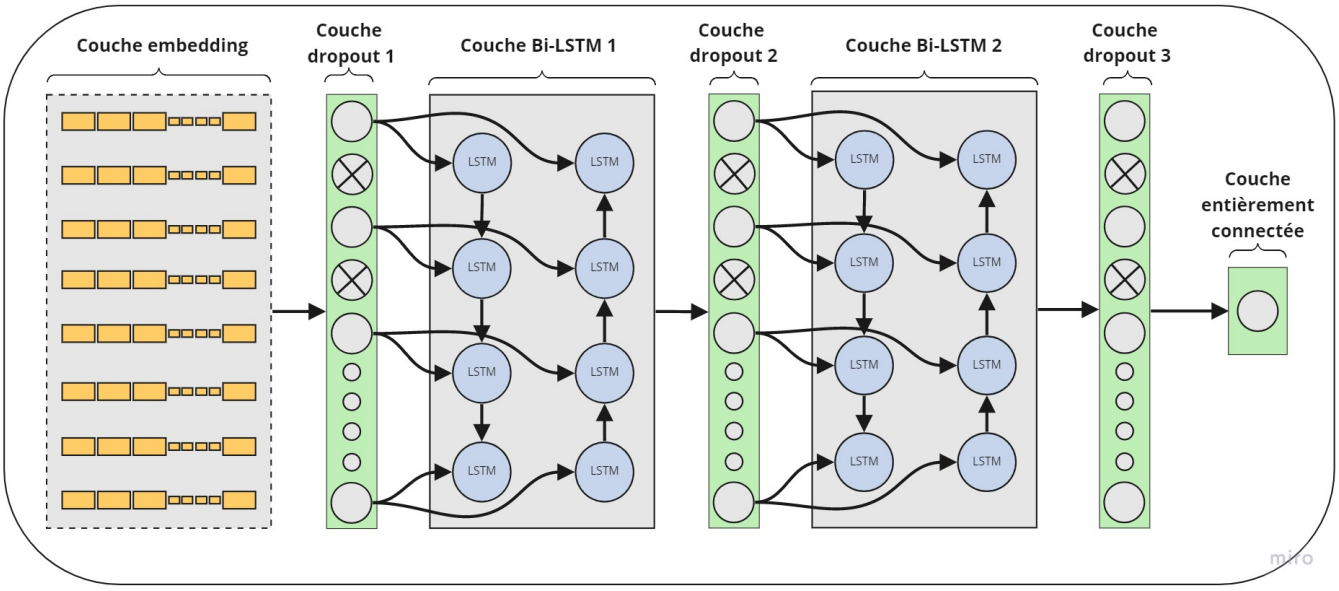

FIGURE 4.5 – Architecture de notre modèle Bi-LSTM

4.2.3.2 Architecture CNN

Le deuxième modèle que nous avons utilisé est un réseau de neurones convolutif (CNN). Comme nous avons vu dans notre étude de l'état de l'art, des architectures neuronales de type CNN ont également été largement utilisées pour la tâche d'analyse des sentiments. Des architectures comme celles qui ont été proposés par Kim Y [38] et Zhang et Wallace [88] ont fait leurs preuves dans différentes tâches liées au TALN. De ce fait, nous avons décidé de tester une architecture similaire dans notre travail (Figure 4.6).

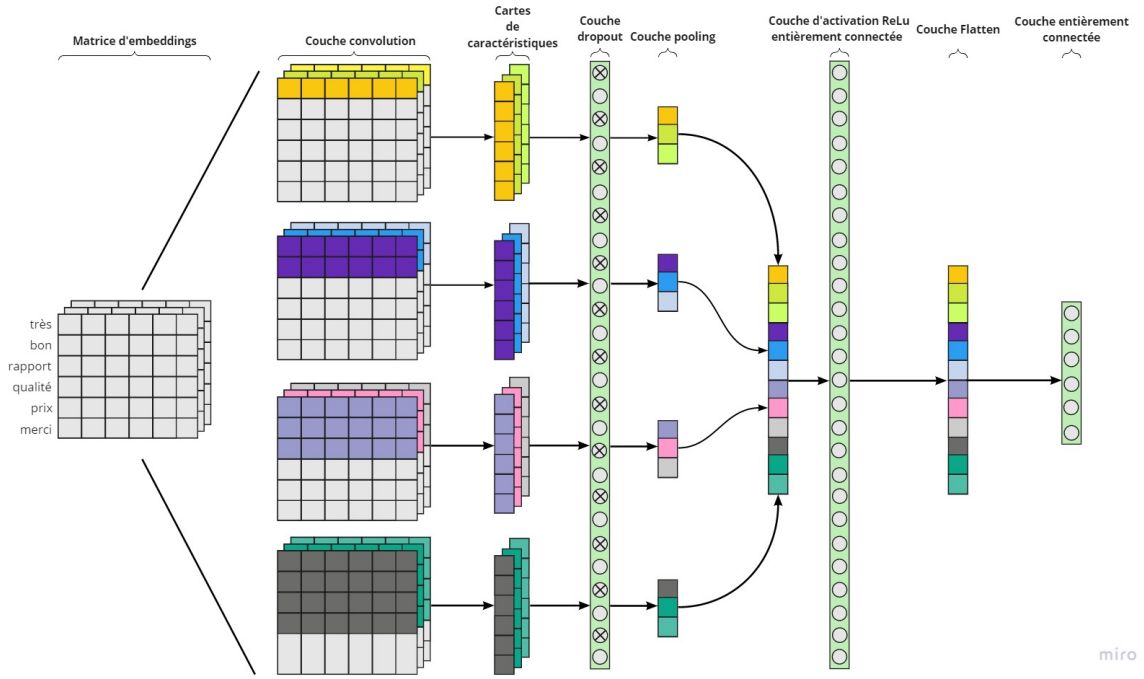

FIGURE 4.6 – Architecture globale de notre modèle CNN [88]

Pour notre système, nous proposons une architecture composée d'une succession de couches convolution et de pooling, suivi par des couches neurones entièrement connectées. L'entrée de ce modèle correspond à une séquence de n mots, où chaque mot est représenté par un vecteur d'embeddings de dimension d, ainsi, le CNN prend en entrée une matrice d'embeddings de dimension $n \times d$. La couche convolution est utilisée pour extraire l'ensemble de caractéristiques ou *features* à partir de la matrice d'embeddings. Il s'agit d'une application d'une convolution de filtres, dont la taille de la fenêtre de chaque filtre est précisée. La couche de pooling (couche d'agrégation en français) quant à elle est utilisée régulièrement entre les couches de convolution et a pour objectif de réduire la taille des cartes de caractéristiques produites par les convolutions pour but de conserver uniquement les caractéristiques pertinentes. Plusieurs méthodes existent pour faire cette tâche et nous utilisons celle qui applique l'opération max au résultat de chaque filtre. Puis, nous avons ajouté une couche d'activation Relu, puis un dropout avant d'ajouté une autre couche appelée Flatten. Enfin, une couche de neurones entièrement connectés où nous

appliquons une fonction d'activation afin d'obtenir les probabilités d'appartenance à chaque classe.

4.3 Collecte de données

Pour réaliser notre système, nous avons besoin de données pour entrainer et tester les modèles de classification que nous avons proposés. Pour cela, un corpus contenant de grandes quantités de données de qualité, dans notre cas, des commentaires étiquetés (annotés) selon leurs polarité, positive ou négative, est souvent nécessaire. Dans ce travail, nous nous situons dans le cadre de la classification binaire dans trois langues différentes, pour cela, nous avons sélectionné trois corpus en trois langues, deux corpus préexistants (anglais et français) et un troisième que nous avons nous même construit (dialecte). Les propriétés détaillées de ces corpus sont présentées dans la suite.

4.3.1 IMDB dataset

Le premier jeu de données que nous avons utilisé dans le cadre de ce travail provient de la base de critiques de film IMBD (Internet Movie Database). Il s'agit d'un ensemble de données de l'université de stanford [45] destiné spécialement à la tâche de classification des sentiments binaires. Il contient 50 000 critiques (avis) de films tirés du site IMDB, dont 25 000 sont de polarité positive (1) et 25 000 de polarité négative (0). Ces données sont accessibles sur le site Stanford AI Lab [4]. Nous fournissons dans ce qui suit quelques exemples de cet ensemble de données :

Exemple 1

"Not a movie for everyone, but this movie is in my top 10. I am a lover of black comedy. With a cast including Richard Dreyfus (Vic), Jeff Goldblum (Mick), Larry Bishop (Nick) and Gabriel Byrne (Ben 'Brass Balls' London) in the leads, the lines can't help but be dry. The supporting cast is nearly dead center. Counting the minor flaws in the movie : Ellen Barkin's make-up gave her face has a washed out look ; there were a couple of gimme cameos by Joey Bishop and Richard Pryor that served no purpose, and Michael J. Pollard's screen time was too short. Over all, the cast was just incredible without egos to wreck a fine script. If you have seen Larry Bishop's (writer, director) film, Underworld (a dark crime flick), you will enjoy this one. His next outing (writer, director, actor) is Hell Ride with Michael Madsen and Quentin Tarantino."

Exemple 2

"I went and saw this movie last night after being coaxed to by a few friends of mine. I'll admit that I was reluctant to see it because from what I knew of Ashton Kutcher

4. https ://ai.stanford.edu/ amaas/data/sentiment/

he was only able to do comedy. I was wrong. Kutcher played the character of Jake Fischer very well, and Kevin Costner played Ben Randall with such professionalism. The sign of a good movie is that it can toy with our emotions. This one did exactly that. The entire theater (which was sold out) was overcome by laughter during the first half of the movie, and were moved to tears during the second half. While exiting the theater I not only saw many women in tears, but many full grown men as well, trying desperately not to let anyone see them crying. This movie was great, and I suggest that you go see it before you judge."

Exemple 3

"Encouraged by the positive comments about this film on here I was looking forward to watching this film. Bad mistake. I've seen 950+ films and this is truly one of the worst of them - it's awful in almost every way : editing, pacing, storyline, 'acting,' soundtrack (the film's only song - a lame country tune - is played no less than four times). The film looks cheap and nasty and is boring in the extreme. Rarely have I been so happy to see the end credits of a film. The only thing that prevents me giving this a 1-score is Harvey Keitel - while this is far from his best performance he at least seems to be making a bit of an effort. One for Keitel obsessives only."

Exemple 4

"This show was an amazing, fresh & innovative idea in the 70's when it first aired. The first 7 or 8 years were brilliant, but things dropped off after that. By 1990, the show was not really funny anymore, and it's continued its decline further to the complete waste of time it is today. It's truly disgraceful how far this show has fallen. The writing is painfully bad, the performances are almost as bad - if not for the mildly entertaining respite of the guest-hosts, this show probably wouldn't still be on the air. I find it so hard to believe that the same creator that hand-selected the original cast also chose the band of hacks that followed. How can one recognize such brilliance and then see fit to replace it with such mediocrity ? I felt I must give 2 stars out of respect for the original cast that made this show such a huge success. As it is now, the show is just awful. I can't believe it's still on the air."

4.3.2 DzReviews dataset

Comme nous l'avons déjà souligné plus haut dans ce travail et à plusieurs reprises, la grande majorité des ressources et des systèmes qui existent dans le domaine du TALN sont destinés à l'anglais. Le manque ou l'insuffisance de données et des ressources linguistiques représentent un problème majeur dans ce genre de travaux. Ce problème est loin d'être exclusif à une langue ou une autre, comme l'arabe ou le français par exemple, le problème se retrouve avec plus ou moins d'importance pour toutes les langues moins dotées ou plus rares sur la Toile. De ce fait, mais à part l'anglais, presque toutes les langues ne disposent pas d'assez de corpus

de textes qui peuvent être utilisées pour entraîner des modèles de classification, et le problème s'aggrave pour les corpus de dialecte et les corpus spécialisés comme dans le cas de notre étude.

