UNIVERSITÉ DE SAAD DAHLEB – 1 –
FACULTÉ DES SCIENCES
DEPARTEMENT D'INFORMATIQUE
SPÉCIALITÉ : TRAITEMENT AUTOMATIQUE DE LA LANGUE

MEMOIRE DE MASTER

ANALYSE DES PUBLICATIONS DANS LES RESEAUX SOCIAUX PAR APPRENTISSAGE PROFOND

Réalisé par

Kheffi Zohra et Ouhaibia Manel Basma

Devant le jury composé de :

BOUMAHDI FATIMA	Université de Blida	*Président*
CHERIGUENE SORAYA	Université de Blida	*Examinateur*
Mme. OUKID LAMIA	Université de Blida	*Promoteur*

Soutenu le 03/10/2021, Blida

Remerciements

En premier lieu nous tenons à remercier Dieu le tout puissant qui nous a donné la force et la patience pour accomplir notre travail.

En second lieu nos remerciements vont tout droit à Mme L. Oukid pour tout ce qu'elle nous a appris, ainsi pour l'ambiance dans laquelle nous avons travaillé, pour sa discipline, sa présence, sa disponibilité, sa gentillesse et son soutien

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'il ont porté à notre mémoire en acceptant d'examiner notre travail.

Nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous apporté leur aide et qui ont contribué à l'élaboration de ce mémoire.

Nous tenons enfin à remercie du plus profond de notre cœur nos chers parents qui nous ont accompagnés et soutenus durant notre cursus.

Dédicaces

Je dédie ce projet :

A mes parents,

Qui n'ont jamais cesser de formuler des prières à mon égard, de me soutenir et de m'épauler pour que je puisse atteindre mes objectifs.

A mon frère et sa femme,

***A ma nièce Eline**,*

Pour leurs soutient moral et leurs conseils précieux tout au long de mes études.

***A mes chères et meilleures amies**,*

Nesrine, Khadîdja, Meriem et Sihem

Pour leurs aides et supports dans les moments difficiles.

***A ma chèrebinôme**,*

Basma

Pour son entente et sa sympathie.

A toute ma famille

A tous mes autres ami(e)s,

A tous ceux que j'aime et ceux qui m'aiment

Melle. Kheffi Zahra

Dédicace

Je dédie ce modeste travail :

A mes chers parents, pour leur soutien indéfectible, leurs encouragements, leurs sacrifices et leurs amours.

A ma sœur Farah et mon frère Abderrahmane, que je les souhaite à leur tour d'arriver là et rendre nos parents fiers.

A ma chère amie et sœur Meriem, pour ses encouragements et son amour.

A ma chère collègue Zahra pour son sérieux et sa gentillesse.

A ma famille et tous mes proches.

Melle. Ouhaibia Basma

RESUME

Aujourd'hui les réseaux sociaux représentent un moyen de communication indispensable, ils permettent la diffusion des différentes informations et nouvelles dans le monde dans différentes langues.

L'objectif de notre travail est l'analyse des publications sur les réseaux sociaux en utilisant des techniques de classification de données textuelle selon deux classes : statuts violents et statuts non violents et cela dans le but de détecter les statuts violents, de propos raciste et menaçants.

Afin d'atteindre notre objectif, nous proposons deux combinaisons entre les algorithmes d'apprentissage profond CNN-LSTM et CNN-BILSTM. Ainsi, Pour régler le problème de déséquilibrage du dataset, nous utilisons une technique de sur-échantillonnage « Oversampling » dite SMOTE. Une étude expérimentale est effectuée. Les résultats obtenus montrent l'intérêt des combinaisons proposées.

Mots clé : Apprentissage profond, Classification supervisée, Réseaux sociaux, Réseaux de neurones, BiLstm, Détection de langage agressive, Corpus déséquilibré, Oversampling.

ABSTRACT

Today social networks represent an essential means of communication, they allow the dissemination of different information and news in the world in different languages.

The objective of our work is the analysis of publications on social networks using techniques of classification of textual data according to two classes: violent statuses and nonviolent statuses and this with the aim of detecting violent statuses, racist comments and threatening.

In order to achieve our goal, we propose two combinations between the deep learning algorithms CNN-LSTM and Cnn-BiLstm. Thus, to solve the problem of data imbalance, we use an “Oversampling” technique called SMOTE. An experimental study is carried out. The results obtained show the value of the proposed combinations.

Keywords: Deep learning, Supervised classification, social networks, Neural networks, BiLstm, Aggressive language detection, Unbalanced corpus, Oversampling.

ملخص

اليوم، تمثل الشبكات الاجتماعية وسيلة اتصال أساسية، فهي تسمح بنشر المعلومات والأخبار المختلفة للعالم بلغات مختلفة.

الهدف من عملنا هو تحليل المنشورات على الشبكات الاجتماعية باستخدام تقنيات تصنيف البيانات النصية وفقًا لفئتين: حالات العنف وحالات اللاعنف وذلك بهدف الكشف عن حالات العنف والتعليقات العنصرية والتهديد.

ولبلوغ هدف بحثنا اقترحنا تركيبتين بين خوارزميات التعلم العميق CNN-LSTMوCnn-BiLstm.
وبالتالي، لحل مشكلة عدم توازن مجموعة البيانات استخدمنا أسلوب أخذ عينات زائدة "oversampling" يسمى SMOTE.
أجرينا دراسة تجريبية. والتي بينت نتائجها أهمية التركيبات المقترحة.

الكلمات المفتاحية: التعلم العميق، التصنيف الخاضع للإشراف، الشبكات الاجتماعية، الشبكات العصبية، BiLstm، الكشف عن اللغة العدوانية، الجسم غير المتوازن، الإفراط في أخذ العينات.

Tables de matières

Liste des tableaux

Liste des figures

Introduction générale

De nos jours l'Internet s'avère plus que jamais un outil indispensable d'échange d'informations. De plus en plus de personnes communiquent et échangent des opinons exprimés sur différentes plateformes, particulièrement les réseaux sociaux. En effet, l'utilisation quotidienne de ces derniers comme Facebook, Twitter ou Instagram ne cesse d'augmenter. Tous ces réseaux sociaux accumulent d'innombrable données, des messages et des images, qui sont enregistrés. Ce phénomène n'est pas lié à internet mais plutôt à des causes sociologiques est que la globalisation nous envahit, spécifiquement avec l'apparition des réseaux sociaux qui sont devenu un espace où les utilisateurs expriment leurs sentiments. Ce qui a poussé à une fouille à travers ces derniers.

Parmi les réseaux sociaux les plus célèbre on retrouve Twitter, un réseau social de micro blogage géré par l'entreprise Twitter Inc., crée en mars 2006 avec un slogan initial « *What are youdoing ?* » (Qu'est-ce que vous faites ?), ensuite remplacé par « *Compose new Tweet…* » (Composer un nouveau tweet). Ce service permet aux internautes de communiquer, de partager leurs expériences, leurs émotions de manière brève. Selon les statistiques faites par l'entreprise twitter, ce dernier compte 330 millions d'utilisateurs actifs chaque mois, chaque seconde environ 5 900 tweets sont expédiés, cela représente 504 millions de tweets par jour ou 184 milliards par an dans différentes langues. Avec cet espace important de données, twitter est devenu une cible de recherche attractive pour les entreprises et surtout les chercheurs en informatique dans le but d'étudier le comportement humain, l'analyse d'opinion, l'analyse des sentiments, l'identification des statuts violents, de propos racistes, menaçants...etc.

Une des problématiques dans le domaine de l'analyse des réseaux sociaux est la détection des publications des trolls. Un troll est un individu ayant un

comportement antisocial qui incite d'autres utilisateurs à agir au sein du même réseau social.[1] En particulier, un troll utilise souvent un langage agressif et a pour but de ralentir l'évolution normale d'une discussion en ligne et éventuellement de l'interrompre.[2]Les trolls influent de manière négatif sur les internautes et représentent éventuellement des personnes dangereuses voir menaçantes, d'où l'intérêt de la détection des publications de ces derniers.
La nécessité de traiter ce problème est donc apparue au fil du temps, ainsi nous nous intéressant à la tâche de détection de trolls en analysant le contenu textuel, en utilisant les outils fournis par l'analyse des sentiments.

Notre objectif à travers ce travail est de proposer une approche pour la détection des publications des trolls à travers l'analyse des statuts agressifs publiés sur les réseaux sociaux. Cela, à travers une classification automatique des publications selon deux classes : statuts agressif et statuts non agressif : en se basant principalement sur des algorithmes d'apprentissage profond. Ainsi, l'approche d'analyse proposée doit être expérimentée en se basant sur une collection de test réelle.

Nous proposons à travers ce mémoire deux solutions d'analyse des statuts violents, à travers deux combinaisons de deux algorithmes d'apprentissage profond puissants, à savoir : CNN-LSTM et CNN-BILSTM. Afin d'améliorer les résultats de ces derniers, nous associant une méthode de sur-échantillonnage (*Over Sampling*) appelé SMOTE. Cette dernière permet un rééquilibrage des classes d'apprentissage et donc d'éviter le phénomène de sur-apprentissage. Une évaluation à travers une collection de tests pour la détection des trolls est effectuée. Des études comparatives entre différentes combinaisons sont présentées. Les résultats obtenus montrent l'intérêt des combinaisons proposées.

[1]https://www.researchgate.net/publication/263568332_Trolling_in_asynchronous_computer-mediated_communication_From_user_discussions_to_academic_definitions
[2]https://aclanthology.org/K15-1032/

La suite du mémoire est organisée comme suit : Le premier chapitre est consacré pour les différentes définitions du domaine de notre étude.

Puis, un deuxième chapitre expose différents algorithmes de classifications et des techniques de prétraitement et extraction de fonctionnalité, ainsi que des techniques de sampling pour le rééquilibrage du corpus.

Par la suite, dans le troisième chapitre nous présenterons des travaux connexes à la problématique.

Par ailleurs un quatrième chapitre définira les analyses des réseaux sociaux par apprentissage profond.

Enfin, dans un cinquième chapitre nous présenterons une étude comparative entres les différents tests réalisés.

Chapitre 1 : L'analyse des sentiments et les trolls

1.1 Introduction

Il existe de nombreux types d'analyses de sentiments allant des systèmes qui se concentrent sur la classification des trolls (agressive, non agressive) aux systèmes qui détectent des émotions (en colère, heureux, triste, etc.) ou identifient des intentions (par exemple, intéressé, pas intéressé). Dans ce mémoire, nous intéressons pour la classification des trolls sur les réseaux sociaux.

1.2 Médias sociaux

1.2.1 Définition :

Les **médias sociaux**[3] sont des applications web qui permettent la création et la publication de contenus générés par l'utilisateur et le développement de réseaux sociaux en ligne en connectant les profils des utilisateurs.

Le terme recouvre les différentes activités qui intègrent la technologie, l'interaction sociale, et la création de contenu. Les médias sociaux utilisent l'intelligence collective dans un esprit de collaboration en ligne. Par le biais de ces moyens de communication sociale, des individus ou des groupes d'individus forment un réseau social, collaborent, créent ensemble du contenu Web, organisent le contenu, l'indexent, le modifient ou font des commentaires, le combinent avec des créations personnelles.

Les termes Web 2.0 et médias sociaux demeurent assez proches et concernent une grande variété de sites différents : les blogs, les wikis, et les réseaux sociaux numériques de tout type. Les médias sociaux utilisent beaucoup de techniques, telles que les flux RSS et autres flux de syndication Web.[4]

[3] https://fr.wikipedia.org/wiki/M%C3%A9dia_social
[4] https://www.vu-du-web.com/community-management/reseaux-sociaux/grands-reseaux-sociaux/.

1.2.2 Types des réseaux sociaux :

Les différents services des réseaux sociaux sont classés selon les six grands usages illustrés dans la figure suivante :

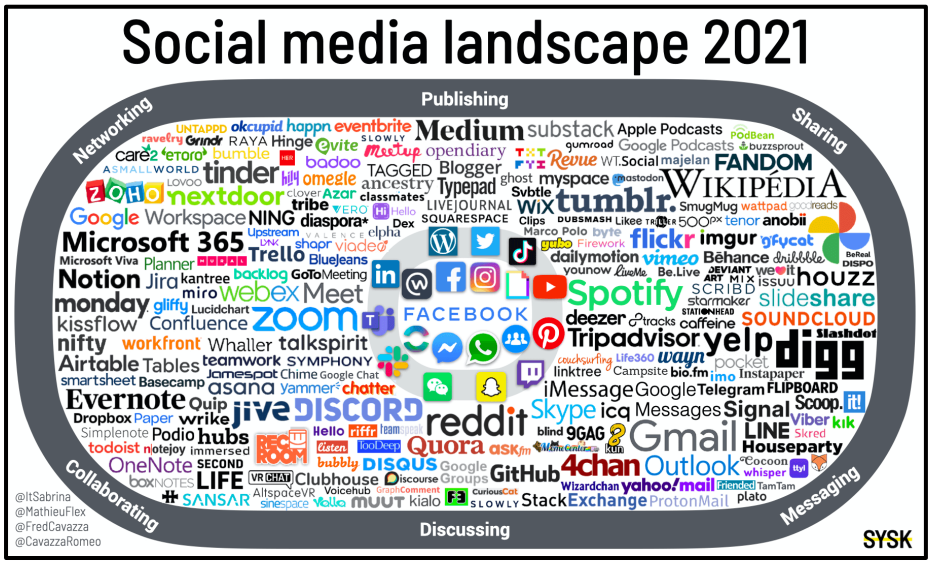


Figure 1: Panorama des média sociaux 2021

[Cavazza, 2021]

- Média sociaux principaux : Facebook, Twitter, Instagram, LinkedIn et YouTube
- Médias sociaux de partage : ils permettent de partager tous types de contenu, aux publics ou à son réseau (photo, vidéo, musique…)
    - Médias sociaux de réseautage : ils permettent la création et le développement d'un réseau, par exemple LinkedIn permet à créer un réseau professionnel
- Médias sociaux de publication : ils permettent à publier du contenue original, des articles, des rapport… il s'agit des plateformes de blogging (WordPress, Blogspot…)
- Médias sociaux de collaboration : ils permettent de collaborer à distance, ils sont très utilisés en gestion de projet.

1.2.3 Réseaux sociaux les plus connu :

De ce qui suit nous allons présenter un exemple d'un des réseaux sociaux publics : [5]

- **Twitter :** Le réseau social Twitter est né le 21 mars 2006 à San Francisco. Jack Dorsey, Evan Williams, Biz Stone et Noah Glass ont souhaité créer une plateforme où les utilisateurs pourraient facilement partager des moments de vie avec leurs amis par l'intermédiaire de messages courts du type « SMS ». C'est pour cette raison que les tweets sont sous forme de messages courts allant jusqu'à 140 caractères. Aujourd'hui, Twitter compte près de 284 millions d'utilisateurs actifs, dont 2,3 millions en France. La majorité des utilisateurs actifs sont sur le mobile, ils représentent 80% des utilisateurs soit 227 millions.

1.2.4 Avantage et les inconvénients des médias sociaux :

La partie ci-dessus présente quelque avantages et inconvénients des réseaux sociaux présenté précédemment :

- **Avantage :**

- Un moyen gratuit et facile pour communiquer avec les autres.
- Permettent une fusion des technologies
- Partager ses pensées, photos, vidéos avec la famille et les amis.
- Ils permettent aux entreprises d'avoir un contact direct avec leurs clients.
- Offrent aux entreprises un grand espace de publicité.

- **Inconvénients :**

- Contenu non désiré comme la violence et les messages haineux.
- Cyberharcèlement
- Risque de vol d'identité et d'information personnel.
- Les faux profils.

[5] https://www.vu-du-web.com/community-management/reseaux-sociaux/grands-reseaux-sociaux/.

Pour lutter contre ses inconvénients, l'un des domaines émergents des technologies de l'information est l'analyse et la détection automatique des trolls via les réseaux sociaux.

1.3 Trolls

1.3.1 Définition :

Le troll[6] est un individu bête et méchant, qui aime générer des polémiques quelques soit le sujet de la conversation. Il souhaite nuire, notamment à travers la perturbation d'un espace d'expression tel que les forums ou les réseaux sociaux, en particulier Facebook et Twitter. Il peut aussi créer des dissensions au sein d'un groupe : « diviser pour mieux régner » est sa devise. Ce type d'internaute est très doué pour exacerber les divergences de points de vue et repérer les membres qui ne partagent pas la même opinion. Étant donné que l'exclusion le guette à chaque phrase, le troll dispose de plusieurs comptes afin d'agir simultanément sur plusieurs forums et réseaux sociaux. Il peut surgir à n'importe quel instant dans une conversation en ligne publique. [7]

1.3.2 Qu'est-ce que le troll sur les réseaux sociaux :

Le troll sur les réseaux sociaux en général, c'est dévier le sujet de discussion de son vrai sens, il cherche à provoquer et créer une réaction émotionnelle négative, d'harceler les gens, lorsque deux groupes sont en dispute un groupe commence se moquer des autres groupes de manière à blesser les sentiments de l'autre groupe. La plateforme Tweeter concentrait une large part d'harcèlement en ligne, cela dû au degré élevé d'anonymat de cette dernière.

[6] https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445206-troll-sur-internet-definition-et-conseils-pratiques-pour-bien-reagir/

[7] https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445206-troll-sur-internet-definition-et-conseils-pratiques-pour-bien-reagir/

1.3.3 L’impact psychologique du troll sur les médias sociaux :

- Le troll est un jeu mental, son principe est d’affecter l’esprit de la personne pour obtenir une certaine réaction d’eux.
- Les effets du troll sont très menaçants, cet impact peut être dangereux conduisant à une depression ou des pensées suicidaires.
- L’impact psychologique avec tout ce qui affecte nos esprits ou nos pensées.

1.4 Analyse de sentiments

1.4.1 Définition :

L’analyse des sentiments consiste essentiellement à classer le sentiment qui se cache derrière un écrit.

Selon [Cambria 2016], L'analyse des sentiments, qu’on appelle aussi la fouille d'opinions, est l'un des domaines de recherche les plus actifs depuis des années 2000 dans le traitement du langage naturel.

Suivant [Pang et Lee, 2008], l’analyse des sentiments d’un texte consiste en plusieurs étapes.

Au moment de concevoir un système autonome d’analyse automatique des sentiments, il importe que celui-ci soit capable de suivre ces différentes étapes de l’analyse de façon indépendante.

Tout d’abord, les textes subjectifs doivent être distingués des textes neutres ou objectifs.

Ensuite, le système détecte les paragraphes ou les phrases comportant des jugements évaluatifs.

Finalement, toutes les informations être présentées dans une analyse globale du sentiment exprimé dans le texte.

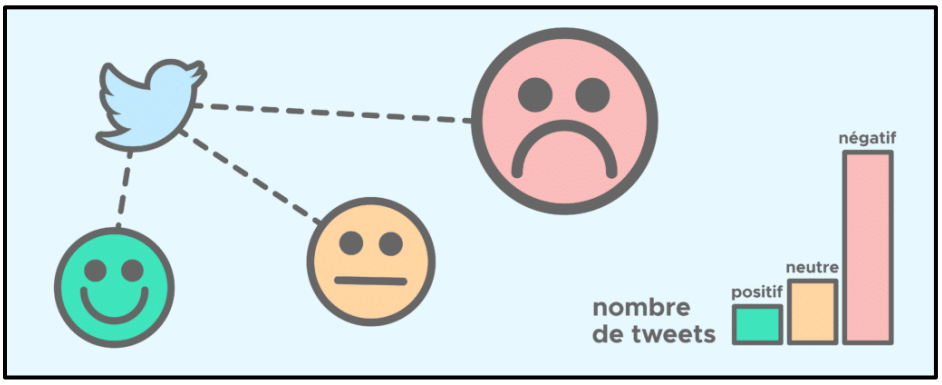


Figure 2: L'analyse de sentiment (positif, négatif, neutre) pour un tweet

[Gary, 2020]

1.4.2 Applications de l'analyse de sentiments :

(MukherjeeS,2012) classe globalement les applications dans les catégories suivantes :

- Applications aux sites Web liés à l'examen critiques de films, critiques de produits, etc.
- Applications en tant que technologie de sous-composant
- Détection de langage antagoniste et passionné dans les e-mails, détection de spam, détection d'informations contextuelles, etc.
- Applications en Business and Gouvernement Intelligence
- Connaître les attitudes et les tendances des consommateurs.
- Applications dans différents domaines
- Connaître les opinions publiques des dirigeants politiques ou leurs notions sur les règles et réglementations en place, etc.

1.4.3 Les défis d'analyse les sentiments :

Les approches d'analyse des sentiments visent à extraire des mots porteurs de sentiments positifs et négatifs à partir d'un texte et classer le texte comme positif, négatif ou bien neutre s'il ne trouve aucuns sentiments porteurs de mots. À cet égard, il peut être considéré comme une tâche de catégorisation de texte. Dans la

classification de texte, il existe de nombreuses classes correspondant à différents sujets alors que dans analyse des sentiments, nous n'avons que 3 grandes classes. Il semble donc que l'analyse des sentiments soit plus facile que la classification de texte, ce qui n'est pas tout à fait le cas.

Les défis généraux peuvent être résumés comme :

- **Sentiment implicite et sarcasme**

Une phrase peut avoir un sentiment implicite même sans la présence de mots porteurs de sentiments. Considérez les exemples suivants.

