RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ MOULOUD MAMMERI DE TIZI OUZOU

FACULTÉ DU GÉNIE ÉLECTRIQUE ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Spécialité SYSTEME INFORMATIQUE

Mémoire

De fin d'étude de master académique

Spécialité : Informatique

Option : système informatique

THÈME

Classification de polarité d'opinions à base d'aspects à l'aide de l'apprentissage profond

RÉALISÉ PAR

KLOUL Nawel

MEDDAH Yasmine

Devant le jury composé de :

Président : M^{r} HAMMACHE Areski-MCB
Examinateur : M^{me} AIT YAKOUB Zina-MCB
Encadreur : M^{r} SAIDANI Fayçal Redha- MCB

ANNÉE UNIVERSITAIRE : 2019 - 2020

REMERCÎMENTS

Nous remercions avant tout dieu tout puissant de nous avoir donné la force, le courage et la patience pour achever ce modeste travail.

Nos remerciement s'adressent, particulièrement à notre promoteur Mr *« SAIDANI Fayçal Redha» d'avoir accepté de diriger ce travail, ainsi pour l'aide et le temps qu'il a bien voulu nous consacrer, son soutien, sa patience et sa disponibilité.*

Nous adressons nos remerciements les plus sincères au membre du jury qui nous font l'honneur de juger notre travail.

Notre profonde gratitude et sincères remerciement vont à tous les Enseignants qui nous ont suivis durant notre parcours d'étude.

Nous tenons à adresser nos vifs remerciements à nos parents pour leur soutien, leur patience et leurs encouragements tout au long de notre parcours universitaire. Qu'ils trouvent dans ces mots les meilleures expressions de remerciement et de respect.

Sans oublier d'exprimer notre reconnaissance envers nos frères et sœurs, et nos amis intimes pour leur encouragement.

DÉDICACES

En toute modestie et en toute reconnaissance, je dédie tous mes efforts traduits dans ce Mémoire :

A

Mes chers parents qui m'ont offert la vie et orienté mes pas, et qui continuent à me guider vers le chemin de la réussite.

Mes grands-parents.

Mes sœurs Sabrina, Nadjet, Imane.

Mon frère Rayan.

Ma binôme YASMINE et sa famille.

Mes amis et a toute ma famille et à tous ceux qui m'aime.

Nawel

Je dédie ce travail à mes chers parents Pour tout l'amour dont vous m'avez entouré pour ce qu'avez fait pour moi, je ferai de mon mieux pour rester un sujet de fierté à vos yeux avec l'espoir de ne jamais vous décevoir Que Dieu vous garde et vous protège.

A mes chères et adorables sœurs Zahia, Zoubida, Farida, Saliha, Mekioussa

A mon cher frère Sofiane

A ma cher binôme et sœur Nawel

A tous mes amis.

Yasmine.

Table des matières

Introduction générale 8
Chapitre 1 : Introduction à l'Analyse d'opinion 10
1.1. Introduction 11
1.2. Processus typique de l'analyse d'opinion 11
1.3. Classification de l'analyse d'opinions 13
1.3.1. Classification de l'analyse d'opinions selon les tâches 13
1.3.2. Classification de l'analyse d'opinions selon la granularité de l'analyse 14
1.3.3. Classification de l'analyse d'opinions selon les approches 15
1.4. Les domaines d'application de l'analyse d'opinion 19
1.5. Exemple d'application 20
1.6. Conclusion 22
Chapitre 2 : L'Apprentissage Profond pour l'Analyse d'Opinion 23
2.1. Introduction 24
2.2. Les réseaux de neurones artificiels 25
2.2.1. Le modèle du perceptron 26
2.2.2. Étapes d'apprentissage d'un réseau de neurones 28
2.3. Modèles du Deep Learning 36
2.3.1. Couches usuelles des réseaux de neurones profonds. 36
2.3.2. Les réseaux de neurones à convolution (CNN) 40
2.3.3. Les réseaux de neurones récurrents 42
2.4. Les représentations de texte usuelles en apprentissage profond 48
2.4.3. Représentation vectorielle de documents 55
2.5. Conclusion 55
Chapitre 3 : Description de l'approche ABSA 56
3.1. Introduction 57
3.2. L'Analyse des Sentiments Basée sur l'Aspect (-ABSA-) 58
3.2.2. Sous tâches de l'ABSA 59
3.2.1. Travaux connexes 61
3.3. Présentation introductive de notre approche d'analyse d'opinions à base d'aspects 63
3.3.1. Présentation du corpus utilisé 63
3.3.2. Sous tâches du corpus SemEval 65
3.4. Intuition de l'approche 65
3.5. Description des paramétrages utilisés pour l'approche 68
3.6. Conclusion 70

Chapitre 4 : Réalisation 71
4.1. Introduction 72
4.2. Présentation des Frameworks utilisés 72
4.3. Présentation des expérimentations réalisées et résultats obtenus 73
4.3.1. Expérimentation 1 : Etudes de l'impact des représentations de textes sur le modèle d'apprentissage utilisé. 73
4.3.2. Expérimentation 2 : Analyse d'opinion à base d'aspect et présentation des résultats de notre approche. 77
4.3.3. Présentation d'une perspective éventuelle d'amélioration de l'approche 80
4.4 Interface web 82
4.5. Conclusion 83
Conclusion générale 84

Liste des figures

FIGURE 1 : PROCESSUS D'ANALYSE D'OPINION 13
FIGURE 2 : NIVEAUX D'ANALYSE D'OPINION 15
FIGURE 3 : LES DOMAINES D'APPLICATION DE L'ANALYSE D'OPINION 19
FIGURE 4 : APPLICATION D'ACHAT EN LIGNE FLIPKART 20
FIGURE 5 : PAGE D'ACCUEIL DU SITE SENTIMENTS VIZ 21
FIGURE 6: PAGE D'ACCUEIL DU SITE QUOT'&VOUS 22
FIGURE 7: PROCESSUS DES MODELES D'APPRENTISSAGE AUTOMATIQUE CLASSIQUE VERSUS APPRENTISSAGE PROFOND (A.I. TECHNICAL - MACHINE LEARNING VS. DEEP LEARNING 2019) 25
FIGURE 8: ILLUSTRATION DU PERCEPTRON 27
FIGURE 9: ILLUSTRATION DU PERCEPTRON MULTICOUCHE 28
FIGURE 10: POIDS DES CONNEXIONS SYNAPTIQUE DU IEME NEURONE DE LA LEME COUCHE ISSUES DU JEME NEURONE DE LA $(L-1)$ IEME COUCHE PRECEDENTE 29
FIGURE 11: SCHEMA DE LA RETRO PROPAGATION DU GRADIENT DE L'ERREUR DEPUIS LA COUCHE DE SORTIE JUSQU'AUX POIDS DES CONNEXIONS SYNAPTIQUE DU IEME NEURONE DE LA LEME COUCHE ISSUES DU JEME NEURONE DE LA $(L-1)$ IEME COUCHE PRECEDENT 31
FIGURE 12: GRAPHIQUE DE LA FONCTION SIGMOÏDE. 34
FIGURE 13: GRAPHIQUE DE LA FONCTION TANGENTE HYPERBOLIQUE 35
FIGURE 14: GRAPHIQUE DE LA FONCTION RELU [3] 35
FIGURE 15: REPRESENTATION GENERALE DES CARTES DE CARACTERISTIQUES 38
FIGURE 16: SCHEMA DE GLISSEMENT D'UNE FENETRE DE FILTRE SUR UN EXEMPLE D'IMAGE D'ENTREE (A.I. TECHNICAL - MACHINE LEARNING VS. DEEP LEARNING 2019) 39
FIGURE 17: LES PARTIES DU CNN 40
FIGURE 18 : MODELE CNN DE BASE AVEC 4 COUCHES. 41
FIGURE 19: MODELE DE BASE DU RESEAU RNN [35] 42
FIGURE 20: CELLULE LSTM [39] 44
FIGURE 21: ETAT DE LA CELLULE FIGURE 22: MODELISATION D'UNE PORTE LSTM 45
FIGURE 23: PORTE D'OUBLIE 45
FIGURE 24: MISE A JOUR DE LA CELLULE 46
FIGURE 25: MISE A JOUR DE L'ANCIEN ETAT DE CELLULE 46
FIGURE 26: LA SORTIE DE RESEAU LSTM 47
FIGURE 27 : UNITE RECURRENTE FERMEE 47
FIGURE 28: ARCHITECTURE EN NEURONES POUR LA MODELISATION DU LANGAGE [55] 51
FIGURE 29: L'ARCHITECTURE CBOW PREDIT LE MOT COURANT EN FONCTION DU CONTEXTE, ET SKIP-GRAM PREDIT LES MOTS EN CONTEXTE EN FONCTION DU MOT COURANT. 53
FIGURE 30: APERÇU D'UN COMMENTAIRE DU CORPUS SEMEVAL 2016 64
FIGURE 31: SCHEMA REPRESENTATIF DES DIFFERENTES ETAPES DE L'ANALYSE D'OPINION A BASE D'ASPECTS 66
FIGURE 32: SCHEMA REPRESENTATIF DE L'ARCHITECTURE DU MODELE LSTM 67
FIGURE 33: GRAPHE REPRESENTE LES DEFERENTES VALEURS DE F1_SCORE EN UTILISANT DIFFERENTES VALEURS D'EPOQUES 75
FIGURE 34: MODELE LSTM 79
FIGURE 35: GRAPHE D'AUGMENTATION DE F1_SCORE EN FONCTION DE NOMBRE D'EPOQUE 80
FIGURE 36: PAGE INDEX 82
FIGURE 37: PAGE RESULTAT 83

Liste des tableaux

TABLEAU 1: TRAVAUX EFFECTUES SUR LE CORPUS SEMEVAL 64
TABLEAU 2: TABLEAU COMPARATIF ENTRE LES DIFFERENTES TECHNIQUES DE REPRESENTATIONS 75
TABLEAU 3: LES RESULTATS CLASSIFICATION DES DEFERENTES CATEGORIES 78
TABLEAU 4: TABLEAU DES RESULTATS OBTENUS POUR LA CLASSIFICATION DES POLARITES 80

Introduction générale

L'analyse d'opinions est une tâche issue du domaine de traitement automatique du langage naturel (TALN) et qui consiste en l'identification et la classification des textes subjectifs en plusieurs catégories d'opinions (polarités).

Beaucoup de travaux se sont penchés sur cette problématique, en prenant le problème sous différents angles (principalement statistique et/ou linguistique). Néanmoins, on retrouve deux grandes catégories de travaux ; ceux reposant sur des approches à base de lexiques et ceux qui exploitent les techniques d'apprentissage automatique pour y opérer une classification. La tâche d'analyse d'opinions opère à plusieurs niveaux de granularité. En effet, on s'intéresse soit aux opinions au niveau des phrases (commentaires sur le web, tweet etc...) soit à des avis et des prises de positions exprimés au sein de textes long ou ensemble de textes. Les objectifs recherchés varient également selon qu'on cherche à distinguer les textes subjectifs, des textes factuels où alors en recherchant à classer les textes opiniâtres selon leur polarité (positive, négative et neutre) ou sur une échelle plus large (très négative, négative, neutre, positive et très positive) afin de jauger l'intensité de l'opinion.

Depuis peu, une autre variante de l'analyse d'opinion nommer « analyse de sentiment à base d'aspects » commence peu à peu à susciter de l'intérêt. Ces opinions basées sur des aspects ont été le sujet d'un important nombre de travaux et ont également fait l'objet de plusieurs challenges dans la campagne d'évaluation SemEval. Cette tâche consiste à déterminer la polarité associée à chaque aspect (caractéristique) d'une cible donnée. Elle est considéré comme étant légèrement plus complexe qu'une analyse d'opinion classique, du fait que, les aspects d'une cible ne sont généralement pas connus a priori, et ils varient d'une cible à une autre.

Etant donné la complexité de la tâche, plusieurs travaux se sont penchés sur l'utilisation d'algorithmes d'apprentissage plus évolués que ceux habituellement utilisés en analyse d'opinion classique. Ces algorithmes dits « d'apprentissage profond ou Deep Learning » repose principalement sur les réseaux de neurones.

Les réseaux de neurones ont prouvé leur efficacité dans divers sous domaines du traitement du langage naturel et en particulier en analyse de sentiment. Dans le cade de notre travail, nous nous sommes intéressée aux réseaux de neurones avec mémoire long terme (LSTM) .Pour se faire nous avons divisé notre approche en deux parties, dans un premier lieu nous

nous sommes intéressées à l'impact de Word Embedding sur la tâche de classification de polarité. Dans un second temps nous avons implémenté une approche d'analyse à base d'aspect en utilisant des réseaux de neurones composés de plusieurs couches afin de de permettre au modèle généré, une meilleure adaptation à notre problématique liée aux aspects.

Ce mémoire est organisé comme suit :

1. Le premier chapitre permet de définir le contexte de l'analyse d'opinion à travers concepts préliminaires liées à d'opinion. Nous avons énuméré les différentes manières de classée l'analyse d'opinion selon les tâches, selon la granularité et selon les approches utilisées, Nous terminons avec une présentation de quelques exemples d'application et usages du domaine.
2. Le deuxième chapitre, présente les principales notions et concepts propre au deep learning et aux réseaux de neurones. Nous avons également présenté quelques modèles couramment utilisé en analyse d'opinion. Enfin, nous terminons avec une brève introduction aux techniques du Word embedding et son utilité pour le deep learning.
3. Le troisième chapitre porte sur l'analyse de sentiment à base d'aspects. ces différentes sous tâche ainsi que quelques travaux connexes au domaine. Nous terminons par une présentation de notre approche.
4. Le quatrième chapitre est composé de deux parties, la première présente les outils utilisés lors de l'implémentation et la réalisation de notre travail. La deuxième partie présente les tests d'évaluation et la discussion des résultats obtenus.

Finalement, nous clôturons ce mémoire par une conclusion générale.

Chapitre 1 : Introduction à l'Analyse d'opinion

1.1. Introduction

Avec l'émergence du Web 2.0, les données textuelles exprimant des informations subjectives (opinions ou sentiments), ne cessent de se multiplier. Ces informations subjectives varient selon la personnalité, les principes et les goûts de chacun. Ainsi, une opinion peut prendre différentes modalités : une opinion positive, négative, neutre ou alors sous forme d'opinion de soutien, de détraction, de doute, de peur, de contentement etc.

L'analyse d'opinions (abrégé : AO), également appelée analyse de sentiments est à juste titre, une tache qui consiste à détecter et classer les données textuelles exprimant ce type d'information subjective. De manière générale, le problème de classification d'opinions, permet d'opposer des classes divergentes comme par exemple objective/subjective, positive/négative/neutre, joie/peur/dégout. Mais, depuis peu, plusieurs travaux s'attachent à réaliser une classification sur des axes d'opinions plus fins à savoir, faiblement positive (resp. négative)/moyennement positive (resp. négative)/fortement positive (resp. négative) [2]

Dans ce chapitre nous présentons les notions liées à l'analyse d'opinions ainsi que les approches les plus utilisées. Nous aborderons également les différents niveaux de granularité d'une analyse d'opinions et puis, nous terminerons par citer les domaines d'applications ainsi que quelques exemples d'application.

1.2. Processus typique de l'analyse d'opinion

Un système d'analyse d'opinions s'appuie, théoriquement, sur un processus en étapes : acquisition, prétraitement, représentation de corpus et classification des données. Toutefois, en fonction de la disponibilité des données, ces étapes peuvent sensiblement variées. En effet, avec l'augmentation constante du nombre de corpus spécialisés pour l'analyse d'opinions, fait que, l'étape d'acquisition des données est de moins en moins traitée par les travaux de recherche sur le sujet. Ainsi, chaque étape consiste à :

Acquisition du corpus : L'objectif de cette phase est d'extraire de manière automatique à partir du web, des documents d'opinions exprimant des avis positifs ou négatifs grâce notamment aux techniques de crawling et de web scraping.

Prétraitement du corpus : étant donné que, les informations disponible sur le web ne sont pas toujours fiable et peuvent être écrites d'une manière incompréhensible. Ainsi, pour améliorer les performances de l'analyse des textes opiniâtres, un prétraitement et un nettoyage de ces textes est indispensable. Cette tâche consiste soit, à l'élimination des doublons, des corrections orthographique, tokénisation, suppression des mots vides etc.

Représentations du corpus et identification des caractéristiques : La représentions des textes est une étape très importante dans le processus de fouille d'opinion, elle nécessite l'utilisation de technique de représentation adéquate notamment pour l'identification des caractéristiques fréquentes et non fréquentes essentiels pour la dernière phase de classification. La représentation couramment utilisée est celle du modèle vectoriel dans laquelle chaque texte est représenté par un vecteur de n termes pondérés. Il existe plusieurs types de représentations possibles pour les données textuelles :

(a) En fonction des approches statistiques :

- **Représentation en sac de mots (bag of words) :** Les textes sont transformés simplement en vecteurs où chaque composante représente un terme (mot). Utiliser les mots a comme avantage d'exclure toute analyse grammaticale pouvant apporter du bruit aux méthodes purement statistiques.
- **Représentation en n-gramme :** Cette méthode consiste à représenter le document par des n-grammes. Un n-gramme est une séquence de *n* (caractères/termes) consécutifs. Cela consiste à découper le texte en plusieurs séquences de *n* (caractères/termes) en se déplaçant avec une fenêtre de *n* éléments.

(b) En fonction des approches lexicales :

- **Représentation en racines lexicales (racinisation)** : Cette méthode consiste à remplacer les mots du document par leurs racines lexicales, et à regrouper les mots de la même racine dans une seule composante.
- **Représentation en lemmes :** La lemmatisation consiste à utiliser l'analyse grammaticale afin de remplacer les verbes par leur forme infinitive et les noms par leur forme au singulier. En effet, Un mot donné peut avoir différentes formes dans un texte, mais leur sémantique reste la même.

Classification des données : Cette dernière phase a pour but de valider l'utilité des termes choisi lors des phases précédentes puis de distinguer de manière automatique l'appartenance de chaque document. Le schéma ci-dessous illustre ces différentes étapes :

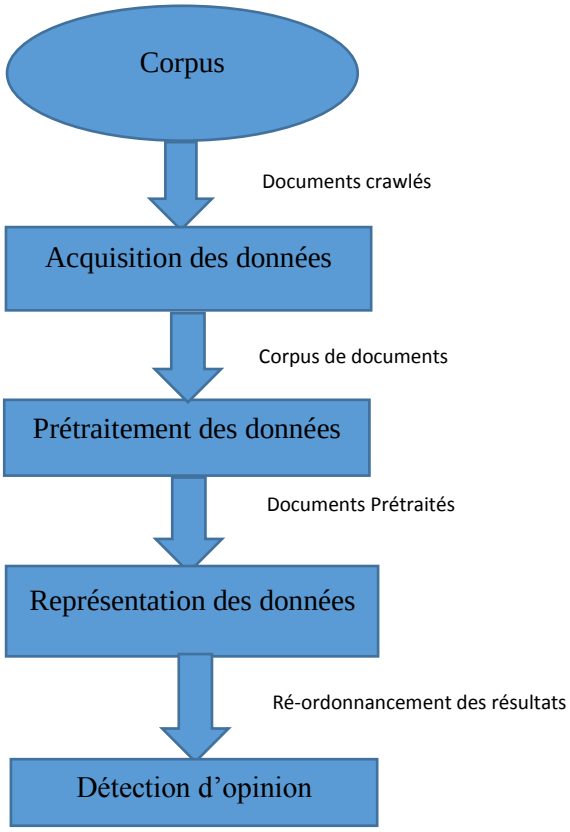


Figure 1 : processus d'analyse d'opinion

1.3. Classification de l'analyse d'opinions

Selon le contexte, l'analyse d'opinions ou de sentiments peut être classée de différentes manières ; un premier classement peut se faire sur la base des techniques de classification utilisées. Ceci peut être divisé en deux approches différentes [11] : celle basée sur l'apprentissage automatique et l'approche basée sur les lexiques. On peut également inclure une troisième classification, dite hybride.

L'AO peut également être classée selon la manière dont les opinions à analyser sont identifiées. Les trois principaux niveaux de classification sont les niveaux de phrase, de document et d'aspect. La dernière manière de faire est de classifier l'AO sur la base de l'objectif à atteindre. En effet, l'étendu de la tâche d'AO varie selon qu'on cherche à déduire le caractère subjectif d'un texte ou alors savoir à quel point un texte est favorable à une opinion (intensité). Toutefois, aucune de ces classifications n'est exclusive.

1.3.1. Classification de l'analyse d'opinions selon les tâches

En fonction de l'objectif à atteindre, on retrouve plusieurs variantes de l'AO : [22]

Détection de la subjectivité : cela consiste à identifier dans un ensemble de données, les textes porteurs d'opinion, ou encore à localiser les passages porteurs d'opinion dans un texte. Plus précisément, on parle ici de classer les textes ou les parties de texte en objectifs ou subjectifs.

Classification de l'axiologie de l'opinion : a pour but d'attribuer une étiquette au texte selon l'opinion qu'il exprime. On considère généralement les classes positives et négatives ou encore positive/négative/neutre.

Classification de l'intensité de l'opinion : le but ici est d'attribuer une étiquette au texte selon l'opinion qu'il exprime avec un degré d'intensité réparti sur une échelle large et précise (très négatif, négatif, neutre, positif et très positif)

Identification de l'objet de l'opinion : cela consiste à rendre l'information rapidement et facilement accessible en mettant en avant les opinions exprimées et les cibles de ces opinions présentes dans le texte.

Identification de la source de l'opinion : consiste à déterminer qui exprime l'opinion.

1.3.2. Classification de l'analyse d'opinions selon la granularité de l'analyse

Les trois principaux degrés d'analyses étudiés en AO sont : au niveau du document, de l'aspect et la phrase. La classification dépend de ces différents niveaux.

Une classification des sentiments au niveau du document cherche à déterminer si le document dans son ensemble a une opinion négative ou positive. En d'autres termes, pour un texte donné, on supposerait que l'ensemble du texte exprime une opinion globale positive ou négative sur une seule entité. Puisque cela suppose qu'il n'y a qu'une seule entité, cette méthode n'est malheureusement pas la plus appropriée pour les textes évaluant plus d'une entité. [18]

Les deux autres types de classifications concernent la phrase et l'aspect. Une analyse au niveau de la phrase est très similaire à celle au niveau du document, à la seule différence que, chaque phrase est analysée individuellement pour voir si elle exprime une opinion négative, neutre ou positive. Ce niveau ajoute plus de flexibilité que le niveau document car il est capable de distinguer les phrases objectives des phrases subjectives, et cela peut être utilisé comme premier filtre à une analyse niveau document. Cependant, il existe des cas particuliers

ou l'on retrouve des phrases objectives exprimant une opinion et des phrases subjectives ne transmettant aucun sentiment, ce qui est malheureusement difficile à gérer.

Et enfin, l'analyse la plus fine concerne les aspects, également appelé niveau de fonctionnalité (features level). À la différence des niveaux de phrase et de document, une analyse sur l'aspect tente de savoir sur quoi porte chaque opinion. La principale différence est que cette analyse trouve une cible pour chaque opinion, au lieu de se concentrer sur les unités linguistiques, comme les phrases, les documents ou les paragraphes. Le but de ce niveau est d'identifier l'opinion ou le sentiment sur les entités et leurs différents aspects. La majorité des systèmes d'analyse des sentiments en temps réel sont basés sur ce niveau d'analyse (Liu 2020). La figure ci-dessous illustre les différents niveaux d'analyses :

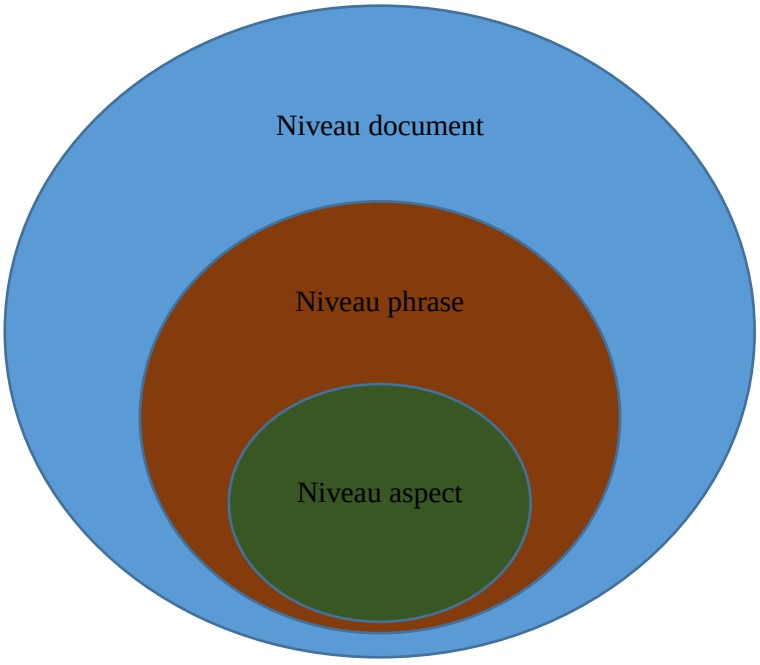


Figure 2 : Niveaux d'analyse d'opinion

1.3.3. Classification de l'analyse d'opinions selon les approches

On retrouve principalement, deux types d'approches en analyse d'opinions, celles basée sur les lexiques et celle basée sur l'apprentissage automatique (Machine Learning). La première approche utilise une liste de mots et d'expressions de sentiment, puis ces derniers permettent de classer les textes en catégories (positif, négatif, neutre …etc.). A contrario, les méthodes de Machine Learning utilisent des algorithmes d'apprentissage pour déterminer l'opinion sans

dépendre d'un quelconque lexique ou base de données de mots, ce qui théoriquement, rend la tâche plus appropriée pour des analyses approfondies. Cependant, ces approches partagent des procédures communes telles que la sélection de caractéristiques, l'intégration des données, le nettoyage des données et le crowdsourcing qui sont des procédures essentielles et indispensables pour améliorer le résultat de l'analyse de sentiments.