Par conséquent, dans le cadre de la partie de notre travail qui consiste à réaliser un modèle d'analyse de sentiments pour le dialecte algérien, nous avons été obligés de construire notre propre corpus de domaine. Etant donné que les corpus de dialecte algérien accessibles existants ne sont pas spécialisés [2, 46], c'est-à-dire trop généraux pour être adaptés à un domaine spécifique tel que le commerce. Un corpus spécialisé doit porter sur un domaine de connaissance ou une situation de communication particuliers. Le langage utilisé dans ce type de corpus doit être représentatif par rapport à la langue utilisé dans le domaine en question [19, 24]. En conséquence, nous avons donc constitué notre propre corpus, celui-ci sera détaillé dans la suite.

Aujourd'hui, Internet représente une importante source de données en ce qui concerne les avis des consommateurs sur différents produits et services, avec la possibilité d'accéder à ces données directement en utilisant quelques outils. Pour cela il existe deux approches : Le scraping et les APIs. Nous avons donc opté pour la première, le scraping.
Le « Scraping » est un terme anglais signifiant littéralement « grattage ». Appliqué au web (web scraping), le terme renvoie à une technique d'extraction du contenu de sites Web de manière automatisée via un programme ou un script. Il s'agit de parcourir les pages d'un site Web et extraire les données, les stocker et les transformer en d'autres formats plus exploitable (excel, csv, etc.) dans le but les réutiliser dans un autre contexte, comme dans notre cas par exemple, où nous avons utilisé cette technique pour construire un corpus issu des commentaires des consommateurs sur des pages de marques algériennes sur Facebook (Sosemie, Aquafine, Molfix, etc.).

Composition du corpus Le corpus DzReviews que nous proposons contient 1 900 critiques de produits et de services sous forme de commentaires écrits en arabe dialectal algérien. Chaque commentaire est associé à une polarité positive ou négative. L'annotation a été faite manuellement par nous-même. Le tableau 4.1 décrit la répartition des commentaires sur les polarités de classement.

	Polarité	
	Positive	Négative
Nombre de commentaires	1136	763

TABLE 4.1 – Distribution du corpus DzReviews

Dans le tableau 4.2 nous montrons quelques commentaires de cet ensemble de données avec leurs polarités.

Commentaire	Polarité
منتجات تستحق الثقة	Positive
هايل عندي عام ملي بديت نستعملو روعة	Positive
جربتو هايل بزاف يعطيكم الف صحة على هاد منتوجات في قمة الروعة	Positive
ذوقها سيء بصراحة	Négative
راهي عندي نصها لي يبغي يديها . ريحتها قاطعة بزاف مستحملتهاش	Négative
تكذبوا لقهاوي اكل عندهم نفس لڤو	Négative

TABLE 4.2 – Quelques exemples de commentaires dans le corpus DzReviews

4.4 Métriques d'évaluation

Afin d'évaluer la performance de notre système, l'usage d'un ensemble de métriques est primordial. Dans ce but, nous avons utilisé dans ce travail les métriques les plus utilisées dans la littérature d'analyse de sentiments. Il s'agit de la précision, le rappel, le F1-mesure ou F1-score et l'accuracy.

4.4.1 Précision

La précision représente le taux de prédictions exactes de chaque classe. Il s'agit d'un ratio entre le nombre d'instances (données) correctement classées et le nombre total d'instances classées. Elle est calculée comme suit :

$$\textbf{\textit{Précision}} = \frac{\text{nombre d'instances correctement classées}}{\text{nombre total de d'instances classées}} \tag{4.1}$$

4.4.2 Rappel

Le rappel permet de mesurer la capacité du modèle à classer correctement toutes les instances d'une classe donnée. Il représente le rapport entre le nombre d'instances correctement classées et le nombre total d'instances existantes au sein du corpus. Cette métrique se calcule comme suit :

$$\textbf{\textit{Rappel}} = \frac{\text{nombre d'instances pertinentes correctement classées}}{\text{nombre total de d'instances pertinentes dans le corpus}} \tag{4.2}$$

4.4.3 F-mesure

Le F1-mesure ou F1-score, une mesure qui représente la moyenne harmonique de la précision et du rappel. Elle est calculée comme suit :

$$\textbf{\textit{F1-mesure}} = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}} \tag{4.3}$$

4.4.4 Accuracy

La dernière mesure que nous avons utilisée est l'accuracy. Cette métrique est couramment utilisée comme alternative à la précision et le rappel pour évaluer un modèle d'une façon global. Il s'agit de nombre d'instances correctement classées sur le nombre total d'instances. Elle se calcule comme suit :

$$\textbf{\textit{Accuracy}} = \frac{\text{nombre d'instances correctement classées}}{\text{nombre total de d'instances}} \tag{4.4}$$

4.5 Conclusion

Dans ce chapitre, nous avons présenté un système neuronal pour l'analyse des sentiments en trois langues, anglais, français et dialecte algérien. Nous avons proposé deux modèles différents, un CNN et un BLSTM, avec différentes word embeddings, Word2Vec, Glove et FastText. Nous avons également présenté les trois corpus utilisés dans ce travail. Enfin, nous avons présenté les métriques d'évaluation que nous avons utilisés pour évaluer les modèles que nous avons réalisés. Dans le chapitre qui suit nous monterons l'utilisation des techniques présentées dans ce chapitre pour l'analyse des sentiments. Par la suite, nous présentons et discutons les différents résultats obtenus sur les différents jeux de données.

Chapitre 5

Implémentation et résultats

5.1 Introduction

Après avoir présenté les modèles proposés dans le chapitre précédent, dans ce chapitre, nous nous consacrons à l'implémentation et l'évaluation de notre approche pour l'analyse des sentiments. Nous commençons en premier lieu par la présentation de notre environnement d'expérimentations, en décrivant les différentes configurations, les langages de programmations et les ressources matérielles utilisées. Dans un second temps, nous allons expliquer pour chaque modèle les différents hyperparamètres que nous avons utilisés afin de dérouler nos tests. Finalement, nous allons discuter les résultats obtenus.

5.2 Environnement et outils de travail

5.2.1 Matériels

L'enesemble de nos expérimentations ont été réalisées sur un PC muni d'un système d'exploiatation Windwos 10 (64 bits) avec un processeur Intel(R) Core(TM) i5-8300H CPU, une fréquence d'horloge de 2.30GHz et une RAM de 12 Go, ainsi que sur la plateforme Google Colab.

Google Colab ou Colaboratory est un service cloud, offert par Google, donnant un accès direct et facile à une interface basé sur Jupyter Notebook [1] permet d'écrire et d'exécuter du code écrit en Python directement depuis le navigateur. Il s'agit d'un environnement déjà pré-configuré, particulièrement adapté au machine learning, il dispose de nombreuses librairies pré-installées telles que Scikit-learn, Numpy, Keras, TensorFlow, etc., prêtes à être utilisées, sans donc avoir besoin d'installer quoi que ce soit sur le cloud. Le plus grand avantage de Google

1. Un environnement de programmation interactif basé sur le web permettant de créer des documents Jupyter Notebook ; Un document Jupyter Notebook est un document Web au format JSON. Il suit un schéma contenant une liste ordonnée de cellules d'entrée/sortie. Celles-ci peuvent contenir du code (exécutable directement dans le document), du texte formaté, des images, etc.

Paramètres	Compte Google Colab gratuit
Modèle CPU	Intel(R) Xeon(R)
Fréquence CPU	2.30GHz
No. CPU Cores	2
RAM	12GB
Espace dique	25GB
Carte Graphique	Nvidia Tesla K80

TABLE 5.1 – Les caractéristiques d'un compte Google Colab gratuit

Colab est qu'il fournit un accès gratuit à des ressources informatiques importantes, dont l'accès à un processeur graphique GPU. Le GPU est aujourd'hui le choix par défaut quand il s'agit d'entraîner des modèles de Deep Learning. L'entraînement d'un modèle de Deep Learning peut prendre beaucoup de temps, de quelques heures à quelques jours, ce qui exige une puissance de calcul considérable pour réduire la durée de ce processus, d'où l'intérêt d'un processus graphique, ou GPU. Les caractéristiques d'un compte Google Colab gratuit sont détaillés dans le tableau 5.1.

Cependant, l'accès n'est pas toujours garanti. L'environnement d'exécution a un délais d'inactivité [2] et un délai d'expiration [3]. De plus, ces limites d'utilisation sont susceptibles de fluctuer. D'après Google, ces contraintes sont nécessaires pour maintenir un accès gratuit aux ressources de Colab [4].

5.2.2 Langage de programmation et APIs

Dans ce travail, nous avons utilisé le langage de programmation Python pour l'implémentation de nos modèles. Il s'agit de langage de programmation le plus utilisé dans le domaine du Machine Learning.

Python [5] est né à la fin des années 1980, il a été créé par Guido van Rossum . Dès sa création, Python a été conçu pour être facile à enseigner et aussi pour mettre à la disposition du plus grand nombre de programmeurs un outil de développement de haut niveau, simple et intuitif. Depuis quelques années, Python a pris une place extrêmement importante, il a même pris le dessus sur de nombreux autre langages, notamment dans le monde de traitement des données avec l'émergence de la data science, le machine learning, le deep learning et l'intelligence artificielle. Par conséquence, en tant que langage lié à ces domaines, l'écosystème Python a abouti au développement rapide de nombreux bibliothèques (packages) et de nombreuses API,

2. Cela signifie que si l'utilisateur n'interagit pas avec son ordinateur pendant une certaine période de temps, la session est automatiquement interrompue ; Certaines sources disent 15 minutes, certaines 30 minutes et d'autres vont jusqu'à 90 minutes, selon notre expérience, on peut affirmer qu'il n'y a pas de délai fixe.
3. De 12 heures, d'après certaines sources.
4. https ://research.google.com/colaboratory/faq.html ?hl=fr
5. https ://www.python.org/

qui font aujourd'hui de ce langage le meilleur outil pour automatiser des traitements de data science, et qui s'intègre parfaitement dans un cadre plus large de l'intelligence artificielle.

Nous présentons dans ce qui suit quelques principales bibliothèques telles que Scikit-Learn, Pandas, Numpy, Keras, Tensorflow, etc. que nous avons utilisé durant la réalisation de ce travail.

5.2.2.1 Numpy

NumPy [6] (abréviation de Numerical Python) est un package absolument central pour Python. Il s'agit d'une bibliothèque open source qui permet de manipuler des matrices ou tableaux multidimensionnels, ainsi qu'effectuer des opérations mathématiques et statistiques. L'élément clé de NumPy est le « array » qui correspond à des tableaux à une ou plusieurs dimensions (ndarry) et permettent stocker des données dans une structure supportant tous types de calculs avancés.

5.2.2.2 Pandas

Pandas [7] est une autre bibliothèque Python open source créé par Wes McKinney dans le but de se rapprocher des structures classiques de l'analyse de données. En effet, comme toute autre bibliothèque Pandas a ses propres structures de données, mais l'une des forces de ce package est qu'il se base sur des structures de type array de la très populaire bibliothèque NumPy mais les enrichit en créant des DataFrame et des Series. Ces structures références de Pandas qui sont la Series et le DataFrame peuvent être considérées comme des versions améliorées de tableaux structurés de NumPy, ont permis un réel changement de statut pour Python.