- Comment peut-on s'asseoir à travers ce film ?
- Il faut s'interroger sur la stabilité d'esprit de l'auteur qui a écrit ce livre.

Les deux phrases ci-dessus ne contiennent pas explicitement de mots porteurs de sentiments négatifs, bien que les deux soient des phrases négatives. Ainsi, l'identification de la sémantique est plus importante en SA que la détection de la syntaxe.

- **Dépendance de domaine**

Il existe de nombreux mots dont la polarité change d'un domaine à l'autre. Considérez les exemples suivants :

- L'histoire était imprévisible.
- La direction de la voiture est imprévisible.
- Allez lire le livre.

Dans le premier exemple, le sentiment véhiculé est positif alors que le sentiment véhiculé dans le second est négatif. Le troisième exemple a un sentiment positif dans le domaine du livre mais un sentiment négatif dans le domaine du film (où le réalisateur est invité à aller lire le livre).

- **Frustrer les attentes**

Parfois, l'auteur met délibérément en place le contexte pour le réfuter à la fin. Considérez l'exemple suivant :

- Ce film doit être génial. Cela ressemble à une excellente intrigue, les acteurs sont de première qualité et les acteurs de soutien sont également bons, et Salim essaie de livrer une bonne performance. Cependant, il ne peut pas tenir le coup.

Malgré la présence de mots d'orientation positive, le sentiment général est négatif en raison de la dernière phrase cruciale, alors que dans la classification de texte traditionnelle, cela aurait été classé comme positif car la fréquence des termes y est plus importante que la présence des termes.

- **Pragmatique** :

Il est important de détecter les pragmatiques de l'opinion des utilisateurs qui peuvent changer le sentiment en profondeur. Considérez les exemples suivants :

- Je viens de finir de regarder le Barca détruit Milan
- Cette finale m'a complètement détruit.

Les majuscules peuvent être utilisées avec subtilité pour désigner le sentiment. Le premier exemple dénote un sentiment positif tandis que le second dénote un sentiment négatif. Il existe de nombreuses autres manières d'exprimer le pragmatisme.

- **Détection de subjectivité :**

Il s'agit de faire la différence entre un texte opiniâtre et un texte sans opinion. Ceci est utilisé pour améliorer les performances du système en incluant un module de détection de subjectivité pour filtrer les faits objectifs. Mais c'est souvent difficile à faire. Considérez les exemples suivants :

- Je déteste les histoires comiques.
- Je n'aime pas le film « Je déteste les histoires ».

Le premier exemple présente un fait objectif tandis que le deuxième exemple décrit l'opinion sur un film particulier.

- **Identification de l'entité :**

Une phrase ou un texte peut avoir plusieurs entités. Il est extrêmement important de connaître l'entité vers laquelle l'opinion est dirigée. Considérez les exemples suivants :

- Samsung est meilleur que Nokia
- Rami a battu Sami au football.

Les exemples sont respectivement positifs pour Samsung et Rami mais négatifs pour Nokia et Sami.

- **Négation :**

La gestion de la négation est une tâche difficile en Afrique du Sud. La négation peut être exprimée de manière subtile même sans l'utilisation explicite d'un mot négatif. Une méthode souvent suivie dans le traitement de la négation explicitement dans des phrases comme « Je n'aime pas le film », consiste à inverser la polarité de tous les mots apparaissant après l'opérateur de négation (comme pas). Mais cela ne fonctionne pas pour "Je n'aime pas le jeu d'acteur mais j'aime la mise en scène".

Nous devons donc également considérer la portée de la négation, qui ne s'étend que jusqu'à mais ici. Ainsi, la chose qui peut être faite est de changer la polarité de tous les mots apparaissant après un mot de négation jusqu'à ce qu'un autre mot de négation apparaisse. Mais il peut toujours y avoir des problèmes. Par exemple, dans la phrase « Non seulement j'ai aimé le jeu d'acteur, mais aussi la direction », la polarité n'est pas inversée après « non » en raison de la présence de « seulement ». Ainsi, ce type de combinaisons de « pas » avec d'autres mots comme « seulement » doit être pris en compte lors de la conception de l'algorithme.

1.4.4 Caractéristiques pour l'analyse des sentiments :

L'ingénierie des fonctionnalités est une tâche extrêmement basique et essentielle pour l'analyse des sentiments. La conversion d'un morceau de texte en un vecteur de caractéristiques est l'étape de base de toute approche de SA basée sur les données. Dans la section suivante, nous verrons certaines fonctionnalités couramment utilisées dans l'analyse des sentiments et leurs critiques.

❖ **Présence à terme vs fréquence à terme**

La fréquence des termes a toujours été considérée comme essentielle dans les tâches traditionnelles de recherche d'informations et de classification de textes. Mais [Pang-Lee et al. ,2002] ont constaté que la présence du terme est plus importante pour l'analyse des sentiments que la fréquence du terme. C'est-à-dire des vecteurs de caractéristiques à valeur binaire dans lesquels les entrées indiquent simplement si un terme se produit (valeur 1) ou non (valeur 0). Ce n'est pas contre-intuitif car dans les nombreux exemples que nous avons vus auparavant, la présence même d'un seul sentiment de chaîne contenant des mots peut inverser la polarité de la phrase entière. Il a également été constaté que l'occurrence de mots rares contient plus d'informations que les mots fréquents, un phénomène appelé Hapax Legomena.

❖ **Caractéristiques N-gramme :**

Les N-grammes sont capables de capturer le contexte dans une certaine mesure et sont largement utilisés dans Natural Tâches de traitement du langage. L'utilité des n-grammes d'ordre supérieur est un sujet de débat. Pang et al. (2002) ont rapporté que les unigrammes surpassent les bigrammes lors de la classification des critiques de films par polarité des sentiments, mais Dave et al. (2003) ont constaté que dans certains contextes, les bigrammes et les trigrammes fonctionnent mieux.

❖ **Parties du discours**

Les parties des informations vocales sont le plus souvent exploitées dans toutes les tâches NLP. L'une des raisons les plus importantes est qu'ils fournissent une forme grossière de désambiguïsation du sens des mots.

1.4.5 Domaine d’application d’analyse des sentiments :

- **Politique :** Grâce à l’analyse des sentiments, les décideurs de politique peuvent prendre l’avis des citoyens sur certaines politiques, afin de

profiter de cette information pour créer une nouvelle politique qui convient avec les citoyens.

- **Prise de décision :** L'opinion et l'expérience des gens sont des éléments très utiles dans le processus de prise de décision.
- **Les systèmes de recommandations :** À travers l'analyse des sentiments on peut classer les opinions des gens positive ou négative, le système définit qui devrait prendre la recommandation et qui ne devrait pas la prendre.

1.5 Traitement automatique du Language naturelle :

1.5.1 Définition :

Le traitement automatique du langage naturel (abr. TALN)[8], ou traitement automatique de la langue naturelle, ou encore traitement automatique des langues (abr. TAL) est un domaine multidisciplinaire impliquant la linguistique, l'informatique et l'intelligence artificielle, Qui s'intéresse aux interactions entre les ordinateur et les langages humains. Parmi les défis qui ont été travaillés dans ce domaine on a la reconnaissance vocale, la compréhension du langage naturel et la détection de la langue.

1.5.2 Objectif :

Le but de la PNL, est « d'accomplir un traitement du langage de type humain ». Le choix du mot « traitement » est très délibéré et ne doit pas être remplacé par 'compréhension'. Car bien que le domaine de la PNL soit à l'origine appelé Naturel Language Understanding (NLU) dans les premiers jours de l'IA, il est bien admis aujourd'hui que si le but de la PNL est le vrai NLU, cet objectif n'a pas encore été atteint. Un NLU complet Le système serait capable de : [9]

- Paraphraser un texte d'entrée
- Traduire le texte dans une autre langue

[8] https://fr.wikipedia.org/wiki/Traitement_automatique_des_langues
[9] https://perso.limsi.fr/anne/coursM2R/intro.pdf

- Répondez aux questions sur le contenu du texte
- Tirez des inférences à partir du texte

1.5.3 Niveaux de traitement du langage naturel :

La méthode la plus explicative pour présenter ce qui se passe réellement dans un système de traitement de la langue utilise l'approche des « niveaux de langue ». C'est également appelé modèle synchronique du langage et se distingue du modèle antérieur modèle séquentiel, qui suppose que les niveaux de traitement du langage humain se succèdent de manière strictement séquentielle.
La description de certain niveau sera présentée comme suit :

- **Phonologie :** Ce niveau traite l'interprétation des sons de la parole à l'intérieur et à travers les mots.
- **Morphologie :** Ce niveau traite de la nature componentielle des mots, qui sont composés de morphèmes qui sont les plus petites unités de sens.
- **Lexical :** À ce niveau, les humains, ainsi que les systèmes de PNL, interprètent le sens des mots individuels.
- **Syntaxique :** Ce niveau se concentre sur l'analyse des mots dans une phrase afin de découvrir la grammaire structure de la phrase. Cela nécessite à la fois une grammaire et un analyseur.
- **Sémantique : ce niveau** concentre sur les interactions entre les significations au niveau des mots dans la phrase.

1.6 Conclusion

Dans ce chapitre nous avons parcouru différentes applications de l'analyse de sentiments, Nous avons de même, présenté l'analyse de sentiments sur les réseaux sociaux pour la détection des trolls.

Dans le chapitre suivant, nous parcourrons les différents algorithmes d'apprentissage et technique d'échantillonnage.

Chapitre 2 : Algorithmes d'apprentissage automatique et techniques d'échantillonnage

2.1 Introduction

Dans ce chapitre nous allons présenter en détails les algorithmes utilisés pour la représentation du texte et la classification supervisée des tweets, ainsi que des techniques adaptées pour le traitement de corpus de données déséquilibrées, et plus précisément les techniques de sur-échantillonnage (*Oversampling)*. Par la suite, des mesures d'évaluations des algorithmes de classification sont présentées.

2.2 Algorithmes de classifications

2.2.1 Représentation du texte :

La donnée brute ne peut pas être manipuler par les algorithmes de classification, c'est pourquoi il est important de trouver une représentation numérique de manière de représenter les mots comme des vecteurs. Chaque mot est représenté par un vecteur qui est appris via un algorithme itératif.

Il existe plusieurs modèles pour la représentation du corpus, comme le vecteur Tf-idf (*Term Frequency-Inverse Document Frequency),* le vecteur d'occurrence de mot (Count Vector) et le Word embedding.

2.2.1.1 TF-IDF :

La TF-IDF (de l'anglais *Term Frequency-Inverse Document Frequency)*[10]est une méthode de pondération la plus utilisée pour le traitement du langage naturel.
C'est une formule qui permet d'évaluer l'importance d'un terme dans le document, relativement à une collection ou un corpus. Il varie en fonction de la fréquence d'un mot dans le corpus.

- Fréquence du terme : le nombre d'occurrence du terme dans le document considéré

- Fréquence inverse de document : une mesure de l'importance du terme dans l'ensemble du corpus. Elle repose sur le calcul du logarithme de l'inverse de la proportion de documents du corpus qui contiennent le terme.

$$Tf\ (t,d)\ =\ \frac{ft,d}{\sum ft',d\ t' \in d}$$

$$Idf\ (t,D)\ =\ log\ \frac{N}{|\{d \in D|t \in d\}|}$$

$$Tf - Idf\ =\ tf\ (t,d)\ \times\ idf\ (t,D)$$

Source : [Salton & McGill, 1983].

- ***t*** : le terme présent dans le document.
- **D** : l'ensemble des documents.
- ***d*** : le document où t est présent.
- ***t'***: les termes présents dans le document.

2.2.1.2 Vecteur d'occurrence :

[10]https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

C’est un moyen d'extraire des caractéristiques du texte à utiliser dans la modélisation.

Une représentation bag-of-words ou sac de mot est la description de document (texte, image...) souvent utilisé dans la recherche d’information, dans cette méthode un document particulier est représenté par l'histogramme des occurrences des mots en comptant simplement le nombre de fois qu’un mot apparait dans le document.

2.2.1.3 Le Word embedding :

Le Word Embedding[11] ou le plongement lexical est une méthode de la représentation des mots et des documents considérés parmi les principales approches d’apprentissage en profondeur utilisé sur les problèmes complexe de traitement du langage naturel.

Il existe deux modèles, le premier vise à prédire le contexte en fonction des mots *Skip-gram*[12]et à l’inverse le deuxième vise à prédire les mots en fonction du contexte (*Continuous Bag Of Words*, CBOW).[13]

- **CBOW**

  Cette architecture vise à prédire un mot en fonction des mots qui l’entourent, c’est-à-dire le contexte, et ceci en fixant la taille d’une fenêtre des mots qui le précèdent et qui le suivent (les mots voisins).

  L’apprentissage des Word Embeddings est réalisé en calculant la somme des Word Embeddings du contexte, puis en appliquant sur le vecteur produit un classifieur log-linéaire pour prédire le mot cible. Enfin, le modèle compare sa prédiction avec la réalité et corrige la représentation vectorielle du mot par rétro propagation du gradient. Formellement, le modèle a pour objectif de maximiser la fonction suivante :

[11]https://dataanalyticspost.com/Lexique/word-embedding/
[12]https://towardsdatascience.com/nlp-101-word2vec-skip-gram-and-cbow-93512ee24314
[13]https://towardsdatascience.com/nlp-101-word2vec-skip-gram-and-cbow-93512ee24314

$$S = \frac{1}{\mathrm{I}} \sum_{i=1}^{I} \log p(m_i | m_{i-n}, \ldots\ m_{i+1}, \ldots\ m_{i+n}$$

Source : [Chaubard, Mundra, &Socher, 2016].

- I : nombre de mots dans le corpus
- n : taille de la fenêtre des mots.
- P : fonction Softmax

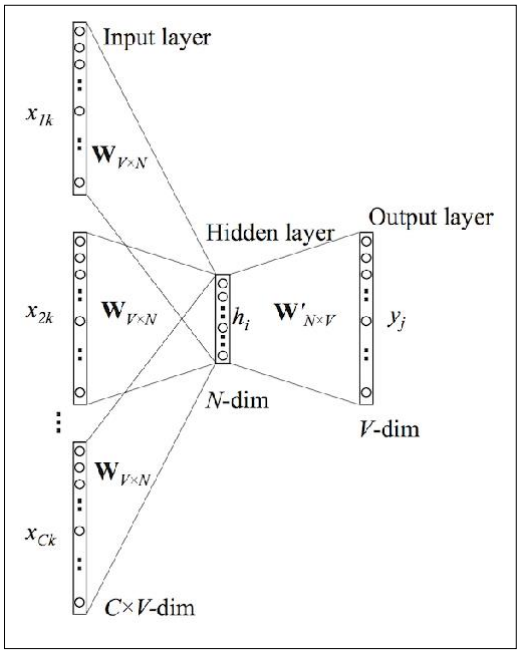


Figure 3: Model Continuous Bag of words

Source : [Meyer,2016]

❖ **Skip-gram**

L'architecture skip-gram est l'inverse de CBOW. Le mot cible$m_i$est utilisé comme entrée, et les mots voisins (le contexte) sont utilisés en sortie. De cette manière, cette méthode vise à prédire, pour un mot donné, le contexte

dont il est issu. La couche cachée va donc prendre le mot cible pour produire un vecteur dans la couche de sortie. Par la suite, ce vecteur est comparé à chaque mot voisin du mot cible, ainsi le model corrige la représentation par rétropropagation du gradient, afin que la représentation vectorielle du mot cible soit proche de celles des mots du même contexte. Formellement, le modèle a pour objectif, de maximiser la fonction suivante :

$$- \quad S = \frac{1}{I}\sum_{i=1}^{I}\sum_{-n\leq j\leq n,\neq 0} \log\, p(m_i|m_{i+j}$$

Source : [Chaubard, Mundra, &Socher, 2016].

- I : nombre de mots dans le corpus
- n : taille de la fenêtre des mots.
- p: fonction Softmax

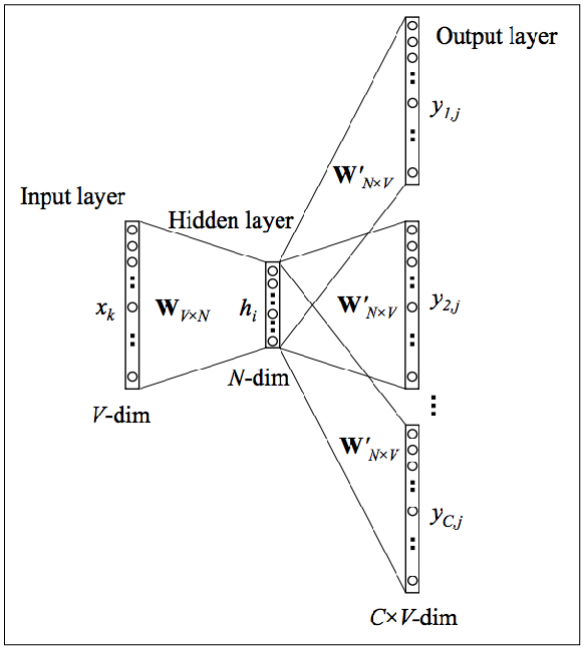


Figure 4: Model Skip-gram

Source: [Meyer,2016]

2.2.1.3.1 Model de langage neuronal

Dans Chollet, 2018], l'auteur montre qu'il existe deux façons d'obtenir des Word embedding, la première c'est avec des modèles pré-entrainées comme avec word2vec ou Glove, et la deuxième c'est d'apprendre les embedding en parallèle avec la tâche principale du réseau de neurones (comme la classification des documents ou la prédiction du sentiment), dans ce cas les vecteurs des mots sont initialisés aléatoirement, et sont appris au fur et à mesure de l'entrainement du réseau de neurones

Cependant il est a noté que la nature des embedding dans ce cas diffère du cas précèdent, vu que ce model ne constitue qu'une couche du réseau de neurone complet, les embedding vont apprendre des caractéristiques plus générales selon la tâche du réseau de neurone.

La couche embedding peut être considérer comme un dictionnaire, qui associe chaque indice (pour chaque mot spécifique dans le corpus) a un vecteur dense de nombre réel, qui prend des entiers en entrée, et retourne les vecteurs associés.

Index du mot ⟶ couche Embedding ⟶ vecteur correspondant.

2.2.2 Apprentissage supervisé :

L'apprentissages supervisées est une tâche qui consiste à apprendre une fonction qui permettra de regrouper des objets en se basant sur un ensemble d'apprentissage étiqueté de paires (entré-sortie).

Il est généralement effectué dans le contexte de la classification et de la régression.

- **Classification :** Un problème de classification survient lorsque la variable de sortie est une catégorie, telle que « rouge », « bleu » ou « maladie » et « pas de maladie ». *Exemples* : Dans le domaine du marketing utilisé pour l'analyse du sentiment de texte (heureux, pas heureux).

❖ **Régression** : Un problème de régression se pose lorsque la variable de sortie est une valeur réelle, telle que « dollars » ou « poids ». *Exemples :* Prédire le prix de l'immobilier.

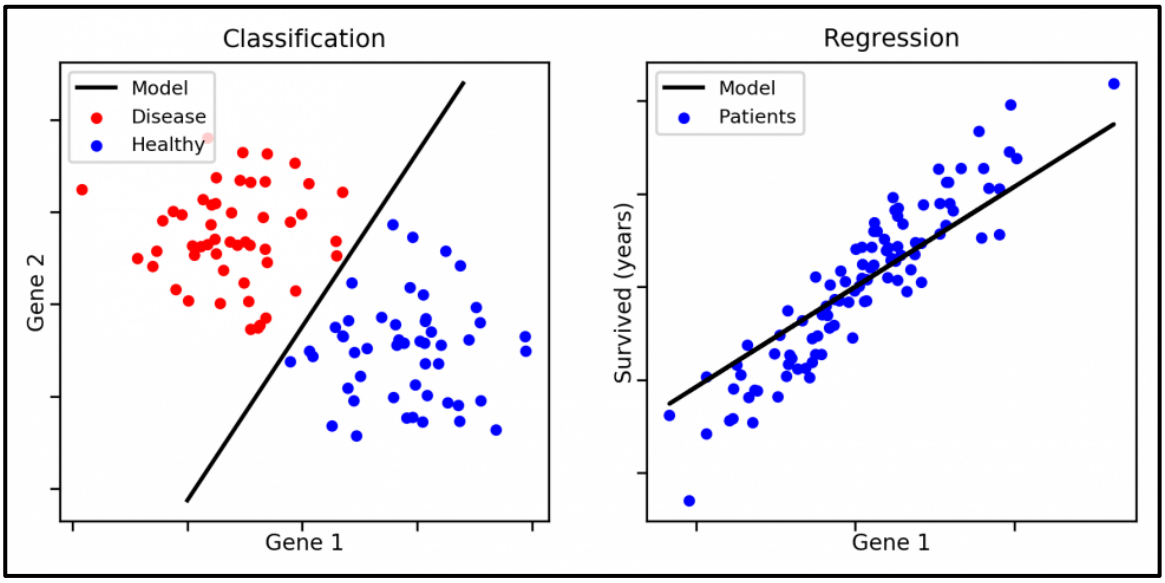


Figure 5: La différence entre la classification et la régression

[Petercour, 2019]

Voici quelques exemples populaires d'algorithmes d'apprentissage automatique Supervisé :

- Méthode des k plus proches voisins.
- Les réseaux de neurones.
- Arbre de décision.
- Classification naïve bayésienne
- Régression linéaire

Dans cette étude on s'intéresse aux réseaux de neurones artificiels ainsi qu'à la méthode des k plus proches voisins

2.2.2.1 Réseau de neurones :

Les réseaux de neurones, communément appelés des réseaux de neurones artificiels sont des **imitations simples des fonctions d'un neurone dans le cerveau humain** pour résoudre des problématiques d'apprentissage de la machine.