1.3.3.1. Approche basée sur les lexiques

Dans ce type d'approche, l'opinion d'un texte est obtenue à l'aide d'une fonction qui tient compte des mots en commun entre ceux du texte et ceux de lexiques spécialisés. Cette fonction peut par exemple, correspondre au nombre de mots positifs divisés par le nombre de négatifs, si le ratio est supérieur à 1, le texte sera positif, s'il est égal à 1, le texte sera considéré comme neutre, sinon il sera considéré comme négatif. Une autre fonction, largement utilisée, est l'orientation sémantique d'adjectifs ou de phrase. C'est cette orientation sémantique qui fera office de mesure de l'opinion au sein du texte. [20]

1.3.3.2. Approche basée sur le Machine Learning non supervisé ou Clustering

De manière générale, l'apprentissage automatique (Machine Learning*)* tente de répondre à deux problématiques, à savoir la classification et la prédiction de données (régression). La classification consiste à mettre en œuvre des algorithmes permettant d'attribuer des catégories à des données (exemple : les catégories d'articles d'informations), tandis que le problème de régression consiste à opérer des prédictions sur des valeurs réelles tel que l'estimation de la valeur future d'une action boursière sur les marchés financiers.

L'analyse d'opinion est considérée comme un problème de classification. Tel que nous l'avons présenté plus haut, ce domaine consiste à analyser puis classer les émotions, les attitudes, les opinions et les sentiments des gens envers différentes entités telles que des produits ou services afin qu'une entreprise sache ce que les gens pensent de ses produits ou services, tandis qu'un particulier aura une idée de ce que les autres pensent d'un certain produit qu'il envisage d'acheter. Par conséquent, l'analyse d'opinions se concentre principalement sur la problématique de classification sur un axe positive ou négative plutôt que de prédire ce que les gens pensent.

A ce propos, les méthodes de classification non supervisées sont employées lorsque les données utilisées ne sont pas étiquetées. Ces méthodes sont généralement basées sur des considérations linguistiques afin de sélectionner les structures grammaticales qui expriment le plus souvent l'opinion et visent à concevoir des modèles structurant l'information. La spécificité dans ce type de méthodes est que les catégories (les classes) des données ne sont pas connues d'avance.

Ainsi, un système d'analyse en clusters prend en entrée un ensemble de données et une mesure de similarité entre ces données, puis l'algorithme produit en sortie un ensemble de clusters. L'algorithme de K-means est parmi ceux les plus utilisés dans ce type d'approches en Analyse d'Opinion.

1.3.3.3. Approche basée sur le Machine Learning supervisé

Les approches basées sur l'apprentissage supervisé visent à distinguer automatiquement des connaissances spécifiques à l'opinion dans de grands ensembles de données, pour ensuite permettre l'extraction d'informations implicites, puis appliquer ces nouvelles connaissances acquises dans le but de prédire de nouvelles informations dans un nouvel ensemble de données.

Les méthodes (classifieurs) utilisées pour ce type d'approche sont réparties en deux catégories : génératives et discriminantes. Les classifieurs discriminants cherchent à modéliser la frontière de décision entre les différentes classes afin d'opérer une discrimination. Les méthodes génératives, quant à elles, modélisent la manière dont les données ont été générées, puis après avoir réalisé une phase d'apprentissage, peuvent être utilisées pour faire des prédictions. À titre d'exemple, Naïve Bayes, considéré comme classificateur probabiliste simple et populaire, est un algorithme génératif appartenant à la famille des classifieurs linéaires. Cet algorithme lorsqu'il est appliqué pour du traitement automatique du texte se base sur des éléments (dits caractéristiques) pour pouvoir modéliser la manière dont les données ont été générées. L'intuition de ce classifieur se base sur le théorème de Bayes et suppose que l'existence d'une caractéristique pour une classe, est indépendante de l'existence d'autres caractéristiques. Ainsi, on y calcule la probabilité à posteriori d'une classe en fonction de la distribution des mots (caractéristiques) dans les documents de la classe.

De manière générale, l'apprentissage supervisé s'appuie sur des classes prédéterminées et sur un certain nombre de paradigmes connus pour mettre en place un système de classement à partir de modèles déjà classées ou étiquetées. Dans ce cas, deux étapes sont nécessaires pour compléter le processus, à commencer par le stade d'apprentissage qui consiste à la modélisation des données cataloguées. Ensuite, on se basera sur les données ainsi définies pour attribuer des classes aux nouvelles données introduites dans le système.

Les réseaux de neurones sont également très répandus en traitement automatique du langage et plus particulièrement en Analyse d'opinion. Au cours de cette dernière décennie, le nombre de travaux ayants intégrés les réseaux de neurones pour l'AO, ont considérablement augmentés. Ces travaux sont pour la plupart répertoriés dans un sous domaine de l'apprentissage automatique nommé apprentissage profond (Deep Learning). Nous aborderons ce domaine dans le chapitre suivant. Ainsi, parmi les premières méthodes supervisées ayant été utilisés en classification de texte, on retrouve les réseaux de neurones convolutionnels (CNN). Une architecture CNN peut être divisée en quatre couches : la première se concentre sur la représentation des phrases ; la seconde est une couche dites convolutive, la troisième réalise ce qu'on appelle du pooling et la dernier est une couche entièrement connectée avec, en sortie, le résultat à prédire.

Plus exactement, la première couche est responsable de la représentation de la phrase à travers des vecteurs de mots. La deuxième couche est l'endroit où se produisent de nombreuses opérations de convolution. Une opération de convolution implique l'utilisation d'un filtre qui est appliqué à chaque fenêtre possible de mots dans la phrase, pour produire ce qu'on appelle « une carte de caractéristiques ». L'avant-dernière couche exécute une opération de mise en pool sur les cartes précédemment calculées par chaque filtre, en prenant la valeur maximale qui correspond à la caractéristique la plus importante de chacun. La dernière couche reçoit les caractéristiques précédentes et les transmet à une couche entièrement connecté dans laquelle tous les nœuds contenus se connectent à tous les nœuds de la couche suivante et dont la sortie est une distribution de probabilité sur les catégories à classer.

En traitement automatique du langage naturel, les CNN se sont avérés très efficaces, atteignant d'ailleurs de bons résultats en analyse sémantique, en recherche d'information ainsi qu'en analyse d'opinion. [6] A ce propos nous tenons à signaler que la thématique de ce mémoire porte justement sur les réseaux de neurones et plus particulièrement les réseaux

récurrents. Par conséquent, nous avons préféré aborder ces notions ainsi que celles de l'apprentissage profond, séparément dans le prochain chapitre.

1.4. Les domaines d'application de l'analyse d'opinion

Comme le rappel Pang Lee « *Opinion Mining and Sentiment Analyse* », le sentiment de « *ce que les autres pensent* » est régulièrement invoqué dans tout processus décisionnel. Que ce soit, en vue de l'achat d'un produit, dans le contexte d'une élection ou encore pour évaluer la réputation d'une marque.

La figure ci-dessous, présente quelques domaines d'application de l'analyse des sentiments.

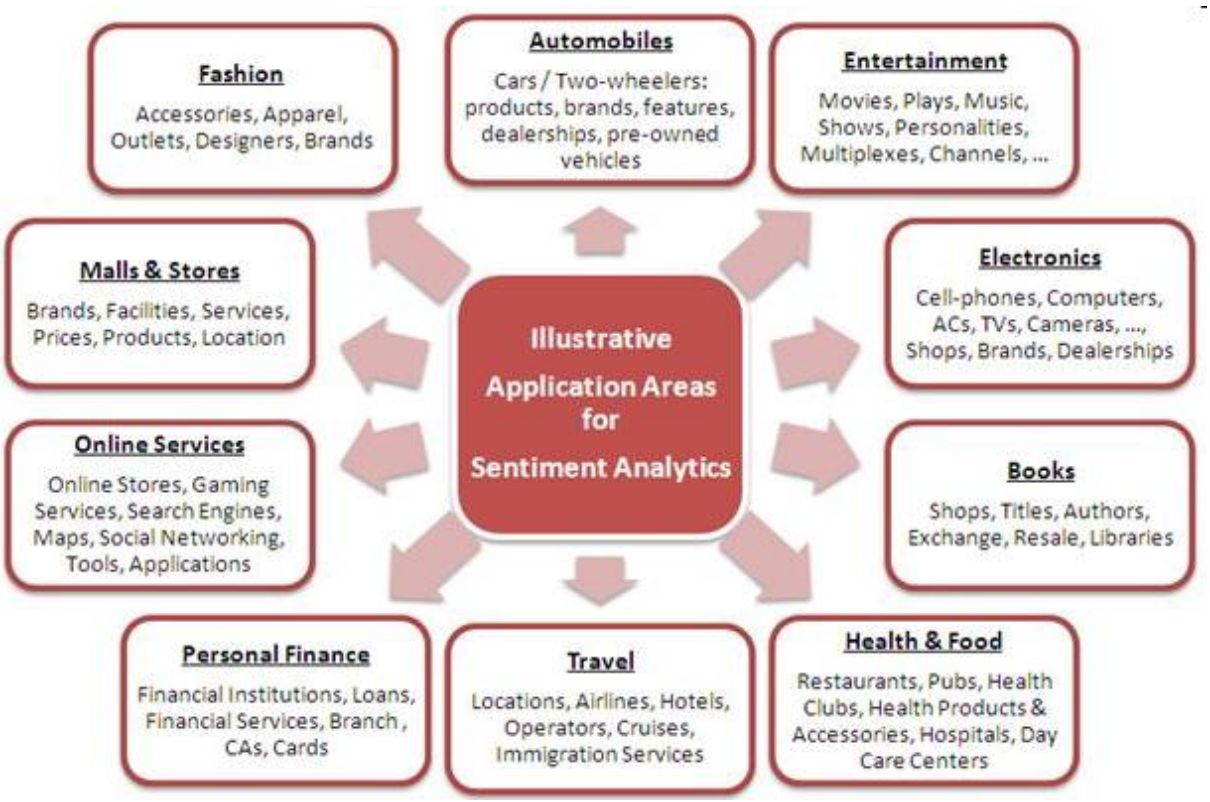


Figure 3 : les domaines d'application de l'analyse d'opinion

1.5. Exemple d'application

Flipkart : Est une application de shopping en ligne crée par une entreprise indienne de Commerce en ligne basée à Bangalore. Cette application utilise le système de classification de produits par étoile afin de les évaluer par les utilisateurs, plus le nombre d'étoile est élevé plus le produit est bon et vice versa. [22]

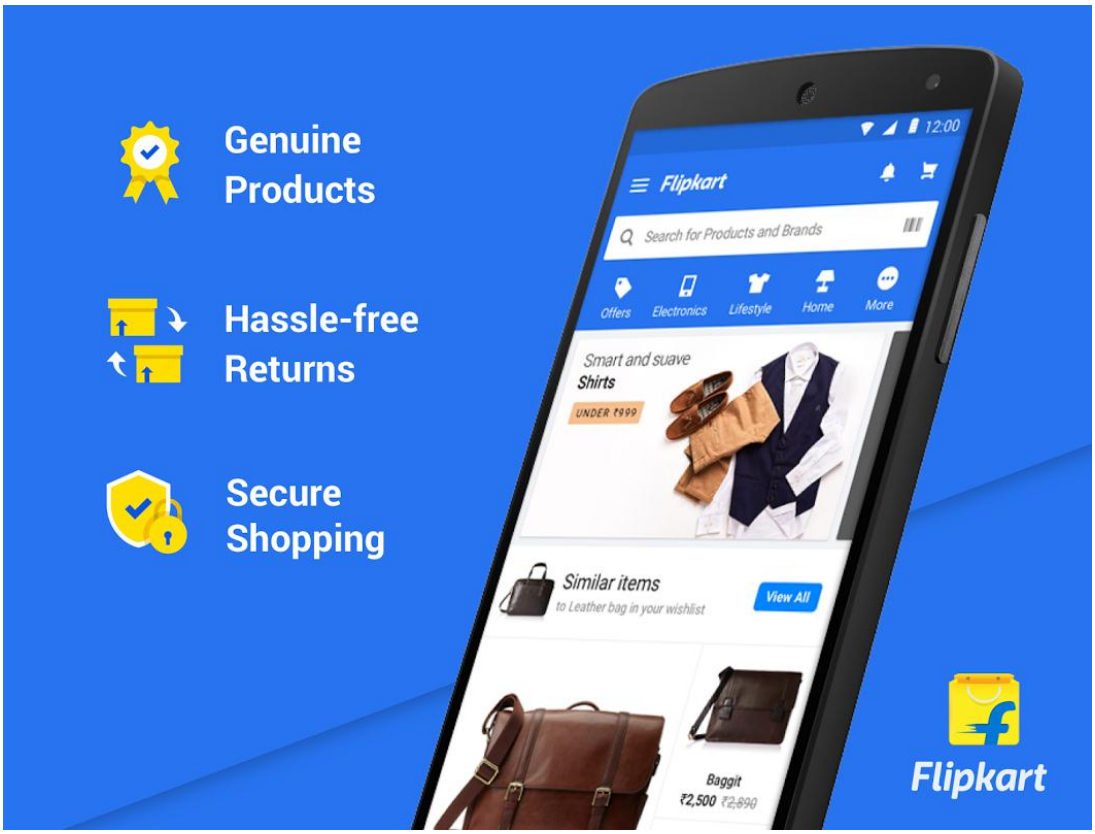


Figure 4 : Application d'achat en ligne Flipkart

Analyse de la tonalité sur Twitter: Le célèbre réseau social et de microblogging Twitter permet à un utilisateur d'envoyer gratuitement de brefs messages, appelés tweets, sur internet, par messagerie instantanée ou par SMS.

Avec un nombre moyen de 500 millions de messages envoyés par jour, Twitter est l'un des lieux privilégiés pour recueillir des opinions spontanées sur des sujets très variées, il existe de nombreux services proposant d'analyser la tonalité des messages partagés sur Twitter, et parmi eux on trouve Twitter Sentiment.

Ce dernier est un outil en ligne gratuit créé par trois étudiants en informatique issue de Stanford. Il s'agit donc d'un projet académique où une timeline est disponible et affiche les courbes de sentiments positifs et négatifs. Un système de retour de pertinence manuel est associé à chacun des résultats et permet d'améliorer le service au fur et à mesure en utilisant l'expertise humaine agrégée des utilisateurs.

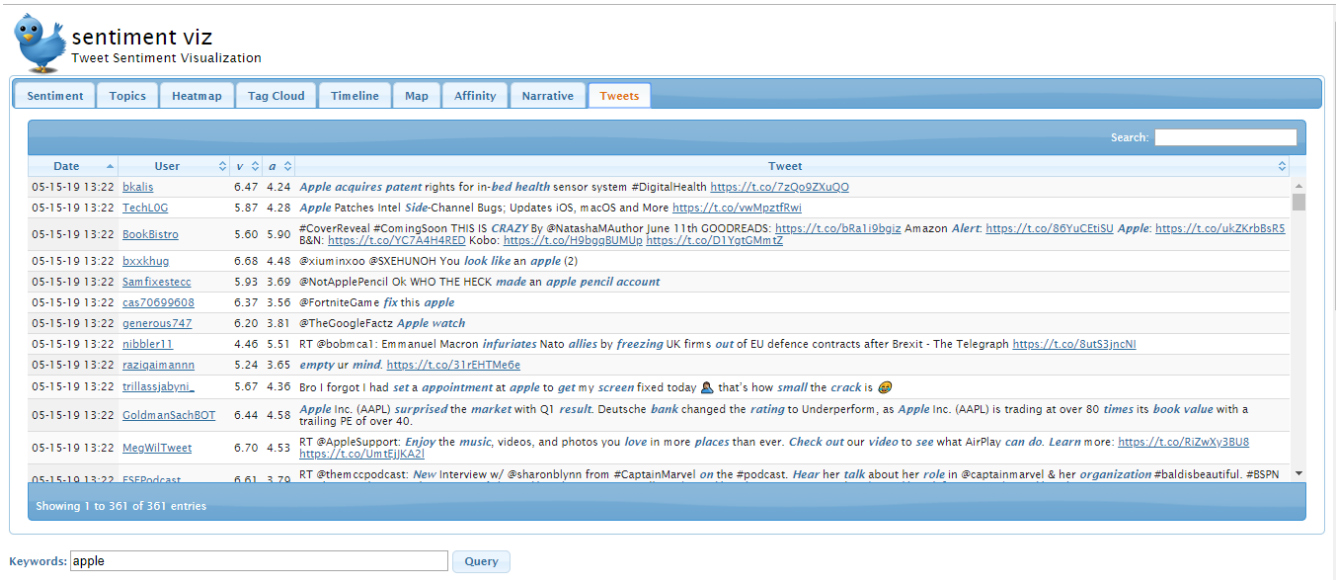


Figure 5 : Page d'accueil du site sentiments viz

Quot'&Vous

Derrière le site internet "Quot'&Vous" se trouve l'entreprise TNS Sofres. Il s'agit du deuxième groupe mondial d'étude de marché et d'opinion après sa fusion en 2008 avec l'un de ses concurrents, le groupe Kantar 2.

Le site Quot'&Vous permet à cette entreprise d'étendre leur immense panel de consommateurs en proposant de répondre à des questionnaires pour donner leurs opinion au sujet d'un produit, d'un service ou de leurs propres habitudes de consommation.

A chaque sondage auquel l'utilisateur répond des points lui seront attribués et lui permettront de participer à un tirage au sort mensuel. Le principe est simple, plus on répond à des questionnaires, plus on obtient de points et donc plus on augmente les chances de gagner un lot lors du tirage au sort. [22]

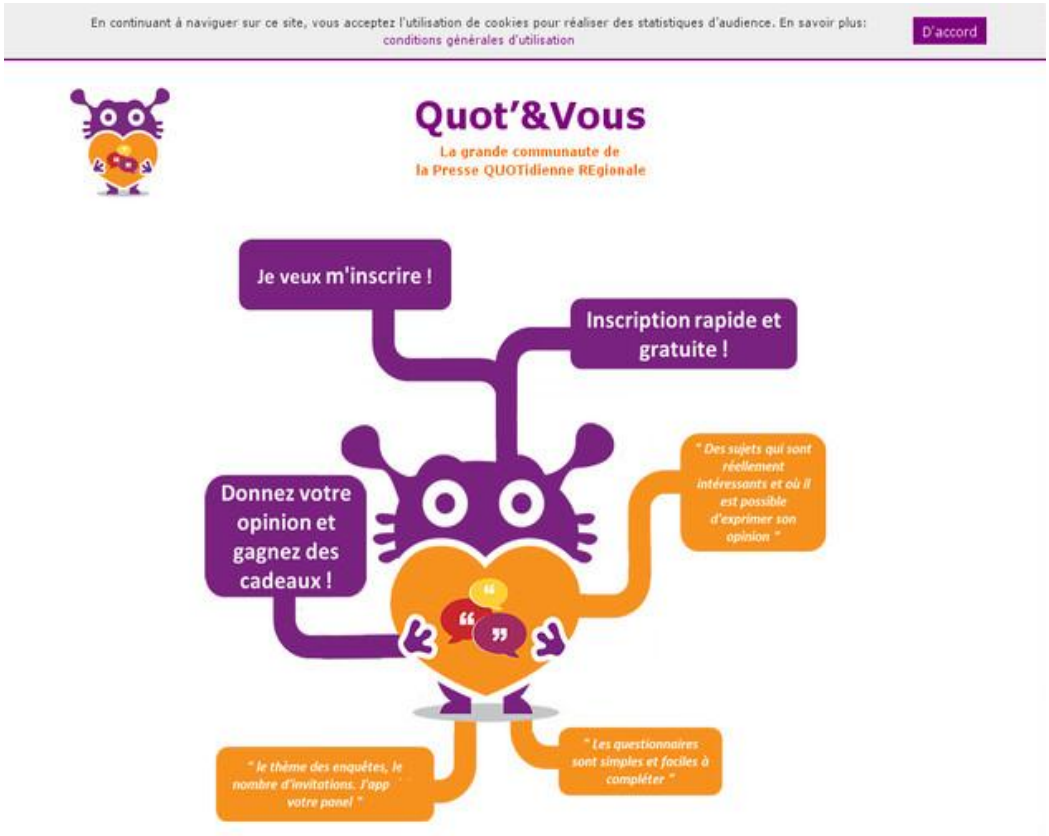


Figure 6: Page d'accueil du site Quot'&Vous

1.6. Conclusion

Dans ce chapitre, nous avons présenté les principales notions et différents concepts propres à l'analyse d'opinions, ensuite nous avons exposé ses différentes étapes, nous avons également présenté les différentes manières de classée l'analyse d'opinion selon les tâches selon la granularité et selon l'approche, et fin nous avons terminé avec les domaines d'application et les exemples d'application de l'analyse d'opinion.

Chapitre 2 : L’Apprentissage Profond pour l’Analyse d’Opinion

2.1. Introduction

Jusqu'en 2013, le domaine de l'analyse d'opinion a connu un nombre conséquent de travaux reposants principalement sur les techniques d'apprentissage automatique classiques (machine learning supervisées et non supervisées). Dans le cadre supervisé, les principaux travaux utilisaient tous types de méthodes (Machines à Vecteurs de Supports (SVM), Naïve Bayes ou encore le principe de l'entropie maximale) et ce conjointement avec des combinaisons de caractéristiques appropriées à l'analyse d'opinions. Alors que, les travaux autours des techniques non supervisées reposent essentiellement sur des considérations linguistiques, notamment avec l'utilisation des lexiques d'opinions, de l'analyse grammaticale et de divers modèles syntaxiques.

A partir de l'année 2013, des méthodes issues du domaine du Deep Learning (anglicisme signifiant l'apprentissage profond) commencent, peu à peu, à se démocratiser et l'on constate un apport significatif de ces méthodes sur l'analyse d'opinion. [9] En effet, ces architectures et algorithmes d'apprentissage profond ont déjà fait leur preuve dans des domaines tels que la vision par ordinateur et la reconnaissance de formes, mais ce n'est récemment que des travaux en TALN, recherche d'information et plus particulièrement en analyse d'opinion commencent à s'intéresser à leur utilisation.

Le Deep Learning (DL) est considéré comme un sous domaine de l'apprentissage automatique. Ainsi, comme toutes techniques d'apprentissage automatique, le DL vise à entraîner un système pour qu'il résolve des situations sans que tous les paramètres nécessaires à la résolution du problème n'aient été calculés par le développeur de la solution. L'objectif est d'entraîner un algorithme à paramètres variables (une « boîte noire ») à prendre une décision correcte concernant une tâche donnée. L'apprentissage se fait en optimisant les paramètres variables pour améliorer les décisions prises. Ceci est possible grâce aux réseaux de neurones artificiels dont la structure imite celle du cerveau humain. Chaque fois que de nouvelles données sont intégrées, les connexions existantes entre les neurones sont susceptibles d'être modifiées et étendues, ce qui a pour effet de permettre au système d'apprendre de manière autonome, tout en améliorant la qualité de ses prévisions. [24]
Dans ce chapitre, nous allons aborder les réseaux de neurones artificiels et expliquer les déférentes étapes d'apprentissage d'un réseau de neurone, présenter les modèles de deep learning les plus utilisés dans le cadre d'analyse des sentiments, et à la fin nous allons voir une technique de représentation très populaire qui est le word embedding.

2.2. Les réseaux de neurones artificiels

Les algorithmes de Deep Learning reposent essentiellement sur le mécanisme de superposition de réseaux de neurones artificiels sous la forme de couches successives. Ces derniers sont inspirés des réseaux de neurones biologiques et permettent de représenter une relation entre des données d'un espace X et un espace de sortie Y. L'unité de calcul de base dans ce type de méthode est le neurone, celui-ci prend en entrée plusieurs signaux et les interprète pour envoyer un nouveau signal vers d'autres neurones du réseau. Ainsi, les résultats d'une première couche de « neurones » servent d'entrée aux calculs d'une deuxième couche et ainsi de suite. Ce principe de superposition de plusieurs couches de neurones appelé « couches cachées » et fonctionnant à l'unisson, correspond au principe de base du Deep Learning.