5.2.2.3 Gensim

Gensim est une autre bibliothèque Python très populaire qui a été principalement développé pour la modélisation sémantique (Topic modeling, l'indexation des documents et la similitude des documents. Il s'agit d'une bibliothèque robuste, efficace et simple à utiliser. Elle regroupe différents types d'outils pour différentes tâches tels que la tokenisation, POS-taggin, NER, lemmatisation etc. C'est pourquoi elle est désormais très utilisée dans différentes tâches du traitement du langage naturel. En outre, Gensim peut effectuer des tâches bien plus complexes telles que le téléchargement des modèles entraînés comme Word2Vec, Glove et FastTex et la génération des word embedding.

6. https ://numpy.org/
7. https ://pandas.pydata.org/

5.2.2.4 Scikit-Learn

Scikit-Learn est une bibliothèque libre Python polyvalente développé initialement à l'INRIA à Saclay en France [8]. Ce package offre une quantité impressionnante d'algorithmes, tels que des algorithmes de classification, de régression et de clustering mais aussi de méthodes de prétraitement, d'analyse et d'exploration de données. Depuis quelques années, ce package est couramment utilisé pour résoudre des problèmes de machine learning et de data science. En fait, le machine learning s'est développé surtout autour de ce package. Cependant, il est à noter que Scikit-Learn ne permet pas de résoudre des problèmes de deep learning, car il ne comprend pas des algorithmes de ce dernier, ce qui nous amène à parler d'autres outils Python pour le machine learning.

5.2.2.5 TensorFlow

Depuis quelques années, le domaine de deep learning se développe à grande vitesse. Ce type d'apprentissage utilise des algorithmes basés sur des réseaux neuronaux profonds qui sont complexes et extrêmement gourmands en puissance de calcul. Pour traiter ce type de modèles, l'environnement TensorFlow a été développé.

TensorFlow [9] est une bibliothèque (appelé aussi Framework) open source dispose d'une puissance de calcul numérique importante permet de résoudre des problèmes mathématiques extrêmement complexe avec aisance, mais c'est surtout ses capacités de deep learning qui sont mises en avant. Elle est particulièrement bien adaptée et optimisée pour l'apprentissage automatique à grande échelle. Ce framework développé par Google , possède une API Python pratique, confortable et très puissante, elle simplifie le processus d'acquisition de donnée, d'entraînement et d'exécution des modèles de deep learning. TensorFlow est aujourd'hui une référence et l'un des outils les plus utilisés dans le domaine du deep learining, il est utilisé à grande échelle chez Google ce qui à contribuer à sa popularité.

5.2.2.6 Keras

Keras [10] est un package de deep learning largement utilisé qui est développé et maintenu par François Chollet . Il s'agit d'une API de haut niveau qui permet de construire, d'entraîner, d'évaluer et d'exécuter des modèles de réseaux de neurones facilement. Cette interface est rapidement devenue populaire en raison de sa facilité d'utilisation et de sa souplesse. Pour effectuer des calculs dans le cadre d'un réseau de neurones, cette implémentation de référence, également nommé Keras, se fonde sur un environnement de deep learning (backend de calcul) tels que TensorFlow, CNTK et Theano [11]. Toutes ces bibliothèques ont contribué grandement à

8. https ://scikit-learn.org/stable/
9. https ://www.tensorflow.org/ ?hl=fr
10. https ://keras.io/
11. Keras multibackend

faire de Python le langage de référence pour la science de données, le machine learning et le deep learning.

5.3 Cadre expérimental et résultats

Après avoir présenté nos modèles proposés dans le chapitre précédent, où le premier était un modèle Bi-LSTM et le second était un modèle CNN. Nous présentons dans cette section les différents paramètres de configuration utilisés dans notre cadre expérimental pour chaque modèle avec les résultats obtenus.

5.3.1 Analyse des sentiments - corpus anglais

5.3.1.1 Données et protocole de test

Pour notre modèle d'analyse d'opinions en anglais, nous avons utilisé l'ensemble de données IMDB qui contient 50 000 critiques de films composées d'un commentaire et d'une polarité associée (positive ou négative). Après une étape de prétraitement et de nettoyage, nous avons décomposé ce jeu de données en trois ensembles : Train, Valid et Test.

— L'ensemble d'entraînement (Train) représente la plus grande partie de notre ensemble de données (80%). Il est utilisé pour entraîner notre modèle de classification.

— L'ensemble de validation (Valid) est utilisé pour mesurer l'erreur de prédiction afin d'ajuster des paramètres du modèle pendant la phase de réglage du modèle en vu de son optimisation. Après l'évaluation, les ajustements s'effectuent en fonction des résultats de prédiction obtenus sur cet ensemble. Pour rendre le modèle robuste, la configuration des paramètres est choisie en fonction des meilleurs résultats observés sur cet ensemble de validation. Cet ensemble représente 10% de notre jeu de données.

— L'ensemble de test (Test), il représente une partie (10% dans notre cas) de jeu de données que le modèle n'a jamais vu au cours de son apprentissage et de son ajustement (phase de réglage). Il permet de mesurer les performances finales du modèle une fois l'apprentissage est terminé. Les résultats obtenus sur cette partie de données estiment sa capacité à s'adapter à de nouvelles données jamais vu auparavant, c'est-à-dire sa capacité de généralisation.

5.3.1.2 Pré-traitement

Afin de pouvoir exploiter les données textuelles de la base IMDB, plusieurs opérations de pré-traitements ont été effectuées. Elles sont détaillées ci-après :

— **Segmentation (Tokenization)** : Pour le corpus anglais nous avons utilisé le Tokenizer natif de NLTK.

— **Suppression des mots vides** : La bibliothèque NLTK dispose d'une liste de mots vides pour la langue anglaise. Nous nous sommes basés sur cette dernière pour la suppression des mots vides. Nous avons aussi utilisé les expression régulière pour éliminer le reste des symboles inutiles tels que '@, !,...,etc'.

5.3.1.3 Extraction des caractéristiques

Pour la modélisation des données textuelles du corpus anglais, nous avons utilisé deux modèles plongement de mot -word embedding- pré-entraînés pour générer nos représentations vectorielles des mots :

— **Word2Vec** : Pour l'implémentation du modèle Word2Vec de Google nous avons utilisé la bibliothèque Gensim.

— **Glove** : Le modèle que nous avons utilisé est celui développé à l'université de Stanford.

Ces deux modèles n'acceptent en données d'entrée qu'une liste de listes de mots (tokens), donc après avoir nettoyé nos données avec différentes techniques de prétraitement (mise en minuscule, suppression de la ponctuation, etc.), le traitement suivant a été rajouté :

— La tokenisation des commentaires afin d'avoir une liste de tokens pour chaque commentaire.

— La construction d'un vocabulaire à partir des listes de tokens, c'est-à-dire l'ensemble des mots dont notre corpus dispose.

— La conversion des commentaires en une séquence d'identifiants basé sur l'index de mot (word index)

— Assurer que tous les commentaires encodés ont la même taille.

Une fois que nous avons terminé ce prétraitement, nous avons téléchargé le modèle pré-entrainé en anglais Word2vec [12]/Glove [13] afin d'initialiser notre matrice d'embedding qui sera entrainée et construite sur notre base de données IMDB.

5.3.1.4 Implémentation des modèles de classification

Dans cette partie, nous allons présenter les différentes configurations de nos différents modèles de classification. Le premier modèle que nous présentons est le modèle Bi-LSTM.

Comme le montre le Tableau 1, notre premier modèle est composé de deux couches LSTM bidirectionnels empilées l'une sur l'autre, avec 256 unités cachées chacune. Nous avons aussi la couche embedding initialisée par la matrice générée par Word2Vec/Glove, dont chaque mot dans la matrice est représenté par un vecteur de dimension 300. De plus, nous avons utilisé un dropout entre chaque sortie d'une couche précédente et l'entrée d'une couche suivante. Le

12. https ://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz
13. http ://nlp.stanford.edu/data/glove.6B.zip

dropout est une technique qui est destinée à empêcher le sur-apprentissage sur les données d'entraînement dans les réseaux de neurones. Elle consiste à désactiver temporairement une partie des unités et leurs connexions pendant l'apprentissage. Pour ce modèle, un dropout de 0.2 est utilisé. Concernant l'algorithme d'optimisation, nous avons utilisé *RMSprop*. Enfin, nous avons appliqué une fonction d'activation *sigmoid* à la couche de sortie pour générer la polarité du commentaire fourni en entrée. Nous récapitulons, dans le tableau 5.2, les différents hyperparamètres de ce modèle Bi-LSTM.

Hyperparamètre	**valeur**
Taille de vecteur	300
Dropout	0.2
Word Embedding	statique
Fonction d'optimisation	RMSprop
Fonction de perte	Binary Cross Entropy
Nombre d'unités LSTM	256

TABLE 5.2 – Valeurs des hyperparamètres du modèle Bi-LSTM

Algorithm 1 Architecture du modèle Bi-LSTM

$model = Sequential()$
$model.add(Embedding(input\_dim = nbr\_words,$
$weights = [embedding\_matrix], output\_dim = embedding\_size,$
$input\_length = max\_len, trainable = False))$
$model.add(Dropout(0.2))$
$model.add(Bidirectional(LSTM(256, return_sequences = True)))$
$model.add(Dropout(0.2))$
$model.add(Bidirectional(LSTM(256)))$
$model.add(Dropout(0.2))$
$model.add(Dense(1, activation =' sigmoid'))$
$model.summary()$

Le deuxième modèle que nous avons proposé il s'agit d'un CNN composé d'une séquence de couches (Tableau 2). D'abord nous avons la couche embedding qui est similaire à la couche d'embedding du modèle LSTM, ensuite quatre couches de convolution de 100 filtres dont la taille est une des valeurs de l'ensemble 1,2,3,4. Pour chacune de ces couches une fonction d'activation ReLu est appliquée. Puis, une couche de Max pooling est insérée à la sortie de chaque couche convolution. Finalement, une couche de sortie avec un fonction d'activation mais cette fois nous avons utilisé *softmax* et non sigmoïde. Quant à l'optimisation, nous avons utilisé l'algorithme d'*Adam*. Un dropout de 0.2 est également utilisé dans ce réseau. Nous récapitulons également les hyperparamètres de ce modèle dans le tableau 5.3.