Le neurone est une unité qui est exprimée généralement par une fonction sigmoïde.

$$f(x) = \frac{1}{1 + e^{-x}}$$

Les domaines d'application des réseaux neuronaux sont souvent caractérisés par une relation entrée-sortie de la donnée d'information :

- La reconnaissance d'image
- Les classifications de textes ou d'images
- Prédiction de données

❖ **Perceptron :**

Le **Perceptron**[14]est un algorithme d'apprentissage automatique linéaire pour les tâches de classification binaire.
Il peut être considéré comme l'un des premiers et l'un des types les plus simples de réseaux de neurones artificiels. Ce n'est certainement pas un apprentissage « en profondeur », mais c'est un élément important.

❖ **Principe perceptron**

La Figure montre un perceptron a trois input (x1, x2, x3) avec des poids « *weights* » (w1, w2, w3), des nombres réels qui exprime l'importance de l'input par rapport à l'output, l'output y étant 0 ou 1 est déterminé par l'application de la fonction de 2.10 au potentiel post-synaptique ou θ est un seuil(*bias*).

[14]https://datascientest.com/perceptron

$$z = \sum_{i=1}^{n} w_i \, . \, x_i$$

$$y = f(z) = \begin{cases} \mathbf{0}\ \boldsymbol{si}\ \boldsymbol{z} \leq \mathbf{0} \\ \mathbf{1}\ \boldsymbol{si}\ \boldsymbol{z} > 0 \end{cases}$$

Source : [Petitjean, 2006]

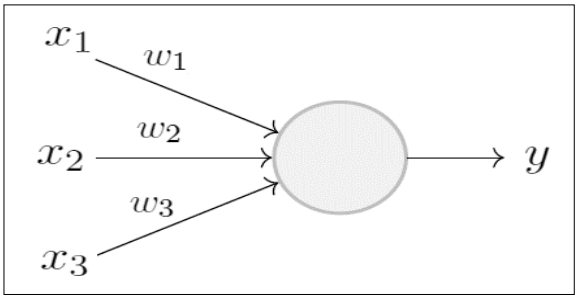


Figure 6: Schéma d'un perceptron

Source : [Minsky-Papert, *1969*]

- **Perceptron multicouche :**

Le Perceptron multicouche est un type de réseau neuronal artificiel organisé en plusieurs couches, dont chacune prend en entré ses entrées sur les sorties de la précédente.

Il s'agit donc d'un réseau à propagation directe.

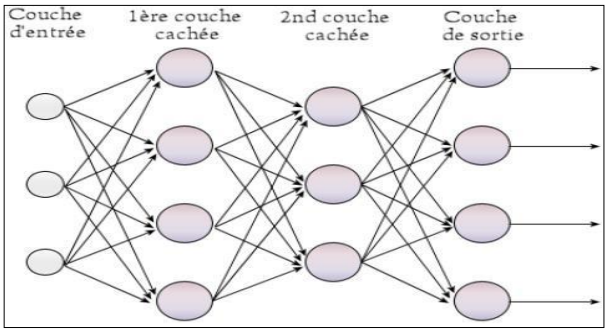


Figure 7: Perceptron multicouche.

Source : [HRcommons, 2009]

L'on peut voir les neurones du perceptron multicouche comme une multitude de perceptrons connectés entre eux. La particularité topologique de ce réseau est que tous les neurones d'une couche sont connectés à tous les neurones de la couche suivante. Chaque neurone a donc n entrées, *n* étant le nombre de neurones présent dans la couche précédente, et une sortie qui est envoyée à tous les neurones de la couche suivante.

À chaque connexion neuronale est associé un poids W, comme pour les entrées du perceptron. Le calcul de la sortie d'un neurone se fait selon la fonction d'activation qui opère une transformation sur la combinaison (c'est-à-dire le produit scalaire entre le vecteur des entrées et le vecteur des poids synaptiques) et renvoie la sortie.

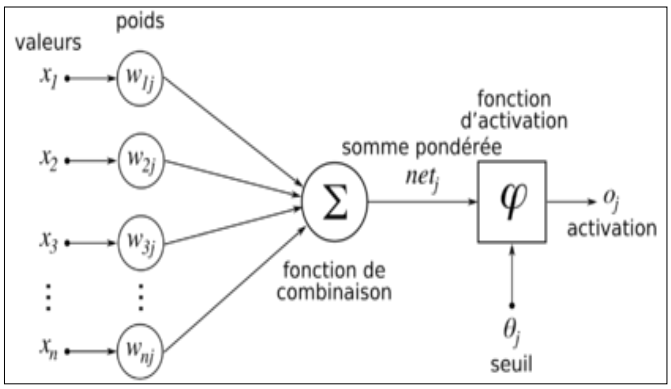


Figure 8: Structure d'un neurone artificiel

Source : [Chrislb, 2005]

2.2.2.1.1 Réseau neuronal récurrent (RNN) :

Les réseaux de neurones récurrents (en anglais RNN pour Recurrent Neural Network)[15]sont un type de réseau de neurones dans lequel les sorties des pas de temps précédents sont

[15]https://builtin.com/data-science/recurrent-neural-networks-and-lstm

alimentées en entrée du pas de temps actuel.

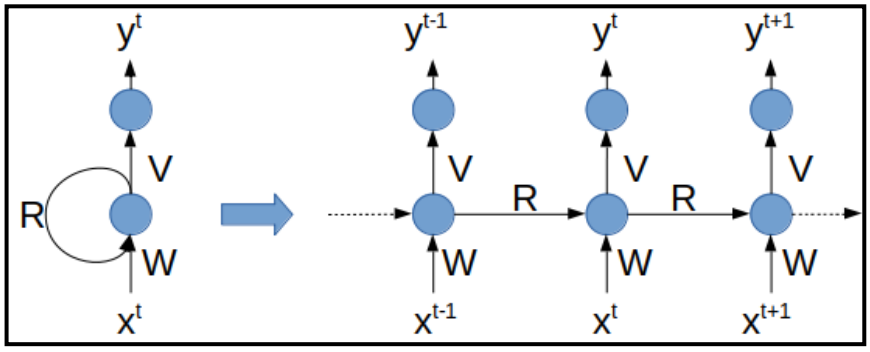


Figure 9: Schéma d'une unité récurrente

Source : [Deloche, 2017a]

- La sortie des étapes précédentes est donnée en entrée de l'étape en cours, où les informations contextuelles traversent le réseau dans son ensemble, affectant ainsi toutes les couches du réseau.
- A la différence des réseaux de neurones traditionnels qui utilisent des paramètres différents dans chaque couche, RNN partage les mêmes paramètres à toutes les étapes.
- Dans la figure 2.8, on a des sorties dans chaque étape, mais ceci ni pas nécessaire, par exemple, si on devait prédire le sentiment d'une phrase, on se souciera que de la sortie final (pas de la sortie qui concerne chaque mot).

Les réseaux de neurones récurrents peuvent modéliser la séquence de données afin que chaque échantillon puisse être supposé dépendant des précédents. Ils sont exposés au problème de disparition du gradient (la disparition du gradient rend la minimisation de la perte difficile) pratiquement, un RNN n'est pas capable de capturer toutes les informations précédentes.

En d'autres termes plus une séquence est longue plus le RNN devient vulnérable. Pour répondre à cette problématique, d'autres types de réseaux de

neurones récurrent sont introduits : Long short- Term Memory LSTM et Gated récurrent unit GRU.
Dans cette étude on s'intéresse à LSTM.

❖ Réseau récurrent à mémoire court et long terme (LSTM) :

Les réseaux de mémoire à long terme (LSTM)[16] sont une version modifiée des réseaux de neurones récurrents, ce qui facilite la mémorisation des données passées en mémoire. Le problème du gradient de fuite de RNN est résolu ici.

➢ Principe LSTM :

LSTM est une cellule composée de trois "portes" : ce sont des zones de calculs qui régulent le flux d'informations (en réalisant des actions spécifiques). On a également deux types de sorties (nommées états), grâce à cet état le réseau pourra transmettre des donnés pertinents et agisse comme une route de transport qui transfère les informations relatives.
La cellule mémoire peut être conduite par **trois portes** de contrôle :

- **Porte d'entrée (Input Gate) :**

  Le rôle de la porte d'entrée est d'extraire l'information de la donnée courante et met à jour l'état de la cellule. Sigmoïde est la fonction d'activation. Pour chaque entrée, la porte fournira en sortie une valeur entre 0 et 1 ensuite elle décide quelles nouvelles informations vont être autorisées à être stockées dans la cellule mémoire.
  Si le résultat est proche de 1, la valeur correspondante sera stockée (importante), au contraire si le résultat est proche de 0 l'information sera bloquée (n'est pas importante).
  A la fin ce résultat sera multiplié par l'état actuel.

[16]https://builtin.com/data-science/recurrent-neural-networks-and-lstm

$$f_t = \sigma\,(W_t[h_{t-1,}X_t] + b_t)C_{t=} f_t \times C_{t-1}$$

- **Porte d'oubli (Forget Gate) :**

  Le rôle de cette porte est de filtrer les informations contenues dans la cellule mémoire précédente d'où elle décide quelle information doit être conservé ou jeté, ensuite ces informations sont passé à la fonction Tanh qui va simplement normaliser les valeurs entre -1 et 1 pour éviter les problèmes de surcharge de l'ordinateurs lors du calcul. Enfin, après la sélection des caractéristiques importantes, une couche sigmoïde décide quelle valeur va être mis à jour.

$$f_{t=} \tanh(W_c[h_{t-1}] + b_c)\, I_{t=} \sigma(W_i[h_{t-1}, X_t] + b_i)$$

La multiplication entre ces deux résultats f_t et I_t et l'ajout à l'état nous permettra de savoir quelles valeurs vont devoir être stocké dans la cellule mémoire.
La nouvelle mémoire sera :

$$C_{t=} f_t + C_{t-1} + f_c \times I_t$$

- **Porte de sortie (Output Gate) :**

  La porte de sortie doit fournir une décision de quel sera le prochain état caché qui contient des informations sur les entrées précédentes du réseau.

  $$O_{t=}\sigma(W_o[h_{t-1}, X_t] + b_o)$$

  Donc, le nouvel état caché est :

  $$h_{t=} O_t \times tanh(C_t)$$

  Source : [Donadio, 2018].

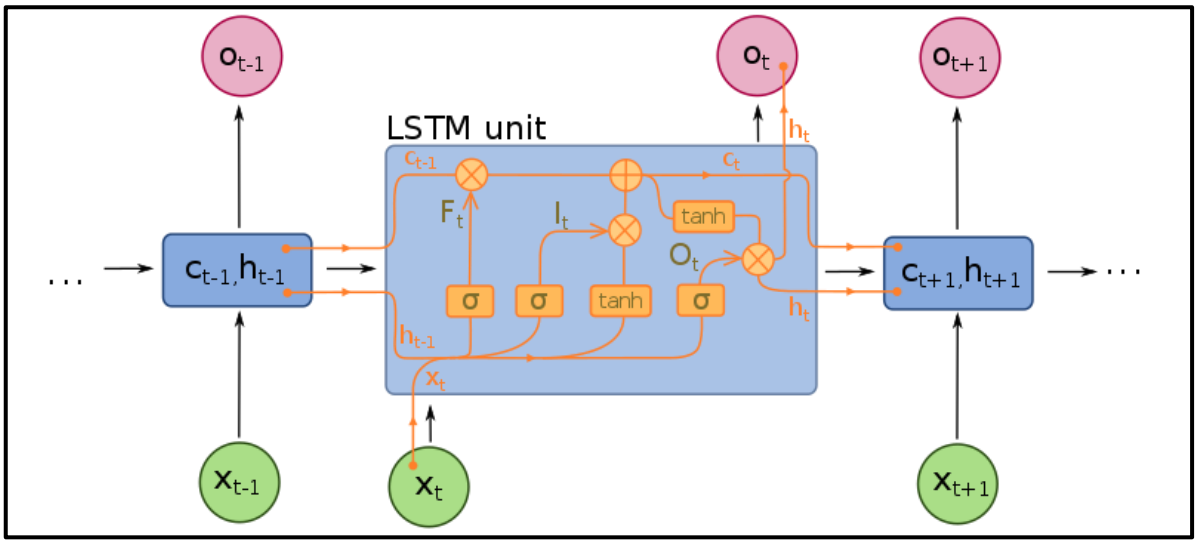


Figure 10: Schéma d'une unité LSTM

Source : [Deloche, 2017b]

❖ **Architecture réseau de neurone avec une couche LSTM**

La première couche c'est la couche *embedding*, suivi d'un dropout pour éviter l'overfitting ensuite une couche LSTM, et enfin, une dernière couche de sortie (dense layer).

Le dropout permet d'éviter le sur-apprentissage (Overfitting).

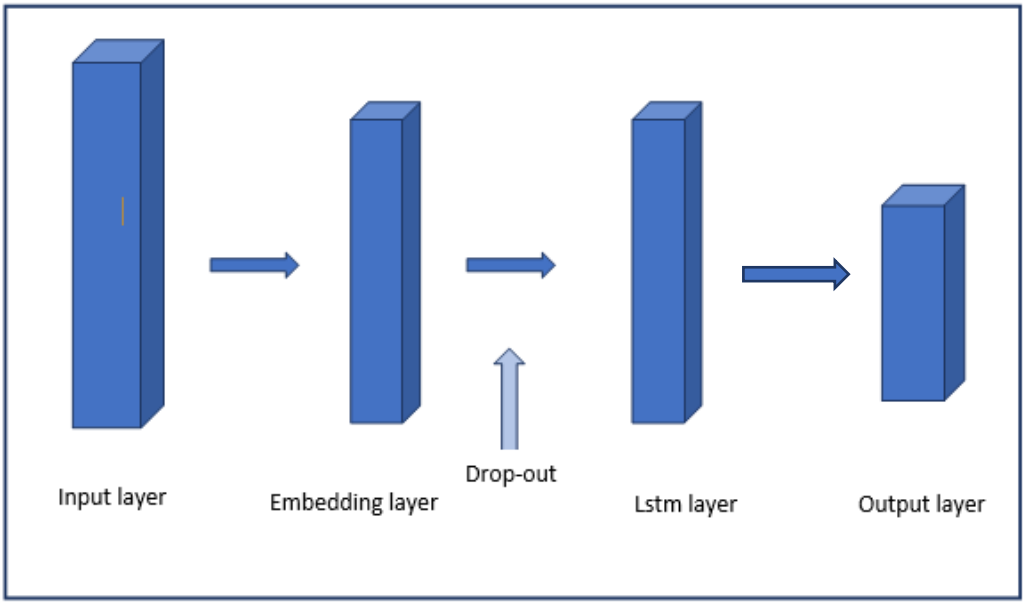


Figure 11: Architecture Réseau de neurones LSTM

Cette couche est utilisée pendant l'apprentissage. Elle permet de désactiver aléatoirement des neurones durant les différentes itérations de l'apprentissage. Chaque neurone étant possiblement inactif pendant une itération d'apprentissage, cela force chaque unité à « bien apprendre » indépendamment des autres et évite ainsi la « Coadaptation ».

Le sur-apprentissage (Overfitting), s'interprète comme un apprentissage « par cœur » des données, un genre de « mémorisation ». C'est le cas quand la perte d'entrainement est très inferieur a la perte de validation, par exemple, si le modèle réussit mieux sur le set d'entraînement que sur le set de test, il est probable que ce soit à cause d'un sur- apprentissage. D'un autre coté il y a aussi le sous-apprentissage (Underfitting) ou la perte d'entrainement est très supérieure à la perte de validation

2.2.2.1.2 Les réseaux de neurones à convolutions (CNN) :

Les réseaux de neurones à convolutions (CNN : Convolutional *Neural Networks*)[17]est une sous-catégorie de réseaux de neurones dont son fonctionnement est inspiré par le cortex visuel des animaux. Leur architecture est composée de deux blocs principaux, le premier bloc représente des cellules dites simple qui peuvent détecter des caractéristiques liées à la forme, l'autre bloc représente des cellules dites complexes qui sont plus sensible à la forme entière. Comparé à d'autre algorithmes de classification d'image, les réseaux de neurones convolutifs utilisent peu de prétraitement ce que signifie que son architecture est responsable de faire évoluer toute seule ses propres filtres. Un autre avantage de ces réseaux est l'utilisation d'un poids unique accordé aux signaux entrant dans tous les neurones d'un même noyau de convolution, cette technique réduit la complexité du modèle et améliore et performances au contraire aux réseaux de neurones multicouches (perceptron) où chaque neurone est indépendant d'où à chaque signal un poids différent est affecté.

- **Principe CNN**

  L'architecture d'un réseau de neurone convolutif est formé par un bloc de traitement qui se compose de quatre types de couches :

- **Couche de convolution (CONV) :** est une méthode qui utilise un filtrage par convolution qui scanne l'entrée suivant ses dimensions. La sortie de cette opération appelé carte d'activation ou feature map.

  Cette technique permet de trouver des parties de l'image qui pourraient être intéressantes.

- **Couche de pooling (POOL) :** c'est une forme de sous-échantillonnage précisément appliqué après la couche de convolution, cette couche est souvent placée entre deux couches convolutif dont le but de réduire le nombre de paramètres du réseau cela diminue le risque de sur-apprentissage et rapporte de gros gain en puissance de calcul.

[17]https://datascientest.com/convolutional-neural-network

Les types de pooling les plus courants sont les suivantes :

- **Le Max-Pooling :** sélectionne la valeur maximale de la surface et garde les caractéristiques détectées.

- **Le Mean-Pooling (Average-Pooling):** sélectionne la valeur moyenne de la surface prenant en compte toutes les caractéristiques et ne rejette aucune information

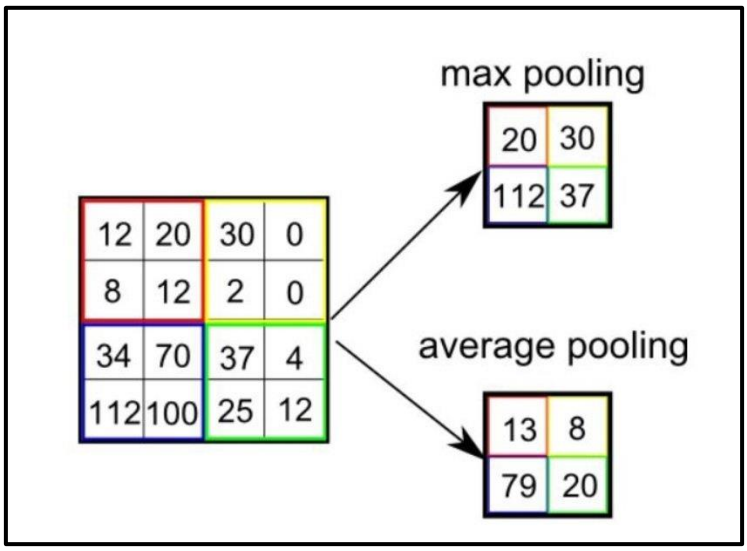


Figure 12: Schéma du fonctionnement du Max-Pooling et Mean-Pooling

Source : [Islam, 2020]

- **ReLU (Rictified Linear Units) :** désigne la fonction réelle non linéaire appelé aussi fonction d'activation non saturante, son rôle est d'augmenter les propriétés non linéaires sans prendre les champs récepteurs de la couche de convolution.
  La fonction ReLU est souvent préférable car elle en résulte la formation du réseau neuronal plus rapidement.

- **La fully-connected :** cette couche constitue toujours la dernière couche d'un réseau de neurone, elle est entièrement connectée car tous les neurones en entré de la couche sont connecté avec celle en sortie, de ce fait

ils ont accès à la totalité des informations d'entrés (comme dans les réseaux réguliers de neurones).

❖ **Fonctions d'activations :**

La fonction d'activation est une fonction mathématique qui permet de transformer le signal entrant dans un neurone artificiel en signal de sortie, elle est souvent une fonction non linéaire qui permet de modifier spatialement leur représentation.
Le choix de la fonction d'activation se fait selon leurs caractéristiques, s'il faut avoir des sorties binaires ou non.
Les fonctions utilisées dans ce travail sont : Sigmoïde, Softmax, TanH et ReLU.

- **Sigmoïde (Logistique) :** la fonction sigmoïde est définie par :

$$\boldsymbol{S(x) = \frac{1}{1 + e^{-x}}}$$

Elle est utilisée pour la classification binaire, elle retourne une probabilité de valeur 1 si l'entrée est un très grand nombre positif et 0 si l'entrée est un très grand nombre négatif.