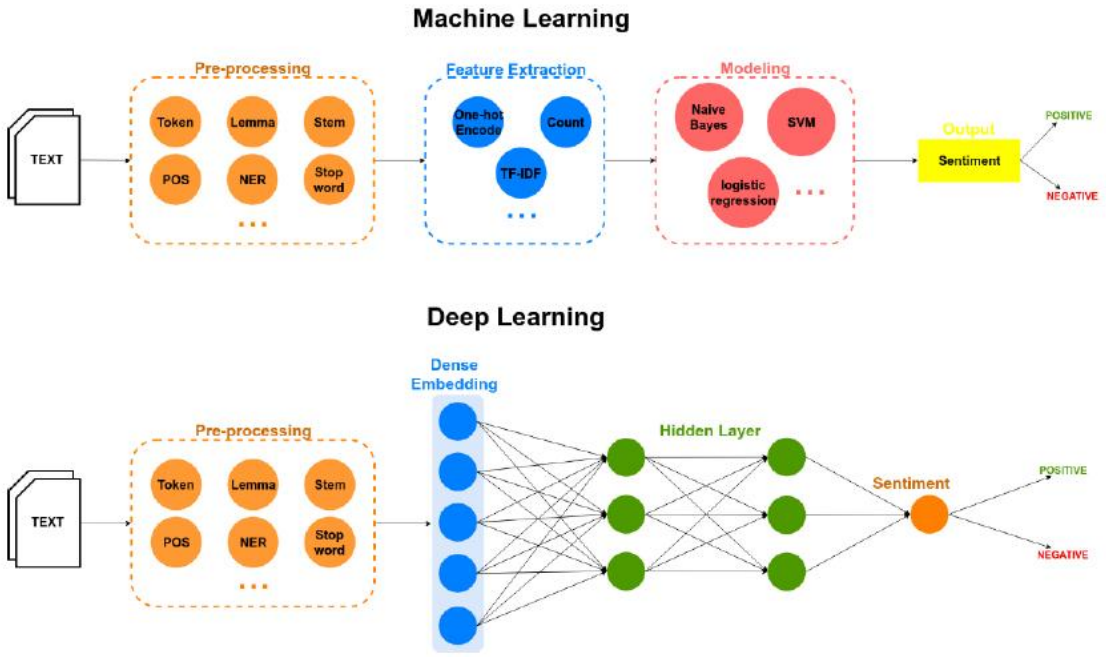


Figure 7: Processus des modèles d'apprentissage automatique classique versus apprentissage profond [65]

La figure 7 montre la différence entre le processus des approches traditionnelles d'apprentissage automatique dont les caractéristiques sont définies et extraites manuellement ou à l'aide de méthodes de sélection de caractéristiques, ainsi que le processus des modèles d'apprentissage profond (Deep Learning) où les caractéristiques sont apprises et extraites automatiquement, ce qui apporte précision et performances à l'apprentissage. Plusieurs modèles de DL seront abordés dans ce chapitre, néanmoins pour mieux comprendre le fonctionnement des réseaux de neurones, nous allons commencer par la présentation d'un

modèle composé d'un seul neurone, appelé modèle du perceptron. Celui-ci va nous permettre de mettre en évidence les mécanismes de base de tout réseau de neurones.

2.2.1. Le modèle du perceptron

Dans sa version la plus simple, le perceptron est un réseau de neurones composé de seulement un neurone, qui prend en entrée *n* données binaires. Chacune de ses entrées *i* est pondérée par un poids noté w_i. Le neurone peut prendre les états "1" ou "0" (respectivement actif ou non-actif) en fonction de ses entrées pondérées et d'un biais noté $\beta \in \mathbb{R}$. Cet état représente la sortie du modèle. Il est donc possible de représenter le perceptron comme une fonction paramétrique $f_\theta : \{0, 1\}^n \rightarrow \{0, 1\}$ avec θ l'ensemble de ses paramètres, c'est-à-dire le biais β et les poids $\mathbf{w} = (w_1, \ldots, w_n)$. La sortie d'un perceptron pour un vecteur $x \in \{0, 1\}^n$ en entrée est calculée tel que : [14]

$$f(x, \boldsymbol{w}) = H(z(x, \boldsymbol{w}) + \beta)$$

Avec $H(t)$ la fonction de Heaviside définie pour tout $t \in \mathbb{R}$ *comme* $H(t) = \mathcal{I}_{\{t>0\}}$ et $z(x, \boldsymbol{w})$ la somme pondéré des entrées :

$$z(x, \boldsymbol{w}) = \boldsymbol{w}^{\mathrm{T}} x = \sum_{j=1}^{n} w_j \, x_j$$

Intuitivement, les poids $w_1, \ldots, w_n$ représentent l'importance accordée à chaque entrée pour l'activation du perceptron. Il est à signaler que, $\mathcal{I}$ correspond à une fonction indicatrice dite d'activation qui est égale à 1 si une condition est vérifiée et 0 sinon. Le biais β peut être vu comme l'ajout d'un seuil à la difficulté d'activation du perceptron. En effet, si la somme pondérée $z(x, \boldsymbol{w})$ dépasse - β (l'opposé du biais), le perceptron s'active, sinon il reste inactif. Pour simplifier les notations, nous allons inclure le biais dans le vecteur de poids en ajoutant une constante en entrée $x_0 = 1$ (le biais devient donc w_0). Nous obtenons la fonction paramétrique [14] : $f_\theta : \{0,1\}^{n+1} \times \mathbb{R}^{n+1} \rightarrow \{0,1\}$ telle que :

$$f(x, \boldsymbol{w}) = H(z(x, \boldsymbol{w}) = H\left(\sum_{j=0}^{n} w_j \, x_j\right)$$

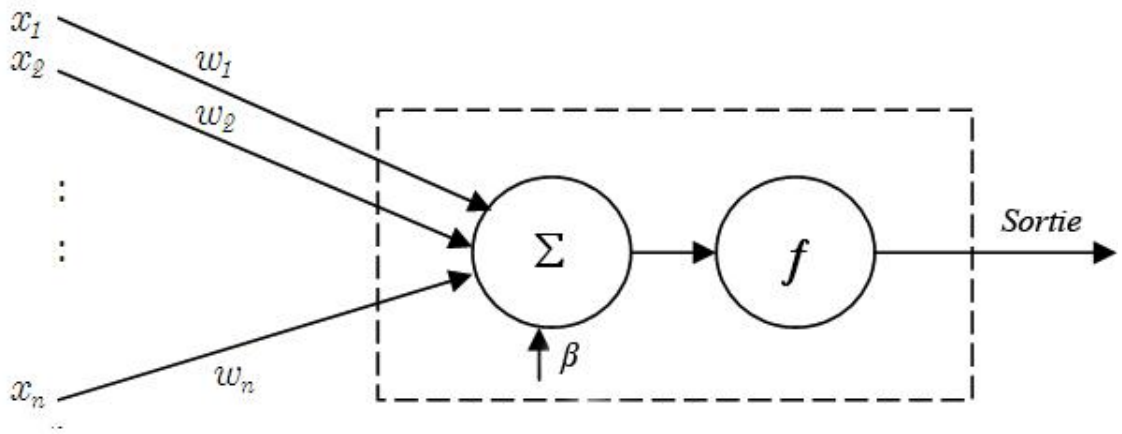


Figure 8: Illustration du perceptron

La figure ci-dessus, montre les trois éléments de base du perceptron :

- Un ensemble de connexions avec les différentes entrées x_i, pondérée chacune par un poids w_i.
- L'additionneur permettant de calculer une combinaison linéaire des entrées x_i pondérées par les coefficients w_i.
- Le biais β qui permet de contrôler l'entrée de la fonction d'activation.
- La fonction d'activation f permettant de délimiter la sortie du neurone.

Cependant, le perceptron de par sa définition est un classifieur dit linéaire, c'est-à-dire qu'il classifie des données à partir d'une combinaison linéaire de ses entrées. Il est donc incapable de classifier des données dans des classes non linéairement séparables. Ainsi, d'autres variantes de réseaux de neurones plus généraux que le perceptron ont vu le jour, tel que les perceptrons multicouches. Ils sont composés d'une multitude de neurones interconnectés et organisés en couches successives. Ces perceptrons multicouches peuvent être représentés par un graphe acyclique dans lequel chaque nœud représente un neurone et les arcs orientés représentent les relations entre chaque neurone.

La Figure ci-dessous montre une représentation graphique d'un perceptron multicouche avec 3 couches ayant respectivement 3, 4 et 3 neurones.

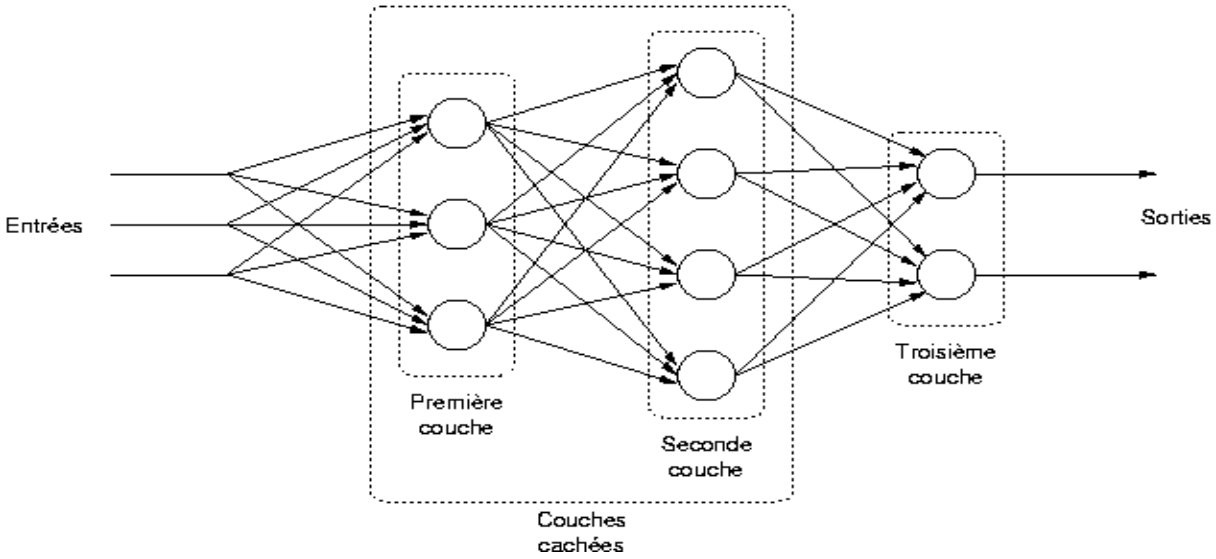


Figure 9: Illustration du perceptron multicouche

Ainsi, sur la base de ces deux modèles, nous pouvons déduire qu'un réseau de neurones est caractérisé par son architecture (l'organisation des neurones), son mode d'apprentissage (méthode de détermination des poids de connexions) et de par sa fonction d'activation [25]. Dans ce suit, nous présentons de manière succincte ce qu'est le Deep Learning, au travers du fonctionnement d'un réseau de neurones. Toutefois, étant donné que le Deep Learning est un sous domaine de l'apprentissage automatique, par conséquent, on distingue deux types d'apprentissage profond :

- L'apprentissage profond supervisé
- L'apprentissage profond non supervisé

Dans le cadre de ce mémoire, nous faisons exclusivement de l'apprentissage profond supervisé, ce qui nous a amené, à ignorer les notions liées au DL non supervisé dans la suite de ce mémoire.

2.2.2. Étapes d'apprentissage d'un réseau de neurones

2.2.2.1. Phase d'initialisation des poids du réseau

Quand on parle des poids d'un réseau de neurones, on entend par-là l'ensemble des poids des connexions synaptiques existantes entre toutes couches confondues. Une connexion synaptique correspond à la connexion entre deux neurones artificiels.

Afin de distinguer entre les notations de la section 2.2.1, on définit les annotations suivantes. Si on considère le troisième neurone d'une couche (numérotée en IV), le poids *w* relatif à sa connexion avec le premier neurone d'une couche précédente (numéroté en III) est annoté $w_{3,1}^{IV}$. Si on généralise : w_{ij}^{l} est le poids qui relie le $i^{éme}$ neurone de la $l^{éme}$ couche au $j^{éme}$ neurone de la $(l - 1)^{éme}$ couche.

Lorsqu'on évoque la somme P_i^l de tous les poids du $i^{éme}$ neurone de la $l^{éme}$ couche, on signifie que c'est la somme de tous les poids des connexions synaptiques existantes entre les neurones de la couche $(l - 1)$ et cet $i^{éme}$ neurone de la $l^{éme}$ couche. Cela peut s'écrire comme : $P_i^l = \sum_j w_{ij}$

Ces connexions sont illustrées dans la Figure numéro 10.

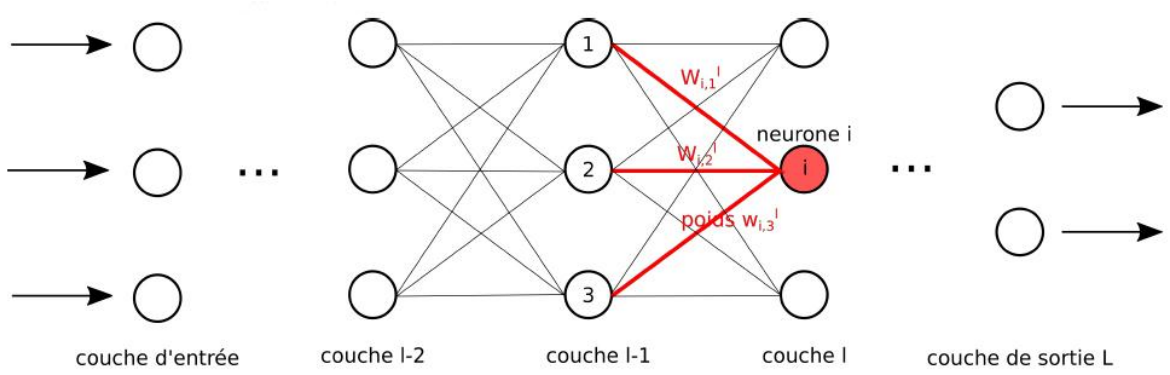


Figure 10: Poids des connexions synaptique du iéme neurone de la léme couche issues du jéme neurone de la (l − 1) iéme couche précédente

Le choix des poids initiaux des couches cachées du réseau est un point fondamental de la phase d'apprentissage. Car, une initialisation avec des poids appropriés peut faire la différence lors de la phase d'apprentissage.

2.2.2.2 Phase de propagation en avant des poids

Afin de mieux comprendre le principe d'un réseau de neurones profond, prenons l'exemple d'un perceptron multicouche qui est un réseau à propagation avant (feedforward neural network). Dans ce réseau, l'information ne se déplace que dans une seule direction, vers l'avant, à partir des nœuds d'entrée, en passant par les couches cachées et vers les nœuds de sortie. Quelques notations pour bien comprendre la suite [14] :

- *L* est le nombre de couches du réseau de neurones.
- ϭ est une fonction d'activation.
- w_{ij}^{l} est le poids qui relie le $i^{éme}$ neurone de la $l^{éme}$ couche au $j^{éme}$ neurone de la $(l - 1)$ $i^{éme}$ couche.

- w^l est une matrice de taille *i x j* (*i* lignes et *j* colonnes) qui contient tous les poids de la $l^{éme}$ couche.
- b_i^l est le biais associé au $i^{éme}$ neurone de la $l^{éme}$ couche.
- b^l est le vecteur qui contient tous les biais de la $l^{éme}$ couche.

L'information de sortie du $i^{éme}$ neurone de la $l^{éme}$ couche se calcule à l'aide des informations d'entrée de ce neurone qui correspondent aux informations de sortie des neurones de la $(l-1)$ $i^{éme}$ couche auxquels il est connectés, ainsi qu'à l'aide des poids des connexions synaptiques mises en jeu. La propagation vers l'avant se calcule à l'aide d'une fonction d'agrégation et d'une fonction d'activation.

$-\ z_i^l$ est la valeur d'agrégation du $i^{éme}$ neurone de la $l^{éme}$ couche, c'est-à-dire la valeur qu'un neurone calcule avant de la passer à la fonction d'activation : $z_i^l = \sum_{j=1}^{n} w_{ij}^l\, a_i^{l-1} + b_i^l$

a_i^l est la valeur d'activation du $i^{éme}$ neurone de la $l^{éme}$ couche, c'est à dire la valeur définitive de sortie du neurone : $a_i^l = 6(z_i^l)$

Avec $z^l = (z_1^l,\ z_2^l, \dots,\ z_n^l)$ et $a^l = (a_1^l,\ a_2^l, \dots,\ a_n^l)$ on peut écrire :

- $z^l = w^l \times a^{l-1} + b^l$
- $a^l = 6(z^l)$

2.2.2.3. Phase de Rétro-propagation du gradient

L'objectif au cours de l'apprentissage est de trouver une configuration de poids qui minimise l'erreur de prédiction. L'algorithme de rétro propagation du gradient (back propagation) permet de calculer l'erreur pour chaque neurone, de la dernière couche vers la première, et ce pour tous les paramètres du réseau, grâce à une méthode développée par Rumelhart, Hinton et Williams. [5] L'algorithme de rétro-propagation du gradient se déroule en deux temps, d'abord (*i*) en calculant l'erreur (sortie) en comparant la sortie avec le résultat attendu puis en (*ii*)propageant cette l'erreur de couche en couche vers l'arrière.

i. Calcul de l'erreur de sortie : Ce calcul obéit au théorème de la dérivation des fonctions composées ou règle de dérivation en chaîne [14], définit comme suit :

Soient $f : \mathbb{R}^n \to \mathbb{R}$ et g : $\mathbb{R}^p \to \mathbb{R}^n$ deux fonctions différentiables.

Écrivons $h = f \circ g$.

D'après la règle de dérivation des fonctions composées nous avons :

$h'(x) = (f°g)'(x) = f'\big(g(xx)\big).g'(x)$

Ou encore $\frac{\partial h}{\partial x_i} = \sum_{i=1}^{n} \frac{\partial f}{\partial g_k}\frac{\partial g_k}{\partial x_i}$

Pour mesurer l'erreur de prédiction, on utilise une fonction appelée fonction de coût *C*, ou encore fonction de perte ou fonction d'erreur. Il existe plusieurs types de fonction de coût selon le problème qu'on cherche à résoudre. Grâce à l'algorithme de rétro-propagation du gradient, nous pouvons savoir chaque erreur individuelle $\frac{\partial C}{\partial w_{ij}^l}$ et $\frac{\partial C}{\partial w_i^l}$ de tous les poids et biais du réseau et leur degré de contribution à l'erreur finale $\sigma^{(sortie)}$. Concernant les poids, selon le théorème suscité, on peut définir pour la couche de sortie *L* du réseau : $\frac{\partial C}{\partial w_{ij}^l} = \frac{\partial z_i^L}{\partial w_{ij}^L}\frac{\partial a_i^L}{\partial z_i^L}\frac{\partial C}{\partial a_i^L}$

Où chaque terme de l'équation correspond à :

- $\frac{\partial C}{\partial a_i^L}$ la variation de la fonction de coût en fonction de la sortie du réseau.
- $\frac{\partial a_i^L}{\partial z_i^L}$ la variation de la fonction d'activation en fonction de l'agrégation qui peut être également obtenue en calculant, la dérivée de la fonction d'activation : $\sigma'(z_i^L) = \frac{\partial a_i^L}{\partial z_i^L}$
- $\frac{\partial z_i^L}{\partial w_{ij}^L}$ la variation de la fonction d'agrégation en fonction d'un seul poids w_{ij}^L. C'est aussi égal à : $a_j^{L-1} = \frac{\partial z_i^L}{\partial w_{ij}^L}$

Par conséquent, on obtient : $\frac{\partial C}{\partial w_{ij}^l} = a_j^{L-1} \times \sigma_i^L$ qui est l'erreur calculée pour la couche de sortie.

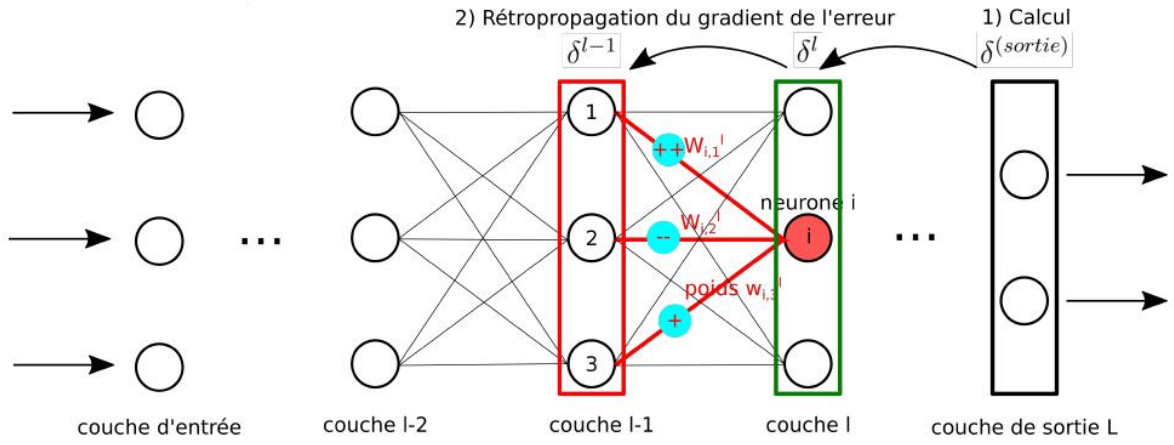


Figure 11: Schéma de la rétro propagation du gradient de l'erreur depuis la couche de sortie jusqu'aux poids des connexions synaptique du iéme neurone de la léme couche issues du jéme neurone de la (l − 1) iéme couche précédent

ii. Phase de rétro-propagation :

La variation de la fonction de coût de la sortie du $i^{éme}$ neurone de la $l^{éme}$ couche peut s'écrire comme : $\frac{\partial C}{\partial a_i^l} = \sum_k \frac{\partial z_k^L}{\partial a_i^l}\frac{\partial C}{\partial z_k^L}$ avec $\frac{\partial C}{\partial z_k^L} = \epsilon_k^L$ et le terme $\frac{\partial z_k^L}{\partial a_i^l}$ peut s'écrire aussi $\frac{\partial z_k^L}{\partial a_i^l} = w_{ki}^L$

On peut résumer les équations nécessaires à l'implémentation de l'algorithme de rétro-propagation du gradient par : [14]

$\epsilon_i^L = \epsilon'(z_i^L) \times C'(a_i^L)$ pour la couche de sortie *L*.

$\epsilon_i^L = \epsilon'(z_i^L) \times \sum_j w_{ji}^{l+1} . \epsilon_i^{l+1}$ pour les couche cachés *l* du réseau.

$\frac{\partial C}{\partial b_i^l} = \epsilon_i^L$ pour les biais

2.2.2.4. Phase d'actualisation des poids

Entraîner un réseau de neurones consiste à répéter l'actualisation et la mise à jour des poids w_{ij}^let biais b_i^l du réseau jusqu'à ce que C (*w*, *b*) converge. Pour se faire, le réseau s'aide de l'algorithme de descente de gradient [14]. Cet algorithme utilise le résultat de la rétro-propagation pour mettre à jour les paramètres. Il cherche à minimiser la fonction de coût *C*(*x*) en modifiant *x*. Il est itératif et procède par améliorations successives.

Son but est de trouver ou de s'approcher d'un point stationnaire ou minimum global, c'est à dire d'un point où le gradient de la fonction de coût est nul et ainsi aboutir à une configuration stable du réseau neuronal. La figure 11 illustre le déplacement le long d'une pente d'erreur vers une valeur d'erreur minimale. A chaque itération d'optimisation du gradient, les erreurs sont corrigées selon l'importance des neurones à participer à l'erreur finale. Les poids synaptiques qui contribuent à engendrer une erreur importante sont modifiés de manière plus significative que les poids qui engendrent une erreur marginale. Les poids sont ainsi soit diminués, soit augmentés, et le modèle est mis à jour. Le biais est une valeur scalaire ajouté en entrée. Cela permet d'avoir toujours des neurones actifs quel que soit la force du signal d'entrée. Ces biais sont modifiés comme les poids au cours de l'apprentissage.

La descente de gradient peut être réalisée de plusieurs manières : [14]

- De manière globale (batch gradient) : on envoie au réseau toutes les données en une seule fois, puis le gradient est calculé et les poids ajustés.

- Par des lots (mini-batch gradient) : on envoie au réseau les données par petits groupes de taille définie par l'utilisateur. Puis l'erreur moyenne est calculée et enfin les poids sont mis à jour.
- De manière unitaire ou stochastique (stochastic gradient) : cette méthode envoie une seule donnée à la fois dans le réseau et donc les poids sont mis à jour à chaque itération.

2.2.2.5. Calcul de la fonction d'activation

Lorsque le potentiel électrique d'un neurone biologique augmente et atteint son seuil d'excitation, cela déclenche un potentiel d'action qui se propage le long neurone : c'est l'influx nerveux. S'il y a déclenchement d'un potentiel d'action, il y a alors propagation du message nerveux à d'autres neurones. La notion de fonction d'activation utilisée en apprentissage profond est justement inspirée du potentiel d'action. La fonction d'activation, grâce à une transformation linéaire ou non-linéaire des entrées, prend la décision de transmettre ou non le signal selon la valeur de sortie obtenue.

Dans les réseaux de neurones profonds, les fonctions d'activations sont des fonctions non-linéaires. [3] Ceci permet de séparer les données non-linéairement séparables et donc d'effectuer des classifications plus poussées qu'en apprentissage automatique classique. Autre point très important, les données traitées par les neurones peuvent atteindre des valeurs étonnamment grandes. L'utilisation d'une fonction linéaire, ne modifiant pas la sortie, les valeurs des données transmises de neurones en neurones peuvent devenir de plus en plus grandes et rendant les calculs beaucoup plus complexes. Afin d'y remédier, les fonctions d'activation non linéaires réduisent la valeur de sortie d'un neurone le plus souvent sous forme d'une simple probabilité. [14] Parmi les principales fonctions d'activation étudiées, on retrouve :

2.2.2.5.1. La fonction sigmoïde

Cette fonction se définie comme suit :

$$f(x) = \frac{1}{1+exp^{-x}} \text{ Pour } x \in \mathbb{R}$$

Si x est un grand nombre positif, exp^{-x} tend vers 0 et donc $f(x) = 1$. À l'inverse, si x est grand nombre négatif, exp^{-x} tend vers l'infini et donc $f(x) = 0$. La fonction sigmoïde convertit ainsi en une probabilité comprise entre 0 et 1 toute valeur d'entrée x. Il faut aussi

noter que seules les valeurs faibles de x influencent réellement la variation des valeurs en sortie.

La fonction sigmoïde est utilisée en apprentissage automatique pour la régression logistique mais peu utilisée en apprentissage profond car sa propriété de tendre rapidement vers 0 ou 1 induit la saturation de certains neurones du réseau entraînant l'arrêt de l'apprentissage. Cependant, nous pouvons la retrouver au niveau des trois portes d'entrée, de sortie et d'oubli d'un neurone LSTM car comme elle génère une valeur comprise entre 0 et 1, elle peut soit ne laisser passer aucun signal, soit compléter le flux d'information via les portes (on ré-abordera ces notions ultérieurement).