Hyperparamètre	valeur
Taille de vecteur	300
Taille de filtres	{1,2,3,4}
Dropout	0.2
Word Embedding	statique
Fonction d'optimisation	Adam
Fonction de perte	Sparse categorical cross entropy

TABLE 5.3 – Valeurs des hyperparamètres du modèle CNN

Algorithm 2 Architecture du modèle CNN

$model = Sequential()$
$model.add(Embedding(input\_dim = nbr\_words,$
$weights = [embedding\_matrix], output\_dim = embedding\_size,$
$input\_length = max\_len, trainable = False))$
$model.add(Dropout(0.2))$
$model.add(Conv1D(filters = 100, kernel\_size = 1, padding =' same', activation ='$
$relu'))$
$model.add(Dropout(0.2))$
$model.add(MaxPooling1D())$
$model.add(Conv1D(filters = 100, kernel\_size = 2, padding =' same', activation ='$
$relu'))$
$model.add(Dropout(0.2))$
$model.add(MaxPooling1D())$
$model.add(Conv1D(filters = 100, kernel\_size = 3, padding =' same', activation ='$
$relu'))$
$model.add(Dropout(0.2))$
$model.add(MaxPooling1D())$
$model.add(Conv1D(filters = 100, kernel\_size = 4, padding =' same', activation ='$
$relu'))$
$model.add(Dropout(0.2))$
$model.add(MaxPooling1D())$
$model.add(Dense(128, activation =' relu')))$
$model.add(Dropout(0.2))$
$model.add(model.add(Flatten()))$
$model.add(Dense(1, activation =' softmax')$
$model.summary()$

5.3.1.5 Résultats

Les résultats des deux modèles sont présentés dans le tableau 5.3.1.5 ci-dessous :

	Modèles	
	Bi-LSTM	CNN
Word Embedding	Word2Vec	Glove
Résultat	90.32	89.78%

TABLE 5.4 – Résutlats des modèles anglais

5.3.2 Analyse des sentiments - corpus dialecte algérien

5.3.2.1 Données et protocole de test

Afin de réaliser notre troisième modèle de classification, nous avons utilisé notre propre corpus de domaine que nous avons construit à partir des pages Facebook de marques algériennes. L'opération d'assemblement de donnée représente une des difficultés majeures lors de l'exploration de données par des techniques de machine learning supervisées. Dans notre travail, nous avons pu collecter 2 905 commentaires de critiques de produits et services. Mais comme toute opération de collecte de données, cette phase requiert un prétraitement, qui consiste à supprimer tout le bruit qui peut exister dans ces données afin de garder que les commentaires écrits totalement en arabe. À la fin, nous avons réussi à garder 1 900 (1113 positifs et 763 négatifs) commentaires prêts pour être utilisé pour construire notre modèle de classification. Nous avons décomposé ce jeu de données en trois ensembles évidemment : Ensemble d'entraînement (80%), ensemble de validation (6%) et ensemble de test (14%) (Tableau 5.5).

	Train	Valid	Test
Nombre de commentaires	1513	141	266

TABLE 5.5 – Répartiton de jeu de données DzReviews

5.3.2.2 Word Embedding

Dans le cadre de cette partie de notre travail qui concerne le dialecte algérien, nous avons expérimenté également deux différentes méthodes de word embedding pour générer les représentations vectorielles des mots. Pour La première méthode, il s'agit de la méthode FastText que nous avons déjà présenté dans la section 3.2 du chapitre 3 et dans la section 4.2.2 du chapitre précédent. Sauf que nous ne nous sommes pas limités au modèle pré-entraîné cette fois, nous avons essayé d'entraîner notre propre modèle dialecte FastText en utilisant notre propre corpus de domaine, en espérant obtenir de meilleurs résultats. Cependant, la deuxième méthode consiste à utiliser la couche embedding pour entraîner notre matrice de word embedding en même temps que l'entraînement du notre modèle pour la tâche de classification. Il s'agit dans ce cas d'initialiser de façon aléatoire les représentations de mots de cette couche, qui sont mis à à jour au fur et à mesure au moment de l'apprentissage. Pour le modèle FastText pré-entraîner

nous avons utilisé un modèle en arabe de dimension 300 [14]. Après avoir préparer nos données d'entrée sous forme de liste de listes de tokens, nous avons passé ces mots au modèle FastText en spécifiant les paramètres présenté dans le tableau 5.6 ci- après :

Paramètres	Valeur
Size	100
Windows	2
Min_count	1
Workers	3
Sg	1

TABLE 5.6 – Paramètres d'entraînement du modèle FastText

Où

— **Size** : la taille de vecteur

— **Windows** : la taille de la fenêtre

— **Min_count** : les mots peu fréquents en fonction du nombre minimum mentionné seront ignorés.

— **Workers** : nombre de threads de processeur à utiliser à la fois pour une formation plus rapide.

— **Sg** : 0 pour CBOW et 2 pour Skip-gram

5.3.2.3 Implémentation des modèles de classification

Dans l'implémentation du ce modèle, nous avons opté pour les mêmes architectures que nous avons utilisés pour l'anglais, c'est-à-dire un réseaux Bi-LSTM et un autre CNN, seulement que dans cette partie nous avons effectué quelques expérimentations afin de trouver un modèle avec de bons résultats par rapport à notre corpus du domaine. Dans ce but, nous avons testé des réseaux de type Bi-LSTM et CNN mais avec des couches, des paramètres et des méthodes de word embedding différentes à chaque fois. Par exemple, dans notre première expérimentation nous avons utilisé le modèle de word embbeding FastText que nous avons entraîné nous-même, alors que nous avons utilisé un modèle pré-entraîné dans la deuxième.

Dans ce qui suit, nous allons présenter les différentes variations des architectures que nous avons testé et leurs configurations, ensuite nous présentons les résultats obtenus de chaque modèle.

5.3.2.4 Expérimentation 1

Dans cette première expérimentation nous avons utilisé une architecture très simple. Il s'agit d'un réseau composé de deux couches Bi-LSTM empilées l'une sur l'autre, avec 256

14. https ://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.ar.300.vec.gz

et 128 unités cachés respectivement. Nous avons également la couche Embedding initialisée par la matrice que nous avons généré en utilisant notre modèle dialecte FastText. Dans ce cas nous avons décidé d'utiliser notre modèle dans sa version statique. De plus, comme le montre le Tableau 3, nous n'avons pas utilisé le décrochage (dropout), contrairement à l'architecture utilisé pour l'anglais. Pour la dernière couche de sortie nous avons utilisé la fonction de sigmoïde.

Algorithm 3 Expérimentation 1

$model = Sequential()$
$model.add(Embedding(input\_dim = nbr\_words,$
$weights = [embedding\_matrix], output\_dim = embedding\_size,$
$input\_length = max\_len, trainable = False))$
$model.add(Bidirectional(LSTM(256, return\_sequences = True)))$
$model.add(Bidirectional(LSTM(128)))$
$model.add(Dense(1, activation =' sigmoid'))$
$model.summary()$

Le tableau 5.7 suivant montre les différents paramètres d'entraînement :

Paramètres	Valeur
La taille du vocabulaire	4154
La taille de séquence	100
La taille d'embedding	100
La taille de batch	32
Le nombre d'epochs	10

TABLE 5.7 – Paramètres d'entraînement pour l'expérimentation 1

Résultats Pour cette première expérimentation, nous n'avons pas de résultats concrète à cause au fait que le modèle a échoué dans son but de trouver une configuration des poids qu'il lui permet de converger vers un état optimal. En effet, durant les 8 itérations avant l'arrêt du processus d'entraînement, l'accuraçy du modèle était toujours entre 54 et 64 pour cent.

5.3.2.5 Expérimentation 2

Dans la deuxième expérimentation nous avons utilisé une architecture similaire à celle de la première, seulement que cette fois nous avons initialisé la couche embedding par une matrice générée à partir du modèle FastText pré-entraîné que nous avons téléchargé et non pas celui que nous avons entraîné. Nous avons aussi, contrairement à l'expérimentation précédente, utilisé un dropout de 0.3 à chaque sortie d'une couche LSTM. Les détails de l'achitecture sont présenté dans le Tableau 4

Algorithm 4 Expérimentation 2

$model = Sequential()$
$model.add(Embedding(input\_dim = nbr\_words,$
$weights = [embedding\_matrix], output\_dim = embedding\_size,$
$input\_length = max\_len, trainable = False))$
$model.add(Bidirectional(LSTM(256, return\_sequences = True)))$
$model.add(Dropout(0.3))$
$model.add(Bidirectional(LSTM(128)))$
$model.add(Dropout(0.3))$
$model.add(Dense(1, activation =' sigmoid'))$
$model.summary()$

Les paramètres d'entraînement de l'expérimentation sont présenté dans le Tableau 5.8.

Paramètres	Valeur
La taille du vocabulaire	4154
La taille de séquence	300
La taille d'embedding	300
La taille de batch	32
Le nombre d'epochs	3

TABLE 5.8 – Paramètres d'entraînement pour l'expérimentation 2

Résultats Les résutlats de cette epérimentation sont présenté dans le tableau 5.9

	Accuracy	Précision	Rappel	F1-score
Résultats	82.70	82.38	87.91	85.06

TABLE 5.9 – Résultats obtenus de l'expérimentation 2

5.3.2.6 Expérimentation 3

Comme il est montré dans le tableau 5, dans cette troisième expérimentation nous avons utilisé une couche LSTM unidirectionnel (128 unités) au lieu de deux couches bidirectionnelles. L'idée derrière ce test est de simplifier l'architecture de notre modèle le maximum afin de tester si nous pouvons obtenir de bons résultats sur notre jeu de données en utilisant un modèle simple. Concernant la couche d'embbeding, nous l'avons initialisé en utilisant le modèle FastText pré-entrainé. Un dropout de 0.3 et d'une fonction de sigmoïde ont également été utilisé pour ce modèle.

Algorithm 5 Expérimentation 3

1: $model = Sequential()$
2: $model.add(Embedding(input\_dim = nbr\_words,$
$weights = [embedding\_matrix], output\_dim = embedding\_size,$
$input\_length = max\_len, trainable = False))$
3: $model.add(Dropout(0.3))$
4: $model.add(LSTM(128))$
5: $model.add(Dropout(0.3))$
6: $model.add(Dense(1, activation =' sigmoid'))$
7: $model.summary()$

Les paramètres d'entraînement pour chaque exécution sont présenté dans le Tableau 5.10

	Paramètres			
	La taille du vocabulaire	La taille de séquence	La taille d'embedding	La taille de batch
Exécution 1	4154	300	300	32
Exécution 2	4154	300	300	32
Exécution 3	4154	300	300	32

TABLE 5.10 – Paramètres d'entraînement pour l'expérimentation 3

Dans cette expérimentation nous avons exécuté l'algorithme trois fois dans le but d'avoir de meilleurs résultats. Les résultats de chaque exécutions est présenté dans le Tableau 5.11

	Résultats			
	Accuracy	Précision	Rappel	F1-score
Résultats de l'exécution 1	82.33	80.35	90.60	85.17
Résultats de l'exécution 2	81.57	78.20	92.60	85.25
Résultats de l'exécution 3	83.08	75.2	92.23	83.23

TABLE 5.11 – Résultats obtenus durant l'expérimentation 3

5.3.2.7 Expérimentation 4

Dans ce réseau nous avons envisagé d'expérimenter le fait de ne pas utiliser un modèle de word embedding pré-entraîné. En revanche, notre modèle de représentation de mots est basé sur la couche d'embedding. De façon que les embeddings de cette couche sont initialisés de façon aléatoire. L'architecture de ce modèle utilisé dans l'expérimentation est présenté dans le Tableau 6. Il s'agit de deux couches Bi-LSTM, de dropout de 0.5, et une couche de sortie avec un fonction de sigmoïde.

Algorithm 6 Expérimentation 4

1: $model = Sequential()$
2: $model.add(Embedding(input\_dim = nbr\_words, output\_dim = embedding\_size, input\_length = max\_len, trainable = True))$
3: $model.add(Dropout(0.5))$
4: $model.add(LSTM(128))$
5: $model.add(Dropout(0.5))$
6: $model.add(Dense(1, activation =' sigmoid'))$
7: $model.summary()$

Les paramètres d'entraînement sont présenté dans le Tableau 5.12 ci-après :

Paramètres	Valeur
La taille du vocabulaire	4154
La taille de séquence	300
La taille d'embedding	300
La taille de batch	32
Le nombre d'epochs	2

TABLE 5.12 – Paramètres d'entraînement pour l'expérimentation 4

Résultats Les résutlats sont présenté dans le tableau 5.13

	Accuracy	Précision	Rappel	F1-score
Résultats	90.31	91.72	91.09	91.40

TABLE 5.13 – Résultats obtenus de l'expérimentation 4

5.3.2.8 Expérimentation 5

Contrairement aux différentes expérimentations précédentes où nous avons contenté d'utiliser pour générer les embeddings soit la couche d'embedding initialisé aléatoirement, ou le modèle FastText pré-entraîné, dans ce test nous avons combiné les deux. Nous avons passé de modèle statique au modèle non statique, c'est-à-dire nous avons initialisé notre couche d'embedding par nos vecteurs pré-entraîné, ensuite ces vecteurs sont mis à jour lors de la phase de l'apprentissage. En outre, nous avons testé un nouveau paramètre que nous n'avons pas utilisé dans les expérimentations précédentes, il s'agit de régularisation l2 qui sert à minimiser la perte dans le but d'améliorer la généralisation lors de l'apprentissage. Le Tableau 7 montre l'architecture utilisé dans cette expérimentation.