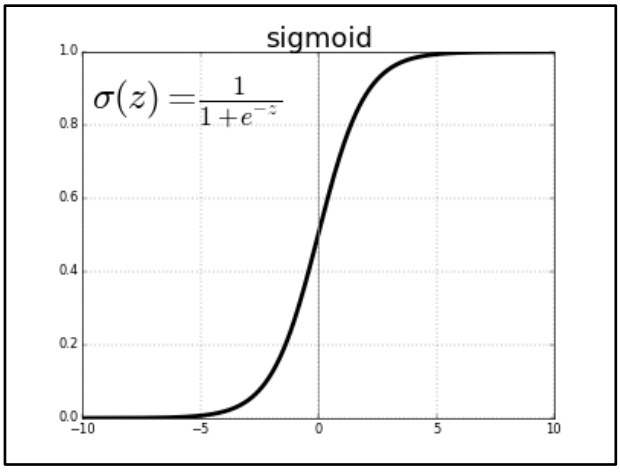


Figure 13: Graphe de la fonction Sigmoïde

[Haochen, 2020]

- **Softmax :**la fonction Softmax est définie par :

$$S(x_j) = \frac{e^{x_j}}{\sum_i e^{x_j}} \qquad \text{Avec j} \in (1, \ldots, K)$$

C'est une généralisation de la régression logistique qui permet de transformer un vecteur réel en vecteur de probabilité. Elle produit en sortie un vecteur de k nombre réel strictement positifs et de somme 1.

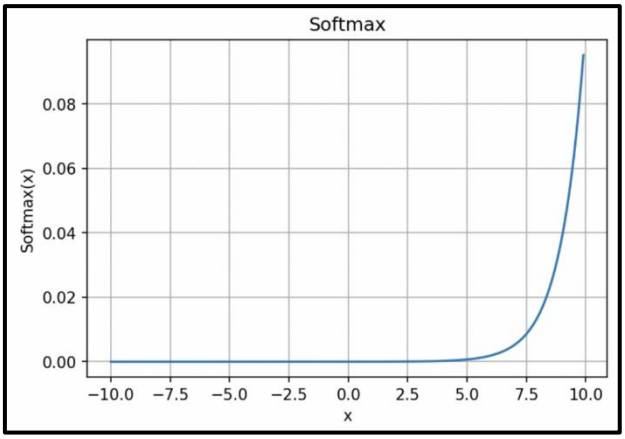


Figure 14: Graphe de la fonction Softmax

[Haochen, 2020]

- **Tanh (Tangente hyperbolique) :** la fonction Tanh est définie par :

$$T(x) = tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Elle produit un résultat compris entre -1 et 1, les grandes entrées négatives tendent vers -1 et les grandes entrées positives tendent vers 1.

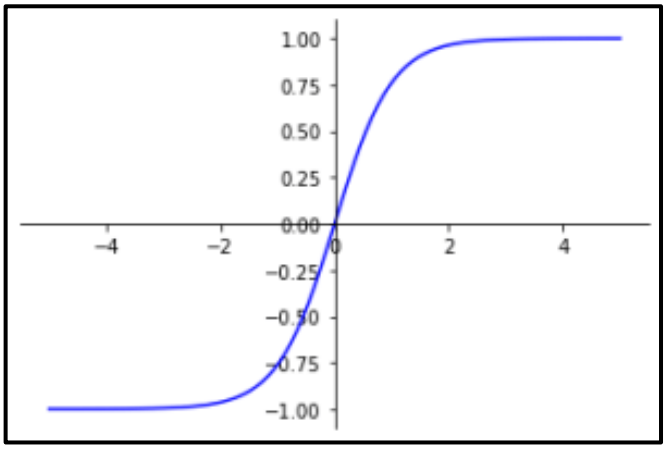


Figure 15: Graphe de la fonction Tanh

[Haochen, 2020]

- **ReLU (Rectified Linear Unit) :**la fonction Relu est définie par :

$$\boldsymbol{R(x)} = \begin{cases} \boldsymbol{0\ pour\ x} < 0 \\ \boldsymbol{x\ pour\ x} > 0 \end{cases}$$

C'est la fonction la plus simple et la plus utilisé, la sortie donne x si l'entrée est supérieure à 0, 0 sinon. Autrement dit la sortie est le maximum entre x et 0

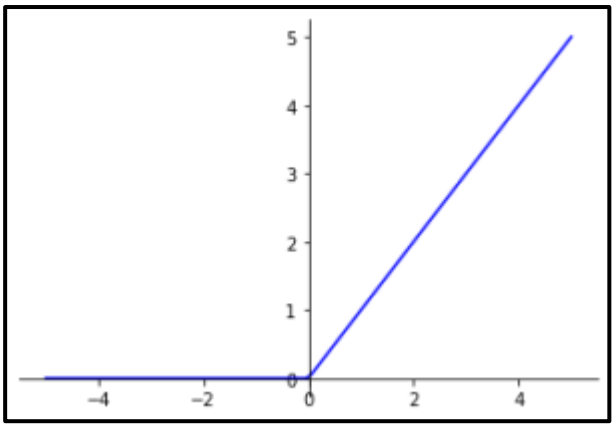


Figure 16: Graphe de la fonction ReLU

[Haochen, 2020]

❖ **Les Fonctions d'optimisation :**

Les fonctions d'optimisation dans les réseaux de neurones ont pour objectif de calculer le gradient dans le but de trouver des poids optimisés, et ceci en modifiant les valeurs dans la direction opposée, cette opération est répétée jusqu'à la minimisation de la fonction objectif.
Dans ce travail les fonctions Adam, Adamax sont testés.

➢ **Principe Adam :**

Adam, *Adaptive Moment optimization,* est un processus d'optimisation stochastique, parmi les plus efficaces pour l'optimisation par descente de gradient.
Pour chaque paramètre, Adam calcule la moyenne exponentielle du gradient ainsi que les carrés, ensuite on multiplie le taux d'apprentissage par la moyenne du gradient et en le divisant par la racine carrée de la moyenne exponentielle des gradients. Enfin la mise à jour est ajoutée.

$$v_t = \beta_1 \times v_{t-1} - (1 - \beta_1) \times g_t$$

$$s_t = \beta_2 \times s_{t-1} - (1 - \beta_2) \times g_t^2$$

$$\Delta w_t = -\eta \frac{v_t}{\sqrt{s_t + \epsilon}} \times g_t$$

$$w_{t+1} = w_t + \Delta w_t$$

Source : [Kingma & Ba, 2015]

- $-\eta$: Learning rate initial.
- g_t: le gradient dans un temps t.
- v_t: la moyenne exponentielle des gradients.
- s_t: la moyenne exponentielle des carrées des gradients.

- $\beta_1, \beta_2, \epsilon$: des hyper paramètres initialisés généralement a 0.90, 0.99 et 1e-10.

- **Principe Adamax :**

  Le processus Adamax est une extension de la version Adam dans le but d'obtenir une convergence plus stable.

$$s_t = \beta_2^p \times s_{t-1} - \left(1 - \beta_2^p\right) \times g_t^p \text{ Avec p} = \infty$$

Source : [Kingma & Ba, 2015]

- **Principe Adadelta :**

  Adadelta définie récursivement une moyenne décroissante de tous les gradients carrés passé, la moyenne mobile à chaque pas de temps dépend uniquement de la moyenne précédente et du gradient actuel. Elle utilise une taille fixe w pour la prise en compte de la somme des carrés de gradients accumulés dans le passé.

- **Principe SGD**

  Le mot « stochastique » désigne un système ou un processus lié à une probabilité aléatoire. Par conséquent, dans la descente de gradient stochastique quelques échantillons sont sélectionnés de manière aléatoire au lieu de l'ensemble des données de chaque itération
  SGD (*Stochastic Gradient Descent*), met à jour les paramètres pour chaque exemple de l'ensemble de données $x_i$et étiquette (*label*)y_i.

  - Initialiser avec x0 (au hasard)
  - Répéter :

$$xt + 1 = xt - \eta \times \nabla f(xt)$$

Jusqu'à convergence

- η : permet de moduler la correction (η trop faible, lenteur de convergence ; η trop élevé, oscillation)
- $\nabla f(x_t)$: Le gradient au point $x_t$montre la direction et l'importance de la pente au voisinage dex_t.

Cette méthode est généralement plus rapide, cependant, en raison des mises à jour fréquentes, la convergence devient plus difficile (trouver le minimum de la fonction objective).

❖ **Fonctions de perte**

Une fonction de perte représente une certaine mesure de la différence entre les valeurs observées des données et les valeurs calculées. C'est la fonction qui est minimisée dans la procédure d'optimisation d'un modèle.

Il existe plusieurs fonctions, et le choix dépend du cas d'utilisation, pour une régression, une classification binaire. Dans cette étude on s'intéresse à la classification binaire.

❖ **Fonction de perte pour un problème de classification**

Cette catégorie regroupe plusieurs méthodes, parmi ces méthodes :

- Negative Log Likelihood.
- Margin Classifier.
- Cross Entropy.
- Poisson.
- Hinge.

Dans cette étude, la perte Cross Entropy est utilisée.

- **Principe Cross Entropy**

  Dans une classification binaire, ou le nombre de classes à prédire est égale à deux, la perte Cross Entropy est calculée comme suit :

$$-(ylog(p) + (1 - y)log(1 - p))$$

  Si le nombre de classes > 2 (classification multiple) la perte distincte est calculée pour chaque étiquette de classe par observation et additionnement du résultat :

$$-\sum_{c=1}^{M} yo, clog(po, c)$$

Source : [Fortuner, Viana, & kowshik, 2018]

- M : nombre de classes
- y : indicateur binaire (0 ou 1) si l'étiquette de la classe c est la classification correcte pour l'observation o.
- p : la probabilité que l'observation o appartient à la classe c

2.2.2.2 Méthode des k plus proches voisins

La méthode des k plus proches voisins (KNN : *K-Nearest Neighbors*), vise à

classifier une instance en fonction de ces K plus proches voisins, et ceci, en calculant une distance quelconque.

- **Pseudo algorithme**

1. Choisir le nombre K : nombre des voisins.
2. Pour chaque instance test :

    - Trouver la distance (Cosinus, Euclidienne …) avec toutes instances d'apprentissage.
    - Trié les distances dans une liste.
    - Choisir les K premier classes (étiquettes) relative au K premières distances.
    - Assigner une classe à l'instance test en se basant sur la majorité des classes des K premier point

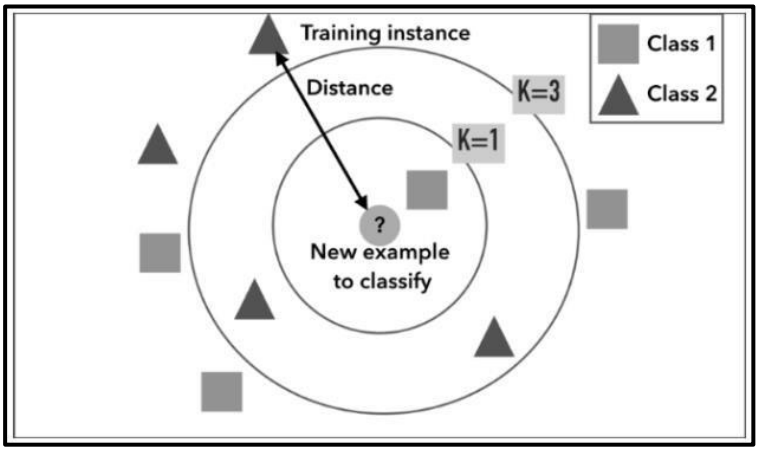


Figure 17: Exemple d'une classification K-NN

Source : [Arnaud, 2018].

2.3 Technique d’échantillonnage

Les méthodes de ré échantillonnage (sampling) sont conçues pour ajouter ou supprimer des exemples de l'ensemble de données d'apprentissage afin de modifier la distribution des classes.

Une fois que les distributions de classes sont plus équilibrées, la suite d'algorithmes de classification d'apprentissage automatique standard peut être adaptée avec succès aux ensembles de données transformés.

Les méthodes de suréchantillonnage (Oversampling) dupliquent ou créent de nouveaux exemples synthétiques dans la classe minoritaire, tandis que les méthodes de sous-échantillonnage (Undersampling) suppriment ou fusionnent des exemples dans la classe majoritaire. Les deux types de rééchantillonnage peuvent être efficaces lorsqu'ils sont utilisés isolément, bien qu'ils puissent être plus efficaces lorsque les deux types de méthodes sont utilisés ensemble.

Dans ce travail, on s’intéresse à *l’oversampling.*

2.3.1 Oversampling

L’oversampling (sur-échantillonnage) est une technique qui rajoute de nouvelles données dans la classe minoritaire. Parmi les techniques de sur-échantillonnage on retrouve un Random Oversampling, SMOTE (Synthetic Minority Oversampling Technique) ou encore ADASYN (Adaptive Synthetic Sampling Approach).

Dans cette étude on s’intéresse à l’oversampling avec SMOTE.

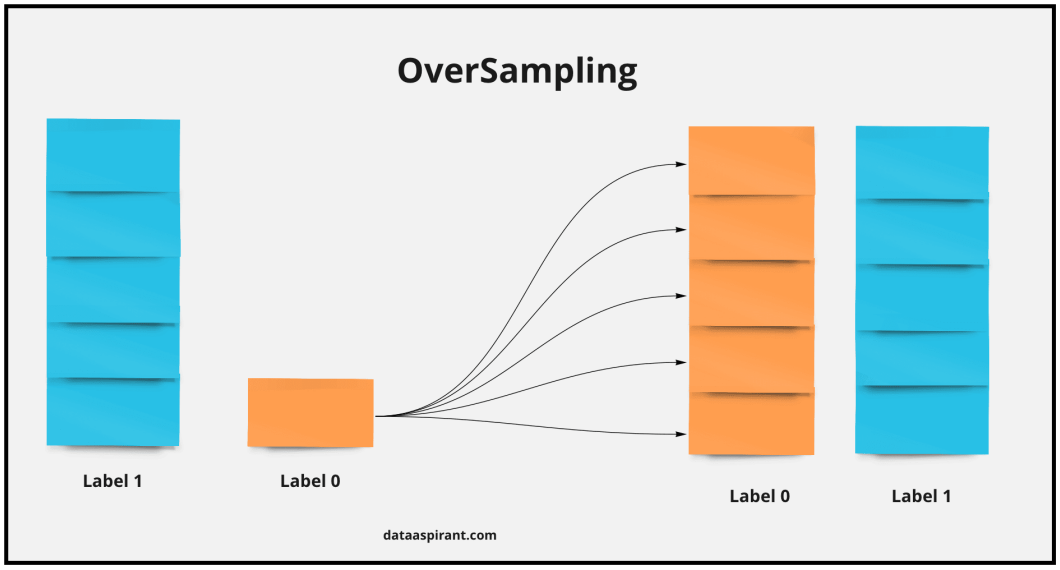


Figure 18: Principe de technique oversampling

[Saimadhu, 2020]

2.3.1.1 SMOTE

(SMOTE)[18] est l'une des méthodes de suréchantillonnage les plus répandues dans la classification déséquilibrée. Elle repose sur l'algorithme des K-plus proches voisins K-NN. SMOTE permet de surmonter le problème de surajustement posé par le suréchantillonnage aléatoire. Il se concentre sur l'espace des fonctionnalités pour générer de nouvelles instances à l'aide d'une interpolation entre les instances positives qui se trouvent ensemble.

[18]https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

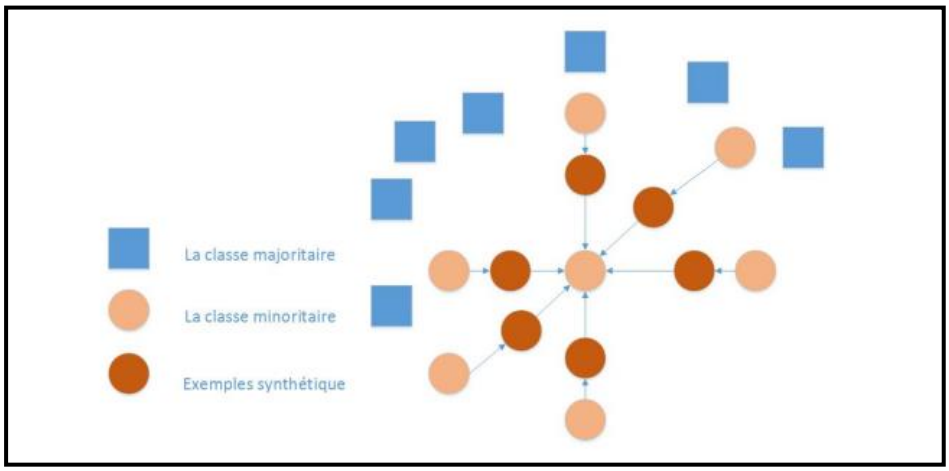


Figure 19: Principe de SMOTE

[Saimadhu, 2020]

Le principe repose sur la différence entre le vecteur voisin et le vecteur du point actuel. Ce résultat est multiplié par un nombre aléatoire entre 0 et 1, et ajouté avec le vecteur du point actuel.

2.3.2 UNDERSAMPLING

L'Undersampling (sous-échantillonnage) est une technique qui supprime des donnés de la classe majoritaire. Parmi les techniques de sous-échantillonnage on retrouve le Random Undersampling et Tomek-links. Elle consiste à supprimer aléatoirement de la base d'apprentissage des individus appartenant à la classe majoritaire, de manière à rééquilibrer la distribution de classes. Cette approche a l'avantage d'être très simple à mettre en œuvre, mais elle risque de supprimer les individus importants pour le concept de la classe majoritaire.

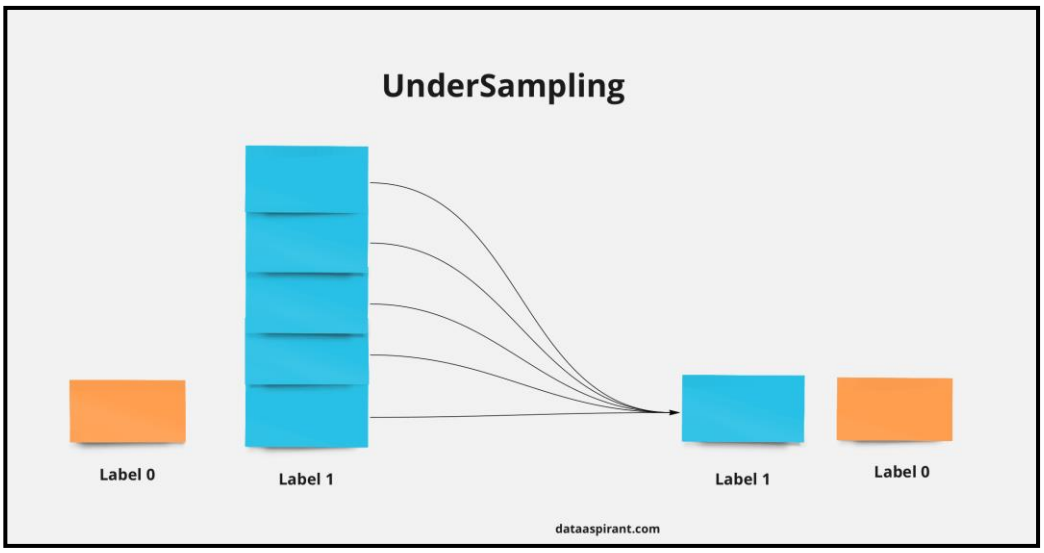


Figure 20: Principe de technique Undersampling

[Saimadhu, 2020]

2.4 Métrique d'évaluations

Il existe plusieurs métriques pour l'évaluation des algorithmes d'apprentissage, et le choix dépend entièrement du cas d'utilisation. L'évaluation consiste à utiliser des *benchmark*s,

Afin de mesurer la différence entre un résultat réel et un résultat obtenu par le ML. Dans cette étude, l'*accuracy*, le rappel, la précision et le F-score sont utilisés.

2.4.1 Matrice de confusion

Une matrice de confusion[19] est un résumé des résultats de prédiction sur un problème de classification.

Le nombre de prédictions correctes et incorrectes est résumé avec des valeurs de comptage et ventilé par chaque classe. C'est la clé de la matrice de confusion.

[19] https://fr.wikipedia.org/wiki/Matrice_de_confusion

Exemple d'une matrice de confusion.

	GoldLabel _A	**GoldLabel_B**	**GoldLabel _C**	
Predicted_A	30	20	10	TotalPredicted_A=60
Predicted_B	50	60	10	TotalPredicted_B=120
Predicted_C	20	20	80	TotalPredicted_C=120
	TotalGlodLabel_A= 100	TotalGlodLabel_B =100	TotalGlodLabel_C=100	

Tableau 1: Exemple d'une matrices de confusion

Dans le tableau on a :

- A, B et C sont les classes.
- La diagonal contient les documents correctement classifiés.
- La somme d'une colonne est le nombre total des instances d'une classe x.

La somme d'une ligne est le nombre des instances attribués à une classe x
Ainsi pour plusieurs classe n le rappel et la précision sont calculés comme suit :

$$recall = \frac{\sum_{i=1}^{n} recall_i}{n}$$

$$precision = \frac{\sum_{i=1}^{n} precision_i}{n}$$

2.4.2 Le rappelle

Le rappel[20] ou sensibilité (Recall en anglais), est défini par le nombre de documents pertinents correctement prédits d'une classe i, au regard du nombre de documents total de la classe i.

Un rappel élevé, signifie que le ML peut prédire correctement un bon nombre de

[20] https://fr.wikipedia.org/wiki/Pr%C3%A9cision_et_rappel

documents pour chaque classe.

$$recall = \frac{\text{nombre de documents correctement attribués a la classe i}}{\text{nombre de documents appartenant a la classe i}}$$

Source : [valuations, 2015].