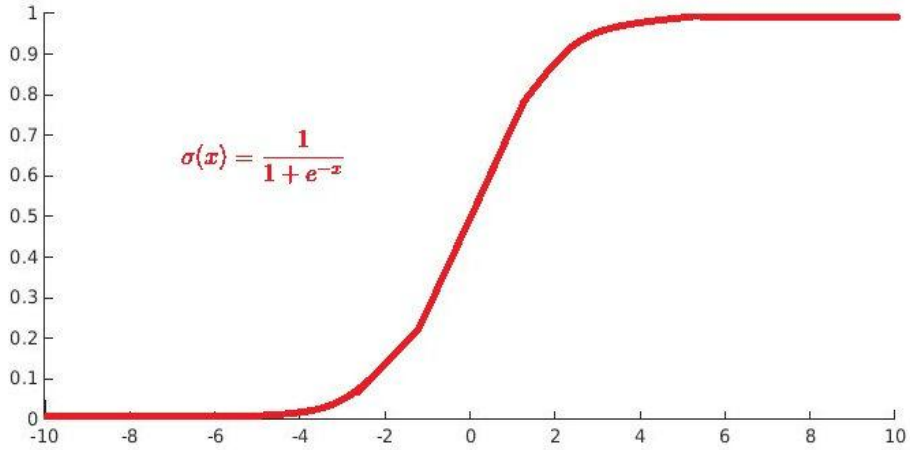


Figure 12: Graphique de la fonction sigmoïde.

2.2.2.5.2. La fonction tangente hyperbolique

La fonction tangente hyperbolique ou *tanh* ressemble quelque peu à la fonction sigmoïde à la différence que son résultat peut être compris entre −1 et 1. Si *x* est un grand nombre positif, $tanh(x) = 1$ et si *x* est un grand nombre négatif $tanh(x) = -1$.

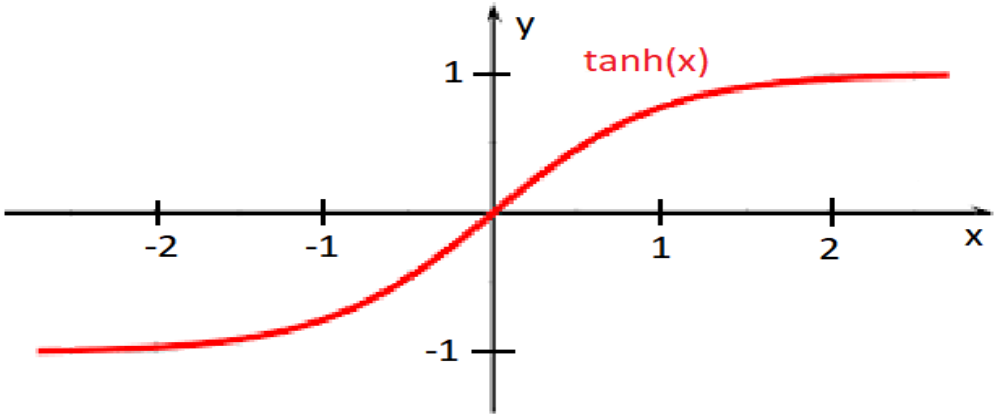


Figure 13: Graphique de la fonction tangente hyperbolique

2.2.2.5.3. La fonction ReLU

La fonction unité de rectification linéaire (Rectified Linear Unit, ReLU) est définie par : $f(x) = \max(0, x)$. Si $x < 0$ alors $f(x) = 0$. Si $x \geq 0$, alors $f(x) = x$.
Cette fonction a la caractéristique d'augmenter les chances de converger du réseau et ne provoque pas de saturation des neurones contrairement aux deux fonctions précédentes *tanh* et *sigmoïde*. Cependant lorsque $x < 0$, cela rend les neurones concernés inactifs donc les poids relatifs ne sont pas mis à jour et le réseau cesse d'apprendre.

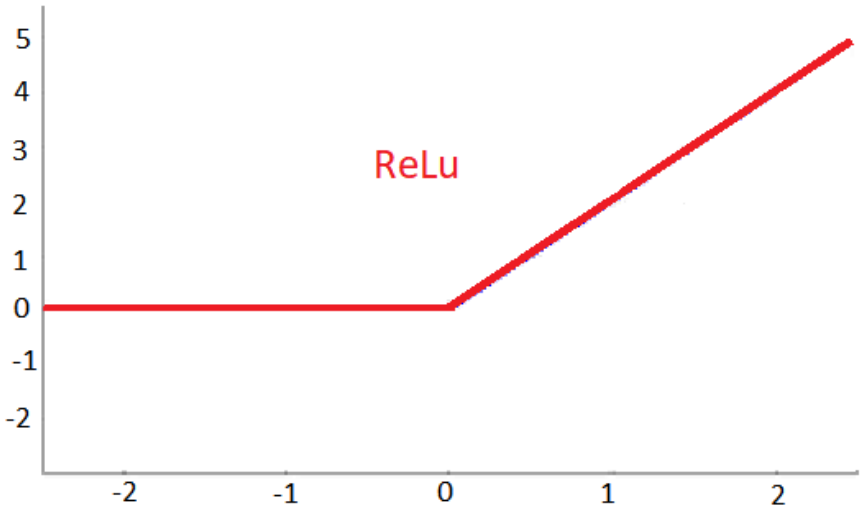


Figure 14: Graphique de la fonction Relu [3]

2.2.2.5.4. La fonction softmax

La fonction softmax ou fonction exponentielle normalisée est définie par :

Fonction softmax $\mathfrak{o}_k\ (u_k) = \frac{\exp(u_k)}{\sum_{i=1}^{k} \exp(u_i)}$ où k désigne la classe considérée.

Le principal intérêt de cette fonction est de prendre en entrée un vecteur u composé de k nombres réels et de renvoyer un vecteur $\mathfrak{o}$ de même longueur :

- Composé de k nombres réels strictement positifs.
- La somme de ces éléments est égale à 1.

La fonction softmax est une non-linéarité utilisée pour la dernière couche d'un réseau neuronal profond créé pour une tâche de classification à k classes car elle permet de sortir k probabilités de prédiction dont la somme totale est égale à 1.

2.3. Modèles du Deep Learning

Il existe plusieurs types de réseaux de neurones artificiels. Ces types de réseaux sont mis en œuvre sur la base d'opérations mathématiques et d'un ensemble de paramètres requis pour déterminer la sortie. Parmi les plus utilisés en analyse d'opinion et de manière générale en TALN, on retrouve les réseaux de neurones à convolution ainsi que les réseaux de neurones récurrents de types LSTM (Long Short-Term Memory).

Dans cette section, nous présentons un aperçu des deux principales catégories de modèles de DL à savoir, les réseaux de neurones à convolution et les réseaux de neurones récurrents. Nous aborderons également les couches communément utilisés dans l'ensemble de ces modèles, on verra leur utilité, leur finalité et fonctionnement respectifs.

2.3.1. Couches usuelles des réseaux de neurones profonds.

2.3.1.1. Couche d'incorporation (Embedding Layers)

En TALN, la couche d'incorporation vise à convertir les index de mots en des représentations denses. Cela se fait généralement, en trois manières différentes. La première consiste à utiliser l'apprentissage pour la représentation de mots, afin de symboliser chaque terme d'un dictionnaire par un vecteur de nombres réels. C'est-à-dire que les word embedding (littéralement traduit par plongement de mots) sont initialisés avec des valeurs aléatoires et seront améliorés au cours du processus d'apprentissage. La seconde consiste à utiliser des

vecteurs de mots pré-entraînés, en gardant les poids d'incorporation statiques. La troisième et dernière manière consiste à utiliser des vecteurs de mots pré-entraînés, mais au lieu de garder leurs poids statiques, ces derniers seront optimisés pendant le processus d'apprentissage.

2.3.1.2. Couche d'abandon (Dropout Layers)

La couche Dropout a pour rôle d'aider le modèle à éviter les problèmes liés au sur-apprentissage. En effet, les réseaux de neurones profonds sont composés de plusieurs couches cachées non linéaires. L'existence de ces types de couches permet aux modèles d'apprendre des relations très complexes entre les entrées et les sorties sur les données d'apprentissage. Cependant, de nombreuses relations seront le résultat de l'échantillonnage du bruit, c'est-à-dire que certaines relations existeront dans l'ensemble d'apprentissage mais pas dans l'ensemble de test. Lorsque cela se produit, le modèle ne se généralise pas bien, ce qui entraîne de mauvais résultats en termes de performances de classification ou prédiction.

Afin d'éviter de problème de sur-apprentissage, de nombreuses méthodes ont été développées, l'une d'entre elles étant la technique d'abandon (dropout). Celle-ci consiste à forcer le réseau de neurones à apprendre plusieurs représentations indépendantes des mêmes données (abandon des neurones pendant la phase d'apprentissage), dans le but de réduire la dépendance à l'ensemble d'entraînement et éviter des co-adaptations complexes sur les données de l'échantillon d'entraînement. Le terme « dropout » se réfère à une suppression temporaire de neurones (à la fois les neurones cachés et les neurones visibles) dans le réseau. Ce dernier se voit ainsi privé d'une partie de ses neurones pendant la phase d'entrainement (leur valeur est considérée à 0) puis ils seront réactivés une nouvelle fois pour tester le modèle.

2.3.1.3. Couches LSTM

Une couche LSTM implémente essentiellement un modèle récurrent de type LSTM, qui est une des variantes des réseaux de neurones récurrents. Ce type de modèle prend en compte trois aspects important qui permettent une amélioration considérable des performances de prédiction. Le premier aspect est que cette couche a une maitrise particulière pour décider quand autoriser l'accès au neurone suivant. Le second est la capacité de contrôler le mécanisme de mémoire et savoir quand se souvenir de ce qui a été calculé dans les couches précédentes. Enfin, ce type de modèle a également la capacité de contrôler quelles parties de

l'information le système il se doit de constituer. Ces aspects seront discutés dans les sections ci-dessous.

2.3.1.4. Couches denses

Une couche dense correspond à une couche qui est entièrement connectée, ou toutes les entrées et sorties sont connectées à tous les neurones de chaque couche. Plus précisément, l'idée de cette couche est de connecter chaque neurone du réseau aux neurones des couches adjacentes.

2.3.1.5. Couche de convolution

Une couche de convolution est l'une des couches les plus pertinentes dans le cas des modèles de réseaux de neurones à convolution. Son objectif est de détecter la présence de caractéristiques dans les données d'entrée. Cela est réalisé grâce à un filtrage par convolution qui consiste à faire glisser une fenêtre représentative de la caractéristique sur les données d'entrée et à calculer le produit de convolution entre la caractéristique et chaque portion des données balayées. Dans ce contexte, le concept de caractéristique est assimilé au filtre. Dans chaque couche convolutive, chaque filtre est répliqué sur tout l'espace de donnée. Ces unités répliquées partagent le même paramétrage (vecteur de poids et biais) et cela forme une carte de caractéristiques où feature map (figure 15). Cela signifie que tous les neurones d'une même couche convolutive répondent aux mêmes caractéristiques. Cette réplication permet ainsi de détecter les caractéristiques quelle que soit leur position dans l'espace de donnée.

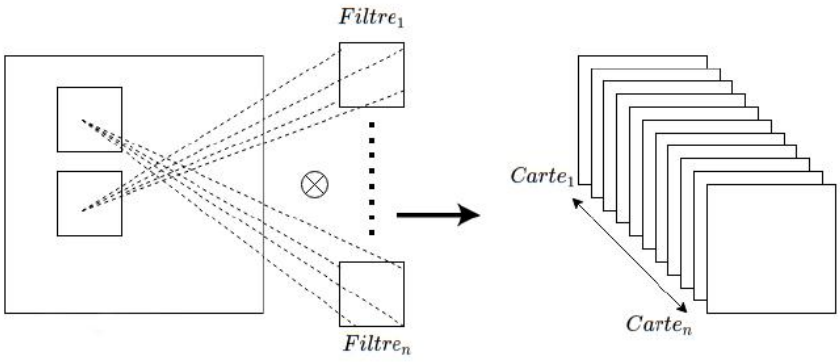


Figure 15: Représentation générale des cartes de caractéristiques

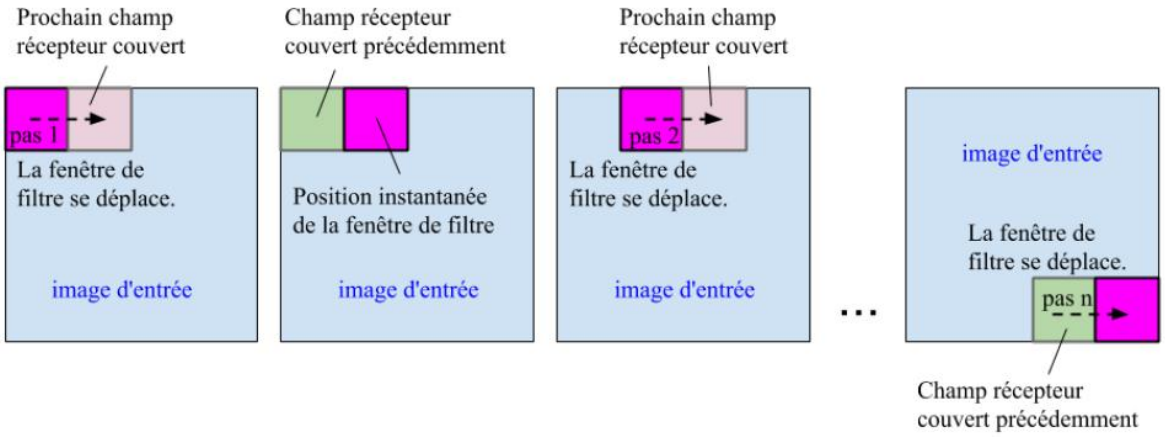


Figure 16: Schéma de glissement d'une fenêtre de filtre sur un exemple d'image d'entrée [65]

2.3.1.6. Couches de regroupement (Pooling layer) et d'aplatissement (Flatten layer)

La couche de regroupement (pooling) se place entre les couches convolutives. Elle permet d'appliquer à chacune des cartes de caractéristiques une réduction de leur taille tout en préservant les caractéristiques les plus importantes (en ne gardant que les valeurs maximales par exemple). Elle permet ainsi de réduire le nombre de paramètres du réseau et donc les calculs nécessaires. Elle permet aussi de rendre le réseau moins sensible à la position des caractéristiques. La fonction la plus simple pour exécuter cette opération est le pooling max. L'idée est de diviser la représentation d'entrée en un ensemble de régions sans chevauchement et pour chacune de ces régions, on sélectionne la valeur maximale. Au final, chaque élément de la nouvelle représentation correspondra à la valeur maximale d'une région dans la représentation initiale. Quant au calque d'aplatissement (flatten), comme son nom l'indique, elle permet d'atténuer l'entrée des neurones. Plus précisément, ces couches aplatissent toutes les dimensions de leurs entrées en une seule dimension, afin d'améliorer l'apprentissage.

2.3.1.7. Couche d'activation

Une couche d'activation à pour rôle d'appliquer une fonction d'activation à une sortie. Les fonctions d'activation (section 2.2.2.5) dans les réseaux de neurones, permettent de restreindre les sorties à une certaine plage de valeurs, mais également rompt le réseau de manière linéaire, lui permettant ainsi d'apprendre des fonctions plus complexes que la régression linéaire. Car, un neurone sans fonction d'activation équivaut à un neurone avec une fonction d'activation linéaire, comme f (x) = x. Lorsque ces fonctions n'ajoutent aucune non-linéarité, l'ensemble du réseau équivaut à un seul neurone linéaire. Par conséquent, cela n'a aucun sens de construire un réseau à plusieurs couche avec des fonctions d'activation linéaires.

2.3.2. Les réseaux de neurones à convolution (CNN)

Un réseau de neurones à convolution, (CNN pour Convolutional Neural Network) un type de réseau de neurones artificiels conçus pour traiter des données se présentant sous la forme de tableaux de valeurs en N-dimensions avec$N \in \mathbb{R}^{+}$. Ils ont initialement été utilisés pour traiter des données sous la forme de tableaux de pixels à multiples dimensions pour la reconnaissance et traitement d'images. Mais de nombreux autres types de données ont été utilisés tel que, des images vidéo, des spectrogrammes, des signaux audio et bien entendu du texte.

Le principe de base d'un CNN pour le traitement d'images est illustré sur la figure numéro 17. Il se compose de différentes couches à savoir la couche d'entrée, la couche de convolution, la couche de regroupement (pooling) et la couche entièrement connectée. En couche d'entrée, on retrouve initialement un calque qui prend les valeurs des pixels de l'image comme entrée. Ensuite, la couche de convolution (CONV) a pour but de filtrer les entrées pour en extraire des caractéristiques. Ainsi, on obtient plusieurs sorties qui peuvent être combinées à l'aide de la couche de regroupement (POOL). Ces opérations ont pour principale rôle de réduire la taille de la représentation (dimensionnalité) de l'image mais également accélérer le calcul. Au final, les couches entièrement connectées s'occuperont des tâches de classification grâces aux données issues des couches précédentes.

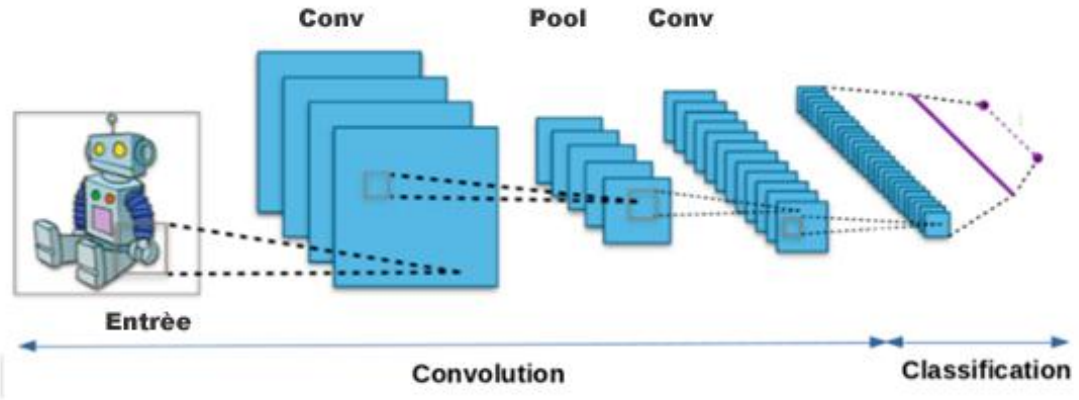


Figure 17: Les parties du CNN.

Comme on peut le voir sur la figure18. un CNN en reconnaissance d'images traite l'image en une succession de couches permettant un redimensionnement de l'image initial.

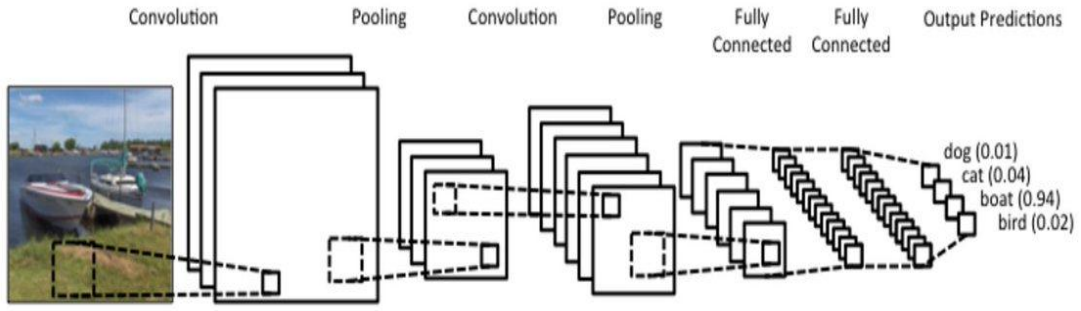


Figure 18 : Modèle CNN de base avec 4 couches.

- **Les réseaux de neurones convolutifs et leur efficacité pour le TALN**

Les réseaux de neurones convolutifs (CNN) ont été initialement conçus pour réaliser un apprentissage en classification d'images et vision par ordinateur et se sont révélés très efficaces pour ce type de tâches. Il s'avère que cette approche fonctionne également de manière efficace en traitement du langage naturel. En 2014, Yoon Kim [8] a publié à propos de l'utilisation des CNN pour la classification des textes. Quatre variations de CNN ont été testées, et les résultats ont montrés que les modèles CNN pouvaient surpasser les approches classiques pour plusieurs tâches de classification.

Au lieu des pixels utilisés en classification d'images, l'entrée de la plupart des tâches TALN sont des phrases ou des documents représentés sous forme vectorielle (matrices). Chaque ligne de la matrice correspond à un jeton, généralement un mot, mais il peut également s'agir d'un caractère. Autrement dit, chaque ligne est un vecteur qui représente un mot. En règle générale, ces vecteurs sont des *incorporations de mots* (word embedding) qui forment des représentations de faible dimension comme word2vec ou GloVe. Ils peuvent également représenter de simples vecteurs indexant le mot dans un vocabulaire.

Puis on utilise généralement des filtres qui glissent sur des lignes complètes de la matrice. Chaque filtre effectue une convolution sur la matrice de phrases et génère des cartes d'entités (de longueur variable). Puis, un regroupement des cartes est effectué, c'est-à-dire que le plus grand nombre de chaque carte d'entités est enregistré. Au final, un vecteur d'entité univarié est généré à partir des cartes, et ces entités sont concaténées pour former un vecteur d'entité pour l'avant-dernière couche. La couche finale (généralement : Softmax) reçoit ce vecteur d'entité en entrée et l'utilise pour classer la phrase. [28]

2.3.3. Les réseaux de neurones récurrents

Un réseau de neurones récurrent (RNN, recurrent neural network) est un type de réseau de neurones artificiels principalement utilisé dans le traitement automatique du langage naturel et de la reconnaissance de la voix. Ce sont des modèles capables de prendre en compte un contexte dans leur fonction de décision. Ils sont particulièrement adaptés à travailler sur des informations séquentielle. L'idée consiste à garder des informations au cours du temps à l'intérieur des couches de neurones afin de donner un contexte aux données analysées. La sortie du RNN, à un instant *t*, va dépendre non seulement de l'entrée à l'instant *t* mais également de l'état du RNN calculé à l'instant $t - 1$.

Dans sa version la plus simple, une couche d'un RNN peut être décrite comme une couche toute connectée *l* qui prend en entrée la couche précédente $l - 1$ à l'instant *t* concaténée à la sortie d'elle même (c'est-à-dire, couche *l*) à l'instant $t - 1$. La Figure 19 (a) un modèle de RNN et la figure 19 (b) représente une couche RNN. [35]

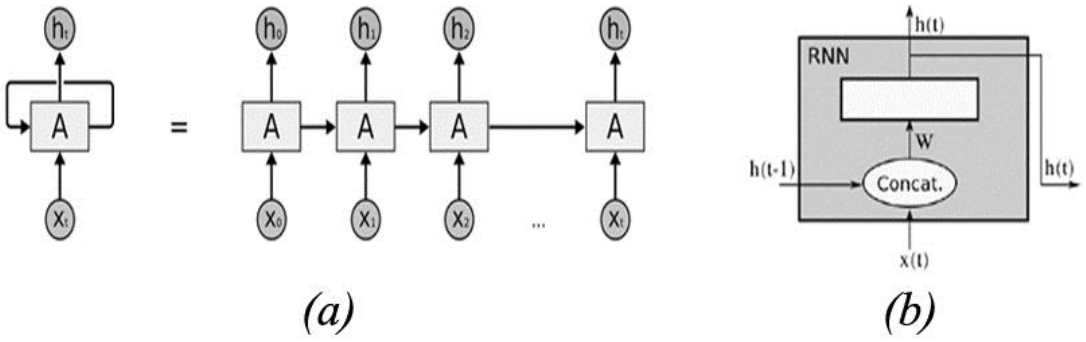


Figure 19: modèle de base du réseau RNN [35]

2.3.3.1. Réseaux de neurones récurrents simples (RNN)

Tout d'abord, on considère A(0) comme étant la séquence d'entrée, puis on obtient en sortie un état h_0 du neurone. Puis A(1) correspondra à l'entrée de l'étape suivante. De même, la sortie h_1 correspondra à l'entrée de A(2) pour l'étape suivante et ainsi de suite. De cette façon, il garde en mémoire le contexte pendant l'entraînement. En termes simples, il y a une couche d'entrée, qui correspond éventuellement à une couche cachée avec certaines activations celle-ci renvoi en sortie une l'état obtenu.

La formule de l'état actuelle est donnée par :

$$h_t = f(h_{t-1}, x_t)$$[36]

Avec :
h_t : État actuel
h_{t-1} : État précédent
x_t : État d'entrée

Puis, la formule généralement utilisée afin d'appliquer une fonction d'activation est donnée par :

$$h_t = \tanh(w_{hh}\ h_{t-1} + w_x\ h_{xt})$$[36]

Avec :
w_{hh} : Poids au neurone récurrent
w_{xh} : Poids au neurone d'entrée

La formule pour calculer la sortie est donnée par :

$$y_t = w_{hy}\ h_t$$ [36]

Avec :

y_t : Sortie du neurone
w_{hy} : Poids de la couche de sortie

Les réseaux RNN posent cependant des problèmes d'apprentissage lorsque les séquences à traiter deviennent trop longues. Le gradient (utile pour tout processus d'apprentissage) peut devenir trop faible lors des dernières itérations du processus d'apprentissage. Par conséquent, le RNN peut *facilement oublier* des données plus anciennes. C'est ce qu'on appelle le problème de la disparition de gradient (*Vanishing Gradient Problem*). Les réseaux LSTM essaye justement de répondre à cette problématique.