Les paramètres d'entraînement sont présenté dans le Tableau 5.14 ci-après :

Algorithm 7 Expérimentation 5

1: $model = Sequential()$
2: $model.add(Embedding(input\_dim = nbr\_words,$
$weights = [embedding\_matrix], output\_dim = embedding\_size,$
$input\_length = max\_len, trainable = True))$
3: $model.add(Bidirectional(LSTM(256, return\_sequences =$
$True, bias\_regularizer = l2(0.0001))))$
4: $model.add(Dropout(0.5))$
5: $model.add(Bidirectional(LSTM(128, bias\_regularizer = l2(0.0001))))$
6: $model.add(Dropout(0.5))$
7: $model.add(Dense(1, activation =' sigmoid'))$
8: $model.summary()$

Paramètres	Valeur
La taille du vocabulaire	4154
La taille de séquence	100
La taille d'embedding	100
La taille de batch	64
Le nombre d'epochs	3
La fonction d'activation	Adam
La fonction de perte	Binary cross entropy

TABLE 5.14 – Paramètres d'entraînement pour l'expérimentation 5

Résultats Les résutlats sont présenté dans le tableau 5.15

	Accuracy	Précision	Rappel	F1-score
Résultats	86.46	87.89	89.03	88.46

TABLE 5.15 – Résultats obtenus de l'expérimentation 5

5.3.2.9 Expérimentation 6

Le tableau 8 montre l'architecture du réseau CNN que nous avons utilisé sur notre jeu de donnée dans cette expérimentation. Il s'agit du même réseau que nous avons adapté à l'anglais avec les même paramètre excepté le modèle de plongement de mots où nous avons utilisé dans ce cas le modèle FastText pré-entraîné bien évidemment. Les paramètres que nous avons utilisé pour l'entraînement de ce modèle sont présenté dans le Tableau 5.16.

Paramètres	Valeur
La taille du vocabulaire	4154
La taille de séquence	300
La taille d'embedding	300
La taille de batch	50
Le nombre d'epochs	100
La fonction d'activation	Adam
La fonction de perte	Binary cross entropy

TABLE 5.16 – Paramètres d'entraînement pour l'expérimentation 6

Algorithm 8 Expérimentation 6

$model = Sequential()$
$model.add(Embedding(input\_dim = nbr\_words,$
$weights = [embedding\_matrix], output\_dim = embedding\_size,$
$input\_length = max\_len, trainable = False))$
$model.add(Dropout(0.5))$
$model.add(Conv1D(filters = 100, kernel\_size = 1, padding =' same', activation =' relu'))$
$model.add(Dropout(0.5))$
$model.add(MaxPooling1D())$
$model.add(Conv1D(filters = 100, kernel\_size = 2, padding =' same', activation =' relu'))$
$model.add(Dropout(0.5))$
$model.add(MaxPooling1D())$
$model.add(Conv1D(filters = 100, kernel\_size = 3, padding =' same', activation =' relu'))$
$model.add(Dropout(0.5))$
$model.add(MaxPooling1D())$
$model.add(Conv1D(filters = 100, kernel\_size = 4, padding =' same', activation =' relu'))$
$model.add(Dropout(0.5))$
$model.add(MaxPooling1D())$
$model.add(Dense(128, activation =' relu')))$
$model.add(Dropout(0.5))$
$model.add(model.add(Flatten()))$
$model.add(Dense(1, activation =' softmax')$
$model.summary()$

Résultats Les résutlats sont présenté dans le tableau 5.17

	Accuracy	Précision	Rappel	F1-score
Résultats	90.52	90.07	93.91	92.30

TABLE 5.17 – Résultats obtenus de l'expérimentation 6

5.3.3 Discussion

5.3.3.1 Cas anglais

D'après les résultats présentés dans le tableau 5.18, nous remarquons que les deux architectures que nous avons proposées pour le cas de l'anglais ont obtenu un résultat légèrement meilleur par rapport aux résultats obtenus dans les travaux de Sharma et al. et Minacc et al.

Auteurs	**Modèle**	**Dataset**	**Accuracy**
Sharma et al. (2020)	CNN	IMDB	82,19%
Minaee et al. (2019)	Bi-LSTM	IMDB	89%
	CNN		89.3%
	Bi-LSTM + CNN		90%
Modèle proposé	Bi-LSTM	IMDB	**90.32%**
	CNN		**89.78%**

TABLE 5.18 – Comparaison des résultats de nos modèles proposé avec certains modèles des travaux antérieures sur le jeu de données IMDB

Sharma et al. ont entrainé leur modèle pendant 10 epochs seulement. Les valeurs métriques à la fin de chaque époque sont présentées dans le tableau 5.19 suivant :

epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
1	0.6767	0.5628	0.5248	0.7395
2	0.4822	0.7668	0.4222	0.8219
3	0.3225	0.8656	0.4291	0.8116
4	0.2039	0.9215	0.4718	0.8051
5	0.125	0.9529	0.57	0.7976
6	0.0843	0.9693	0.656	0.7994
7	0.0579	0.9788	0.7111	0.7891
8	0.0404	0.9844	0.7736	0.7919
9	0.0295	0.9898	0.8214	0.7976
10	0.0272	0.9907	0.9089	0.7994

TABLE 5.19 – Les valeurs métriques à la fin de chaque époque durant la phase d'apprentissage [72]

Selon les auteurs, le modèle qui ont proposé a donné une accuracy de 99.07% pour les données d'apprentissage, et 82.19% pour les données de test. Pour cette dernière, il s'agit d'une accuracy obtenue à la deuxième itération qui représente la 2ème epoch, sauf que nous pouvons voir qu'après l'epoch 2, le modèle semble commencer son phénomène d'overfitting

(sur-apprentissage) [15], le val_accuracy est en diminution et le val_loss augmente. On remarque aussi que à la fin de l'entraînement le loss sur les données d'entraînements est beaucoup plus faible (2.72%) que celle sur les données de test (90.89%), ce qui signifie que le modèle a trop appris les données d'entraînement. De cette manière, il risque de ne pas savoir généraliser à des données inconnues.

Cependant, notre modèle CNN que nous avons proposé dans notre travail a été entrainé pendant 100 itérations. Le tableau 5.20 montre les 10 premières epochs.

epoch	**loss**	**accuracy**	**val_loss**	**val_accuracy**
1	0.6452	0.5928	0.4246	0.8210
2	0.4277	0.8088	0.3918	0.8348
3	0.3891	0.8267	0.3574	0.8560
4	0.3608	0.8425	0.3363	0.8582
5	0.3461	0.8502	0.3239	0.8690
6	0.3362	0.8556	0.3264	0.8642
7	0.3361	0.8560	0.3121	0.8686
8	0.3196	0.8651	0.3058	0.8690
9	0.3135	0.8663	0.2970	0.8770
10	0.3168	0.8664	0.2905	0.8786

TABLE 5.20 – Les valeurs métriques des 10 premières époques durant la phase d'apprentissage de notre modèle CNN proposé

Nous pouvons constater sur le Tableau 5.20 que les résultats obtenus par notre modèle sont meilleurs que celui de Sharma et al. Par exemple, à la deuxième itération seulement, notre modèle à obtenu une accuracy de 83.34%, ce qui supérieur au résultat obtenu par Sharma et al. après le même nombre d'itérations. Notons également, contrairement au modèle de Sharma et al., notre modèle ne souffre pas de phénomène d'overfitting après la 2ème epoch ni même après la 10ème comme le montre le tableau 5.21, en effet, l'accuracy continu d'augmenter, pendant que val_loss continue de diminuer, pour atteindre à la fin de l'entrainement une accuracy de 93.54% sur l'ensemble de données d'entrainement et une accuracy de 89.58% sur l'ensemble de validation, tandis que nous avons obtenu une accuracy de 89.78% sur l'ensemble de test, qui représente des données non vus pendant l'entraînement par notre modèle, ce qui veut dire que notre modèle est capable de généraliser sur de nouvelles données, ce qui signifie qu'il a été bien entrainé.

15. L'overfitting est le risque pour un modèle d'apprendre "par cœur" les données d'entraînement. De cette manière, il risque de ne pas savoir faire des prédictions sur de nouvelles données inconnues.