2.4.3 La précision

La précision[21] est le nombre de documents pertinents retrouvés rapporté au nombre de documents total proposé par le ML. Si elle est élevée, cela signifie que la plupart des documents d'une classe i sont correctement classifiés, ainsi ce dernier peut être considéré comme « précis »

$$précision = \frac{\text{nombre de documents correctement attribués a la classe i}}{\text{nombre de documents attribués a la classe i}}$$

Source : [valuations, 2015].

2.4.4 Le F-score

L'apprentissage à partir de données déséquilibrées revient souvent à faire un compromis entre le rappel et la précision du modèle.

Avoir une haute précision signifie que la plupart des documents attribués à une classe i appartiennent à cette classe. Par contre, avoir un taux de rappel élevé signifie qu'on peut identifier la plupart des individus pour chaque classe du corpus (de cette manière on est capable de détecter des individus d'une classe minoritaire)

[21] https://fr.wikipedia.org/wiki/Pr%C3%A9cision_et_rappel

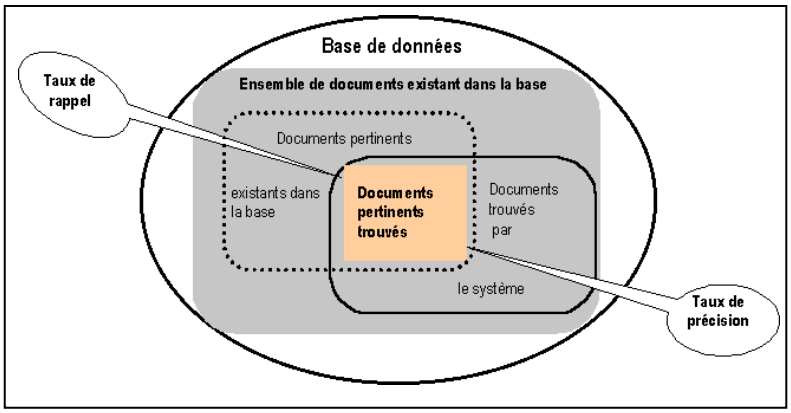


Figure 21: Taux de rappel et taux de précision dans un système d'information

Source : [ROUISSI, 2001].

$$F\_score = 2.\frac{\text{précision.recall}}{\text{précision} + \text{recall}}$$

Source : [valuations, 2015].

2.4.5 L'accuracy

C'est la métrique le plus utilisés et la plus intuitive pour l'évaluation des algorithmes d'apprentissage

$$accuracy = \frac{\text{nombre de documents correctement clasifiés}}{\text{nombre total de documents}}$$

Source : [valuations, 2015].

Parfois, cette métrique est très biaisée. Lorsque 89 % des instances appartiennent à une classe A et 11% des instances à une classe B, il est très probable que le ML (*Machine Learning algorithm)* prédira que toutes les instances appartiendront à la classe A, ainsi l'*accuracy* sera de 89%.

2.5 Conclusion

Dans ce chapitre, nous avons exposé des techniques de représentation du texte comme le Word embedding, ainsi que des algorithmes de classification supervisée. Essentiellement nous avons présenté la classification supervisée

avec deux types de réseaux de neurones. Le premier est un réseau de neurones à convolution CNN. Le deuxième, étant les réseaux de neurones récurrents RNNs de type LSTM.

Nous avons aussi présenté les techniques d'échantillonnages des corpus déséquilibrés. Nous avons exposé les deux principales méthodes d'échantillonnage, le sur-échantillonnage (oversampling) ainsi que le sous-échantillonnage (Undersampling). Nous avons aussi présenté les mesures d'évaluation qui sont plus appropriées aux algorithmes de classification à partir de données déséquilibrées.

Dans le chapitre suivant, quelque travaux précédents serrant exposer.

Chapitre 3 : Travaux antérieurs

3.1 Introduction

Comme dans tous les projets de recherche, l'étude des travaux antérieurs sur la problématique étudiée est très importante. Dans ce chapitre nous présentons les différents travaux récents effectués dans le domaine de l'analyse des réseaux sociaux. Nous commençons tout d'abord par les différentes techniques du prétraitement du texte, ainsi que les méthodes d'extraction de fonctionnalité. Par la suite, nous présentons un ensemble de travaux antérieurs qui se base sur les techniques de classification. Nous les regroupons en deux familles : ceux qui utilisent la classification supervisée et non supervisée. A la fin, nous discutons sur les différentes approches présentées.

3.2 Prétraitement du texte :

Il existe plusieurs types de donnés et pour chacun de ces types, avant de procéder à une analyse, il est nécessaire de prétraiter les données brutes afin de ne laisser que ceux qui présentent des informations pertinentes. Dans notre cas l'analyse concerne le filtrage des tweets.

Dans la littérature du domaine de la fouille de texte, différentes méthodes de prétraitements sont présentes.

Dans la majorité des travaux de la classification des tweets la phase de nettoyage de données est présente comme dans [Gao, 2014 et Lee, 2011] tel que l'élimination des liens, signes, identifiant Twitter, aussi la suppression des mots vides appelé aussi stop words.

[Beverungen et Kalita, 2011] ont proposé des techniques pour la normalisation des vecteurs caractéristiques afin d'améliorer la performance de la classification automatique des tweets.

La lemmatisation **[**Korenius et al. 2004] est une technique de normalisation qui consiste à réduire les mots à leurs lemmes, infinitif ou masculin singulier…etc. La racinisation (stemming) [Kantrowitz, Mohit, et Mittal. ,2000] est une autre technique qui consiste à transformer des flexions en leurs radicales. Le but derrière ces techniques de normalisation est de regrouper les mots qui ont le même sens et qui sont écrit de manière différente, par exemple les verbes conjugués.

3.3 Représentation du texte :

La majorité des algorithmes de classification prennent en entré des données qui se présentent sous format numérique (vecteur numérique) et non pas sous leurs formats bruts. De ce fait une transformation représentative sera importante pour effectuer un traitement analytique sans trop perdre d'information.

Parmi les transformations les plus connue on retrouve TF-IDF (*Term Frequency-Inverse Document frequency)*[22], c'est une méthode statistique basé sur l'occurrence des mots /termes dans un document en prenant compte les autres documents du corpus, l'utilisation de cette dernière a été appliqué sur Twitter dans [Kathy, Lee et al., 2011], D'où le type de donnée (tweets) convient avec le principe de cette méthode, où l'importance du terme vient de sa fréquence d'emploi.

Une autre méthode, Le Word Embedding « plongement de mots en français »[23], c'est une méthode d'apprentissage d'une représentation de mots utilisée notamment en traitement automatique des langues, Cette théorie linguistique fondée par « Zelling Harris » basé sur l'hypothèse distribution elle *distributionnel hypothesis*, qui considère qu'un mot est caractérisé par son contexte, autrement dit les mots qui partagent des contextes similaires partagent

[22]https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089
[23]https://dataanalyticspost.com/Lexique/word-embedding/

également des significations apparentées, cette représentation permet à ces mots d'avoir des vecteurs relativement proches peu distant dans l'espace vectoriel.

L'une des techniques les plus populaires du Word Embedding qui génère une représentation vectorielle le Word2Vec [Mikolov, Sutskever, Chen, Corrado, et Dean, 2013], cette dernière a été appliquée sur des tweets dans [Acosta, Lamaute, Luo, Finkelstein, et Cotoranu, 2017]. Aussi la méthode Glove Global Vectors for Word Représentation [Pennington, Socher, et Manning, 2014] dans [Dasgupta, Kumar, Das, Naskar, et Bandyopadhyay, 2016].

L'analyse sémantique latente ou « LSA »[24] est l'une des techniques de base de la représentation du texte qui permet d'établir une relation entre un ensemble de documents et les termes qu'ils contiennent, en construisant des concepts liés aux document et aux termes, cette analyse est utilisée dans plusieurs recherches comme [Weigend, Wiener, et Pedersen, 1999] et [Steinberger et Jezek, 2004], où cette méthode permet de faire ressortir les relations sémantiques entre les termes. Deux termes peuvent être considérés sémantiquement proches s'ils sont utilisés dans des contextes similaires.

3.4 Mesures de similarités

Dans [Kusner et al. 2015] les auteurs proposent une approche serait de transformer la phrase en vecteur grâce à un outil comme word2vec et ensuite calculer la distance entre les deux vecteurs.

Afin de mesurer la similarité sémantique entre deux phrases, [Sravanthi et Srinivasu, 2017] proposent une méthode qui consiste d'abord à comparer les deux phrases à un niveau syntaxique, ensuite à un niveau de *synset* (synonymes), après à un niveau sémantique où on considère la définition des mots.

Dans [Lesk, 1986], l'auteur propose un algorithme qui est basé sur le fait que les mots dans un même voisinage vont partager le même concept. Une autre

[24] https://www.tandfonline.com/doi/abs/10.1080/01638539809545028

version de Lesk a été adapté [Banerjee et Pederson,2002]. La lacune de cet algorithme est justement le voisinage, si on a un voisinage riche ça serait plus facile de déterminer le concept en d'autres termes l'absence de certain mot peut radicalement changer le résultat.

3.5 Approches d'analyse des réseaux sociaux par classification

3.5.1 Approches supervisées

Les réseaux de neurones convolutifs (CNN) ont permis de résoudre plusieurs problèmes de traitement d'images et de traitement automatique du langage naturel tels que l'analyse d'opinion, les réponses aux questions, le résumé de texte, etc.

Comme dans [Dos Santos et Gatti, 2014], où les auteurs proposent de nouveaux réseaux de neurones convolutifs profonds qui exploitent les informations de caractère à phrase pour effectuer une analyse des sentiments dans des textes courts.

[Kim, 2014] a appliqué un CNN simple avec une seule couche convolutive à l'aide d'un modèle non supervisé, sur plusieurs bases de données, Il a utilisé un petit hyper paramètre, qui a généré des résultats très puissants. Il a également utilisé différentes tailles de filtres et testé plusieurs modèles CNN pour extraire les données importantes. Il a conclu que CNN-static est le modèle qui a donné les meilleures performances.

[Wang et al., 2016] ont travaillé sur la combinaison de CNN et RNN pour l'analyse d'opinion dans les phrases. Ils voulaient profiter des avantages du CNN et du RNN afin d'avoir plus de précision. Ils ont appliqué le CNN, qui est insensible à l'emplacement des mots dans la phrase. Ensuite, ils ont utilisé la sortie du CNN comme entrée pour le RNN formé avec rétropropagation dans le temps.

[Zhou et al.,2016] ont appliqué LSTM bidirectionnel avec max pooling bidimensionnel sur la base de données Stanford Sentiment Treebank (STS), de sorte que chaque vecteur est représenté par une matrice de dimension 2. Ainsi, ils ont changé l'utilisation habituelle du pooling 2D pour échantillonner des

caractéristiques plus pertinentes. Pour les tâches de modélisation de séquence. De plus, ils ont utilisé la convolution 2D pour mettre en évidence les informations les plus importantes sur la matrice. La combinaison BLSTM-2DPooling a atteint une performance de 88,3% tandis que la combinaison BLSTM-2DCNN a atteint une performance de 89,5% sur la base de données SST2.

[Yenter et Verma, 2017] ont proposé un modèle CNN-LSTM pour l'analyse d'opinion à partir de la base de données IMDB. Ce travail diffère du précédent car ils ont concaténé les résultats après l'application de la couche LSTM. Ce modèle a atteint une précision de 89 %.

[Shen et al.,2017] ont proposé une conception spéciale qui combine les modèles CNN et BiLstm pour des performances optimales. Ils ont constaté que cette combinaison donnait une précision de 89,7%, meilleure que la précision de l'un ou l'autre modèle individuellement.

[Yoon et al., 2017] ont proposé une architecture Cnn-BiLstm pour prédire les sentiments au niveau du document avec des intégrations multicanaux en utilisant l'intégration de mots Word2vec. Les modèles ont été appliqués sur différents jeux de données et ont obtenu des performances prometteuses mais modestes comprises entre 51,97 % et 70,08 %

Dans [Yoon, 2014], l'auteur utilise un réseau neuronal convolutif (CNN) multi canal pour la classification des phrases, l'auteur utilise des vecteurs pré-entrainés de mots avec word2vec, ces vecteurs ont été entraînés sur 100 billions de mots grâce à Google News, l'idée est de capturer des caractéristiques génériques. Plusieurs variantes du model sont testées : CNN-rand ou l'initialisation est aléatoire ensuite modifiée durant l'apprentissage, CNN-static ou les vecteurs pré-entrainé word2vec sont utilisés sans modification dans l'entrainement, CNN_no_nstatic ou la modification est activé et enfin un CNN_ multi canal ou les mots sont représentés grâce à deux vecteurs l'un étant statique et l'autre dynamique. Ces modèles sont régularisés aussi avec un drop-out de 2% à 4% (une technique pour éviter le sur-apprentissage). Les résultats mettent en

évidence qu'une représentation word2vec pré-entrainé donne de meilleurs résultats.

[Lökk et Hallman, 2016] ont utilisé deux méthodes différentes d'analyse des sentiments naïve bayes et maximum Entropy, pour la détection des trolls.

Pour classer les tweets avec naïve bayes ils ont utilisé la fonction de classification de probabilité antérieure et un lexique des mots et leur classification sentimentale ainsi de renvoyer une liste des tweets avec une classification ajoutée de négatif, neutre ou positif pour l'Entropy qui a fourni une répartition plus juste des sentiments et peut-être une classification plus réaliste.

[Miao L et al ,2020] dans la détection des tweets trolls dans un corpus bilingue, ils ont considéré Naïve Bayes (NB), K-Nearest Le classificateur Voisin (KNN) et Support Vector Machine (SVM) pour détecter les tweets troll basés sur des fonctionnalités stylo métriques. Ils ont utilisé le classifieur de régression logistique pour les fonctions n-gram. Et pour les méthodes d'apprentissage profond ils ont utilisé CNN, RNN et CNN+RNN. Ils ont obtenu les meilleurs résultats lors de l'utilisation d'algorithmes d'apprentissage en profondeur de la combinaison CNN+RNN

3.5.2 Approches non supervisées

[Friedemann, 2015] l'auteur utilise Kmeans avec une mesure de similarité pour faire un clustering des clients d'une compagnie. On calcule le coefficient de la silhouette maximal pour choisir le nombre de topic k, puis faire un regroupement en utilisant la distance euclidienne avec l'algorithme Kmeans, [Forgy, 1965]. Cette technique a donnée des résultats satisfaisants.

Un travail a été réalisée par [Purwitasari et al, 2015] pour résumer les nouveautés sur Twitter, les auteurs ont choisis de garder les retweets et de laisser les hashtags.

Puis, ils ont calculé la fréquence des termes dans un tweet après appliquer l'algorithme

K-medoids [kaufman and Rousseeuw, 1987] pour le regroupement (clustering).

Les résultats n'étaient pas satisfaisants à cause de l'inclusion des retweets qui ont affecté la qualité du clustering.

Une étude récente [Popovici et al., 2014], ils proposent une méthode de clustering des tweets en ligne, pour identifier un cluster, le tweet avec le meilleur degré de pertinence sémantique c'est à dire avec une similarité maximale dans un cluster est sélectionné comme topic.

Pour le regroupement des tweets avec la distance euclidienne et la distance Manhattan sur un dataset étiqueté [Zhao, 2011] utilise Kmeans et K-Medoids. L'auteur initialise le nombre de K selon le nombre des classes pour examiner l'homogénéité des clusters, en prenant les tops mots fréquents dans chaque cluster. Il déduit que les clusters obtenus sont bien harmonieux.

[Engelin, M et al, 2016], ils proposent une méthode de clustering des sur la détection des fermes de trolls sur Twitter. Le but de cette recherche est de tester si les algorithmes de clustering peuvent être utilisés pour détecter les fermes de trolls dans les réseaux sociaux. Les fermes de trolls sont des organisations professionnelles qui diffusent de la désinformation en ligne via de faux personnages. La recherche comprend une étude comparative de deux algorithmes de regroupement différents et un ensemble de données d'utilisateurs et de publications de Twitter qui comprend une ferme de trolls fabriquée.

En comparant les résultats et les implémentations des Kmeans ainsi que de l'algorithme DBSCAN
L'algorithme DBSCAN a des variables qui gèrent la taille minimale acceptable pour un cluster et la distance de seuil. Ces variables doivent être spécifiées et après quelques expérimentations avec différentes valeurs.
Ils ont obtenu que l'algorithme DBSCAN soit une meilleure approche que k-means

3.6 Technique de Sampling et d'évaluation :

Diffèrent mesures existe pour évaluer différentes caractéristiques du ML

(*Machine Learning algorithm*). Dans [Sokolova et al., 2006], les auteurs expliquent que ça dépendra du cas d'utilisation. Parfois c'est utile d'avoir un taux de réussite général, mais dans certaines situations, on s'intéresse plutôt à un taux de réussite spécifique.

C'est généralement le cas lorsqu'une classe minoritaire doit être détectée. Dans [Somasundaram et Reddy, 2016], ils rappellent que dans la détection d'anomalies, la classification d'images, les gènes et les documents, les motifs d'intérêt représentent généralement moins de 1% des données complètes. Dans cette recherche de détection de fraude par carte de crédit/débit, des techniques d'échantillonnage sont d'abord utilisées pour évaluer l'algorithme d'exercice d'arbre de décision (classificateur de forêt aléatoire) utilisé par l'auteur : précision, taux vrai négatif (TNR), valeur de prédiction négative (VAN) et le score F proposé par Rijsbergen (1979).

Le problème du dataset déséquilibré est notamment adressé dans [Kaur et Gosain, 2018], les auteurs expliquent que dans plusieurs scenarios sérieux comme la détection de tumeurs, les algorithmes de machine Learning échouent. Les auteurs comparent deux approches, l'oversampling avec SMOTE (Synthetic Minority Oversampling) [Bowyer et al., 2002], dont le but est de créer des données synthétiques similaires aux k-plus proches voisins d'un ensemble de données aléatoire d'une classe minoritaire et l'Undersampling avec RUS (Random Undersampling). Les auteurs utilisent un dataset qui contient 70% de bruit et observent que l'oversampling donne de meilleurs résultats.

Dans [Hacibeyoglu et Ibrahim, 2018], les auteurs trouvent que grâce à un *oversampling* avec SMOTE l'algorithme de classification donne de meilleurs résultats qu'un *Undersampling* pour prédire des patients malades

Dans [Ah-Pine et Soriano-Morales, 2016], ils s'intéressent au problème du dataset déséquilibré, et proposent d'effectuer un oversampling synthétique SMOTE sur des vecteurs TF-IDF pour la classification binaire du sentiment des tweets en utilisant l'apprentissage par arbre de décision ainsi que la régression logistique. Les auteurs utilisent trois dataset publique

« Obama-McCain Debate », « Health Care Reform » et « Imagiweb », pour

l'évaluation le F- score, l'accuracy global et l'accuracy moyenne sont utilisés. Toutefois les auteurs s'aperçoivent qu'après un oversampling, l'accuracy global diminue dans tous les tests, et mettent en évidence que l'accuracy global favorise la classe majoritaire par contre le F-score augmente après un oversampling car l'algorithme a pu détecter la class minoritaire, l'accuracy moyenne a augmenté dans un seul cas.

Dans [Imran et Srinivasa, 2018] et [Triguero, Galar, Bustince, et Herrera, 2017], les auteurs travaillent sur les big data, utilisons des méthode d'Undersampling, où [Río, López, Benítez, et Herrera, 2014] trouvent que ces techniques convient aux grands corpus.

3.7 Discussion :

Selon les articles contenus dans ce chapitre :

En général, les auteurs utilisent des représentations au niveau des mots (ou tokens) pour la représentation vectorielle à partir du texte. La représentation vectorielle avec TF-IDF est largement utilisée pour la modélisation de documents, et il existe des techniques plus modernes telles que l'incorporation de mots (Word Embedding), cette dernière permettre de capturer les relations sémantiques entre les mots.
De nombreux auteurs travaux le Word Embedding pré-entrainé pour apprendre les caractéristiques générales de la langue, mais cela peut ne pas convenir aux situations sociales, qui contiennent souvent des orthographes et des stéréotypes.

Les réseaux de neurones sont très utilisés pour la classification supervisée du texte et donnent de bons résultats. Les auteurs utilisent généralement des réseaux de neurones convolutifs pour le traitement d'images, mais les CNN sont plus récemment utilisés spécifiquement pour classer le texte, en raison de leur capacité à découvrir des caractéristiques locales importantes.

Des travaux récents utilisent notamment des réseaux de neurones récurrents tels que les LSTM et les GRU. Ces réseaux permettent de maintenir la dépendance temporelle sémantique, qui s'adapte parfaitement aux données séquentielles telles que les données textuelles.

Pour le *clustering* du texte les auteurs utilisent plus souvent Kmeans.

Pour l'échantillonnage, le suréchantillonnage et plus particulièrement SMOTE donne de meilleurs résultats que le sous-échantillonnage. Le sous-échantillonnage est préféré dans le cas où le corpus est très grand, car le nombre des données après un sous- échantillonnage est toujours suffisant pour l'apprentissage du ML.