2.3.3.2. Réseaux Long Short-Term Memory (LSTM)

Les réseaux de mémoire courte à long terme (LSTM) sont une extension pour les réseaux neurones récurrents, Ils ont été introduits par Hochreiter Schmidhuber en 1997. Le but étant de faire face aux problèmes de disparition du gradient lorsque l'élément courant et son contexte sont trop éloignés dans le temps. L'idée principale des cellules LSTM est de garder un état de mémoire composée de trois « portes », qui correspondent à des zones de calculs régulant le flot d'informations en réalisant des actions spécifiques (*Forget gate* ou porte d'oubli, *Input gate* ou porte d'entrée et enfin *Output gate* pour porte de sortie).

Comme on peut le constater sur la figure 20, la cellule mémoire est presque entièrement pilotée par les trois portes de contrôle ou chacune peut être vue comme une vanne qui a pour rôle :

Input gate : décide si l'entrée doit modifier le contenu de la cellule.

Forget gate : décide s'il faut remettre à 0 le contenu de la cellule.

Output gate : décide si le contenu de la cellule doit influer sur la sortie du neurone.

Le mécanisme des trois portes est strictement similaire. L'ouverture/la fermeture de la vanne est modélisée par une fonction f qui est généralement une sigmoïde et cette sigmoïde est appliquée à la somme pondérée des entrée.

On retrouve également deux types de sorties, nommées états (*Hidden state* et *Cell state*)

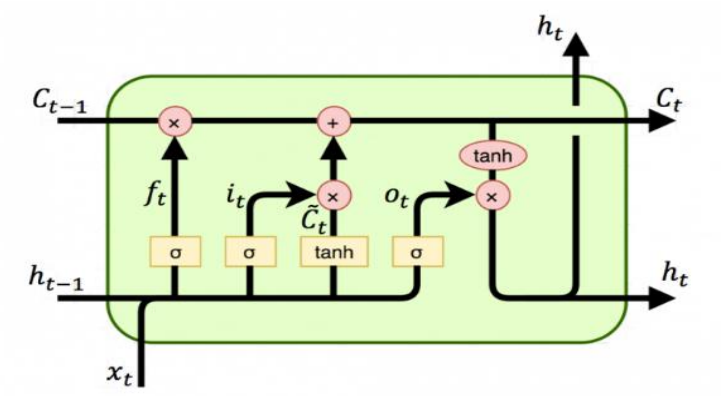


Figure 20: cellule LSTM [39]

Les couches de réseau

Fonctionnement point par point

Vecteur de transfert

Concaténation

copie

Fonctionnement du réseau LSTM

Dans ce qui suit nous essayerons de manière simple, de démystifier le fonctionnement d'une cellule LSTM :

Un des éléments clé des LSTM correspond à l'état de la cellule, la ligne horizontale passant par le haut du diagramme. Les portes sont un moyen de laisser éventuellement passer les informations. Elles sont composées d'une couche de réseau de neurones sigmoïde et d'une opération de multiplication ponctuelle.

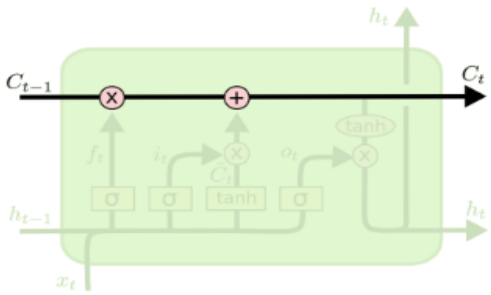


Figure 21: état de la cellule

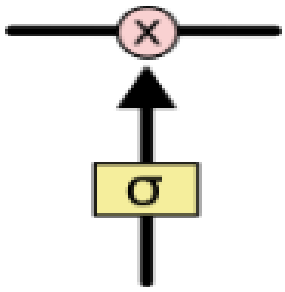
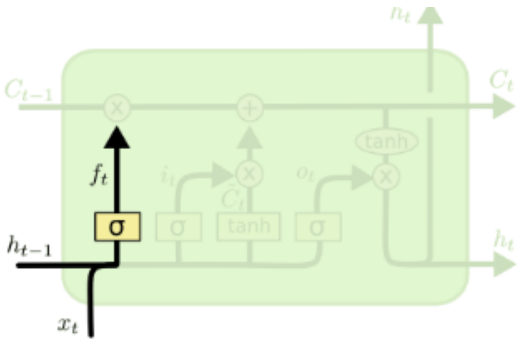


Figure 22: modélisation d'une porte LSTM

La première étape du LSTM consiste à décider quelles informations seront ignorées (oubliées) de l'état de la cellule. Cette décision est prise par une couche sigmoïde appelée couche de porte d'oubli.

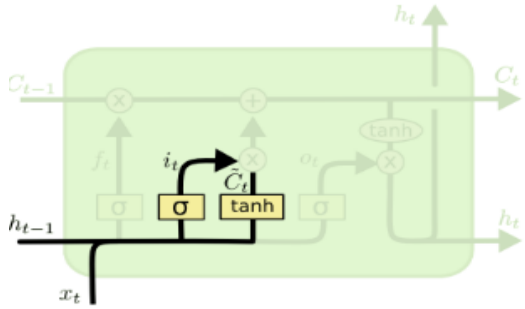


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure 23: porte d'oublie

Avec :
h_{t-1} : La sortie a l'instant *t-1*
x_t : L'entrée courante à l' instant t
b_f : Un vecteur de bais
W_f : Une matrice de poids
σ : Fonction sigmoïde utilisée

L’étape suivante consiste à décider quelles seront les nouvelles les informations à stocker dans l'état de la cellule, c’est-à-dire la mise à jour de l’état.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

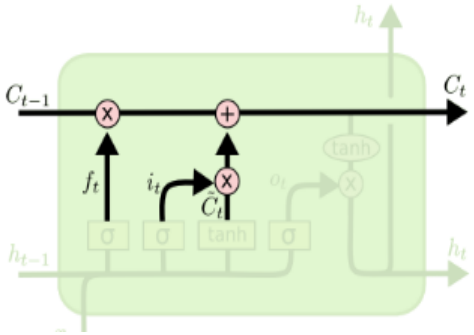
Figure 24: Mise à jour de la cellule

Avec :

tanh : La fonction d’activation tangente hyperbolique

C_t : Valeur candidate

Puis, l’ancien état (reçu en entrée) sera mis à jours dans le nouvel état de cellule.



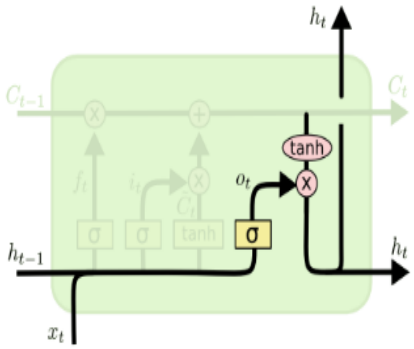
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure 25: Mise à jour de l’ancien état de cellule

Avec :

C_t : État interne

Au final, on obtient une sortie à partir de l'état courant de la cellule



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 26: la sortie de réseau LSTM

Avec :

h_t : Sortie obtenue

Plusieurs variantes de LSTM ont été misent au point.

2.3.3.3. Unité récurrente fermée (GRU)

Cho et al. [2014] ont proposé une simplification de l'architecture LSTM sous le nom de Gated Recurrent Unit (GRU). Ils n'utilisent p connexions peepholes ni fonctions d'activation de sortie, mais ont choisi de fusionner les portes d'entrée et d'oubli en une seule porte, dite de mise à jour. Enfin, ils ont modifié la fonction de la porte de sortie (renommée aussi porte de réinitialisation) afin qu'elle ne soit plus en charge que de la transmission des connexions récurrentes à l'entrée de bloc. [40]

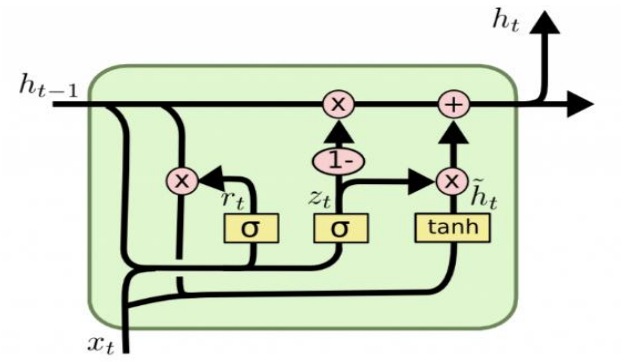


Figure 27 : Unité récurrente fermée

Avec :

- h_t : vecteurs de couche cachés.
- x_t : vecteur d'entrée.
- σ, tanh : fonctions d'activation

2.4. Les représentations de texte usuelles en apprentissage profond

L'un des principaux problèmes de l'analyse des sentiments consiste à créer des représentations de texte adéquates pour l'analyse, ce qui revient à préparer le texte pour répondre aux exigences d'entrée des systèmes de classification. En effet, après avoir converti les données dans un format structuré, il est nécessaire d'avoir un modèle de représentation de texte efficace pour construire un système de classification efficace. Nous avons abordé dans la section 1.2, quelques modèles couramment utilisés en apprentissage automatique classique. Pour le cas du deep learning, ces méthodes dépendent beaucoup de la représentation des données, puisque ces représentations peuvent coder différents types de relations cachées (syntaxiques, sémantiques, etc.) dans les données. Les tâches conventionnelles de traitement de langage naturel utilisent souvent la représentation one-hot, qui consiste à représenter chaque mot du vocabulaire par un vecteur de la taille du vocabulaire. Toutefois, cette représentation simple du mot se heurte à plusieurs difficultés. Dans cette section, nous essayerons d'aborder les trois principaux types de représentation utilisés en apprentissage profond.

2.4.1. Représentation en sac de mots

Nous avons brièvement abordé ce modèle dans la section 1.2. Celui-ci est l'une des techniques de représentation la plus utilisée en TALN. Compte tenu d'un ensemble de documents, nous identifions d'abord l'ensemble des mots utilisés dans l'ensemble de la collection, communément appelé vocabulaire, puis cet ensemble sera réduit, en ne conservant que les mots qui sont utilisés dans au moins deux documents. A partir du moment où le vocabulaire est défini, chaque document peut être représenté comme un vecteur (avec des entrées numériques) de longueur *m*, avec *m* la taille du vocabulaire.

Afin de calculer la longueur *n* du vecteur de document, nous pouvons utiliser la formule suivante :

$$n = \sum_{j=1}^{m} x_j$$

Où, x est la représentation vectorielle du document et x_j est le nombre d'occurrences du mot *j* dans le document.

Le schéma de sac de mots présente néanmoins certaines limites. En effet, l'ordre des mots est perdu, ce qui fait que différentes phrases peuvent avoir exactement la même représentation,

tant que les mêmes mots sont utilisés. Dans le but de réduire l'impact de cette limitation, une variante particulière, dite sac de n-gramme, a été mise en œuvre et où l'on identifie les expressions de plusieurs mots présentes dans le document, garantissant l'ordre des mots notamment dans le contexte des textes courts. Des expressions telles que « Quelle drôle d'idée » ou le terme anglais « United States » seront détectées comme des unités uniques lors de l'utilisation d'un sac de trigrammes ou d'un sac de bi-grammes, respectivement.
Malheureusement, la représentation sac de mots et sa variante sac de n-grammes ont peu de sens sur la sémantique des mots, et ils souffrent tous deux de problèmes de rareté des données et de la haute dimensionnalité des données obtenus en résultat.

2.4.2. Représentation vectorielle des mots

Tel que décrit ci-dessus, le modèle en sac de mot est une représentation simple du mot qui se heurte à plusieurs difficultés. La plus critique est que cette représentation ne peut pas capter de relations entre les données, même s'il existe une forte corrélation sémantique ou syntaxique entre certains d'entre elles.

D'autres stratégies ont émergé pour capturer les similarités sémantiques et syntaxiques dans ces données. La plupart s'appuie sur le fait que les mots dans des contextes similaires ont des significations semblables. En se focalisant sur ce principe, de nombreuses méthodes ont été étudiées dans la communauté TALN pour la mise en place de méthodes de représentations adéquate.

En analyse d'opinion, plusieurs travaux se sont penchés sur le problème.[17] Il en ressort que les représentations dites distribuées tels que le Word Embedding, offrent des résultats relativement intéressants en fonction du type d'algorithme de DL utilisé.

A juste titre, les word embeddings ont été utilisés avec succès en tant qu'informations supplémentaires dans plusieurs tâches, notamment pour le TALN (étiquetage morpho-syntaxique, le regroupement en syntagme, la reconnaissance d'entités nommées etc.)

Les words embeddings constituent une projection des mots du vocabulaire dans un espace de faible dimension de manière à préserver les similarités sémantiques et syntaxiques. Ainsi, si les vecteurs de mots sont proches les uns des autres en terme de distance, les mots doivent être sémantiquement ou syntaxiquement proches. Chaque dimension représente une caractéristique latente du mot, qui peut capter des propriétés syntaxiques et sémantiques.

L’un des principaux avantages par rapport à la représentation en sac de mots décrite ci-dessus sont : la réduction de la dimensionnalité, qui rend la représentation plus efficace, mais également la similitude contextuelle qui permet une représentation plus expressive. Compte tenu de la réduction de la dimensionnalité et contrairement à l'approche sac de mots, sur laquelle la taille des vecteurs croît de façon exponentielle avec la taille de la collection de documents, les word embeddidng visent à créer des représentations vectorielles avec une dimensionnalité beaucoup plus faible.

Actuellement, le word embeddings le plus populaires dans la littérature est fournis par la boite à outils Word2vec. Les auteurs dans Word2vec [12] ont proposé au préalable deux architectures (CBOW et Skip-gram) pour l’apprentissage des word embeddings dont l’avantage est leur faible temps de calcul. Pour mieux appréhender l’amélioration apportée dans Word2vec, un léger éclaircissement s’impose.

- **Principe du Word Embedding**

Initialement, Les word embeddings ont été introduits à travers la construction de modèles de réseaux neuronales [54]. Cela consiste à apprendre un réseau de neurones pour estimer la probabilité du prochain mot, en s’appuyant sur la représentation continue des mots précédents. Cette représentation est apprise au fur et à mesure au moment de l’apprentissage du réseau de neurones.

Deux tâches sont opérées, d’abord, la projection des indices des mots du contexte dans un espace continu afin d’obtenir leurs représentations vectorielles. Ces représentations sont ensuite concaténées pour construire l’entrée de la couche cachée. Deuxièmement, la probabilité de chaque mot dans une liste de mots est calculée en sortie du réseau.

Ces deux tâches peuvent être effectuées par un réseau de neurones composé de deux couches cachées. La première correspond à la représentation continue de tous les mots du contexte et la deuxième couche cachée est nécessaire pour obtenir une estimation de probabilité.

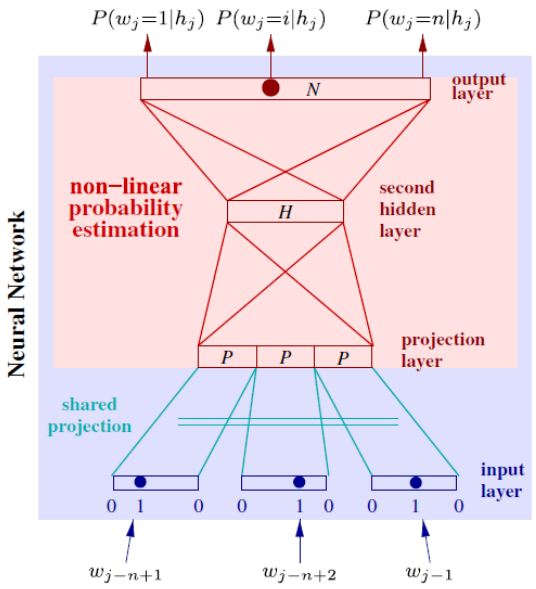


Figure 28: Architecture en neurones pour la modélisation du langage [55]

Dans la couche de sortie, chaque neurone correspond à la probabilité d'un mot en fonction du contexte (calculée par la fonction Softmax afin de garantir que la somme de toutes les probabilités soit égale à 1). Une approche intuitive consiste à ce que, la couche de sortie contient tous les mots du vocabulaire. Avec cette approche, l'utilisation d'un grand vocabulaire peut être très couteuse en termes de temps d'exécution, le temps de calcul étant linéaire par rapport à la taille du vocabulaire. C'est pour cette raison que Schwenk dans [55] a proposé dans son modèle nommé CSLM (Continuous Space Language Models), de se restreindre à une liste des mots fréquents (short list).

Collobert et Weston (C&W) [56] ont par la suite présentés une autre approche pour la modélisation neuronale du langage. Ils proposent une solution au problème soulevé ci-dessus. Pour éviter le calcul de la couche de sortie sur tout le vocabulaire, ils ont proposé d'utiliser une autre fonction de coût pour l'apprentissage.

- **Modèle Word2vect**

Considéré comme le modèle le plus populaire actuellement, celui-ci a été introduit par [54]. Les auteurs ont proposés d'abord deux architectures (CBOW et Skip- gram) pour l'apprentissage des word embeddings qui sont moins coûteuses en termes de temps de calcul que les modèles précédents. Puis, ils ont amélioré ces modèles en employant des stratégies supplémentaires pour améliorer la vitesse d'apprentissage et la performance des word

embeddings [57]. Ces architectures offrent deux principaux avantages par rapport aux modèles CSLM et C&W :

- Ils ne contiennent pas de couches cachées ;
- Ils permettent au modèle de langage de prendre en compte un contexte supplémentaire.

Ainsi [12] ont introduit :

- **CBOW**

Ce modèle consiste à prédire un mot w_i en fonction de son contexte, qui correspond aux mots précédents et aux mots suivants. Dans cette architecture, la couche de projection est partagée par tous les mots : tous les mots sont projetés dans la même position. Ceci est différent de l'approche utilisée pour le modèle de langage (CSLM), où les mots sont projetés dans des positions différentes.

Ce modèle est nommé CBOW pour sac-de-mots continus ou Continuous Bag of Words pour deux raisons. D'une part, l'appellation sac-de-mots indique que l'ordre des mots du contexte n'a pas d'influence sur la projection. D'autre part, l'objectif souligne l'utilisation d'une représentation continue (word embeddings) des mots en contexte, ce qui diffère des modèles sac-de-mots standards.

- **Skip-gram**

L'architecture skip-gram est l'inverse de CBOW. Le mot cible m_i est maintenant utilisé comme entrée, et les mots de contexte sont utilisés en sortie. Elle consiste à prédire, pour un mot donné, le contexte dont il est issu.

Dans cette architecture le mot en entrée est projeté vers la couche de sortie pour produire un vecteur. Ce vecteur est ensuite comparé à chacun des mots du contexte et le réseau se corrige par rétro-propagation du gradient. De cette manière, la représentation vectorielle du mot d'entrée va être proche de celles des mots qui produisent le même contexte que lui.

En outre, le contexte ne se limite pas au contexte immédiat, et les instances d'apprentissage peuvent être créées en ignorant (sautant) un nombre de mots dans son contexte, par exemple, m_{i-4}, m_{i-3}, m_{i+3}, m_{i+4}, d'où le nom de Skip-gram.

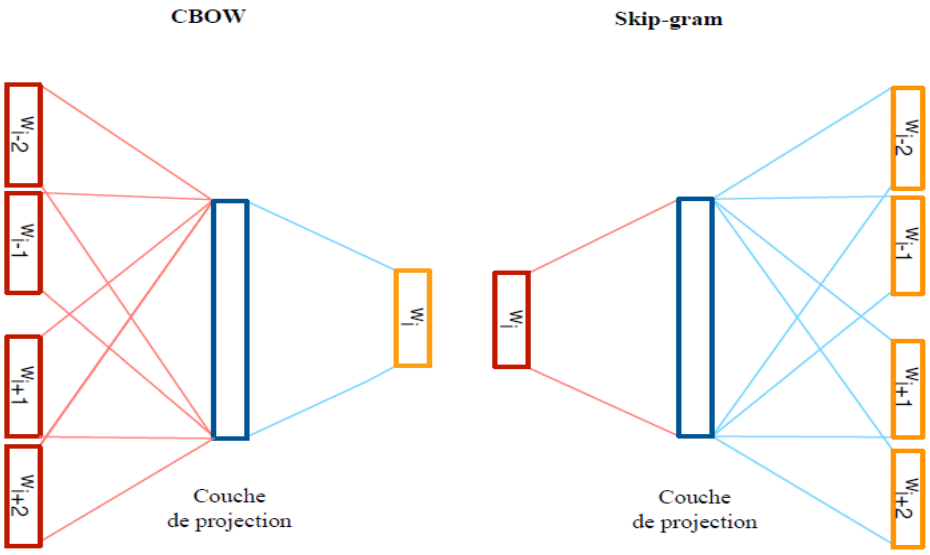


Figure 29: L'architecture CBOW prédit le mot courant en fonction du contexte, et Skip-gram prédit les mots en contexte en fonction du mot courant.

➢ Modèle Glove

Il existe d'autres variantes de word embeddings. A titre d'exemple, GloVe est une approche plus récente que Word2vect proposée par (Pennington, Socher, et Manning 2014) et réunissant deux types approches : la factorisation de matrice « count-based » et les modèles prédictifs ou neuronaux. Malheureusement, nous ne sommes pas intéressés en détail à ce type d'approche vu que cela ne concerne pas le sujet de notre mémoire.

Néanmoins, ce qui est important de préciser est que, ce modèle repose sur la construction d'une matrice de co-occurrence globale des mots *MG*, en traitant un corpus de données en utilisant une fenêtre contextuelle glissante.[14] Ainsi, chaque élément MG_{ij} représente le nombre de fois où un mot m_j apparaît dans le contexte du mot m_i.

GloVe est un modèle d'apprentissage non supervisé qui prend en compte toute l'information portée par le corpus et non pas la seule information portée par une fenêtre de mots, d'où le nom GloVe, pour VEcteur GLObal. Une fois la matrice *MG* calculée, un modèle de régression est entrainé pour construire des représentations vectorielles $\underset{m_i}{\rightarrow}$ et $\underset{m_j}{\rightarrow}$. Ces représentations doivent conserver des informations utiles sur la cooccurrence de pair de mots m_i et m_j , telles que : $\underset{m_i}{\rightarrow}^T \underset{m_j}{\rightarrow} + b_i + b_i = logMG_i$ Où b_i et b_j sont les vecteurs biais associés respectivement pour les mots m_i et m_j.

- **Le Word Embedding pour l'analyse d'opinions**

L'avantage du modèle Glove est qu'il peut être formé rapidement sur plus de données car l'implémentation peut être parallélisée. D'autre part, au lieu d'apprendre la représentation du mot complet, char2vec [58] peut apprendre l'incorporation associée à chaque caractère d'un mot. C'est ce qui a poussé plusieurs travaux à exploiter son utilisation en analyse d'opinions.

Plus récemment, de nombreux travaux proposent de nouvelles approches sur la représentation des mots pour l'analyse des sentiments, vu que les méthodes citées ci-dessus apprennent des distributions de mots indépendantes de toute tâche spécifique. Pour l'analyse des sentiments, ce problème peut être surmonté en utilisant les connaissances préalables disponibles sous la forme de label d'opinions ou de mots annotés issus de lexiques de sentiments.

[61] ont proposé un framework qui combine différents niveaux de connaissances préalables dans les words embedding pour l'analyse des sentiments. Les résultats expérimentaux sur des données du monde réel démontrent que la méthode de représentation de mots proposée améliore les performances des systèmes d'analyse des sentiments par rapport aux approches de word embedding classique.

[62] ont proposé une nouvelle approche en appliquant des plongements (embedding) stochastiques pour la classification des sentiments inter domaines, qui préserve les structures de similitude dans l'espace d'intégration. Néanmoins, les résultats obtenus sont moins attrayants que ceux espérés par l'auteur.

Et enfin, [63] ont proposé un modèle qui apprend l'intégration des sentiments (sentiment embedding) en utilisant des scores d'intensité d'opinions à partir de lexiques de sentiment. Cela a nettement amélioré les vecteurs de mots car ils sont d'un point de vu sémantique et opiniâtre plus proches de mots relativement similaires.

2.4.3. Représentation vectorielle de documents

[64] ont proposé une extension de l'approche Word2Vec qui vise à construire des words embedding à partir de documents entiers. Cette nouvelle extension est nommée Doc2Vec ou Paragraph2Vec.

Si l'on se réfère à l'analyse d'opinions, les systèmes habituellement utilisés pour la représentation à partir de documents entiers exigent généralement que l'entrée de texte soit représentée sous forme de vecteurs de longueur fixe. Ainsi, en raison de sa simplicité, de son efficacité et de sa précision parfois surprenante, la technique de sac de mots a souvent été appliquée pour l'obtention de vecteurs de longueur fixe.

Toutefois, des techniques comme Doc2Vec ont vu le jour pour remplacer les insuffisances des sacs de mots sur cette spécificité de longueur. Sommairement, Doc2Vec consiste à combiner les vecteurs de mots fournis par l'algorithme Word2Vec. L'idée principale est de se retrouver avec un seul vecteur d'agrégation qui représente la sémantique de l'ensemble du document.

Cette technique nous permet de représenter un document dans un petit espace de données, c'est-à-dire en quelques centaines de dimensions. L'objectif principal est d'obtenir un bon équilibre où des documents similaires seront regroupés, laissant suffisamment d'espace pour discerner un cluster des autres.

Cette propriété a une grande pertinence pour les systèmes de classification. En effet, appliquer Doc2Vec à une collection de documents et en définissant une dimensionnalité raisonnable, peut permettre un apport considérable aux algorithmes de deep learning, afin qu'ils puissent mieux définir la frontière qui sépare les différentes catégories que l'on veut distinguer.