epoch	loss	accuracy	val_loss	val_accu	epoch	loss	accuracy	val_loss	val_accu	epoch	loss	accuracy	val_loss	val_accu
1	0.6452	0.5928	0.4246	0.8210	35	0.2416	0.9012	0.2859	0.8830	69	0.1873	0.9223	0.2568	0.8950
2	0.4277	0.8088	0.3918	0.8348	36	0.2435	0.9011	0.2579	0.8906	70	0.1842	0.9260	0.2621	0.8928
3	0.3891	0.8267	0.3574	0.8560	37	0.2335	0.9056	0.2615	0.8920	71	0.1827	0.9264	0.2604	0.8928
4	0.3608	0.8425	0.3363	0.8582	38	0.2350	0.9030	0.2559	0.8940	72	0.1886	0.9222	0.2596	0.8920
5	0.3461	0.8502	0.3239	0.8690	39	0.2285	0.9060	0.2604	0.8928	73	0.1844	0.9265	0.2871	0.8868
6	0.3362	0.8556	0.3264	0.8642	40	0.2279	0.9071	0.2563	0.8938	74	0.1852	0.9238	0.2568	0.8938
7	0.3361	0.8560	0.3121	0.8686	41	0.2272	0.9068	0.2571	0.8912	75	0.1866	0.9261	0.2641	0.8924
8	0.3196	0.8651	0.3058	0.8690	42	0.2316	0.9046	0.2654	0.8914	76	0.1843	0.9248	0.2586	0.8938
9	0.3135	0.8663	0.2970	0.8770	43	0.2233	0.9089	0.2576	0.8940	77	0.1850	0.9236	0.2558	0.8956
10	0.3168	0.8664	0.2905	0.8786	44	0.2219	0.9092	0.2623	0.8914	78	0.1791	0.9279	0.2563	0.8984
11	0.3071	0.8698	0.2941	0.8752	45	0.2216	0.9110	0.2555	0.8938	79	0.1823	0.9254	0.2589	0.8930
12	0.2968	0.8744	0.2935	0.8762	46	0.2182	0.9108	0.2551	0.8952	80	0.1817	0.9268	0.2593	0.8954
13	0.2971	0.8744	0.2869	0.8806	47	0.2214	0.9107	0.2619	0.8908	81	0.1796	0.9263	0.2720	0.8926
14	0.2914	0.8786	0.2906	0.8828	48	0.2184	0.9113	0.2545	0.8930	82	0.1765	0.9276	0.2608	0.8940
15	0.2943	0.8767	0.2769	0.8816	49	0.2176	0.9121	0.2551	0.8918	83	0.1730	0.9298	0.2616	0.8918
16	0.2841	0.8813	0.2786	0.8836	50	0.2185	0.9115	0.2627	0.8900	84	0.1736	0.9308	0.2628	0.8920
17	0.2829	0.8809	0.2725	0.8826	51	0.2130	0.9119	0.2569	0.8948	85	0.1746	0.9296	0.2615	0.8948
18	0.2755	0.8842	0.2747	0.8834	52	0.2169	0.9123	0.2654	0.8902	86	0.1755	0.9278	0.2617	0.8964
19	0.2776	0.8851	0.2707	0.8852	53	0.2126	0.9124	0.2542	0.8956	87	0.1774	0.9264	0.2595	0.8946
20	0.2726	0.8871	0.2737	0.8882	54	0.2067	0.9163	0.2536	0.8938	88	0.1729	0.9297	0.2633	0.8934
21	0.2669	0.8893	0.2696	0.8842	55	0.2115	0.9132	0.2559	0.8938	89	0.1697	0.9329	0.2583	0.8944
22	0.2678	0.8911	0.2678	0.8858	56	0.2062	0.9165	0.2577	0.8942	90	0.1660	0.9323	0.2565	0.8974
23	0.2643	0.8896	0.2759	0.8872	57	0.2045	0.9170	0.2623	0.8898	91	0.1671	0.9343	0.2597	0.8960
24	0.2635	0.8922	0.2673	0.8866	58	0.2014	9189	0.2556	0.8928	92	0.1640	0.9328	0.2592	0.8968
25	0.2559	0.8943	0.2645	0.8854	59	0.2018	0.9181	0.2548	0.8936	93	0.1672	0.9333	0.2670	0.8934
26	0.2626	0.8927	0.2720	0.8846	60	0.1971	0.9186	0.2550	0.8942	94	0.1656	0.9341	0.2601	0.8980
27	0.2589	0.8919	0.2627	0.8884	61	0.1948	0.9223	0.2542	0.8964	95	0.1655	0.9342	0.2677	0.8940
28	0.2530	0.8967	0.2634	0.8870	62	0.1980	0.9200	0.2534	0.8954	96	0.1633	0.9340	0.2660	0.8948
29	0.2520	0.8963	0.2602	0.8894	63	0.1960	0.9214	0.2545	0.8948	97	0.1628	0.9330	0.2724	0.8922
30	0.2479	0.8961	0.2687	0.8878	64	0.1986	0.9186	0.2538	0.8962	98	0.1635	0.9351	0.2688	0.8918
31	0.2448	0.8998	0.2626	0.8876	65	0.1985	0.9174	0.2749	0.8874	99	0.1624	0.9339	0.2668	0.8948
32	0.2484	0.8959	0.2617	0.8878	66	0.1959	0.9202	0.2556	0.8974	100	0.1592	0.9354	0.2700	0.8958
33	0.2428	0.9014	0.2581	0.8910	67	0.1943	0.9213	0.2573	0.8938					
34	0.2460	0.8993	0.2568	0.8922	68	0.1904	0.9222	0.2616	0.8906					

TABLE 5.21 – Les valeurs métriques à la fin de chaque époque durant la phase d'apprentissage de notre modèle CNN proposé

Dans le travail de Minaee et al., les auteurs ont testé trois architectures : Bi-LSTM, CNN et une combinaison des deux (CNN+Bi-LSTM). Comme le montre le tableau 5.18, les résultats qui ont obtenu sont 89%, 89.3% et 90% respectivement. Les modèles ont été entraînés pendant 100 epochs. Ces résultats sont légèrement en dessous des résultats obtenus par les deux architectures que nous avons proposé dans notre travail. Avec notre modèle Bi-LSTM qui a été entrainé pendant 10 epchs seulement, nous avons réussi a atteindre une accuracy de 90.44% sur l'ensemble de données de validation et une accuracy de 93.46% sur l'ensemble d'entrainement, avec une perte de 25.52% et 17.17% respectivement. Le tableau 5.22 montre les résultats du modèle après chaque itération.

epoch	loss	accuracy	val_loss	val_accuracy
1	0.6176	0.6727	0.4692	0.7980
2	0.4363	0.8058	0.3136	0.8680
3	0.3116	0.8718	0.2656	0.8912
4	0.2822	0.8835	0.2565	0.8968
5	0.2576	0.8958	0.2487	0.8974
6	0.2357	0.9063	0.2459	0.9032
7	0.2210	0.9129	0.2836	0.8918
8	0.2006	0.9206	0.2551	0.8970
9	0.1810	0.9311	0.2739	0.9010
10	0.1717	0.9346	0.2552	0.9044

TABLE 5.22 – Les valeurs métriques à la fin de chaque époque durant la phase d'apprentissage de notre modèle Bi-LSTM proposé

Pour l'ensemble de test, nous avons obtenu une accuracy de 90.32%. Le tableau 5.23 montre le résultat obtenu pour chaque métrique de l'évaluation de modèle sur cet ensemble.

	Accuracy	Précision	Rappel	F1-score
Résultat (%)	90.32	91.74	88.92	90.31

TABLE 5.23 – Les résultats du modèle sur l'ensemble de test

Notons également que l'accuracy obtenue par notre modèle proposé n'est pas meilleur que le modèle CNN et Bi-LSTM proposé par Monacc et al. seulement, mais aussi meilleur que leur modèle CNN+BiLSTM proposé.

5.3.3.2 Cas dialecte : première expérimentation

Dans notre première expérimentation, nous nous somme basés sur l'idée que la qualité des embeddings de mots utilisés pour entrainer notre modèle affecte les résultats. Nous avons alors décidé d'entrainer notre Bi-LSTM avec notre propre modèle de word embbeding que nous avons entrainé en utilisant notre propre corpus de dialecte. Il s'agit de modèle de word embedding FastText. Le tableau 5.24 montre les résultats du modèle après chaque epoch.

epoch	loss	accuracy	val_loss	val_accu
1	0.6704	0.5995	0.6678	0.5804
2	0.6514	0.6146	0.6627	0.5893
3	0.6404	0.6341	0.6627	0.5982
4	0.6414	0.6287	0.6763	0.5893
5	0.6352	0.6370	0.6427	0.6250
6	0.6345	0.6482	0.7065	0.5446
7	0.6746	0.6043	0.7034	0.5446
8	0.6633	0.6183	0.6857	0.5446

TABLE 5.24 – Les valeurs métriques à la fin de chaque époque durant la phase d'apprentissage

Pour le paramètre relatif au nombre d'époques, dans cette première expérience nous avons choisi la valeur 10, sauf que l'entrainement est arrêté à la 8eme itération car nous avons utilisé l'arrêt prématuré (en anglais early stopping). Il s'agit d'une technique de régularisation qui consiste à stopper l'étape d'entraînement dès que le loss de validation atteint un plateau ou commence à augmenter. Dans notre cas, pour le paramètre "patience" relatif à la condition d'arrêt qui représente le nombre d'époques avant de s'arrêter un fois que la perte comme à augmenter, nous avons choisi la valeur 3.

Dans la phase d'entrainement, à chaque époque, le but est de minimiser la fonction de coût (loss), et comme nous pouvons le voir dans le tableau 5.24, notre modèle semble avoir des difficultés à converger vers le minimum, ce qui explique les faibles résultats que nous avons obtenus dans cette première expérience (54.46% pour l'ensemble de validation et 61.83% pour l'ensemble d'entrainement).

La faible performance de notre modèle dans cette première expérimentation peut être expliqué par le fait que l'entrainement d'un modèle de plongement de mots à partir de zéro est un problème difficile pour deux raisons principales :

— Pas assez de donnée ou la sparcité des données.

— Un grand nombre de paramètres entraînables.

Pour notre cas, la première raison est la plus probable. Notre corpus contient 1900 critiques seulement, ce qui rend la capture de la signification sémantique et syntaxique difficile, ce qui affecte la qualité des représentations mots. Par contre, entrainer un modèle de plongement de mots sur un plus grand corpus avec un plus grand nombre de données d'apprentissage donne généralement de meilleures représentations.

5.3.3.3 Cas dialecte : deuxième expérimentation

En prenant en considération les résultats de la première expérimentation, dans la deuxième, nous avons décidé de tester le même modèle de plongement de mots qui est FastText mais cette fois en utilisant un modèle pré-entainé au lieu de l'entrainer nous même. Le tableau 5.25 montre les résultats du modèle après chaque epoch.

epoch	loss	accuracy	val_loss	val_accu
1	0.6285	0.6121	0.4213	0.7895
2	0.3887	0.8226	0.4061	0.8070
3	0.3217	0.8693	0.4205	0.8070

TABLE 5.25 – Les valeurs métriques à la fin de chaque époque durant la phase d'apprentissage

En utilisant un modèle pré-entrainé, nous remarquons que les résultats sont améliorés d'une façon significative. nous avons obtenu un accuracy de 86.93% et 80.70% pour l'ensemble d'entrainement et l'ensemble de validation respectivement, comparé à 61.83% et 54.46% pour les mêmes ensembles dans la première expérimenation.

En ce qui concerne l'ensemble de test, nous avons obtenu une accuracy de 82.70%, ce qui est pas mal vu la taille de notre corpus. Le tableau 5.26 montre le résultat obtenu pour chaque métrique de l'évaluation de modèle sur cet ensemble.

	Accuracy	Précision	Rappel	F1-score
Résultat (%)	82.70	82.38	87.91	85.06

TABLE 5.26 – Les résultats du modèle sur l'ensemble de test

Le tableau 5.27 présente la matrice de confusion. En général, nous remarquons que notre modèle performe plutôt bien, mais si nous regardons un peu plus près les résultats de cette matrice, nous allons remarquer que notre modèle a du mal à prédire la classe négative avec 28 critiques prédit positifs alors qu'ils sont négatifs, soit 23.93% de la classe négative. Cependant, le modèle performe mieux sur la classe positive, avec seulement 18 critiques prédit négatifs alors qu'ils sont positifs, soit 12.08% de la classe positive. Cela probablement dû au fait que nous avons un corpus déséquilibré. Il arrive souvent dans les cas de classification que l'une des classes soit minoritaire par rapport à la population globale. Les classes avec des difficultés de prédiction sont généralement celles avec des proportions faibles dans le corpus d'apprentissage par rapport aux classes bien prédites.
Il existe différentes techniques permettant d'équilibrer les corpus. Nous citons par exemple le souséchantillonnage (undersampling) et le sur-échantillonnage (oversampling).

		Classes prédites	
		Négatif	Positif
Classes de référence	Négatif	89	28
	Positif	18	131

TABLE 5.27 – Matrice de confusion sur l'ensemble de test

5.3.3.4 Cas dialecte : troisème expérimentation

Les tableaux 5.28, 5.30 et 5.32 montrent les résultats des trois exécutions, tandis que les tableaux 5.29, 5.31 et 5.33 présentent les matrices de confusion. Dans le cadre de cette expérimentation, nous remarquons que les résultats de la première et la troisième exécution présentés dans les matrices de confusion 5.29 et 5.33 sont quasiment identiques. Nous remarquons aussi que les résultats obtenus après ces deux exécutions sont meilleurs dans la prédiction de la classe négative avec 33 et 32 critiques respectivement prédit positifs alors qu'ils sont négatifs, soit 28.20% et 27.35% respectivement de la classe négative, comparativement à 39 critiques prédit positifs après la deuxième exécution alors qu'ils sont négatifs, soit 33.33% de la classe positive. Cependant, le tableau 5.30 montre que les résultats de la 2ème exécution sont légèrement meilleurs dans la prédiction de la classe positifs avec 10 critiques prédit négatifs alors qu'ils sont positifs, soit 6.71% de la classe positive comparé à 9.39% et 8.72% après la 1ère et la 3ème exécution respectivement.