3.8 Conclusion :

Dans ce chapitre, nous avons présenté les travaux connexes par rapport à notre sujet qui traite l'analyse des données textuelle. Nous avons abordé les études les plus pertinentes. Précisément celle qui traitent les données sur les réseaux sociaux notamment Twitter qui est devenu la plateforme de micro blogage la plus étudiée dans les recherches.

Nous avons exposé les différentes solutions proposées dans la littérature, le traitement des données ainsi que la représentation textuelle, suivi par des techniques de conversion de texte et un comparative sur les algorithmes de classification

Dans le chapitre suivant nous allons présenter notre solution pour l'analyse des réseaux sociaux par apprentissage profond.

Chapitre4 : Analyse des réseaux sociaux par apprentissage profond

4.1 Introduction

Apres l'étude de l'état de l'art, nous proposons dans ce chapitre une approche pour l'analyse des tweets. Cela, en s'inspirant des algorithmes d'apprentissage profond qui ont fait leurs preuves en matière d'analyse de textes. Notre cas d'étude concerne la détection des publications des trolls à travers une collection de tests réel (*Trolls Detection*[25]).

Nous proposons deux solutions :

- Une combinaison entre les algorithmes d'apprentissage profond CNN et LSTM
- Une deuxième combinaison entre les algorithmes CNN et BiLstm

A ces deux solutions est associée une méthode d'augmentation de données (*oversampling)* appelé SMOTE.

4.2 Architecture générale :

La figure ci-dessous montre l'architecture générale de la solution proposée :

[25]https://www.kaggle.com/dataturks/dataset-for-detection-of-cybertrolls

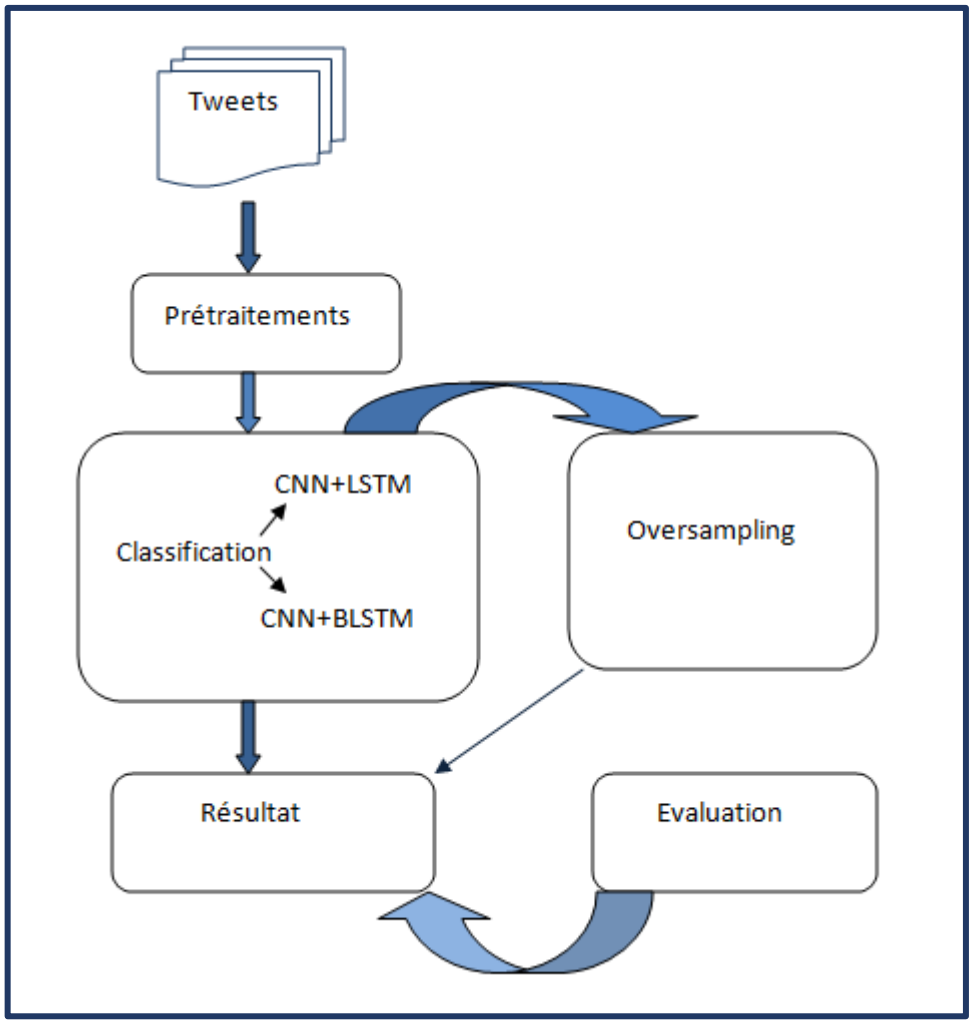


Figure 22: Architecture générale de la solution proposée

4.2.1 Prétraitement

Le prétraitement est une étape très importante avant de procéder à une représentation vectorielle qui consiste à réduire les informations indésirables.

.

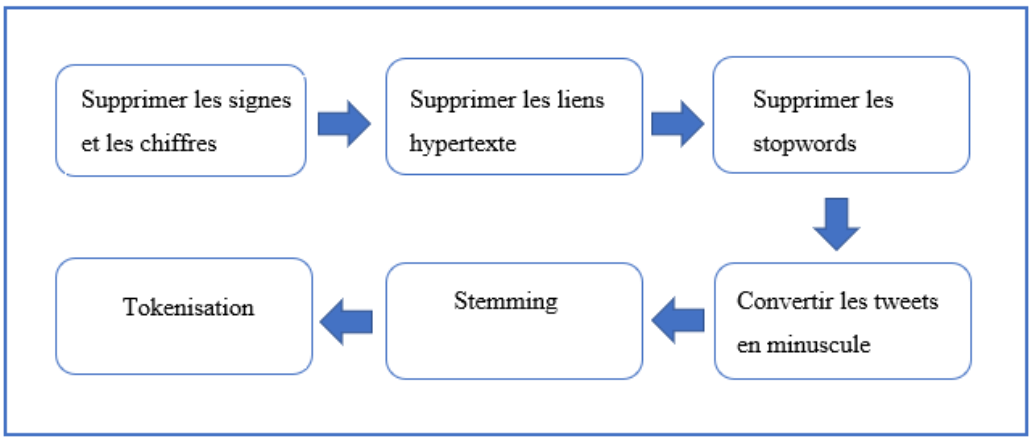


Figure 23 : Schéma de la procédure pour le prétraitement des tweets

➢ **Procédure de prétraitement :**

- La conversion de tous les tweets en minuscule.
- Les chiffres, les identifiant (@username) ainsi que les hashtags sont supprimé.
- Les ponctuations comme les points, virgules, point d'exclamation, etc. sont supprimé.
- Les stop words ou mots vides (and, to, a, etc.…) sont des mots qui n'apportent aucun sens sémantique donc il est inutile de les indexer
- Un procédé stemming est accompli, où les flexions sont transformées en leur radicale (racine).
- Tous les tweets sont découpés en token (Tokenisation).

4.2.2 L'Oversampling (SMOTE)

Parmi les paramètres qui peuvent influencer les résultats d'algorithmes d'apprentissage, le déséquilibre entre le nombre d'objets (tweets dans notre cas) entre classes. En effet, les algorithmes d'apprentissage seront insuffisamment entrainés pour identifier des objets qui appartiennent aux classes minoritaires.
L'oversampling est une technique utilisé pour ajuster la distribution des classes d'un ensemble de donnée, c'est une méthode d'augmentation de donnée. SMOTE

consiste à créer des données synthétiques de la classe minoritaire. (Voir chapitre 2, section 2.2)

- **Pseudo code de l'algorithme SMOTE**

```
1 Algorithme : SMOTE
2 Pour un nombre suffisant d'instance synthétique faire
3 |   Sélectionner une instance minoritaire A
4 |  Sélectionner une des instances les plus proche voisins B
5 |   Sélectionner un poids aléatoire entre 0 et 1 Créer la nouvelle
  |   Instance
  |   Synthétique c
6     Pour chaque attribut faire
7 |       attValue_C = attValue_A + (attValue_B - attValue_A) * W
8     Fin Pour
9 Fin Pour
```

4.2.3 Classification CNN+LSTM

Une combinaison entre CNN et LSTM est proposée. La motivation derrière ce travail, serait de bénéficier des propriétés des deux couches CNN et LSTM. Comme il a été expliqué précédemment, les Cnns peuvent extraire des caractéristiques locales importantes, et les Lstms soutiennent la prédiction de séquence dans le temps.

- **Architecture CNN-LSTM**

La figure ci-dessous montre l'architecture de notre combinaison CNN-LSTM

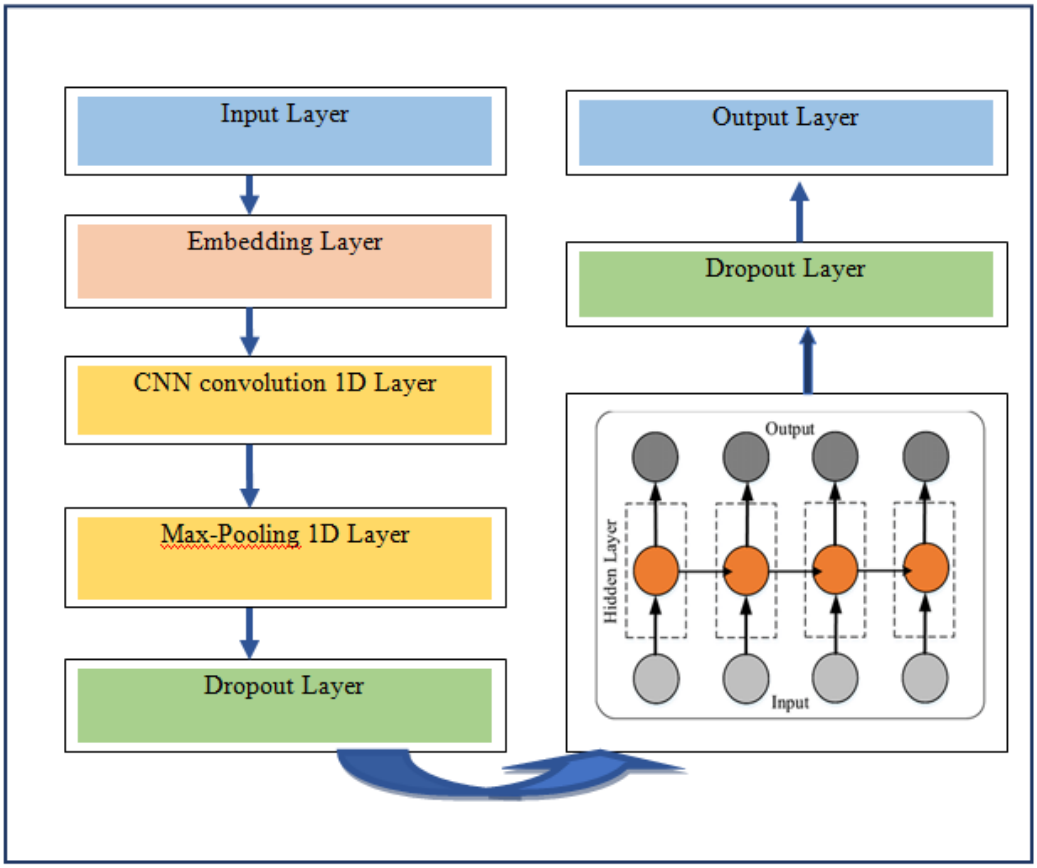


Figure 24 : Architecture de notre approche Cnn-Lstm

- Une couche *embedding* est d'abord introduite. Elle va servir à améliorer la capacité des réseaux à apprendre à partir de données textuelles, en représentant ces données sous forme de vecteurs de dimension inférieure.
- Une couche de convolution CNN suivi d'un *Max-Pooling* est ajoutée. Le *Max-Pooling* est utilisé en raison du fait que ça considère que les caractéristiques les plus importantes, contrairement à l'*Average-pooling* qui considère la totalité de l'input.
- Une couche LSTM
- Un Dropout est ajouté pour éviter le sur-apprentissage.

Il a été jugé plus intuitive d'introduire une couche CNN avant la couche

LSTM, et ce, pour extraire des caractéristiques importantes ensuite de « s'en souvenir » grâce à LSTM

- **Pseudo algorithme :**

1- **Un modèle Sequential ()** est approprié pour une simple pile de couches où chaque couche a exactement un tenseur d'entrée et un tenseur de sortie.

2- **La couche Embedding ()** est initialisée avec des poids aléatoires et apprendra une incorporation pour tous les mots de l'ensemble de données d'apprentissage. Il faut 3 arguments :

- Input_dim : C'est la taille du vocabulaire dans les données textuelles qui est de **Vocab size** dans notre cas.
- Output_dim : C'est la taille de l'espace vectoriel dans lequel les mots seront intégrés. Il définit la taille des vecteurs de sortie de cette couche pour chaque mot. Nous l'avons fixé à 100.
- Input_length : Longueur des séquences d'entrée, lorsqu'elle est constante. Dans notre cas, c'est 120.

3- **Dans la couche Conv1D (),** nous apprenons un total de **32 filtres** avec une taille de fenêtre convolutive **kernel_size = 3**. Nous utiliserons la fonction d'activation **ReLu**

4- **MaxPool1D ()** sous-échantillonne la représentation d'entrée en prenant la valeur maximale sur la fenêtre définie par **pool_size qui est 2** dans le cas de ce réseau de neurones.

5- **Couche Dropout** utilisé pour éviter overfitting (0.5)

6- **Couche Lstm** avec (filters = 128 et function d'activation Tanh) qui a 3 portes :

- La porte d'entrée décide si l'entrée doit modifier le contenu de la cellule
- La porte d'oubli décide s'il faut remettre au contenu de la cellule
- La porte de sortie décide si le contenu de la cellule doit influer sur la sortie du neurone

7- **Couche Dropout** utilisé pour éviter overfitting 0.5

8- **Couche dense** avec filters= 1 et function d'activation sigmoïde

4.2.4 Classification Cnn + BiLstm :

Nous avons proposé une combinaison entre Cnn et BiLstm, pour nous bénéficier des propriétés des deux couches Cnn et BiLstm, les Cnns permet d'extraire des caractéristiques locales importantes et les BiLstms permet au modèle d'apprendre des fonctionnalités plus abstraites.

L'utilisations des BiLstm comparant à Lstm, généra l'exécution des entrées de deux manières, l'une du passé au futur et l'autre du futur au passé, ils sont donc capables à tout moment de préserver les informations du passé et du futur.

❖ **Architecture Cnn-BiLstm :**

Le schéma suivant montre l'architecture de notre approche

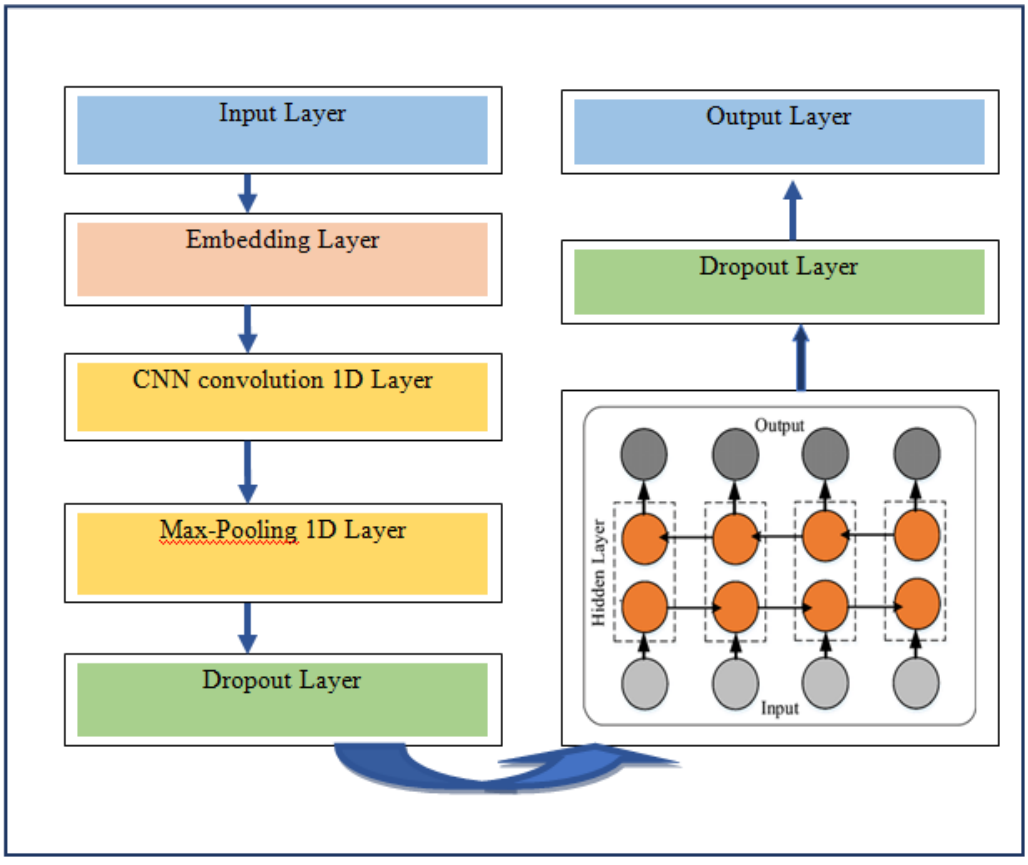


Figure 25 : Architecture de notre approche Cnn-BiLstm

- Une couche d'entrée embedding qui va servir à améliorer la capacité des réseaux à partit des données textuelles.
- Une couche de convolution qui traite les données d'un champ récepteur.
- La couche pooling qui permet de compresser l'information.
- La couche entièrement connectée (FC) qui est une couche de type perceptron.
- Une couche BiLstm est ajouté pour capturer les informations sur les entrées environnantes.
- Dropout pour éviter le sur-apprentissage.

➢ **Pseudo algorithme :**

1- **Un modèle Sequential ()** est approprié pour une simple pile de couches où chaque couche a exactement un tenseur d'entrée et un tenseur de sortie.

2- **La couche Embedding ()** Il faut 3 arguments :

- Input_dim : C'est la taille du vocabulaire dans les données textuelles qui est de **Vocab size** dans notre cas.
- Output_dim : C'est la taille de l'espace vectoriel dans lequel les mots seront intégrés. Il définit la taille des vecteurs de sortie de cette couche pour chaque mot. Nous l'avons fixé à 100.
- Input_length : Longueur des séquences d'entrée, lorsqu'elle est constante. Dans notre cas, c'est 120.

3- **Dans la couche Conv1D (),** nous apprenons un total de **32 filtres** avec une taille de fenêtre convolutive **kernel_size = 3**. Nous utiliserons la fonction d'activation **ReLu**

4- **MaxPool1D ()** avec **pool_size qui est 2** dans le cas de ce réseau de neurones.
5- **Couche Dropout** utilisé pour éviter overfitting (0.5)
6- **Couche Bidirectional Lstm** avec (filters = 200 et function d'activation Tanh).
7- **Couche Dropout** utilisé pour éviter l'overfitting (0.5)
8- **Couche dense** avec filters= 1 et function d'activation sigmoïde

4.2.5 Evaluation des résultats :

Nous nous basons sur les mesures : l'accuarcy, la précision, le rappelle (recall) et le F-score pour l'évaluation de nos approches.

4.3 Conclusion

Dans ce chapitre, nous avons présenté les combinaisons entre les algorithmes de classification CNN et LSTM ensuit entre le CNN et le BiLstm, ainsi que la technique de sur-échantillonnage (oversampling) SMOTE. Dans le chapitre suivant différant tests sont présenté afin d'évaluer les solutions exposées.

Chapitre 5 : Expérimentation et comparaison des résultats

5.1 Introduction

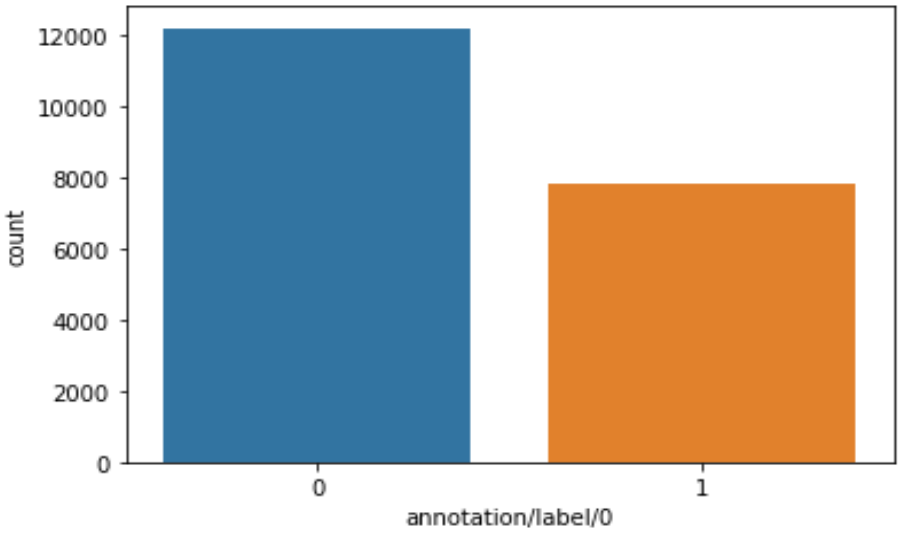
Dans ce chapitre nous allons exposer les résultats des différents test établis en utilisant les techniques présentées dans les chapitres précédent.