2.5. Conclusion

Dans ce chapitre nous avons survolé les principales notions liées aux réseaux de neurones artificiels. Dans un premier temps nous avons illustré les principales étapes d'apprentissage d'un réseau de neurone à travers l'exemple du perceptron. Nous avons également décrit les principales couches qui entrent en jeu dans un processus de deep learning ainsi que quelques modèles de réseaux de neurones communément utilisés. Nous avons également abordé quelques techniques de représentation usuelles en apprentissage profond et en particulier la notion de word embedding qu'on aura à utiliser dans la suite de notre travail.

Chapitre 3 : Description de l’approche ABSA

3.1. Introduction

Depuis ses débuts, le domaine de l'analyse de sentiment a vu se complexifier toutes les taches intervenants dans le processus de détection et d'identification de l'opiniion réorientant ainsi les objectifs et les méthodes proposées. Au tout début, les principaux travaux se sont attachés à la résolution des problèmes de classification : classification binaire en termes de polarité, de phrases et de textes. Ceci s'est fait en utilisant différents algorithmes d'apprentissage automatique (classifieur naïf bayésien, Machine à Vecteur de Support, classifieur d'entropie maximale) et en définissant des caractéristiques pertinentes pour l'expression d'opinions (fréquence des mots en unigramme ou n-gramme, catégories grammaticales, informations sur les dépendances syntaxiques, lexiques évaluatifs etc...)

Ces méthodes ont permis de répondre à différentes problématiques liées à l'opinion, allant d'une simple détection au sein de la phrase à des analyses complexes au sein des textes. Néanmoins, Une problématique subsiste toujours lorsqu'il est question d'analyser les opinions en lien avec une cible précise au sein du texte ainsi que les caractéristiques (aspects) de cette cible. En effet, au sein d'un même texte on retrouve fréquemment des passages subjectifs qui font références à des sujets et entités cibles relativement distincts, mais également à propos de différentes caractéristiques ou aspects d'une cible. La tâche qui consiste à détecter et classer les opinions des différentes caractéristiques (ou aspects) d'une cible dans le texte, est connue sous le nom d'analyse d'opinion basée sur les aspects (Aspect Based Sentiment Analysis - ABSA-). Il s'agit donc d'une analyse approfondie dont le but est de déduire la polarité du sentiment exprimé à propos d'entités mentionnées dans les textes ou de certains de leurs aspects.

Dans ce chapitre nous présentons l'approche d'analyse d'opinions basée sur les aspects ainsi que ses différentes sous taches. Nous aborderons également les différents travaux connexes qui utilisent analyse d'opinions et puis, nous terminerons par la représentation de notre approche.

3.2. L'Analyse des Sentiments Basée sur l'Aspect (-ABSA-)

Bien que l'analyse d'opinion au niveau du document et au niveau de la phrase soit utile dans de nombreux cas, elle laisse encore beaucoup à désirer. En effet, un texte subjectif à connotation positif sur une entité (cible) particulière ne signifie pas que l'auteur a des opinions positives sur tous les aspects de l'entité. De même, un texte négatif pour une entité ne signifie pas que l'auteur n'aime pas tout ce qui concerne l'entité. Habituellement sur les sites de e-commerce ou sur des sites dédiés aux critiques de produits, l'émetteur d'un commentaire ou d'une critique écrit généralement les aspects positifs et négatifs du produit, bien que le sentiment général sur le produit puisse être positif ou négatif. Pour obtenir une analyse d'opinion plus fine, il est essentiel d'approfondir l'analyse au niveau de l'aspect.

Ainsi, pour une entité (cible) donnée, l'identification de la polarité des opinions basées sur les aspects consiste à déterminer la polarité associée à chaque aspect de la cible. Cependant, les aspects d'une cible ne sont généralement pas connus a priori, et ils varient d'une cible à une autre. Par exemple, les aspects d'un téléphone mobile sont sa batterie, son appareil photo, son poids, etc., alors que les aspects d'un film de cinéma sont son histoire, son jeu d'acteurs, ses décors, etc. De ce fait, une difficulté additionnelle en ABSA est d'extraire les aspects dans un premier temps. La seconde étape, similaire à la classification de la polarité d'opinions (générale), associe une polarité aux différents aspects extraits.

Les opinions basées sur des aspects ont été le sujet d'un important nombre de travaux et ont également fait l'objet de plusieurs tâches dans la campagne d'évaluation SemEval [59]. Cette tâche présente également plusieurs avantages dans divers domaines tel que le commerce électronique, les réseaux sociaux, les déférentes entreprises...etc. Contrairement à une analyse de sentiments classique, l'ABSA permet :

- **Analyse en temps réel :** L'analyse des sentiments basée sur les aspects permet aux entreprises de se concentrer sur les aspects d'un produit pour entrevoir les forces et les faiblesses de leurs produits et services pour enfin diagnostiquer les éventuels d'extraire les problèmes et les bugs majeur et agir le plus tôt possible (voir même en temps réel).
- **Meilleure compréhension du client :** Il est plus facile de classer un texte comme positif ou négatif que de passer du temps à analyser chaque phrase d'un texte. Mais en

utilisant un système automatisé d'analyse des sentiments basés sur les aspects, les entreprises peuvent acquérir une compréhension plus approfondie de produits et services spécifiques rapidement et facilement, et se concentrer vraiment sur les besoins et les attentes de leurs clients.

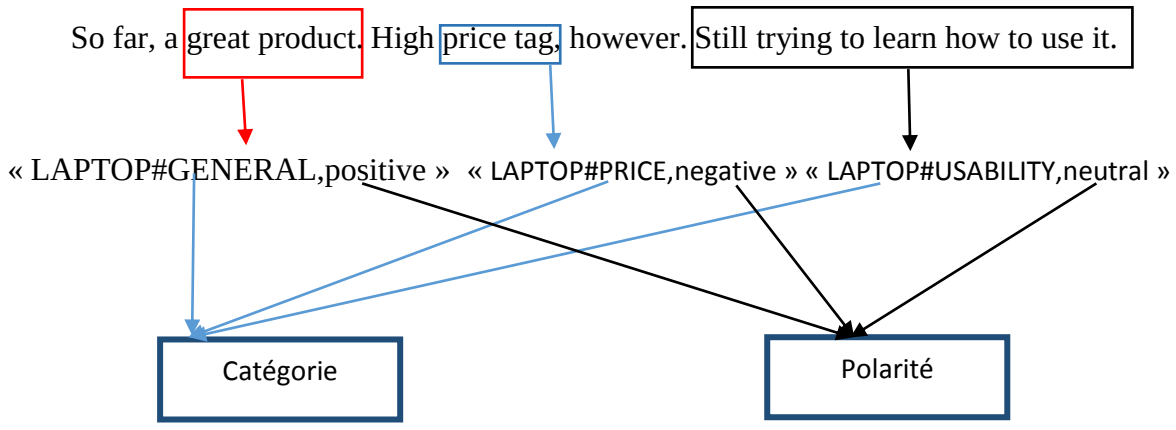
On distingue dans la littérature trois types de travaux sur l'ABSA et que l'on essayera de détailler ci-dessous.

3.2.2. Sous tâches de l'ABSA

L'analyse de sentiment basée sur les aspects comporte trois sous tâches qui sont l'extraction de catégorie d'aspect, l'extraction de terme d'aspect et la classification des polarités au niveau des aspects

- *Détection de catégorie d'aspect (ACD)* : est une sous-tâche importante de l'analyse des sentiments basée sur l'aspect. Étant donné un ensemble de catégories d'aspects prédéfini et un ensemble de phrases, l'ACD a pour fin d'attribuer à chaque revue l'ensemble des catégories correspondantes. Parmi les diverses approches ayant été appliquées, on retrouve des méthodes basées sur des règles grammaticales ainsi que sur l'apprentissage automatique, dont la plupart sont de nature statistique [47]

  Dans une ACD, une ou plusieurs « catégories d'aspects » évoquées dans une phrase peuvent être identifiées. Dans l'exemple numéro 1 on constate que



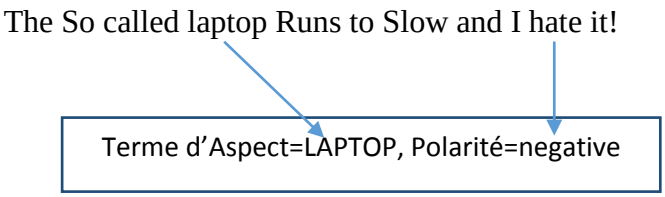
Exemple numéro 1

Le client dans la phrase a exprimé plusieurs polarités {positive, négative, neutre} qui correspondent à plusieurs catégories {LAPTOP#GENERAL, LAPTOP#PRICE, LAPTOP#USABILITY}. Nous remarquons qu'il est satisfait du produit en générale (great product) par contre il le trouve cher (price tag) et il exprime un avis neutre concernent l'usabilité du laptop. Cette analyse distinctive est qui est l'avantage de l'ABSA par rapport à l'approche classique dans laquelle les avis sont typiquement consacrés à une seule caractéristique du produit ou service

- *Extraction de termes d'aspect (OTE)*

  Cette sous tâche consiste à extraire l'entité dont parle l'avis et vise donc à identifier tous les mots qui expriment un aspect, ou bien les termes aspectuels. Pour réaliser cette tâche. [48]

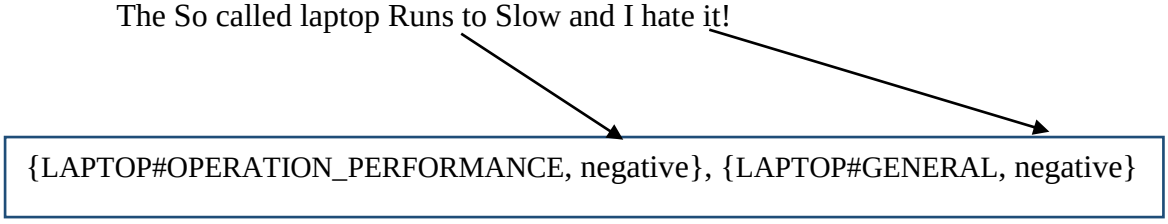
  Cela peut se faire de manière explicite ou implicite. Implicite sous-entend que l'entité ou le sujet a déjà été mentionné au sein de la phrase en question ou une autre phrase. Dans le cas d'une référence implicite, l'entité est attribuée la valeur « NULL ». [48]



Exemple numéro 2

- *la classification des sentiments au niveau des aspects (SP)*

  Polarité du sentiment (SP) aborde le problème de la classification de texte dans le contexte où ces catégories prédéfinies se voient attribuer des sentiments comme positifs, négatifs, neutre ou conflit. Il est beaucoup plus difficile de trouver des sentiments à un niveau d'aspect par rapport au niveau global de la phrase, car la même phrase peut avoir des polarités de sentiment différentes pour différents aspects. [47]



Exemple numéro 3

3.2.1. Travaux connexes

Beaucoup de travaux se sont intéressé à l'analyse de sentiment niveau aspect. Ainsi, on retrouve :

- *Soujanya Poria, Erik Cambria :* Dans leur recherche, Soujanya Poria, Erik Cambria ont présenté une approche d'apprentissage en profondeur de l'extraction d'aspect. Pour la réaliser ils ont utilisé un réseau neuronal convolutif profond à 7 couches pour marquer chaque mot dans des phrases d'opinion comme un mot aspect ou non. Ils ont également développé un ensemble de modèles linguistiques dans le même but, et les avons combinés avec le réseau neuronal. Le classificateur d'ensemble résultant, associé à un modèle d'inclusion de mots, a permis à cette approche d'obtenir une précision nettement meilleure que d'autres méthodes.[41]
- *Toh & Su (2016) :* c'est une étude qui a pour but de détecter les catégories d'aspect, Le principe est de traiter le problème comme un problème de classification multi-classes où les catégories d'aspect sont prédites via un ensemble de classificateurs binaires. Chaque classificateur est formé à l'aide d'un réseau à action directe à une seule couche. [44]
- Wei Xue, Tao Li(2018) : ils ont proposé un modèle basé sur des réseaux de neurones convolutifs. Premièrement, les nouvelles unités Gated Tanh-ReLU peuvent produire de manière sélective le sentiment en fonction de l'aspect ou de l'entité donné. L'architecture est beaucoup plus simple que la couche d'attention utilisée dans les modèles existants. Deuxièmement, les calculs de modèle pourraient être facilement parallélisés pendant la formation, car les couches convolutives n'ont pas de dépendance temporelle comme dans les couches LSTM, et les unités de porte fonctionnent également indépendamment. Les expériences sur les ensembles de données SemEval démontrent l'efficacité de ces modèles. [42]
- *Taha Shangipour Ataei, Kamyar Darvishi , Soroush Javdan , Behrouz Minaei-Bidgoli , Sauleh Eetemadi :* Cet article fournit un ensemble de données persan annoté manuellement, Pars-ABSA, qui est vérifié par 3 locuteurs natifs du persan. L'ensemble de données comprend 5 114 échantillons positifs, 3 061 négatifs et 1 827 échantillons de données neutres provenant de 5 602 avis uniques. De plus, à titre de référence, cet article rapporte les performances de certaines méthodes d'analyse des sentiments basées sur les aspects avec un accent sur l'apprentissage en profondeur, sur Pars-

ABSA. Les résultats obtenus sont impressionnants par rapport à un état de l'art anglais similaire.

- *Zhuang Chen, Tieyun Qian (2020) :* dans cette étude ils ont proposé un cadre RACL (Relation-Aware Collaborative Learning) qui permet aux sous-tâches de l'ABSA (l'extraction de termes d'aspect, l'extraction de catégories d'opinion et la classification des sentiments au niveau des aspects) de travailler de manière coordonnée via les mécanismes d'apprentissage multitâches et de propagation de relations dans un réseau multicouche empilé. Des expériences approfondies sur trois ensembles de données du monde réel démontrent que RACL surpasse considérablement les méthodes de pointe (state-of-the-art methods) pour la tâche ABSA complète. [43]
- *Murtadha Ahmed, Qun Chen, Yanyan Wang, Zhanhuai Li (2019) :* Dans ce travail, ils ont proposé une nouvelle approche de réseau de neurones qui vise à explorer la connexion entre un aspect et son contenu sémantique dans la phrase. Le mécanisme d'attention est conçu pour se concentrer sur différentes parties d'une phrase en fonction des indicateurs des aspects. Ils ont expérimenté sur des ensembles de données de référence (restaurant SemEval 2014 tâche 4 et ordinateur portable SemEval 2016 tâche 5), et les résultats montrent que le modèle atteint des performances considérables sur la tâche d'identification d'aspect. [44]
- *Mengting Hu, Shiwan Zhao, Honglei Guo, Renhong Cheng, Zhong Su (2019) :* Dans cet article, ils ont proposé une approche qui détermine les positions de début et de fin de l'extrait d'opinion, et sélectionne les mots entre ces deux positions pour la prédiction des sentiments. Plus précisément, ils apprirent des associations profondes entre la phrase et l'aspect, et les dépendances à long terme En particulier, les résultats expérimentaux démontrent l'efficacité de la méthode et prouvent que cette approche plus performante que d'autres approches lors du traitement de phrases multi-aspects. [45]
- *Feiyang Ren, Liangming Feng, Ding Xiao, Ming Cai, Sheng Cheng :* Cet article présente le réseau de distillation (DNet), un modèle d'analyse des sentiments léger et efficace basé sur des réseaux de neurones convolutifs. En combinant la convolution gated empilée avec le mécanisme d'attention. Des expériences sur les ensembles de données Twitter SemEval 2014 Task 4 et ACL14 démontrent que cette approche permet d'atteindre des performances très élevés. De plus, par rapport au modèle basé

sur BERT, DNet réduit la taille du modèle de plus de 50 fois et améliore la réactivité de 24 fois. [46]

3.3. Présentation introductive de notre approche d'analyse d'opinions à base d'aspects

Tel que décrit ci-dessus, l'analyse des sentiments basée sur l'aspect est une technique qui présente certains avantages par rapport à l'analyse des sentiments classique. Notre travail, a justement pour objectif de classer les polarités liées aux différents aspects inclus au sein de textes courts.

3.3.1. Présentation du corpus utilisé

3.1.1.1. Présentation de la campagne SemEval

SemEval est une série d'ateliers de recherche internationaux sur le traitement du langage naturel (TALN) dont le but est de faire progresser l'état actuel de l'art en analyse sémantique et d'aider à créer des ensembles de données annotés de haute qualité dans une gamme de problèmes de plus en plus complexes dans la sémantique du langage naturel. L'atelier présente chaque année une collection de tâches partagées dans lesquelles des systèmes d'analyse sémantique computationnelle conçus par différentes équipes sont présentés et comparés. [19]

L'ABSA a été introduit en tant que tâche SemEval en 2014 (SE-ABSA14) fournissant des ensembles de données de référence des revues en anglais et un cadre d'évaluation commun. Avec l'objectif fixé d'identifier tous les aspects abordés d'une entité ou un produit cible ainsi que sa polarité correspondante dans des commentaires clients en anglais, à propos de données des restaurants et des ordinateurs portables [59]. Puis en 2015 SemEval a fourni La tâche 12 d'analyse des sentiments basée sur les aspects (SE-ABSA15) qui représente une continuation de la tâche 4 de SemEval-2014 (SE-ABSA14). Cette tâche a traité des avis dans trois différents domaines : les restaurants, les ordinateurs portables et les hôtels.

La tâche SemEval ABSA 2016 (SE-ABSA16) donne l'opportunité aux participants d'expérimenter les données anglaises (critiques) des domaines de SE-ABSA15 en fournissant de nouveaux ensembles de données de test. En outre, SE-ABSA16 fournira également des ensembles de données dans d'autres langues que l'anglais. Pour chaque domaine, un ensemble commun de directives d'annotation sera utilisé dans toutes les langues. En outre, SE-ABSA16

comprend des annotations au niveau du texte ainsi qu'une méthodologie d'évaluation appropriée. [50]

3.3.1.2. Principaux travaux recensés avec SemEval [60]

Etude	Ensemble de donnée	Domaine	Langue	Modèle utilisé	Résultats obtenue
[44]	SemEval 2016	Restaurant	Anglais	RNN + WE	F1-score : 72.34%
[66]	SemEval 2016	Hôtels	Arabe	RNN SVM	F1-score : 48% 89%
[67]	SemEval 2014, SemEval 2016	Restaurant	Anglais	Memory interaction networks basé sur LSTM	F1-score: 77,58%,73,44%
[68]	SemEval 2014, SemEval 2015	Restaurant, laptop, service web	Anglais	LSTM hiérarchique	F1-score : 77.9%, 76.6%,