	Accuracy	**Précision**	**Rappel**	**F1-score**
Résultat (%)	82.33	80.35	90.60	85.17

TABLE 5.28 – Les résultats de la première exécution sur l'ensemble de test

		Classes prédites	
		Négatif	Positif
Classes de référence	Négatif	84	33
	Positif	14	135

TABLE 5.29 – Matrice de confusion de la première exécution sur l'ensemble de test

	Accuracy	**Précision**	**Rappel**	**F1-score**
Résultat (%)	81.57	78.20	92.60	85.25

TABLE 5.30 – Les résultats de la deuxième exécution l'ensemble de test

		Classes prédites	
		Négatif	Positif
Classes de référence	Négatif	78	39
	Positif	10	139

TABLE 5.31 – Matrice de confusion de la deuxième exécution sur l'ensemble de test

	Accuracy	Précision	Rappel	F1-score
Résultat (%)	83.08	75.2	92.23	83.23

TABLE 5.32 – Les résultats de la troisième exécution sur l'ensemble de test

		Classes prédites	
		Négatif	Positif
Classes de référence	Négatif	85	32
	Positif	13	136

TABLE 5.33 – Matrice de confusion de la troisième exécution sur l'ensemble de test

5.3.3.5 Cas dialecte : quatrième expérimentation

Les tableaux 5.34 et 5.35 montrent le résultat obtenu pour chaque métrique de l'évaluation de modèle et la matrice de confusion sur l'ensemble de test respectivement. Nous remarquons dans cette expérimenation que le modèle a moins du mal à prédire la classe négative avec 12 critiques prédit positifs alors qu'ils sont négatifs, soit 10.71% de la calsse négative.

	Accuracy	Précision	Rappel	F1-score
Résultat (%)	90.31	91.72	91.09	91.40

TABLE 5.34 – Les résultats du modèle sur l'ensemble de test

		Classes prédites	
		Négatif	Positif
Classes de référence	Négatif	100	12
	Positif	13	136

TABLE 5.35 – Matrice de confusion sur l'ensemble de test

5.3.3.6 Cas dialecte : cinquième expérimentation

Les résultats obtenus pour chaque métrique de l'évaluation de modèle et la matrice de confusion sur l'ensemble de test sont présentés dans les tableaux 5.36 et 5.37 respectivement. Dans cette expérimentation, nous remarquons que les résultats ne sont pas très différents des autres expérimentations pour ce qui concerne la prédiction de la classe positive. En ce qui concerne la prédiction de la classe négative nous remarquons que les résultats sont un peu meilleurs par rapport aux résultats obtenus dans la 2ème et la 3ème expérimentation, avec 19 critiques prédit positifs alors qu'ils sont négatifs, soit 17.17% de la classe négative.

	Accuracy	Précision	Rappel	F1-score
Résultat (%)	86.46	87.89	89.03	88.46

TABLE 5.36 – Les résultats du modèle sur l'ensemble de test

		Classes prédites	
		Négatif	Positif
Classes de référence	Négatif	92	19
	Positif	17	138

TABLE 5.37 – Matrice de confusion sur l'ensemble de test

5.3.3.7 Cas dialecte : sixième expérimentation

Les résultats de cette dernière expérimentaion sont dans les deux tbleaux 5.38 et 5.39 ci-dessous.

	Accuracy	Précision	Rappel	F1-score
Résultat (%)	90.52	90.75	93.91	92.30

TABLE 5.38 – Les résultats du modèle sur l'ensemble de test

		Classes prédites	
		Négatif	Positif
Classes de référence	Négatif	64	11
	Positif	7	108

TABLE 5.39 – Matrice de confusion sur l'ensemble de test

Nous récapitlons, dans le tableau 5.40, les résultats obtenus dans les différentes expérimentations.

Récapitulation des résultats					
		Accuracy (%)	**Précision (%)**	**Rappel (%)**	**F1-score (%)**
Expérimentation N°1		54.46	-	-	-
Expérimentation N°2		82.70	82.38	87.91	85.06
Expérimentation N°3	**Exécution 1**	82.33	80.35	90.60	85.17
	Exécution 2	81.57	78.20	92.60	85.25
	Exécution 3	83.08	75.2	92.23	83.23
Expérimentation N°4		**90.31**	**91.72**	**91.09**	**91.40**
Expérimentation N°5		86.46	89.03	89.03	88.46
Expérimentation N°6		**90.52**	**90.07**	**93.91**	**92.30**

TABLE 5.40 – Récapitulation des résultats des modèles dialecte

5.4 Conclusion

Dans ce chapitre, nous avons commencé en premier lieu par présenter l'environnement d'expérimentations ainsi que les différentes configurations, les langages de programmations et les ressource matérielles que nous avons utilisées, puis nous avons présenté les différents hyperparamètres que nous avons utilisés pour les tests que nous avons effectués sur les différents corpus. Enfin, nous avons présenté et discuté les résultats obtenus par les différentes architectures que nous avons testées.

Conclusion et perspectives

Dans ce travail nous voulions réaliser un système d'analyse d'opinion multilingual afin de répondre à un problème majeur dans le métier du Community manager qui s'agit d'analyser manuellement les commentaires laissés par les consommateurs sur des pages Facebook de marques algériennes pour identifier leurs polarités (positive ou négative). Ce problème représente une tâche qui consomme pas mal du temps et d'énergie. De ce fait, et dans le but d'offre une solution à cette problématique, nous avons exploité différentes méthodes, techniques et approches utilisées dans le domaine de TALN et d'apprentissage automatique, notamment celles basé sur l'apprentissage profond, principalement les réseaux de neurones récurrents et les réseaux de neurones convolution. Nous avons également expérimenté différentes approches qui représente aujourd'hui l'état de l'art pour la tâche d'analyse des sentiments

Notre travail a porté sur deux langues, l'anglais et le dialecte algérien. Premièrement, nous avons met en avant différentes méthodes de word embedding (la couche embedding, Word2Vec, Glove et FastText) les plus utilisés pour la tâche de l'analyse des sentiments. Nous avons vu l'importance de ces techniques et leurs influences sur les résultats obtenus par le modèle. Deuxièmement, nous avons présenté différentes architectures neuronales pour l'analyse des sentiments, il s'agit de l'architecture des réseaux de neurones récurent et les réseaux de neurones convolutifs. Dans un premier temps, c'est deux architectures ont été testés en utilisant différentes méthodes de plongement de mots notamment Word2Vec et Glove. Afin de trouver le modèle avec le meilleur résultat, nous avons testé un ensemble différent de paramètres plusieurs fois. Dans un deuxième temps, afin de gérer la complexité du dialecte, nous avons proposé un modèle FastText que nous avons entraîné nous-même en utilisant notre corpus de domaine. Comme nous l'avons fait pour l'anglais, nous avons également réaliser plusieurs tests, avec plusieurs architectures telles que Bi-LSTM, CNN et LSTM, avec différents paramètres dans le but d'avoir une meilleure performance de classification pour un dialecte assez complexe comme le dialecte algérien.

La contribution exposée dans ce travail réside dans le fait que nous avons réussi à obtenir de bon résultat concernant le dialecte algérien qui représente un réel challenge dans ce genre de travaux à cause de sa complexité qui se montre clairement dans sa riche morphologie,

son lexique et ses règles linguistiques appropriées qui ne suivent aucune règle grammaticale. Sachant que pour atteindre un niveau de précision acceptable, un modèle de Deep Learning nécessitent l'accès à d'immenses quantités de données d'entraînement, cependant avoir accès à des ressources de données suffisante pour pouvoir entraîner un modèle d'apprentissage profond pour un dialecte reste une tâche difficile.

Enfin, selon différentes expérimentations que nous avons réalisées, nous pouvons tirer les conclusions suivantes :

— Il n'y pas de règle fixe pour choisir les paramètres d'entrainement du modèle,

— La qualité et la quantité des données ont une grande influence sur les performances obtenues par le modèle

— Entraîner son propre modèle de word embedding tels que Word2Vec, Glove ou FastText nécessitent beaucoup de données pour avoir de bonnes représentations de mots

Bibliographie

[1] (2020). Visualizing named entities. https://notebook.community/rishuatgithub/MLPy/nlp/UPDATED_NLP_COURSE/02-Parts-of-Speech-Tagging/03-Visualizing-NE.

[2] Abdelli, A., Guerrouf, F., Tibermacine, O., and Abdelli, B. (2019). Sentiment analysis of arabic algerian dialect using a supervised method. In *2019 International Conference on Intelligent Systems and Advanced Computing Sciences (ISACS)*, pages 1–6. IEEE.

[3] Abdulla, N. A., Ahmed, N. A., Shehab, M. A., and Al-Ayyoub, M. (2013). Arabic sentiment analysis : Lexicon-based and corpus-based. In *2013 IEEE Jordan conference on applied electrical engineering and computing technologies (AEECT)*, pages 1–6. IEEE.

[4] Ahuja, R., Chug, A., Kohli, S., Gupta, S., and Ahuja, P. (2019). The impact of features extraction on the sentiment analysis. *Procedia Computer Science*, 152 :341–348.

[5] Al-Hajjar, D. and Syed, A. Z. (2015). Applying sentiment and emotion analysis on brand tweets for digital marketing. In *2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, pages 1–6.

[6] Al Omari, M., Al-Hajj, M., Hammami, N., and Sabra, A. (2019). Sentiment classifier : Logistic regression for arabic services' reviews in lebanon. In *2019 international conference on computer and information sciences (iccis)*, pages 1–5. IEEE.

[7] Baccouche, A., Garcia-Zapirain, B., and Elmaghraby, A. (2018). Annotation technique for health-related tweets sentiment analysis. In *2018 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, pages 382–387. IEEE.

[8] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv :1607.04606*.

[9] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5 :135–146.

[10] Brill, E. (2000). Part-of-speech tagging. *Handbook of natural language processing*, pages 403–414.

[11] Caetano, J., Seixas Lima, H., Santos, M., and Marques-Neto, H. (2018). Using sentiment analysis to define twitter political users' classes and their homophily during the 2016 american presidential election.

[12] Chamlertwat, W., Bhattarakosol, P., Rungkasiri, T., and Haruechaiyasak, C. (2012). Discovering consumer insight from twitter via sentiment analysis. *J. Univers. Comput. Sci.*, 18(8) :973–992.

[13] Chapelle, O. and Vapnik, V. (1999). Model selection for support vector machines. *Advances in neural information processing systems*, 12 :230–236.

[14] Clark, E. M., James, T., Jones, C. A., Alapati, A., Ukandu, P., Danforth, C. M., and Dodds, P. S. (2018). A sentiment analysis of breast cancer treatment experiences and healthcare perceptions across twitter. *arXiv preprint arXiv :1805.09959*.

[15] Croce, D., Castellucci, G., and Basili, R. (2016). Injecting sentiment information in context-aware convolutional neural networks. In *7th Italian Information Retrieval Workshop, IIR 2016*, volume 1653. CEUR-WS.

[16] de la Fuente-Tomás, L., Sierra, P., Sanchez-Autet, M., Arranz, B., García-Blanco, A., Safont, G., and García-Portilla, M. P. (2020). A clinical staging model for bipolar disorder : longitudinal approach. *Translational psychiatry*, 10(1) :1–9.

[17] Deng, L. and Yu, D. (2014). Deep learning : Methods and applications. *Found. Trends Signal Process.*, 7(3–4) :197–387.