5.2 Jeu de données

L'ensemble de donnée utilisé dans ce travail pour comparer nos algorithmes pour la classification des tweets :

Trolls Detection[26] : L'ensemble de données contient 20001 tweets, c'est une collection de tests composé de Tweets étiqueté selon deux catégories :
1 (classe agressive) : 7822 tweets et 0 (classe non agressive) : 12179tweets.

Les figures suivantes (figures 26 et 27) montrent respectivement la distribution des Tweets selon les deux classes, ainsi qu'un nuage des mots les plus fréquents dans Dataset Trolls.



[26]https://www.kaggle.com/dataturks/dataset-for-detection-of-cybertrolls

Figure 26 : Distribution de l'ensemble de données

Figure 27 : Nuages des mots les plus fréquents dans Dataset trolls Original

5.3 Protocole expérimentale :

- ❖ Python 3.7 est utilisé pour l'implémentation des algorithmes.

- ❖ Librairie utilisée pour la réalisation de ce travail :

  - NLTK (Natural Language Toolkit).
  - Tensorflew (L'implémentation des réseaux de neurones).
  - Numpy (Manipulations des tableaux et matrices).
  - Pandas (L'analyse des données).

5.3.1 Prétraitement

Les techniques utilisées pour le nettoyage des tweets, qui est une partie importante pour garder que les informations pertinentes :

- Suppression des chiffres et des signes.

- Suppression des lien hypertexte.
- Suppression des stopwords en utilisant la librairie NLTK.
- Faire un stemming en utilisant Stemmer dans la librairie NLTK.
- Conversion des tweets en minuscule.
- Faire une Tokenisation.

5.3.2 Implémentation et tests réalisé :

Nous avons effectué plusieurs tests sur le dataset Trolls et ceci pour :

- Le dataset original (déséquilibré).
- Le dataset après Oversampling.

Afin d'évaluer ces derniers, l'accuracy, la précision, le recall et le F-score sont utilisé.

❖ **Etude comparative entre les algorithmes de classification** :

Afin de comparer entre les résultats des algorithmes d'apprentissage et les combinaisons proposées, nous avons implémenté les réseaux de neurone suivants :

- Réseau de neurone à convolution CNN
- Réseau de neurone récurrent LSTM.
- Réseau de neurone récurrent BiLstm
- CNN-LSTM
- CNN-BLSTM

Pour l'augmentation de nos données nous avons testé la méthode de sur-échantillonnage (oversampling) SMOTE avec des vecteurs TF-IDF sur le dataset Trolls.

❖ **Liste des paramètre et notation :**

Cette partie présente les différentes notations et paramètres utilisées pour les tests.

❖ **Réseaux de neurone avec CNN :**

Nous avons utilisé 5 couches :

- **La couche Embedding ()** : est initialisée avec des poids aléatoires et apprendra une incorporation pour tous les mots de l'ensemble de données d'apprentissage. Il faut 3 arguments :
- Input_dim : C'est la taille du vocabulaire dans les données textuelles qui est le **Vocab size** a été calculé dans notre cas.
- Output_dim : C'est la taille de l'espace vectoriel dans lequel les mots seront intégrés. Il définit la taille des vecteurs de sortie de cette couche pour chaque mot. Nous l'avons fixé à 100.
- Input_length : Longueur des séquences d'entrée, lorsqu'elle est constante. Dans notre cas, c'est 120.

- **La couche Conv1D () :**
  Le nombre d'unité (modifié dans chaque test) avec une taille de fenêtre convolutive **kernel_size = 4**. Nous utiliserons la fonction d'activation **ReLu**
- **La couche MaxPool1ing1D ()** : avec pool_size est 2
- **La couche Flatten** ()
- **La couche Dense ()** : est la couche de réseau de neurones profondément connectée. La couche de sortie est une couche dense avec 1 neurone car nous prédisons une seule valeur. La fonction sigmoïde est utilisée car elle existe entre (0 à 1) et cela nous facilite la prédiction d'une entrée binaire.
- **Batch_size** : modifié dans chaque test
- **Epoch** : modifié dans chaque test.

❖ **Réseaux de neurones avec Lstm :**

- Lstm-output : nombre d'unité (modifié dans chaque test).

- Activation : Tanh (fonction d'activation par défaut).
- Dropout : modifié dans chaque test.
- Optimizer : Adam (mentionné si modifié).

❖ **Réseaux de neurones avec BiLstm :**

- BiLstm-output : nombre d'unité (modifié dans chaque test).
- Activation : Tanh (fonction d'activation par défaut).
- Dropout : modifié dans chaque test.
- Optimizer : Adamax (mentionné si modifié).

❖ **Oversampling**

➢ **SMOTE**

Définir une instance SMOTE avec des paramètres par défaut qui équilibreront la classe minoritaire, puis l'ajusteront et l'appliqueront en une seule étape pour créer une version transformée de notre ensemble de données

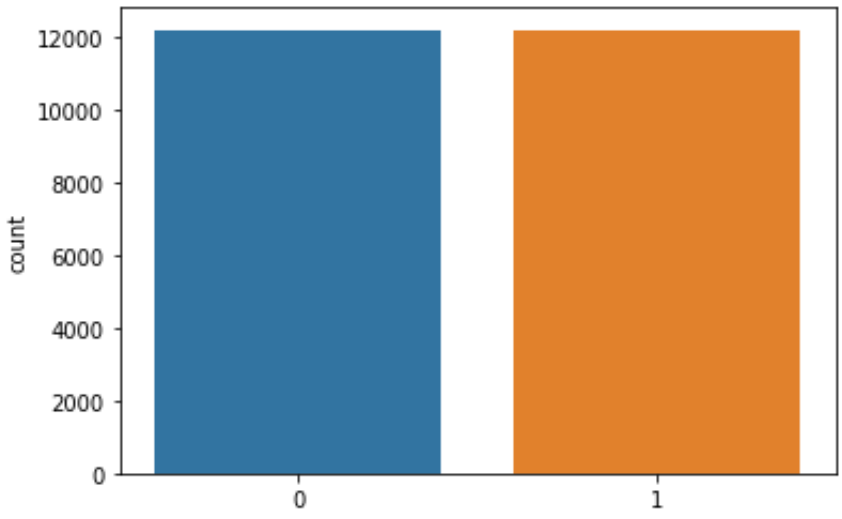


Figure 28 : Distribution de l'ensemble de données après oversampling (SMOTE)

❖ **Division du dataset :**

- Le dataset a été divisé en 20% pour les tests et 80% pour l'apprentissage.
- Random_state (de division) =50.

❖ **Evaluation de la combinaison CNN-LSTM :**

- Kernel_size : 3. (Mentionné si modifié)
- Pool_size : 2. (mentionné si modifié)
- Filters (Cnn_dim) : 32.
- LSTM (filters) : 128 (mentionné si modifié)
- Dropout 1 : 0.3.
- Dropout 2 : 0.3
- Batch size : modifié dans chaque test
- Epochs : modifié dans chaque test
- Optimizer : Adam (mentionné si modifier).

❖ **Evaluation de la combinaison Cnn-BiLstm :**

- Kernel_size : 3. (Mentionné si modifié)
- Pool_size : 2. (mentionné si modifié)
- Filters (Cnn_dim) : 32.
- BILSTM(Filters) : 200 (mentionné si modifié)
- Dropout 1 : 0.3.
- Dropout 2 : 0.3
- Batch size : modifié dans chaque test
- Epochs : modifié dans chaque test
- Optimizer : Adam (mentionné si modifier).

5.4 Résultats

Nous présentons les résultats des tests effectués sur les algorithmes de classification présentée précédemment.

5.4.1 Résultats des tests Cnn :

Le tableau suivant résume les résultats des tests effectués sur notre collection en utilisant l'algorithme CNN avant est après Oversampling (SMOTE).

	Accuracy%		Précision%		Recall%		F1_score%	
	Original	SMOTE	Original	SMOTE	Original	SMOTE	Original	SMOTE
Epoch =10 Batch_size =50 Cnn_dim=32 Embedding_dim =100	75.80	79	0 : 82 1 : 68 Total : 67.5	0 :82 1 :75 Total75 :	0 :77 1 :73 Total : 73.3	0 :73 1 :85 Total :85	0 :80 1 :70 Total : 70.3	0 :77 1 :80 Total :80
Epoch =10 Batch_size =128 Cnn_dim=32 Embedding_dim =100	77.60	77.87	0 :82 1 :71 Total :77.8	0 :78 1 :76 Total : 76	0 :81 1 :73 Total :72.5	0 :75 1 :79 Total :79	0 :81 1 :72 Total :71.7	0 :77 1 :78 Total :77
Epoch=30 Batch_size= 50 Cnn_dim=32 Embedding_dim =100	**85.42**	82.58	0 :90 1 :79 Total : 79.12	0 :83 1 :82 Total : 76	0 :86 1 :85 Total : 85.23	0 :81 1 :83 Total :86	0 :88 1 :82 Total : **82.06**	0 :82 1 :82 Total :81
Epoch = 50 Batch_size =50 Cnn_dim=32 Embedding_dim =100	85,20	86.03	0 :90 1 :78 Total : 78.40	0 :85 1 :80 Total :82	0 :85 1 :86 Total : 85.81	0 :78 1 :86 Total :83	0 :87 1 :82 Total : 81.94	0 :81 1 :83 Total :82

Tableau 2: Résultat CNN sur dataset

5.4.2 Résultat des tests Lstm :

Le tableau suivant montre les résultats des tests effectués sur notre collection en utilisant l'algorithme LSTM avant est après Oversampling.

	Acc%		Précision%		Recall%		F1_score%	
	Original	SMOTE	Original	SMOTE	Original	SMOTE	Original	SMOTE

Epoch =10 Batch_size =50 lstm_dim=128 Embedding_dim =100	88.07	87	0 : 95 1 : 80 Total : 79.94	0 : 95 1 : 78 Total : 74	0 : 85 1 : 93 Total : 93.31	0 : 83 1 : 94 Total : 94	0 : 90 1 : 86 Total : 86.11	0 : 89 1 : 85 Total : 83
Epoch =10 Batch_size =128 lstm_dim=200 Embedding_dim =100	85.47	0,86	0 : 94 1 : 76 Total : 76.44	0 :94 1 :77 Total :76	0 : 81 1 : 92 Total :92.28	0 : 81 1 : 93 Total :92	0 : 87 1 : 84 Total : 83.62	0 : 87 1 : 84 Total :84
Epoch=30 Batch_size= 50 lstm_dim=200 Embedding_dim =100	85.50	88	0 : 93 1 : 76 Total :75.4	0 :95 1 :79 Total :79	0 : 84 1 : 88 Total : 88.20	0 : 84 1 :94 Total : 93	0 : 88 1 : 83 Total : 84.03	0 :89 1 :86 Total :86
Epoch=50 Batch_size=50 lstm_dim=200 Embedding_dim =100	**87.6**	88	0 : 97 1 : 80 Total : 81.3	0 :96 1 :78 Total :78	0 : 84 1 : 96 Total : 96.30	0 :84 1 :96 Total :95	0 : 90 1 : 84 Total : 84.60	0 :89 1 :86 **Total :86**

Tableau 3: Résultat LSTM sur dataset

5.4.3 Résultat des tests BiLstm

Le tableau suivant résume les résultats des tests effectués sur notre collection en utilisant l'algorithme BiLstm avant est après Oversampling.

	Accuracy%		Précision%		Recall%		F1_score%	
	Original	SMOTE	Original	SMOTE	Original	SMOTE	Original	SMOTE
Epoch =10 Batch_size =512 blstm_dim=200 Embedding_dim =100	84.05	83	0 : 93 1 : 77 Total : 77.29	0 :90 1 :74 Total :77	0 :83 1 :85 Total : 85.37	0 :80 1 :87 Total :92	0 :87 1 :83 Total : 82.13	0 :85 1 :80 Total :80

Epoch =10 Batch_size =128 blstm_dim=200 Embedding_dim =100	85.47	86	0 :94 1 :78 Total : 78.50	0 :92 1 :79 Total :79	0 :82 1 :90 Total : 89.09	0 :84 1 :89 Total : 89	0 :87 1 :83 Total : 83.23	0 :88 1 :83 Total :83
Epoch=30 Batch_size=50 blstm_dim=200 Embedding_dim =100	87,47	88	0 :96 1 :79 Total :80.03	0 :95 1 :79 Total :79	0 :83 1 :94 Total : 94.06	0 :84 1 :94 Total :94	0 :89 1 :86 Total :86.56	0 :84 1 :86 Total :86
Epoch=50 Batch_size=50 blstm_dim=200 Embedding_dim =100	**88.20**	87	0 : 95 1 : 80 Total : 82.20	0 :96 1 :78 Total :78	0 :84 1 :94 Total : 94.09	0 :82 1 :95 Total :95	0 :90 1 :86 Total : **86.69**	0 :89 1 :86 Total :86

Tableau 4: Résultat BLSTM sur dataset

❖ **Discussion :**

- Dans ce travail le F-score renvoyé par les réseaux de neurones (CNN : [0.7-0.82], LSTM : [0.83-0.86] et BiLstm : [0.8-0.86]) à marquer de bonnes performances même dans le cas du dataset déséquilibré.
- Les résultats après l'oversampling (SMOTE) sont meilleurs**.**

5.4.4 Résultat approche (Cnn+Lstm) :

Dans ce qui suit, nous présentons les résultats des tests effectués sur la combinaison CNN-LSTM avant est après Oversampling

❖ **Résulta Cnn+Lstm sur dataset Trolls original :**

	Accuracy%	Précision%	Recall%	F1_score%
Epoch =10 Batch_size =50 Embed_dim 100 Cnn_dim 32 Lstm 128 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	87.70	0 : 95 1 : 79 Total : 79.17	0 :84 1 :94 Total : 93.56	0 :89 1 :86 Total : 85.77

Epoch =10 Batch_size =50 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	88.02	0 :95 1 :76 Total :76.38	0 :81 1 :94 Total : 93.69	0 :88 1 :84 Total : 84.15
Epoch=30 Batch_size= 50 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	**90.07**	0 :95 1 :80 Total : 79.68	0 :84 1 :94 Total :93.81	0 :90 1 :86 Total : **86.17**
Epoch =50 Batch_size=50 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	87.70	0 :96 1 :77 Total : 77.05	0 :81 1 :95 Total : 94.24	0 :88 1 :85 Total : 85.19
Epoch 50 Batch_size 250 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	89.30	0 :96 1 :82 Total : 81.32	0 :86 1 :94 Total : 93.41	0 :91 1 :88 Total : 87.49
Epoch 20 Batch size 512 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3	87.65% --- --- ---	0 :95 1 :80 Total :79.56 _____ 0 :90 1 :76	0 :84 1 :93 Total :92.61 --- --- --- --- 0 :82	0 :89 1 :86 Total :85.59 --- -- -- -- 0 :86

Dropout 2 0.3 Optimizer adam, adamax,	83.77	Total :76.23	1 :86 Total :85.8	1 :81 Total :80.73

Tableau 5: Résulta Cnn+Lstm sur dataset Trolls original

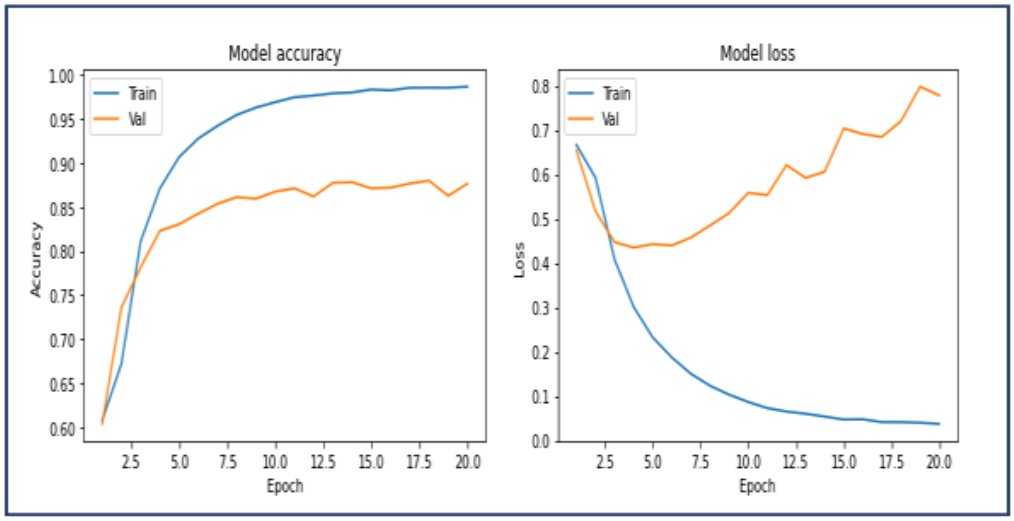


Figure 29 : L'accuracy et perte de validation (avec Adam)

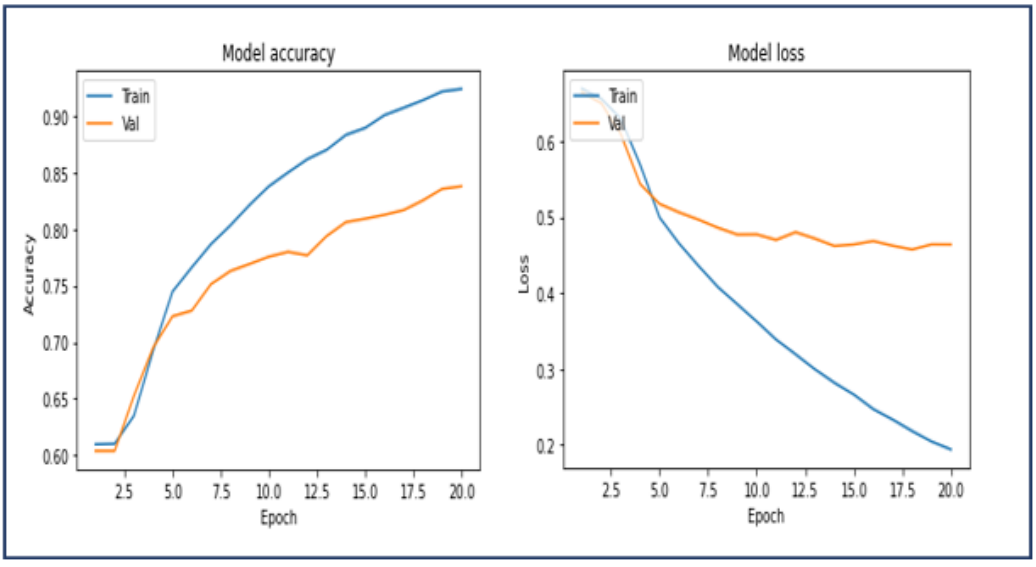


Figure 30 : L'accuracy et perte de validation (avec Adamax)

❖ **Résulta Cnn+Lstm sur dataset Trolls après oversampling (SMOTE) :**

	Acc%	Précision%	Recall%	F1_score%
Epoch =10 Batch_size =50 Embed_dim 100 Cnn_dim 32 Lstm 128 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	88	0 :96 1 :80 Total :79	0 :84 1 :95 Total :95	0 :89 1 :87 Total :86
Epoch =10 Batch_size =128 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	88.25	0 :96 1 :80 Total :80	0 :84 1 :94 Total :94	0 :90 1 :87 Total :86
Epoch=30 Batch_size= 50 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	88.25	0 :96 1 :80 Total :80	0 :84 1 :94 Total :94	0 :90 1 :87 Total :86
Epoch =50 Batch_size=50 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3	**89**	0 :96 1 :81 Total : 81	0 :85 1 :94 Total :94	0 :90 1 :87 Total : 87

Dropout 2 0.3 Optimizer adam				
Epoch 50 Batch_size 250 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	87	0 :98 1 :77 Total :77	0 :81 1 :95 Total :95	0 :88 1 :85 Total :85

Tableau 6: Résultat Cnn+Lstm sur dataset Trolls après oversampling (SMOTE)

5.4.5 Résultat approche (Cnn+BiLstm) :

Dans ce qui suit, nous présentons les résultats des tests effectués sur la combinaison Cnn-BiLstm avant est après Oversampling.