Tableau 1: Travaux effectués sur le corpus SemEval

3.3.1.3. Description de corpus

L'ensemble de données que nous avons utilisé est spécifiques au domaine des ordinateurs portables, En particulier, les ensembles de données d'apprentissage et de test SE-ABSA16 qui sont mis à disposition en tant que données de formation. Ils se composent de 450 textes de révision (2500 phrases) annotés avec 2923 tuples {E # A, polarité} Chaque paire E # A est considérée comme une catégorie d'aspect du texte donné.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Reviews>
    <Review rid="348">
        <sentences>
            <sentence id="348:0">
                <text>Most everything is fine with this machine: speed, capacity, build.</text>
            </sentence>
            <sentence id="348:1">
                <text>The only thing I don't understand is that the resolution of the screen isn't high enough for
                some pages, such as Yahoo!Mail.</text>
            </sentence>
            <sentence id="348:2">
                <text>Yes, I have it on the highest available setting.</text>
            </sentence>
        </sentences>
        <Opinions>
            <Opinion category="LAPTOP#GENERAL" polarity="positive"/>
            <Opinion category="LAPTOP#OPERATION_PERFORMANCE" polarity="positive"/>
            <Opinion category="HARD_DISC#DESIGN_FEATURES" polarity="positive"/>
            <Opinion category="LAPTOP#QUALITY" polarity="positive"/>
            <Opinion category="DISPLAY#QUALITY" polarity="negative"/>
        </Opinions>
    </Review>
```

Figure 30: Aperçu d'un commentaire du corpus SemEval 2016

3.3.2. Sous tâches du corpus SemEval

SemEval fournit chaque année des taches pour différents domaines, ces tâches sont à leurs tours réparties en différentes sous tâches. Notre travail a consisté au traitement de la sous tâche 2 liée à la tâche 5 du corpus SemEval 2016. Étant donné un ensemble d'avis sur une entité cible (Exemple : un ordinateur portable), l'objectif de cette sous tâche est de pouvoir identifier un ensemble de tuples {catégorie, polarité} qui résument les opinions exprimées dans chaque avis.

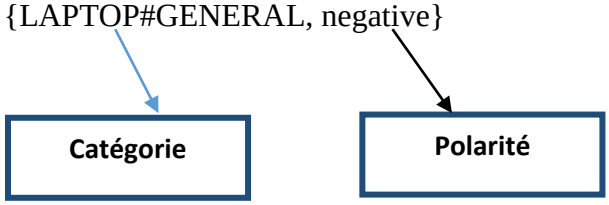
Soit la revue suivante :

"The So called laptop runs to Slow and I hate it! Do not buy it! It is the worst laptop ever."

Le but est d'obtenir les résultats suivants :

{LAPTOP#OPERATION_PERFORMANCE, negative}

{LAPTOP#GENERAL, negative}



Exemple numéro 3

3.4. Intuition de l'approche

Afin de respecter les consignes de la sous tâche 2 introduite sous SemEval 2016 (décrite ci-dessus), nous avons réparti notre travail en deux parties (figure 31). Ainsi, la première phase s'intéressera à la détection des catégories d'aspects (ACD) dans laquelle, on essayera d'explorer et identifier les catégories évoquées dans une phrase, puis la sortie obtenue sera utilisée comme entrée pour l'étape suivante (SP).

Pour se faire, un certain nombre de prétraitements de base sont opérés (tokenisation, suppression de mots vides, corrections de formes etc…). Puis, une fois les données prêtes, nous procédons à l'apprentissage du modèle de catégorisation d'aspects. Ce modèle doit pouvoir prédire la catégorie d'aspect compte tenu du vocabulaire obtenu lors de la tokenisation. A cet effet, nous mettons une approche à base d'une pile de plusieurs couches LSTM. L'idée est de former un réseau profond non seulement en termes de couches mais aussi en dimensions récurrentes.

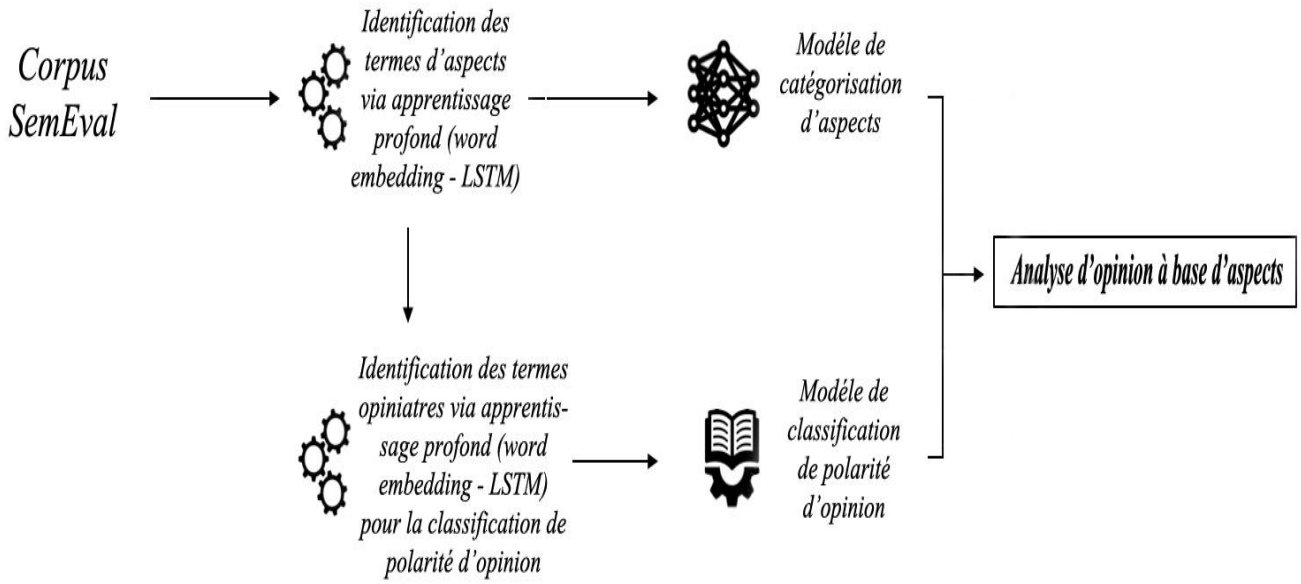


Figure 31: schéma représentatif des différentes étapes de l'analyse d'opinion à base d'aspects

L'intuition est que les couches LSTM supérieures peuvent capturer des concepts abstraits dans des séquences qui pourraient aider dans notre tâche de catégorisation d'aspect mais également, de manière générale, pour l'analyse d'opinions. Pour mieux comprendre le procédé, la figure 32 présente l'architecture du modèle LSTMs employé.

En bref, la figure 32 montre un modèle composé d'une couche d'intégration, de divers couches LSTM et d'une couche d'activation. Aussi, il est à signaler qu'une couche d'abandon avec une probabilité de 0,20 a également été introduite entre la plupart des couches afin d'éviter les problèmes de sur-apprentissage, comme décrit dans le chapitre précédent.

Concernant la couche d'incorporation (embedding), quatre paramètres de données ont été spécifiés. La dimensionnalité d'entrée, la dimensionnalité de sortie, la longueur d'entrée et les poids fourni par le modèle Glove.

La dimensionnalité d'entrée a été définie aléatoirement sur la base du vocabulaire et a été fixé à 100). La longueur d'entrée a également été définie sur la base d'une moyenne des longueurs maximales des phrases recensées au sein du corpus d'apprentissage (fixée à 100 mots). Puis, les poids sous forme de liste de tableaux ont été obtenus via Glove afin de définir les poids d'incorporation (Embedding) initiaux.

D’un autre côté, les couches LSTM présentent principalement trois paramètres, la dimensionnalité de la sortie, l'activation et l'initialisation des poids. La dimensionnalité de sortie est définie comme étant la taille des représentations vectorielles. L'activation est réglée sur une fonction *Relu* et les poids sont initialisés à zéro.

La couche d'activation n'a qu'un seul paramètre. Une fonction sigmoïde est utilisée. La dimensionnalité de sortie correspond au nombre de dimensions de sortie du modèle (obtenu sur la base du nombre de catégories d’aspects à prédire).

Une fois le modèle de catégorisation d’aspects obtenu, le processus passe à la seconde partie de l’approche, où les résultats obtenus à la phase précédente seront exploités et dans lesquelles on retrouve une ou plusieurs « catégories d’aspects » identifiées. Cette seconde phase peut être considérée comme une tâche de classification de polarité multi-classe où chaque texte d’entrée doit être classé comme positif, négatif, neutre ou conflictuel.

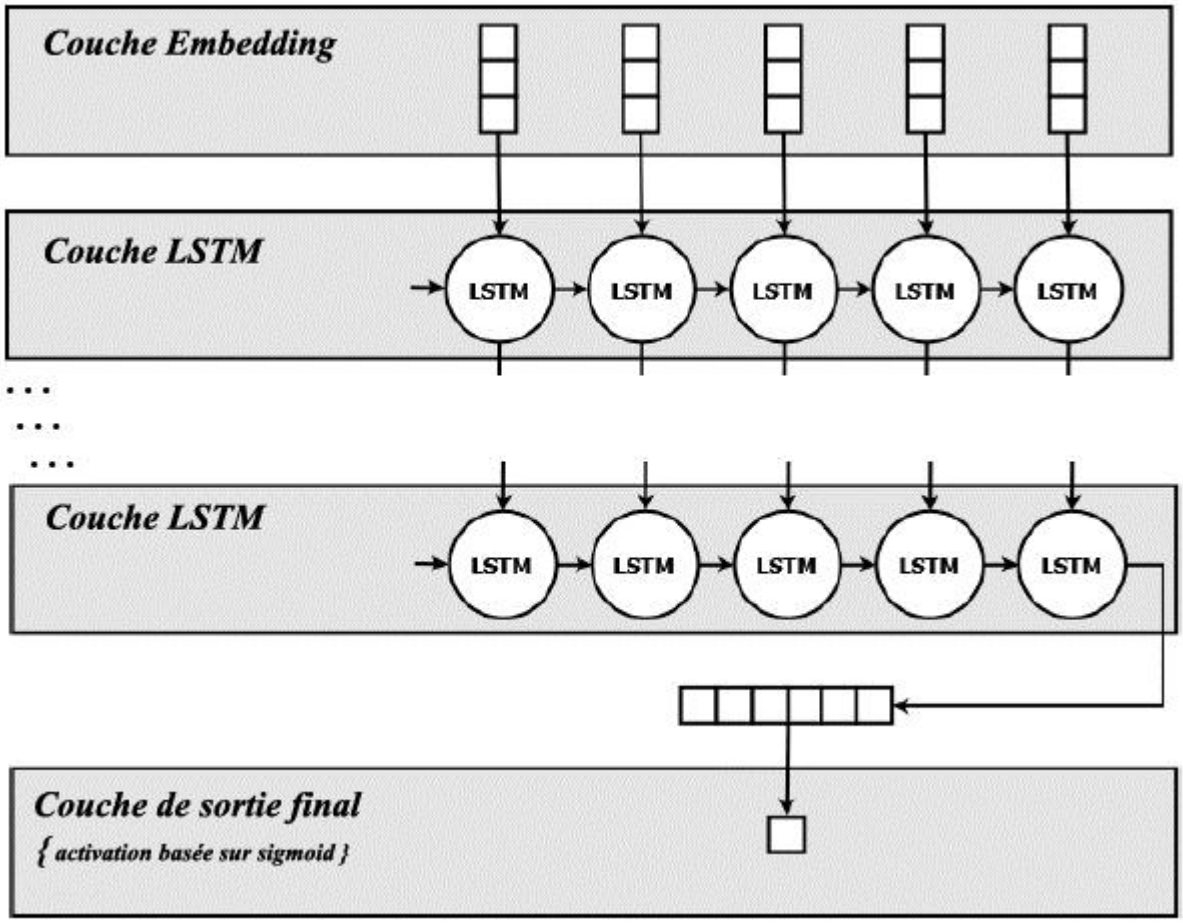


Figure 32: schéma représentatif de l’architecture du modèle LSTM

3.5. Description des paramétrages utilisés pour l'approche

Dans cette partie nous allons présenter les principaux paramètres utilisés pour les phases d'apprentissages.

Tel que décrit précédemment, un certain nombre de prétraitement de données ont été réalisés, où les textes ont d'abord subis une transformation en minuscule sur l'ensemble des mots.

Pseudo code de la phase d'extraction d'aspects

Input : $Corpus_{initial}$ = Ensemble de textes {t1, t2, t3 … t_n}
Output : $Modèle_{aspects\_catégorisation}$

$Corpus_{prétraité}$ = Prétraitement_des_textes ($Corpus_{initial}$)
$Liste_{label}$ = Récupération_des_labels ($Corpus_{initial}$)
$Corpus_{train}$, $Corpus_{test}$ = Méthode_split ($Corpus_{initial}$)
$Liste_{vocabulaire}$ = génération_vocabulaire_corpus($Corpus_{train}$)
$Corpus_{train\_sequence}$, $Corpus_{test\_sequence}$ = Encodage_des_textes_en_séquence_d'entier ($Corpus_{train}$, $Corpus_{test}$)
$Matrice_{embedding}$ = Génération_de_matrice_word_embedding(Modèle_Glove, $Liste_{vocabulaire}$)
$Modèle_{aspects\_catégorisation}$ = Modèle_LSTM($Corpus_{train\_sequence}$, $Matrice_{embedding}$, $Liste_{label}$)

Evaluation du modèle : $Modèle_{aspects\_catégorisation}$($Corpus_{test\_sequence}$)

Puis, une suppression de la ponctuation a été nécessaire afin d'épurer le corpus. Et enfin, un étiquetage grammatical a été réalisé afin de ne sauvegarder que les configurations de type « nom, nom propre et pluriel » ainsi que certaines formes verbales utilisées pour distinguer les catégories d'aspects. Le pseudo algorithme ci-dessous, illustre les différents traitements opérés lors que la phase d'ACD.

Pour l'étape de représentation des données, nous avons opté pour l'utilisation du modèle pré-entrainé de Word Embedding « Glove » au vu des bons résultats (chapitre 4) obtenus grâce à cette technique. Enfin, pour ce qui est du modèle, nous avons adopté un réseau de neurones de type LSTM à plusieurs couches. Ce modèle doit permettre l'extraction des catégories d'aspect compte tenu des données en entrée. Nous l'avons instancié à l'aide de Keras Sequential, qui permet d'ajouter un groupe de couches séquentielles et les empile dans un seul modèle. Nous y avons également ajouté :

- *Couche d'intégration* : Qui correspond à une transformation des séquences numérique (la sortie du word embedding) en vecteurs de taille fixe. Cette couche n'a été utilisée uniquement comme première couche dans notre modèle. La couche Embedding est initialisée grâce aux pondérations de Glove et prend les paramètres suivants :
  - input_dim : Qui correspond à la taille du vocabulaire dans les données de texte.
  - output_dim : Qui correspond à la taille de l'espace vectoriel dans lequel les mots seront incorporés. Il définit la taille des vecteurs de sortie de cette couche pour chaque mot.
  - input_length : Il s'agit de la longueur des séquences d'entrée.

- *Couche LSTM :* En fonction du matériel d'exécution disponible et des contraintes, cette couche choisira différentes implémentations (basées sur cuDNN ou pure-TensorFlow) pour maximiser les performances. Le premier argument est le nombre de nœuds dans les couches cachées. Le deuxième correspond au return_sequences, afin de garantit que la cellule LSTM renvoie toutes les sorties de la cellule déroulée dans le temps.
- *Couche d'abondons* : après chaque couche LSTM nous avons défini une couche d'abondons (Dropout) qui définit de manière aléatoire les unités d'entrée sur 0 à chaque étapes durant le processus d'entraînement, ce qui permet d'éviter le sur-apprentissage.
- *Couches d'activation* : Nous y avons appliqué une fonction d'activation à la sortie des couches LSTM. Etant donnée le fait que nous avons réalisé une classification multi labels qui consiste à obtenir pour chaque commentaire plusieurs catégories en sortie. Ainsi, nous avons favorisé l'utilisation de la fonction sigmoïde. Elle prend comme argument le nombre des catégories (81 catégories au total) et renvoie une valeur entre 0 et 1.

  Toutefois, vu l'intervalle que cette fonction renvoi (entre [0,1]), ainsi nous avons mis en place une procédure d'arbitrage afin de sélectionner au mieux, un seuil optimum permettant de retenir les catégories d'aspect les plus discriminantes.

Une fois l'extraction des catégories d'aspects réalisée, nous procédons à la classification de polarité sur chaque type d'aspect obtenu. La configuration du modèle LSTM décrit ci-dessus a été reprise pour cette tâche. Ses entrées correspondent aux revues concaténées avec leurs catégories annotées dans le corpus, ces données sont ensuite prétraitées puis encodées en

transformant chaque phrase du corpus en une séquence d'entier, et en effectuant un encodage One-hot pour les labels, produisant ainsi un vecteur de longueur égale au nombre de polarité initiales du corpus (soit 4 polarité). Si un point de données appartient à la $i^{ème}$ catégorie, les composantes de ce vecteur reçoivent la valeur 0 pour l'$i^{ème}$ composante et 1 pour le reste.
L'étape suivante consiste à la création de notre matrice d'incorporation (word embedding) qui contient les mots de notre corpus et leurs valeurs correspondantes à partir des incorporations GloVe.

Concernant, la couche d'activation, nous avons utilisé la fonction Softmax, étant donné que le problème traité correspond à une classification multi classe. Elle permet de faire ressortir k probabilités de prédiction dont la somme totale est égale à 1, les k classes retourné dans notre cas correspondent à liste {positive négative neutre ou conflit}

3.6. Conclusion

Dans ce chapitre nous avons présenté un aperçu de l'analyse d'opinions à base d'aspects, ces différentes sous taches ainsi que les avantages qu'elle présente par rapport à l'analyse d'opinions classique. Par la suite, nous avons introduit une présentation de notre approche d'analyse à base d'aspects et les configurations utilisées pour ce type de tâche.

Chapitre 4 : Réalisation

4.1. Introduction

Dans ce chapitre nous allons définir les déférents outils utilisés pour la mise en œuvre de notre application d'analyse d'opinions à base d'aspects. Nous présenterons les résultats obtenus au cours des différentes expérimentations et discuterons ces résultats.

4.2. Présentation des Frameworks utilisés

Pour les besoins de notre approche, nous avons opté pour l'utilisation d'un certain nombre de framworks écrits en Python tel que Tensorflow, keras etc. Ces derniers sont largement utilisés pour la mise en œuvre de modèles en deep learning. Ainsi, on retrouve :

- *Tensorflow* : C'est une bibliothèque de logiciels open source, crée par l'équipe Google Brain pour mener des recherches sur le ML et le Deep Learning. Il s'agit d'une boîte à outils permettant de résoudre des problèmes mathématiques extrêmement complexes avec aisance. Elle permet aux chercheurs de développer des architectures d'apprentissages expérimentaux et de les transformer en logiciels. [52] Nous avons utilisé tensorflow pour importer l'API tf.keras,cela est dû au fait que la bibliothèque Keras est étroitement intégrée à TensorFlow .
- *Keras* : Keras est une bibliothèque open source écrite en python et permettant d'interagir avec les algorithmes de réseaux de neurones profonds et de machine Learning, notamment Tensorflow et Theano. Elle a été initialement écrite par François Chollet [51]. Dans ce travail, Nous avons utilisé la bibliothèque Keras principalement pour la phase de classification ainsi la phase de préparation des données pour l'entrainement. Cependant à fin de convertir les labels encoder en tableau numpy nous avons importé la méthode to_categorical () de la bibliothèque numpy de keras. Nous avons également importé la classe tokenizer qui est utilisé pour encoder les documents, et pad_sequence pour que toutes les séquences ayant la même longueur. Pour construire le modèle d'apprentissage en profondeur, le module « Sequential » est importer pour initialiser le réseau de neurone, le module « Dense » est également importer pour ajouter les différentes couches, commençant par une couche Embedding en suite la couche LSTM suivi des couches d'activation(Relu) et entre chaque couche on trouve une couche Dropout et en fin une couche d'activation (Softmax ,Sigmoid) est appliquer.

- *Numpy* : Est une bibliothèque permettant d'effectuer des calculs numériques avec Python. Elle introduit une gestion facilitée des tableaux de nombres, des fonctions sophistiquées. [51]
  Nous avons utilisé Numpy pour convertir les revues et les labels en tableau numpy, ainsi numpy.stack() pour joindre la séquence des deux tableau ,et aussi np.delete pour supprimer les revues vides.
- *Panda* : pandas est une bibliothèque open source sous licence BSD fournissant des structures de données hautes performances et faciles à utiliser, ainsi que des outils d'analyse des données pour le langage de programmation Python.
  Dans ce travail nous avons utilisé les pandas DataFrame pour représenter nos données sous forme de data frame.
- *Scikit-learn* : est une bibliothèque libre Python pour effectuer de l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment par des instituts français d'enseignement supérieur et de recherche comme Inria et Télécom ParisTech. Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support. Elle est conçue pour s'harmoniser avec d'autres bibliothèques libres Python, notamment **NumPy** et **SciPy**. [53]
  Dans le cadre de ce travail nous avons utilisé la bibliothèque Scikit-learn pour encoder les labels en important le module LabelEncoder, Devisé les données en ensemble de d'entrainement et de teste avec le module train_test_split. Importer les différentes métriques de teste telle que f1_score, precision_score.

4.3. Présentation des expérimentations réalisées et résultats obtenus

4.3.1. Expérimentation 1 : Etudes de l'impact des représentations de textes sur le modèle d'apprentissage utilisé.

Afin d'évaluer l'apport du word embedding sur les performances de notre méthode de classification à base de réseaux LSTM, nous avons opéré une série de tests en variant à chaque expérimentation la représentation numérique du corpus fourni à l'algorithme de deep learning. Ainsi, nous avons opté pour quatre types de représentations puis étudier l'impact de chaque type sur le processus d'apprentissage profond.

Dans un premier temps, le modèle basique de représentation en sac de mots a été utilisé. Les textes du corpus ont été traité afin de découvrir combien de fois chaque mot est apparu dans la phrase. Nous utilisons pour cela une vectorisation, dans laquelle on tokenise une collection de documents texte, puis construisons un vocabulaire de mots avec lesquelles seront encodés de nouveaux documents candidats lors des phases de test.

Dans un second temps, une représentation en bi-grammes et trigrammes de mots a été utilisé. L'intérêt à vouloir séparer les textes en groupes de plusieurs mots (en couples et en groupes de 3), réside dans la nature des textes. En effet, pour étudier idéalement le sens d'un mot il faudrait l'observer dans son contexte. Il existe donc dans un texte (et par extension dans le langage) une forme de dépendance plus ou moins grande entre les mots. C'est justement ce qu'on cherche à capturer à travers ces bi-grammes et trigrammes de mots.

Dans un troisième temps, nous avons opté pour une représentation en pondération tf.idf. L'intérêt d'utiliser cette représentation est double. D'abord, ce type a montré de bons résultats dans divers travaux en TALN ainsi qu'en analyse d'opinion. La seconde raison est que, cela nous permet de varier le type de capture d'information utilisé. En effet, dans les modèles en (bi/tri)-grammes et sac de mots utilisés, seule les fréquences d'apparition des termes ont de l'importance. Le problème est que si l'on veut vraiment représenter un document par les n-grammes qu'il contient, il faudrait le faire relativement à leur apparition dans les autres documents. En effet, si un mot apparait dans d'autres documents, il est donc moins représentatif du document qu'un mot qui apparait uniquement dans ce document, c'est le principe même de la pondération tf idf.

Enfin, la dernière représentation numérique du corpus a été réalisée à l'aide du word embedding issue du modèle Glove. Ce modèle exploite à la fois :

- Les méthodes de factorisation matricielle globale comme l'analyse sémantique latente (LSA) pour générer des représentations de mots de faible dimension
- Les méthodes de fenêtre contextuelle locale telles que le modèle skip-gram de [12]

Pour comparer l'impact des représentations citées ci-dessus sur la performance de l'algorithme LSTM utilisé, nous avons fait le choix d'utiliser la mesure F1-score qui correspond à la moyenne harmonique du taux de précision et du rappel.
Le tableau suivant résume les résultats obtenus en utilisant les déférentes techniques de représentation

Représentations / Mesures	Rappel	Précision	Micro F1_score
Bag-of-Words (Sac de mots)	0.66	0.67	0.67
TF_IDF	0.66	0 .67	0.67
Bigram	0.66	0.67	0.67
trigram	0.67	0.66	0.67
Word Embedding	0.86	0.87	0.86

Tableau 2: Tableau comparatif entre les différentes techniques de représentations

La figure ci-dessous, montre les différentes valeurs de F1_score en variant le nombre d'époques en utilisant la représentation word embedding :

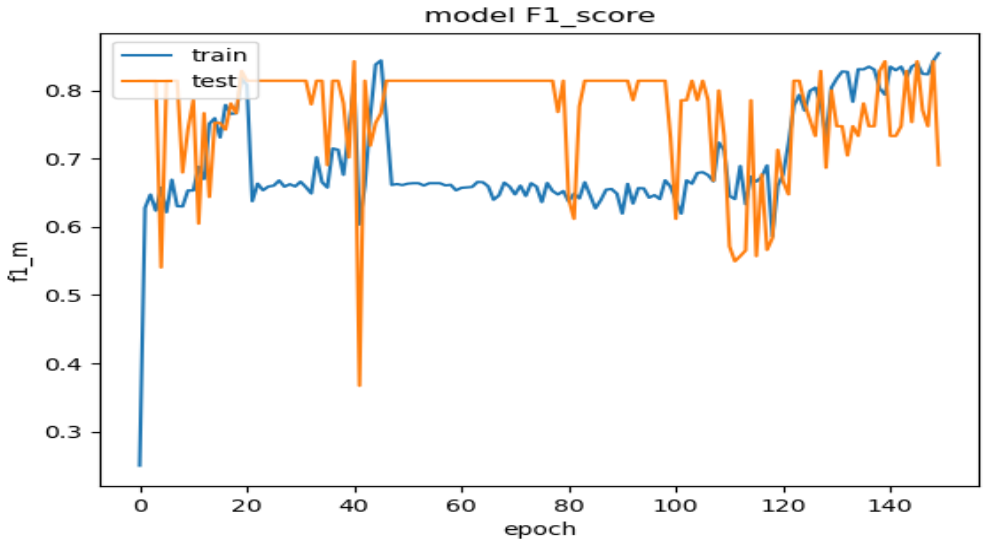


Figure 33: graphe représente les déférentes valeurs de F1_score en utilisant différentes valeurs d'époques

Discussion des résultats

Les résultats obtenus montrent clairement la nette amélioration du taux de F1-score lors de l'utilisation du word embedding contrairement aux autres types de représentations. Ainsi, plusieurs configurations ont été testées durant lesquelles, nous avons varié le nombre d'époques d'apprentissage afin de constater son éventuel impact sur le degré de sur-apprentissage du LSTM. Pour se faire, nous avons aléatoirement choisie un nombre initial de 150 époques, puis nous incrémentons avec un seuil de 10 à chaque nouvelle expérience, afin de constater si l'augmentation du nombre d'époques permettait l'obtention d'un meilleur taux de F1-score. Dans le cas contraire une décrémentation à hauteur de 10 époques est également opérée. Ce procédé est itéré jusqu'à aboutir à nombre d'époques optimum. Puis nous réitérons le processus en diminuant successivement le seuil utilisé.

Fort est de constater que le seuil ayant permis le meilleur taux de F1-score était de 10 époques. Tels que nous pouvons le remarquer dans le tableau numéro 2, les meilleurs score ont été obtenus grâce avec un nombre de 10 époques ajouté à un taux de dropout ajusté à 0.25, le tout a été appris à l'aide de 4 couches d'entrés lstm jointes à 01 couche d'activation.

Dans le tableau numéro 2, on constate que le meilleur résultat de F1-score a été enregistré à l'aide de la représentation word embedding avec un taux de 86%. Ce taux de classification est relativement intéressant au vue des résultats obtenus habituellement dans divers travaux en classification de polarité d'opinion.
Cependant, par manque de travaux ayant étudiés séparément cette tâche à l'aide du corpus Semeval 2016, ainsi il nous est impossible d'affirmer clairement que les résultats obtenus sont probants. Néanmoins, fort est de constater que le word embedding se distingue clairement des autres représentations utilisés.

En contrepartie, le mauvais taux de F1-score a été obtenu avec la représentation en sac de mots. Ceci pourrait s'expliquer du fait qu'avec cette représentation, chaque document est formalisé comme un vecteur de comptage de mots, malheureusement ces mots ne sont pas présents dans chaque document (le ratio *vocabulaire / taille de document* est trop élevé). De plus, vu qu'il n'y a pas eu de tri approfondi car les mots sont traités comme des unités singulières, c'est-à-dire qu'il n'y a pas de notion de similitude entre eux, ainsi certains mots

non pertinents peuvent apparaitre fréquemment (en dehors des mots vides) et d'autres plus pertinents, peuvent apparaitre plus rarement (dans des contextes précis).
Pour conclure, au vu des résultats obtenu, nous avons délibérément prit la décision d'incorporer une couche intégré (embedding layer) à l'aide du modèle Glove lors de notre processus d'analyse d'opinions à base d'aspects.

4.3.2. Expérimentation 2 : Analyse d'opinion à base d'aspect et présentation des résultats de notre approche.

Dans le but d'évaluer notre approche d'analyse à base d'aspects, nous avons effectué un ensemble de tests en utilisant la configuration présenté au chapitre 3. Tel que nous l'avons déjà mentionné, nous avons devisé notre travail en deux parties : Phase d'extraction de catégories d'aspect et Phase de classification de polarité.

4.3.2.1. Résultat de l'extraction des catégories d'aspects

Dans un premier lieu, nous avons voulu étudier l'impact du nombre de catégories d'aspects utilisés sur les performances de notre modèle d'apprentissage à base de réseaux LSTM. Pour cela, nous avons délibérément procéder à une réduction du nombre de catégories d'aspects à prédire. Premièrement, cela permet de mieux ajuster notre procédure d'arbitrage utilisée pour la sélection du seuil optimum fourni par la couche d'activation sigmoïde. Deuxièmement, cela permet d'alléger drastiquement le temps d'apprentissage du modèle.

Ainsi, nous avons d'abord pris en compte l'ensemble des catégories d'aspects (soit 81). Puis nous avons procédé à une réduction aléatoire, jusqu'à atteindre 40% de réduction sur le nombre de catégories initiales. Cette réduction a été opérée sur les aspects rares. En effet, On recense environs 44 catégories d'aspects auxquelles est lié un nombre très limité de textes. Ainsi, nous jugeons qu'il est difficile pour n'importe quel algorithme d'apprentissage (que ça soit classique ou profond) de prédire un aspect avec seulement quelques occurrences lors de la phase d'apprentissage.

Par conséquent, nous avons conservé les catégories les plus prédominantes au sein du corpus et combiné le reste dans une nouvelle catégorie d'aspect «Autre» et ce de manière décroissant. La réduction du nombre de catégories induit également des modifications au sein des paramètres du modèle (nombre de neurones de sorties en diminution), ce qui est sensé

améliorer les performances du modèle, et le rendre moins sensible au sur-apprentissage. Le tableau suivant montre les différents résultats obtenus :

Nombre des catégories	F1_score	Précision	Rappel
81(100%)	0.53	0.87	0.20
64(80%)	0.51	0.9747	0.1915
57(70%)	0.51	0.6194	0.4403
40(50%)	0.49	0.7911	0.25
32(40%)	0.46	0.75	0.20

Tableau 3: les résultats classification des déférentes catégories

Discussion des résultats :

Les résultats obtenus montrent clairement que, la diminution du nombre de catégories n'a aucun impact sur les performances du modèle d'apprentissage. Le meilleur résultat a été obtenu en tenant compte de l'ensemble des catégories. Toutefois, on constate que le modèle obtient un très faible résultat de $F1_{score}$ malgré la prise en compte de 100% de catégories.

Ajouté à cela, le temps de calcul a drastiquement augmenté. Concernant cette latente de calcul, cela peut s'expliquer du fait que, les LSTM ont généralement besoin d'une bande passante (mémoire) élevée en raison de la pile de couches LSTM superposé, ce qui induit probablement cette latence. Ainsi, au niveau matériel, les LSTM deviennent assez inefficaces à force que le nombre de données augmente.

Aussi, d'un point de vue performance, une forte accumulation de couches LSTM superposé peut éventuellement induire à une perte d'information de contexte au sein de la mémoire LSTM, au fur et à mesure que l'on s'éloigne du contexte initial.

Concernant, le faible résultat de $F1_{score}$, nous avons eue l'occasion de recenser lors de notre prospection de l'état de l'art, que la perte de performance serai probablement due au fait que, les LSTM sont de nature à être affectés par les différentes initialisations de poids opérés lors de la phase d'apprentissage. Ce qui les amène à se comporter de manière assez similaire à celle des réseaux de neurones à rétro-propagation (souffrant du vanishing gradient). Cela est probablement vrai, au vue du nombre important de catégories (labels) à prédire.

4.3.2.2. Résultat de la classification de polarité d'aspects

Après avoir effectué nos tests sur la première partie du code et choisit le nombre adéquat de catégories. Nous nous sommes intéressées à la deuxième partie du travail qui consiste à classer la polarité de chaque catégorie d'acpects.

L'image ci-dessus montre la structuration du modèle LSTM avec les différentes couches et paramètres correspondant.