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

[19] Dubreil, E. (2006). *La dimension argumentative des collocations textuelles en corpus électronique spécialisé au domaine du TAL(N)*. Theses, Université de Nantes.

[20] Duque, T. (2020). Nlp preprocessing pipeline . what, when, why ? `https://medium.com/analytics-vidhya/nlp-preprocessing-pipeline-what-when-why-2fc808899d1f`.

[21] Education, B. I. C. (2020). What are recurrent neural networks ? `https://www.ibm.com/cloud/learn/recurrent-neural-networks`.

[22] El-Din, D. M. (2016). Enhancement bag-of-words model for solving the challenges of sentiment analysis. *International Journal of Advanced Computer Science and Applications*, 7(1).

[23] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2) :179–211.

[24] Goeuriot, L. (2009). *Découverte et caractérisation des corpus comparables spécialisés*. Theses, Université de Nantes.

[25] Grave, E., Bojanowski, P., Gupta, P., Joulin, A., and Mikolov, T. (2018). Learning word vectors for 157 languages. *arXiv preprint arXiv :1802.06893*.

[26] Grishman, R. and Sundheim, B. (1995). Design of the muc-6 evaluation. In *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, page 1–11, USA. Association for Computational Linguistics.

[27] Harris, Z. S. (1954). Distributional structure. *Word*, 10(2-3) :146–162.

[28] Hermanto, A., Adji, T. B., and Setiawan, N. A. (2015). Recurrent neural network language model for english-indonesian machine translation : Experimental study. In *2015 International conference on science in information technology (ICSITech)*, pages 132–136. IEEE.

[29] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8) :1735–1780.

[30] Holzinger, A., Malle, B., Kieseberg, P., Roth, P. M., Müller, H., Reihs, R., and Zatloukal, K. (2017). Machine learning and knowledge extraction in digital pathology needs an integrative approach. In *Towards Integrative Machine Learning and Knowledge Extraction*, pages 13–50. Springer.

[31] Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1) :106–154.

[32] Hughes, T. and Mierle, K. (2013). Recurrent neural networks for voice activity detection. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7378–7382. IEEE.

[33] K, A. K. (2020). Training a single perceptron. `https://levelup.gitconnected.com/training-a-single-perceptron-405026d61f4b`.

[34] Kane, B., Jrad, A., Essebbar, A., Guinaudeau, O., Chiesa, V., Quénel, I., and Chau, S. (2021). Cnn-lstm-crf for aspect-based sentiment analysis : A joint method applied to french reviews. In *ICAART (1)*, pages 498–505.

[35] Kasaraneni, C. K. (2020). Understanding nlp pipeline. `https://medium.com/analytics-vidhya/understanding-nlp-pipeline-9af8cba78a56`.

[36] Kauffmann, E., Peral, J., Gil, D., Ferrández, A., Sellers, R., and Mora, H. (2019). Managing marketing decision-making with sentiment analysis : An evaluation of the main product features using text data mining. *Sustainability*, 11(15) :4235.

[37] Kaur, S., Sikka, G., and Awasthi, L. K. (2018). Sentiment analysis approach based on n-gram and knn classifier. In *2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pages 1–4. IEEE.

[38] Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

[39] Laksono, R. A., Sungkono, K. R., Sarno, R., and Wahyuni, C. S. (2019). Sentiment analysis of restaurant customer reviews on tripadvisor using naïve bayes. In *2019 12th International Conference on Information & Communication Technology and System (ICTS)*, pages 49–54. IEEE.

[40] Lilleberg, J., Zhu, Y., and Zhang, Y. (2015). Support vector machines and word2vec for text classification with semantic features. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pages 136–140. IEEE.

[41] Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1) :1–167.

[42] Liu, Y., Yu, X., Liu, B., and Chen, Z. (2014). Sentence-level sentiment analysis in the presence of modalities. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 1–16. Springer.

[43] Ljajić, A. and Marovac, U. (2019). Improving sentiment analysis for twitter data by handling negation rules in the serbian language. *Computer Science and Information Systems*, 16(1) :289–311.

[44] Lundqvist, K., Liyanagunawardena, T., and Starkey, L. (2020). Evaluation of student feedback within a mooc using sentiment analysis and target groups. *International Review of Research in Open and Distributed Learning*, 21(3) :140–156.

[45] Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics : Human language technologies*, pages 142–150.

[46] Mataoui, M., Zelmati, O., and Boumechache, M. (2016). A proposed lexicon-based sentiment analysis approach for the vernacular algerian arabic. *Research in Computing Science*, 110(1) :55–70.

[47] Mazari, A. C. and Djeffal, A. (2021). Deep learning-based sentiment analysis of algerian dialect during hirak 2019. In *2020 2nd International Workshop on Human-Centric Smart Environments for Health and Well-being (IHSH)*, pages 233–236. IEEE.

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

[49] Minaee, S., Azimi, E., and Abdolrashidi, A. (2019). Deep-sentiment : Sentiment analysis using ensemble of cnn and bi-lstm models. *arXiv preprint arXiv :1904.04206*.

[50] Minard, A.-L., Raymond, C., and Claveau, V. (2018). Participation de l'irisa à deft 2018 : classification et annotation d'opinion dans des tweets. In *DEFT 2018-Défi Fouille de Textes*, pages 1–13.

[51] Minsky, M. and Papert, S. A. (2017). *Perceptrons : An introduction to computational geometry*. MIT press.

[52] Monika, R., Deivalakshmi, S., and Janet, B. (2019). Sentiment analysis of us airlines tweets using lstm/rnn. In *2019 IEEE 9th International Conference on Advanced Computing (IACC)*, pages 92–95. IEEE.

[53] Moudjari, L. and Karima, A.-A. (2020). An experimental study on sentiment classification of algerian dialect texts. *Procedia Computer Science*, 176 :1151–1159.

[54] Naeem, S., Logofătu, D., and Muharemi, F. (2020). Sentiment analysis by using supervised machine learning and deep learning approaches. In *International Conference on Computational Collective Intelligence*, pages 481–491. Springer.

[55] Odhiambo, J., Weke, P., and Ngare, P. (2021). A deep learning integrated cairns-blake-dowd (cbd) sytematic mortality risk model. *Journal of Risk and Financial Management*, 14(6) :259.

[56] Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up ? sentiment classification using machine learning techniques. *arXiv preprint cs/0205070*.

[57] Paredes-Valverde, M. A., Colomo-Palacios, R., Salas-Zárate, M. d. P., and Valencia-García, R. (2017). Sentiment analysis in spanish for improvement of products and services : A deep learning approach. *Scientific Programming*, 2017.

[58] Pavlopoulos, I. (2014). Aspect based sentiment analysis. *Athens University of Economics and Business*.

[59] Pednault, E. P. (1997). *Statistical learning theory*. Citeseer.

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

[61] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv :1802.05365*.

[62] Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., Al-Smadi, M., Al-Ayyoub, M., Zhao, Y., Qin, B., De Clercq, O., et al. (2016). Semeval-2016 task 5 : Aspect based sentiment analysis. In *International workshop on semantic evaluation*, pages 19–30.

[63] Qazi, A., Raj, R. G., Hardaker, G., and Standing, C. (2017). A systematic literature review on opinion types and sentiment analysis techniques : Tasks and challenges. *Internet Research*.

[64] Rajeswari, A., Mahalakshmi, M., Nithyashree, R., and Nalini, G. (2020). Sentiment analysis for predicting customer reviews using a hybrid approach. In *2020 Advanced Computing and Communication Technologies for High Performance Applications (ACCTHPA)*, pages 200–205. IEEE.

[65] Rani, S. and Kumar, P. (2017). A sentiment analysis system to improve teaching and learning. *Computer*, 50(5) :36–43.

[66] Ratz, A. V. (2020). Classifying data using artificial intelligence k-means clustering algorithm. `https://www.codeproject.com/Articles/5256294/Classifying-Data-Using-Artificial-Intelligence-K-M`.

[67] Rhanoui, M., Mikram, M., Yousfi, S., and Barzali, S. (2019). A cnn-bilstm model for document-level sentiment analysis. *Machine Learning and Knowledge Extraction*, 1(3) :832–847.

[68] Rish, I. et al. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46.

[69] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088) :533–536.

[70] Sarkar, D. (2019). *Text analytics with Python : a practitioners guide to natural language processing*. Apress.

[71] Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11) :2673–2681.

[72] Sharma, A. K., Chaurasia, S., and Srivastava, D. K. (2020). Sentimental short sentences classification by using cnn deep learning model with fine tuned word2vec. *Procedia Computer Science*, 167 :1139–1147.

[73] Sinaga, K. P. and Yang, M.-S. (2020). Unsupervised k-means clustering algorithm. *IEEE Access*, 8 :80716–80727.

[74] Sindhu, I., Daudpota, S. M., Badar, K., Bakhtyar, M., Baber, J., and Nurunnabi, M. (2019). Aspect-based opinion mining on student's feedback for faculty teaching performance evaluation. *IEEE Access*, 7 :108729–108741.

[75] Singh, P. K. and Paul, S. (2021). Deep learning approach for negation handling in sentiment analysis. *IEEE Access*, 9 :102579–102592.

[76] Soumeur, A., Mokdadi, M., Guessoum, A., and Daoud, A. (2018). Sentiment analysis of users on social networks : overcoming the challenge of the loose usages of the algerian dialect. *Procedia computer science*, 142 :26–37.

[77] Suthar, N., Indr, P., and Vinit, P. (2013). A technical survey on dbscan clustering algorithm. *Int. J. Sci. Eng. Res*, 4(5) :1775–1781.

[78] Takaoka, K., Hisamoto, S., Kawahara, N., Sakamoto, M., Uchida, Y., and Matsumoto, Y. (2018). Sudachi : a Japanese tokenizer for business. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

[79] Turney, P. D. and Littman, M. L. (2003). Measuring praise and criticism : Inference of semantic orientation from association. *acm Transactions on Information Systems (tois)*, 21(4) :315–346.

[80] Vapnik, V. and Vapnik, V. (1998). Statistical learning theory wiley. *New York*, 1(624) :2.

[81] Vijayarani, S., Ilamathi, M. J., Nithya, M., et al. (2015). Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, 5(1) :7–16.

[82] Wang, J.-H., Liu, T.-W., Luo, X., and Wang, L. (2018). An lstm approach to short text sentiment classification with word embeddings. In *Proceedings of the 30th conference on computational linguistics and speech processing (ROCLING 2018)*, pages 214–223.

[83] Wu, D. and Fung, P. (1994). Improving chinese tokenization with linguistic filters on statistical lexical acquisition. In *Fourth Conference on Applied Natural Language Processing*, pages 180–181.

[84] Xiang, J., Qiu, Z., Hao, Q., and Cao, H. (2020). Multi-time scale wind speed prediction based on wt-bi-lstm. In *MATEC Web of Conferences*, volume 309, page 05011. EDP Sciences.

[85] Xie, X., Ge, S., Hu, F., Xie, M., and Jiang, N. (2019). An improved algorithm for sentiment analysis based on maximum entropy. *Soft Computing*, 23(2) :599–611.

[86] Xu, G., Meng, Y., Qiu, X., Yu, Z., and Wu, X. (2019). Sentiment analysis of comment texts based on bilstm. *Ieee Access*, 7 :51522–51532.

[87] Zainuddin, N. and Selamat, A. (2014). Sentiment analysis using support vector machine. In *2014 international conference on computer, communications, and control technology (I4CT)*, pages 333–337. IEEE.

[88] Zhang, Y. and Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv :1510.03820*.