```
model2 = Sequential()

embedding_layer = Embedding(vocab_size2, 100, weights=[embedding_matrix2],
                            input_length=maxlen , trainable=False)
model2.add(embedding_layer)

model2.add(LSTM(512,return_sequences=True))
model2.add(Dropout(0.25))

model2.add(LSTM(512,return_sequences=True))
model2.add(Dropout(0.25))
model2.add(Flatten())

model2.add(Dense(128, activation='relu'))
model2.add(Dropout(0.25))

model2.add(Dense(128, activation='relu'))
model2.add(Dropout(0.25))

model2.add(Dense(128, activation='relu'))
model2.add(Dropout(0.25))

model2.add(Dense(128, activation='relu'))
model2.add(Dropout(0.25))

model2.add(Dense(81, activation='sigmoid'))
print(model.summary())
model2.compile(loss='categorical_crossentropy', optimizer='adam')

model2.fit(review_train2, label_train2, batch_size=10,
           epochs=45, verbose=1, validation_split=0.2)
```

Figure 34: Modèle LSTM

Le tableau suivant montre les différents résultats obtenus. Il est à signaler que nous avons également varié le nombre d'époques, de manière similaire à la l'expérimentation 1 :

Nombre d'époque	F1_score	Précision	Rappelle
150	0.7600	0.7968	0.7146
100	0.75	0.79	0.71
45	0.7248	0.8727	0.5755
10	0.63	0.48	0.94

Tableau 4: tableau des résultats obtenus pour la classification des polarités en utilisant différentes époques

Le schéma suivant montre les résultats du $F1_{score}$, ainsi que l'augmentation de celui-ci en fonction de nombre d'époque utilisé :

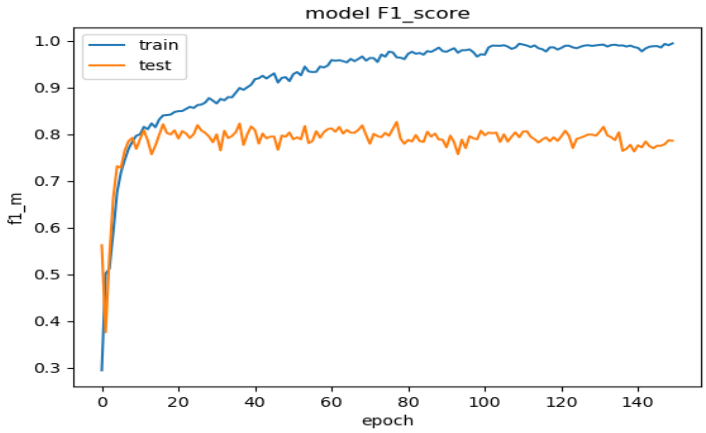


Figure 35: graphe d'augmentation de F1_score en fonction de nombre d'époque

De la même manière que précédemment nous avons obtenue des résultats à partir des tests en utilisant initialement un nombre d'époque de 150 puis en diminuant à chaque fois le nombre par 10 époque, nous remarquons que F1_score diverge, de telle sorte qu'a chaque fois qu'on augmente les nombres d'époque on obtient de meilleur résultat et vice versa.

4.3.3. Présentation d'une perspective éventuelle d'amélioration de l'approche

Nous avons eu l'occasion de constater dans la section précédente, que d'un point de vue performance, notre modèle LSTM avait certaines difficultés à performer notamment lors de la tâche d'extraction des catégories d'aspects. Nous avions supposés qu'une forte accumulation

de couches LSTM superposés aurait pu induire à une perte d'information de contexte au sein de la mémoire LSTM et ce, au fur et à mesure que l'on s'éloigne du contexte initial.

Pour essayer de répondre à ce problème, nous avons exploré les solutions et mécanismes qui pourraient améliorer cela. C'est ce qui nous a amené à s'intéresser au mécanisme d'attention.

Ce dernier a fait sa première apparition dans le domaine de la traduction de langages et plus généralement dans le domaine du traitement du langage naturel.

Pour rappel, dans les modèles utilisant l'architecture LSTM encodeur/décodeur, la séquence en entrée (par exemple, dans notre cas ça sera la phrase), est passée objet par objet dans l'encodeur LSTM. On obtient un état caché par objet présent dans la séquence (ou mot dans le cas d'une phrase). Le dernier état caché produit est censé, en théorie, grâce au mécanisme de mémorisation du LSTM, représenter le sens de l'ensemble de la phrase. En pratique, plus la phrase est longue, moins le dernier état caché est capable de représenter l'ensemble de la phrase et plus il représente la fin de celle-ci. On se rend bien compte que pour prédire la catégorie d'une phrase, il serait plus optimal de se concentrer en alternance sur différents mots de la phrase en entrée.

Chacun des états cachés obtenus représente le mieux le dernier objet (ou dernier mot) considéré par le LSTM. Le but initial d'un module d'attention est ainsi de choisir étant donné le contexte, quel objet de la séquence (quel mot de la phrase) en entrée utiliser pour effectuer la prochaine prédiction.

Dans le cadre de ce travail, nous avons essayé d'implémenter une version simple de l'attention dite « soft attention ». La bibliothèque Keras offre certaines facilitations pour l'intégration d'une telle couche aux modèles LSTM. Ainsi, nous avons défini une classe nommée Attention et qui hérite de la classe Layer en python. Nous y avons défini quatre fonctions selon la règle de génération de couche personnalisée Keras. Nous avons défini les poids et biais pour le calcul du contexte selon les règles de l'attention. Malheureusement, les résultats obtenus ne présentaient pas d'améliorations notables par rapport à l'approche simple sans l'ajout de l'attention (environs 51% de $F1_{score}$ pour la phase d'extraction des catégories d'aspects). C'est la raison pour laquelle nous avons fait le choix de ne pas présenter les résultats obtenus. Néanmoins, en perspective, cela nous motive à continuer l'expérimentation de plusieurs autres variantes d'attention et étudier leur impact sur notre modèle d'analyse d'opinions.

4.4 Interface web

Pour mettre notre modèle à la disposition des utilisateurs, nous avons développé une interface web simple permettant d'évaluer des phrases (séparément). Ci-dessous, un aperçu de l'interface principale.

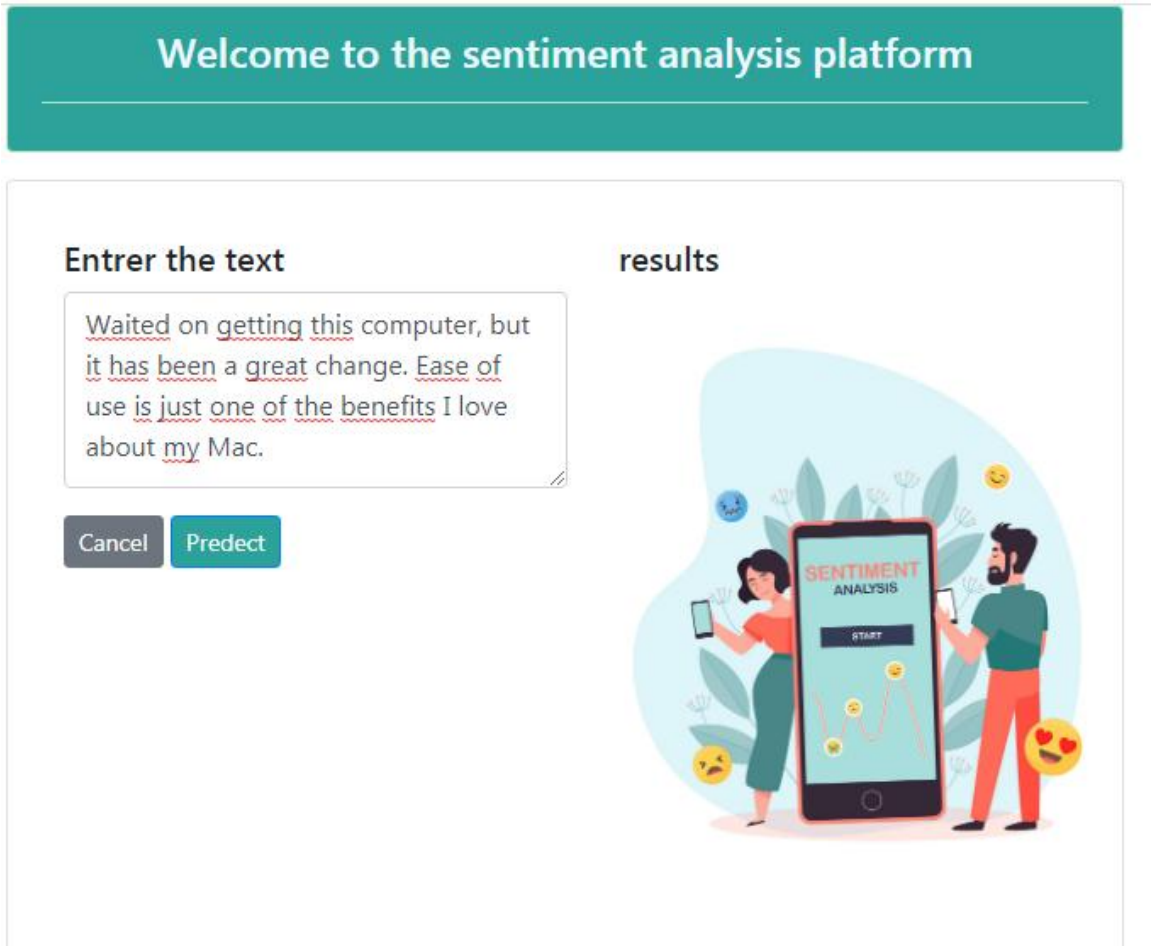


Figure 36: page index

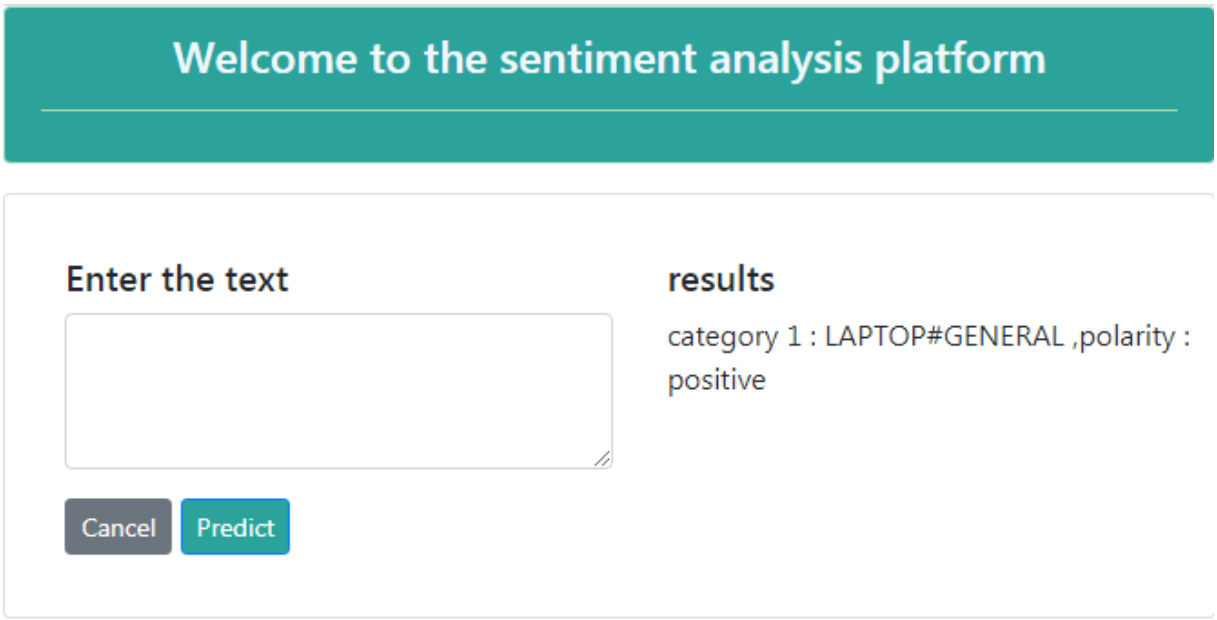


Figure 37: page résultat

4.5. Conclusion

Dans ce chapitre nous avons présenté et discuté les résultats obtenus dans expérimentations. On constate à travers ces résultats que le processus d'extractions de catégories d'aspects présente une certaine faiblesse à discriminer correctement les catégories. Ceci ce reflète à travers le taux de $F1_{score}$ obtenue. Pour la phase de classification de polarité, les résultats obtenus sont relativement acceptable et se situe dans les normes obtenus durant la campagne SemEval 2016. Nous avons également présenté au début du chapitre, l'ensemble des frameworks python utilisés.

Conclusion générale

Le travail présenté dans ce mémoire s'insère dans le domaine du traitement automatique du langage naturel et plus particulièrement dans le cadre de l'analyse d'opinions à base d'aspects. Cette dernière s'intéresse à l'étude des opinions en lien avec les caractéristiques (aspects) d'une cible précise au sein du texte.

Pour mieux comprendre le travail présenté dans ce mémoire, nous avons d'abord introduit, de manière non exhaustive, le domaine de l'analyse d'opinions. Puis, une introduction aux notions de l'apprentissage profond a été exposée au deuxième.

Enfin, dans les dernières parties, nous avons présenté notre travail qui consistait à mettre en œuvre une solution afin de détecter la polarité d'opinion des aspects (caractéristiques) de produits informatiques de type ordinateurs portables. Pour ce faire, nous nous sommes basés sur un algorithme d'apprentissage profond (LSTM) permettant d'abord d'extraire les catégories d'aspects présentent au sein du texte. Puis, sur la base de ces dernières, une classification de polarité a été réalisée.

La technique du word embedding a également été utilisée, afin d'améliorer les performances de notre solution. A ce propos, nous avons au préalable réalisé une expérience pour évaluer l'apport du word embedding sur la phase d'apprentissage. Les résultats obtenus ont clairement affirmé l'avantage de l'utilisation du word embedding.

Concernant les résultats de l'analyse d'opinions à base d'aspects, nous avons constaté que ceux de la phase de classification de polarité étaient relativement acceptables. Cependant, les résultats obtenus pour l'extraction des catégories d'aspects ont été moins concluants. Ce qui nous amenait à s'intéresser au mécanisme d'attention afin d'apporter plus détail lors de l'apprentissage des LSTM.

Ainsi, en guise de perspectives, nous nous intéresserons en détails au mécanisme d'attention et ses différentes variantes afin de permettre aux réseaux de neurones utilisés, de se focaliser sur des parties spécifiques du texte et à apprendre sur quoi "se concentrer" lorsqu'il fait des prédictions.

Liste des abréviations :

ABSA	Aspect-Based Sentiment Analysis
SVM	Support Vector Machine
CNN	Convolution Neural Networks
AI	Artificial Intelligence
RNN	Recurrent Neural Networks
ANN	Artificial Neural Networks
NLP	Natural Language Processing
TALN	Traitement Automatique du Langage Naturel
TAL	Traitement Automatique des Langages
TNL	Traitement de Langage Naturel

Annexe

Corpus : ensemble limité d'éléments (énoncés) sur lesquels se base l'étude d'un phénomène linguistique, ensemble de textes réunis qui serve de base à une étude quantitative.

n-gramme : Succession de N éléments des mêmes types extraits d'un texte, d'une séquence ou d'un signal, les éléments pouvant notamment être des mots ou des lettres. Les N grammes sont beaucoup utilisés en traitement automatique du langage naturel mais aussi en traitement du signal.

K-means : Les K-means est une technique de classification par apprentissage automatique non supervisé, utilisée pour simplifier des ensembles de données volumineux en ensembles de données simples et plus petits.

Naïve Bayes : Un classificateur Naif Bayes est un modèle d'apprentissage automatique probabiliste utilisé pour la tâche de classification. Le nœud du classificateur est basé sur le théorème de [Bayes, 1763], avec une forte indépendance (dite naïve) des hypothèses.

SVM : Les machines à vecteurs de support (Support Vector Machine, SVM) appelés aussi séparateurs à vaste marge sont des techniques d'apprentissage supervisées destinées à résoudre des problèmes de classification et de régression. La tâche principale de l'algorithme est de trouver la ligne la plus correcte, ou Hyperplane, qui divise les données en deux classes.
L'entropie maximale : Il consiste à représenter une connaissance imparfaite d'un phénomène par une loi de probabilité, en identifiant les contraintes auxquelles cette distribution doit répondre dans un premier lieu, puis choisir de toutes les distributions répondant à ces contraintes celle ayant la plus grande entropie au sens de Shannon.
Crawling : signifie Le parcourt, en permanence, de façon autonome et automatique, les différents sites et pages Internet à la recherche de contenus nouveaux ou d'éventuelles mises à jour de contenus déjà explorés par le passé.

Web scraping : est une technique permettant l'extraction des données d'un site via un programme, un logiciel automatique ou un autre site. L'objectif est donc d'extraire le contenu d'une page d'un site de façon structurée. Le scraping permet ainsi de pouvoir réutiliser ces données.

Bibliographie

[1] : Al-Azani, Sadam, et El-Sayed M. El-Alfy. 2017. « Using Word Embedding and Ensemble Learning for Highly Imbalanced Data Sentiment Analysis in Short Arabic Text. » In *ANT/SEIT*, 359‑66.

[2]: Chaturvedi, Iti, Ranjan Satapathy, Sandro Cavallari, et Erik Cambria. 2019. « Fuzzy Commonsense Reasoning for Multimodal Sentiment Analysis ». *Pattern Recognition Letters* 125: 264‑70.

[3]: Farhadi, Farnoush. 2017. « Learning activation functions in deep neural networks ». École Polytechnique de Montréal.

[4]: Goodfellow, Ian, Yoshua Bengio, Aaron Courville, et Yoshua Bengio. 2016. 1 *Deep learning*. MIT press Cambridge.

[5] : James, L., et David E. Rumelhart. 1986. « PARALLEL DISTRIBUTED PROCESSING, EXPLORATIONS IN THE MICROSTRUCTURE OF COGNITION: VOL 2, PSYCHOLOGICAL AND BIOLOGICAL MODELS ».

[6]: Kalchbrenner, Nal, Edward Grefenstette, et Phil Blunsom. 2014. « A convolutional neural network for modelling sentences ». *arXiv preprint arXiv:1404.2188*.

[7] : Kandasamy, Ilanthenral, W. B. Vasantha, Jagan M. Obbineni, et F. Smarandache. 2020. « Sentiment Analysis of Tweets Using Refined Neutrosophic Sets ». *Computers in Industry* 115: 103180.

[8]: Kim, Yoon. 2014. « Convolutional neural networks for sentence classification ». *arXiv preprint arXiv:1408.5882*.

[9]: Kulkarni, Akshay, et Adarsha Shivananda. 2019. « Deep Learning for NLP ». In *Natural Language Processing Recipes*, Springer, 185‑227.

[10]: Liu, Bing. 2020. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge university press.

[11]: Maynard, Diana, et Adam Funk. 2011. « Automatic detection of political opinions in tweets ». In *Extended Semantic Web Conference*, Springer, 88‑99.

[12]: Mikolov, Tomas et al. 2013. « word2vec ». *URL https://code. google. com/p/word2vec* 22.

[13]: Mou, Lili et al. 2015. « Natural language inference by tree-based convolution and heuristic matching ». *arXiv preprint arXiv:1512.08422*.

[14]: Nielsen, Michael A. 2015. 2018 *Neural networks and deep learning*. Determination press San Francisco, CA.

[15]: Pennington, Jeffrey, Richard Socher, et Christopher D. Manning. 2014. « Glove: Global vectors for word representation ». In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, , 1532‑43.

[16] : Poirier, Damien, Cécile Bothorel, Emilie Guimier De Neef, et Marc Boullé. 2011. « Automating opinion analysis in film reviews: the case of statistic versus linguistic approach ». In *Affective Computing and Sentiment Analysis*, Springer, 125‑40.

[17]: Rezaeinia, Seyed Mahdi, Rouhollah Rahmani, Ali Ghodsi, et Hadi Veisi. 2019. « Sentiment analysis based on improved pre-trained word embeddings ». *Expert Systems with Applications* 117: 139‑47.

[18]: Sharma, Richa, Shweta Nigam, et Rekha Jain. 2014. « Opinion mining of movie reviews at document level ». *arXiv preprint arXiv:1408.3829*.

[19]: Strubell, Emma, Ananya Ganesh, et Andrew McCallum. 2019. « Energy and policy considerations for deep learning in NLP ». *arXiv preprint arXiv:1906.02243*.

[20] :Taboada, Maite et al. 2011. « Lexicon-based methods for sentiment analysis ». *Computational linguistics* 37(2): 267‑307.

[21]: Tang, Duyu et al. 2014. « Learning sentiment-specific word embedding for twitter sentiment classification ». In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, , 1555‑65.

[22] : Yousfi, Yasmine, et Yasmine Bellahoues. 2019. *Sélection de caractéristiques pour la classification de polarité*.

[23] : Zhao, Yan, Suyu Dong, et Leixiao Li. 2014. « Sentiment analysis on news comments based on supervised learning method ».

[24]: https://le-datascientist.fr/5-apprentissage-supervise

[25]: https://fr.quora.com/Quelle-est-la-diff%C3%A9rence-entre-la-r%C3%A9gression-logistique-et-la-classification-automatique

[26]: https://dataanalyticspost.com/Lexique/svm/

[27]: https://meritis.fr/ia/deep-learning/

[28]: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

[35]: https://blog.octo.com/les-reseaux-de-neurones-recurrents-des-rnn-simples-aux-lstm/

[37]: https://datasciencetoday.net/index.php/fr/machine-learning/148-reseaux-neuronaux-recurrents-et-lstm

[38]: https://penseeartificielle.fr/comprendre-lstm-gru-fonctionnement-schema/

[39]: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[40] : Ikram Chraibi Kaadoud, Apprentissage de séquences et extraction de règles de réseaux récurrents : application au traçage de schémas techniques, L'université de Bordeaux

[41] : https://www.sciencedirect.com/science/article/abs/pii/S0950705116301721

[42] : https://arxiv.org/abs/1805.07043

[43] : https://www.aclweb.org/anthology/2020.acl-main.340/

[44] : https://link.springer.com/chapter/10.1007/978-3-030-29911-8_44

[45] : https://arxiv.org/abs/1909.11297

[46] : https://www.sciencedirect.com/science/article/abs/pii/S0957417420302177#!

[47] : Manon Vermeersch, L'ANALYSE DES SENTIMENTS AU NIVEAU DES ASPECTS POUR DES AVIS SUR DES HOTELS EN NEERLANDAIS ET FRANÇAIS, Universiteit Gent.

[48] : https://onlinelibrary.wiley.com/doi/pdf/10.1111/coin.12327

[49] : https://semeval.github.io/

[50] : http://alt.qcri.org/semeval2016/task5/

[51] : DIALLO Nene Adama Dian, La reconnaissance des expressions faciales, Université 8 Mai 1945 – Guelma –

[52] :https://www.lebigdata.fr/tensorflow-definition-tout savoir

[53] : https://www.ambient-it.net/formation/python-machine-learning/

[54] : Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, et Christian Jauvin. 2003. « A neural probabilistic language model ». Journal of machine learning research 3(Feb): 1137‑55.

[55]: Schwenk, Holger. 2007. « Continuous Space Language Models ». Computer Speech & Language 21(3): 492‑518

[56]: Collobert, Ronan, et Jason Weston. 2008. « A unified architecture for natural language processing: Deep neural networks with multitask learning ». In Proceedings of the 25th international conference on Machine learning,, 160‑67.

[57]: Nielsen, Michael A. 2015. 2018 Neural networks and deep learning. Determination press San Francisco, CA.

[58]: Cao, Kris, et Marek Rei. 2016. « A joint model for word embedding and word morphology ». arXiv preprint arXiv:1606.02601.

[59]: Pontiki, Maria et al. 2015. « Semeval-2015 task 12: Aspect based sentiment analysis ». In Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015), , 486‑95.

[60]: Hai Ha Dohaiha , PWC Prasad , Angelika Maag , Abeer Alsadoon, Deep Learning for Aspect-Based Sentiment Analysis: A Comparative Review,(2018).

[61]: Li, Yang et al. 2017. « Learning word representations for sentiment analysis ». Cognitive Computation 9(6): 843‑51.

[62]: Hao, Yanbin et al. 2019. « Cross-domain sentiment encoding through stochastic word embedding ». IEEE Transactions on Knowledge and Data Engineering.

[63]: Yu, Liang-Chih, Jin Wang, K. Robert Lai, et Xuejie Zhang. 2017. « Refining word embeddings using intensity scores for sentiment analysis ». IEEE/ACM Transactions on Audio, Speech, and Language Processing 26(3): 671‑81.

[64]: Le, Quoc, ET Tomas Mikolov. 2014. « Distributed representations of sentences and documents ». In International conference on machine learning,, 1188‑96.

[65]: A.I. technical - Machine Learning vs. Deep Learning 2019

[66]: MohammadAl-Smadi, OmarQawasmeh, MahmoudAl Ayyoub, YaserJararweh, BrijGuptac. « Deep Recurrent neural network vs. support vector machine for aspect-based sentiment analysis of Arabic hotels' reviews » 2017.

[67]: Xin Li and Wai Lam, Deep Multi-Task Learning for Aspect Term Extraction

with Memory Interaction , 2017.

[68]: Feiyang Rena, Liangming Feng, DingXiaoa Ming, CaiaSheng Cheng, DNet: A lightweight and efficient model for aspect based sentiment analysis, 2017.