- **Résulta Cnn+BiLstm sur dataset Trolls original**

	Acc%	Précision%	Recall%	F1_score%
Epoch =10 Batch_size =50 Embed_dim 100 Cnn_dim 32 Lstm 128 Dropout1 0.3 Dropout 2 0.3 Optimizer adam, adamax	88.57 --- 81.50	0 :94 1 :82 Total :81.68 --- 0 :85 1 :76 Total :76	0 :87 1 :92 Total :91.73 --- 0 :84 1 :78 Total :77.7	0 :90 1 :86 Total :86.41 --- 0 :85 1 :77 Total :76
Epoch =10 Batch_size =128 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3	85.40	0 :92 1 :77 Total :77.45	0 :83 1 :89 Total :89.08	0 :87 1 :83 Total :82.86

Dropout 2 0.3 Optimizer adam				
Epoch=30 Batch_size= 50 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	88.27	0 :96 1 :81 Total :80.25	0 :85 1 :95 Total :94.07	0 :90 1 :87 Total :88.21
Epoch =50 Batch_size=50 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	**89**	0 :96 1 :81 Total :81	0 :85 1 :94 Total :93.41	0 :90 1 :87 Total :87.32
Epoch 20 Batch size 512 Embed_dim 100 Cnn_dim 32 Lstm 128 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	88.57	0 :95 1 :81 Total :81	0 :85 1 :94 Total :94	0 :90 1 :87 Total :87
Epoch 50 Batch_size 250 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	86.69	0 :96 1 :77 Total : 77.03	0 :81 1 :94 Total :93.56	0 :88 1 :85 Total :84.09

Tableau 7: Résulta Cnn+BiLstm sur dataset Trolls original

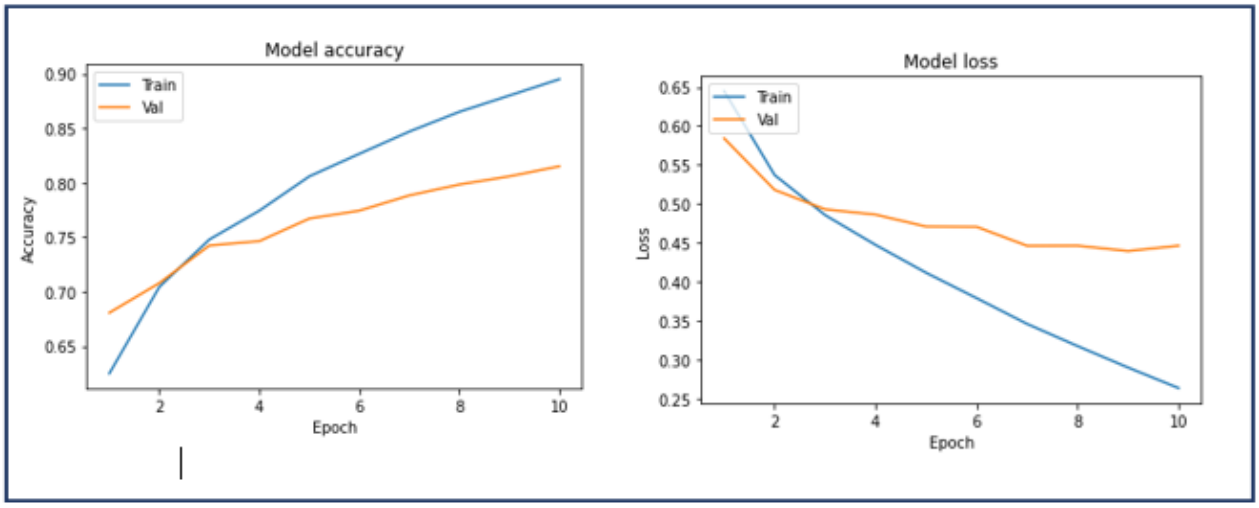


Figure 31 : L'accuracy et perte de validation (avec Adamax)

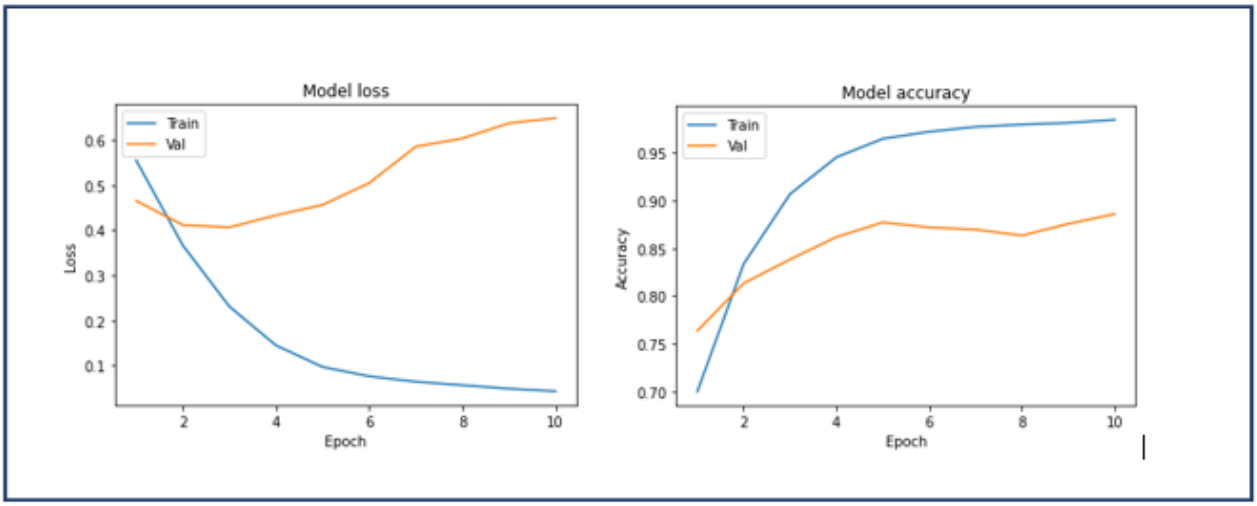


Figure 32 : L'accuracy et perte de validation (avec Adam)

- **Résultat Cnn+BiLstm sur dataset Trolls après oversampling (SMOTE) :**

	Acc%	Précision%	Recall%	F1_score%

Epoch =10 Batch_size =50 Embed_dim 100 Cnn_dim 32 Lstm 128 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	87	0 :95 1 :79 Total : 79	0 :83 1 :94 Total :94	0 :89 1 :86 Total : 86
Epoch =10 Batch_size =128 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	88	0 :95 1 :81 Total :81	0 :85 1 :94 Total :93	0 :90 1 :87 Total : 87
Epoch=30 Batch_size= 50 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	**89.58**	0 :96 1 :82 Total : 82	0 :86 1 :94 Total : 94	0 :91 1 :88 Total : **88**
Epoch =50 Batch_size=50 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	88	0 :96 1 :79 Total :78	0 :83 1 :95 Total :95	0 :89 1 :86 Total : 86

Epoch 50 Batch_size 250 Embed_dim 100 Cnn_dim 32 Lstm 200 Dropout1 0.3 Dropout 2 0.3 Optimizer adam	89.2	0 :96 1 :82 Total : 82	0 :86 1 :94 Total : 94	0 :91 1 :88 Total : 87

Tableau 8: Résultat Cnn+BiLstm sur dataset Trolls après oversampling (SMOTE)

5.4.6 Récapitulatif des Résultats :

Nous présentons un tableau comparatif des résultats pour les diffèrent algorithmes :

	F-Score %		Accuracy %	
	Original	SMOTE	Original	SMOTE
Cnn	**82.06**	81	**85.42**	82.58
Lstm	84.60	**86**	87.6	**88**
BiLstm	**86.69**	86.58	**88.20**	88
Cnn-Lstm	86.17	87	**90.07**	89
Cnn-BiLstm	87.32	**88**	89	**89.58**

Tableau 9: Récapitulation des meilleurs résultats avec corpus Trolls

La figure suivante montre une comparaison entre l'accuracy pour les différents algorithmes :

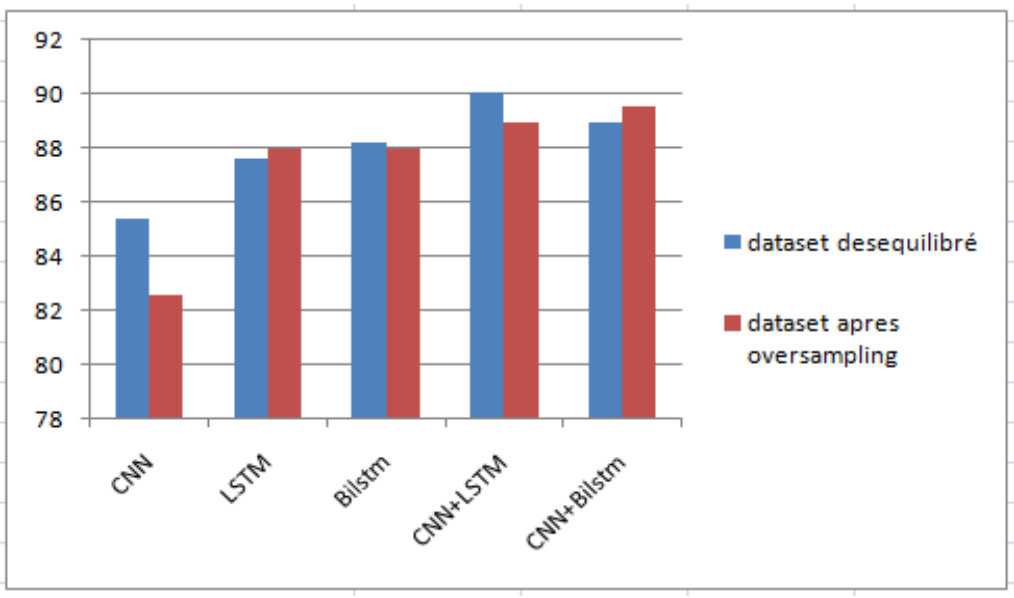


Figure 33 : Comparaison de l'accuracy pour les différentes approches (Trolls)

La figure suivante montre une comparaison entre Le F-score pour les différents algorithmes :

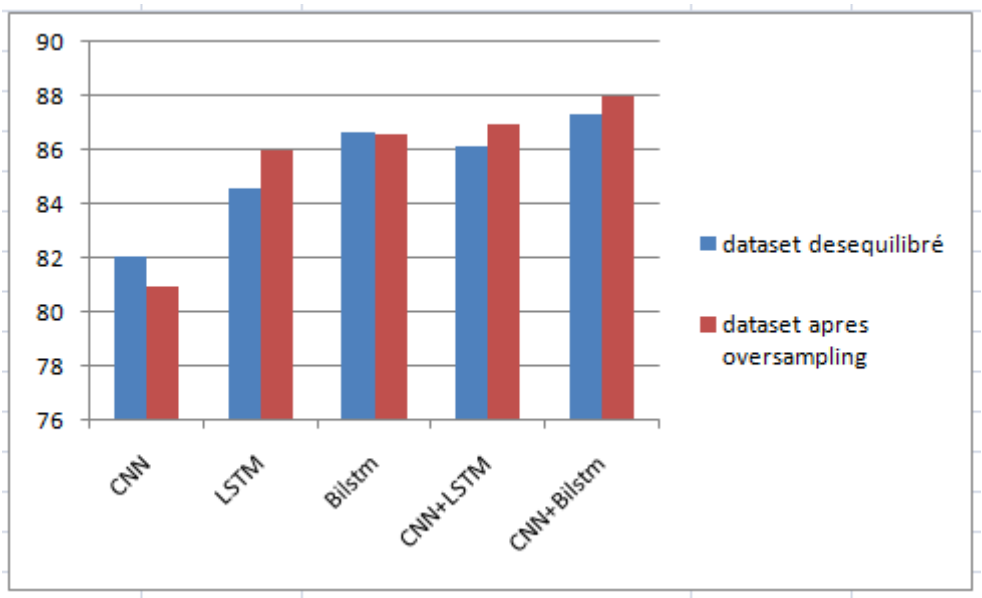


Figure 34 : Comparaison du F-Score pour les différentes approches (Trolls)

5.5 Analyse des résultats :

D'après les tableaux et les graphes précédents :

- La Combinaison Cnn-BiLstm a obtenu les meilleurs résultats pour la mesure accuracy
- Les scores d'accuracy des combinaisons CNN-LSTM et Cnn-BiLstm sont meilleurs que les algorithmes originaux implémenté séparément.
- Le meilleur score d'*accuracy* dans CNN, BLSTM et CNN-LSTM dans le cas du dataset déséquilibré dépasse légèrement celle avec un *Oversampling.*
- Par contre pour la combinaison Cnn-BiLstm le meilleur score d'*accuracy* est obtenu avec le dataset après Oversampling.
- Le meilleur résultat pour F-score LSTM, BLSTM, CNN-LSTM et CNN-BLSTM et dans le cas du dataset après Oversampling.
- Dans la plupart des tests, le dropout s'est avéré utile pour éviter l'*overfitting* et a présenté une amélioration d'accuracy pour la plupart des tests.
- Deux Optimizer ont été testé pour CNN+BLSTM et CNN+LSTM d'après les résultats, les Optimizer qui ont convergé le plus vite sont Adam suivi Adamax Cependant il n'y a pas eu un Optimizer meilleur qu'un autre.

5.6 Conclusion

Dans ce chapitre, nous avons effectué différents tests pratiques sur le dataset Trolls, ces tests ont été accomplie avant le sur-échantillonnage (Oversampling) et après.

La combinaison des deux algorithmes de classification CNN-LSTM à donner les meilleurs résultats sur le dataset.

Après la réalisation de la méthode d'Oversampling, les algorithmes de classifications ont pu améliorer les valeurs du F-score.

Conclusion générale

Dans cette étude nous nous sommes intéressées à l'analyse des publications dans les réseaux sociaux, nous avons réussi à mettre en œuvre un système d'apprentissage automatique pour la classification des tweets. Dans un premier temps, une phase de prétraitement a été appliquée afin de nettoyer les tweets des stops words et ponctuations pour ne laisser que les mots pertinents. En second lieu, on a implémenté une représentation vectorielle pour modéliser les tweets. Et enfin on a proposé deux approches basées sur une combinaison CNN-LSTM et CNN-BILSTM et comparée à d'autres approches de la littérature.

Le problème du corpus déséquilibré a été traité, de ce fait, nous avons appliqué une technique de sur-échantillonnage « *Oversampling* » pour rééquilibrer les classes.

Nous avons réalisé une étude comparative entre différents algorithmes de classification. Les performances de la combinaison CNN-LSTM et CNN-BLSTM proposée ont pu dépasser celles des algorithmes LSTM et le CNN et BLSTM, et ceci pour le dataset Trolls. Ainsi, CNN-LSTM à légèrement surpasser Cnn-BiLstm.

En dernier lieu, les résultats d'analyse des tweets avec les réseaux de neurones pourraient éventuellement être améliorés en utilisant un Automatic Machine Learning, le but serait d'apprendre au model, à régler les paramètres automatiquement ainsi obtenir le meilleur score possible.

Bibliographie

Acosta, J.M., Lamaute, N., Luo, M., Finkelstein, E., & Cotoranu, A. (2017). Sentiment Analysis of Twitter Messages Using Word 2 Vechttps://www.semanticscholar.org/paper/Sentiment-Analysis-of-Twitter-Messages-UsingWord-2-Acosta-Lamaute/784d1b2aebda3b80567bf8244e89499c31cf42a9 Consulté le 05-07-2021

Beverungen G., Kalita J. (2011). « Evaluating Methods for Summarizing Twitter Posts », Proceedings of the 5th AAAI ICWSM.

Brownlee, J, (2021). SMOTE for imbalanced classification with python, https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/Consulté le 13-09-2021

Cambria, E. (2016). Affective computing and sentiment analysis, IEEE Intell. Syst. 31 (2), pp. 102–107.

Cavazza, F, (2021). Panorama des média sociaux 20121. https://fredcavazza.net/2021/05/06/panorama-des-medias-sociaux-2021/. Consulté le 25-04-2021

Chaumartin, F, (2020). *Le traitement automatique des langues,* https://fr.wikipedia.org/wiki/Traitement_automatique_des_languesConsulté le 14-04-2021

Chavent, M. (2013). Classification automatique. Université de Bordeaux.

Chrislb (2005). Structure d'un neurone artificiel. https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels#/media/Fichier:ArtificialNeuronModel_francais.png .Consulté le 05-08-2021

Dasgupta, S., Kumar, A., Das, D., Naskar, S.K., & Bandyopadhyay, S. (2016). Word Embeddings for Information Extraction from Tweets. FIRE.

Deloche, F. (2017). A diagram for a one-unit recurrent neural network (RNN). From bottom to top: input state, hidden state, output state. U, V, W are the weights of the network. Compressed diagram on the left and the unfold version of it on the right. https://upload.wikimedia.org/wikipedia/commons/b/b5/Recurrent_neural_network_unfold.svg. Consulté le 10-08-2021

Donadio, J. (2018). Intelligence artificielle RAPPORT THEORIQUE ET PRATIQUE.

Donges, N, (2021). Understanding Recurrent Neural Networks and Lstm Networks,

Dos Santos, C.; Gatti, M. Deep convolutional neural networks for sentiment analysis of short texts. In Proceedings of the COLING 2014, the 25th International Conference on Computational Linguistics.

Engelin, M., & amp ; De Silva, F. (2016). Troll detection: A comparative study in detecting troll farms on Twitter using cluster analysis (Unpublished Dissertation). KTH, School of Computer Science and Communication, Sweden.

François, Y, (2010). Une petite introduction au Traitement Automatique des Langues Naturelles, https://perso.limsi.fr/anne/coursM2R/intro.pdfConsulté le 19-04-2021

Friedemann, V. (2015). Clustering a customer base using Twitter data. *CS-229*.
Gao, D., Li, W., Cai, X., Zhang, R., and Ouyang, Y. (2014). *Sequential summarization: A full view of twitter trending topics*.

Gary, B, (2020). Convolutional neural network, https://datascientest.com/convolutional-neural-networkConsulté le 19-05-2021

Hardaker, C, (2010). Trolling in asynchronous computer-mediated communication: From user discussions to academic definitions, https://www.researchgate.net/publication/263568332_Trolling_in_asynchronous_computer_mediated_communication_From_user_discussions_to_academic_definitions. Consulté le 05-06-2021

Gary, B, (2020). NLP Twitter-Analyse de sentiment. https://datascientest.com/nlp-twitter-analyse-de-sentiment. Consulté le 17-04-2020

Haochen, W, (2020). Fonction de perte et d'activation. https://atcold.github.io/pytorch-Deep-Learning/fr/week11/11-1/. Consulté le 28-07-2021

HRcommons, (2009). Perceptron multicouche. https://fr.wikipedia.org/wiki/Perceptron_multicouche#/media/Fichier:Perceptron_4layers.png. Consulté le 05-06-2021

Niklas, D, (2021). Recurrent neural networks and Lstm https://builtin.com/data-science/recurrent-neural-networks-and-lstmConsulté le 25-07-2021

Imran, M., & Srinivasa, V. (2018). A Novel Technique on Class Imbalance Big Data using Analogous under Sampling Approach. *International Journal of Computer Applications*, *179*(33),

Islam, M. A. (2020). Reduced Dataset Neural Network Model for Manuscript Character Recognition.

K. Dave, S. Lawrence, and D. M. Pennock, Mining the peanut gallery: Opinion extraction and semantic classification of product reviews, In Proceedings of the 12th international conference on World Wide Web (pp. 519-528).

kantrowitz, M., Mohit, B., and Mittal, V. (2000)" Stemming and Its Effects on TFIDF Ranking", In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 357–359, Athens, Greece.

Kaufman, L., Hopke, P. K., & Rousseeuw, P. J. (1987). *Using a parallel computer system for statistical resampling methods*. Technische Universiteit Delft.

Kim, Y. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*; Association for Computational Linguistics: Doha,

Kingma, D.P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Korenius, T., J. Laurikkala, K. Jarvelin, et M. Juhola (2004). Stemming and lemmatization in the clustering of finnish text documents. In CIKM '04: Proceedings of the thirteenth ACM conference on Information and knowledge management, New York, NY, USA, pp. 625-633. ACM Press.

Kulshrestha, R, (2019). Word2vec-Skipgram and CBow, https://towardsdatascience.com/nlp-101-word2vec-skip-gram-and-cbow-93512ee24314Consulté le 11-07-2021

Landauer, T. K., Foltz, P. W., &Laham, D. (1998). An introduction to latent semantic analysis, https://www.tandfonline.com/doi/abs/10.1080/01638539809545028. Consulté le 07-06-2021

Laudet, F, (2019). Community management et réseaux sociaux, https://www.vu-du-web.com/community-management/reseaux-sociaux/grands-reseaux-sociaux/. Consulté le 14-05-2021.

Lee, K., Palsetia, D., Narayanan, R., Patwary, Md. M. A., Agrawal, A., &Choudhary, A. (2011). Twitter Trending Topic Classification. 2011 IEEE 11th International Conference on Data Mining Workshops.

Liddy, E, (2015). Scholarship 2 1 Natural Language Processing, https://surface.syr.edu/cgi/viewcontent.cgi?article=1019&context=cnlpConsulté le 19-04-2021

Lin Miao, Mark Last, Marina Litvak, (2020). Detecting Troll Tweets in a Bilingual Corpus

Lökk, A., & Hallman, J. (2016). Viability of sentiment analysis for troll detection on twitter: A comparative study between the naive bayes and maximum entropy algorithm

Margot, P, (2021). Perceptron : qu'est-ce que c'est et à quoi ça sert ? https://datascientest.com/perceptronConsulté le 20-08-2021

Mihaylov, T.; Georgiev, G.; Nakov, P, (2015). Finding opinion manipulation trolls in news community forums. https://aclanthology.org/K15-1032/. Consulté le 20-06-2021

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. 2013. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS 2013.

Minsky-Papert. (*1969*). Perceptron: the artificial neuron. https://towardsdatascience.com/perceptron-the-artificial-neuron-4d8c70d5cc8d

Pang, B. & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(1-2), 1-135.

Pang, Bo and Lee, Lillian and Vaithyanathan, Shivakumar, Thumbs up? Sentiment Classification using Machine Learning Techniques, Proceedings of EMNLP 2002

Paul, G, (2007). Média social. https://fr.wikipedia.org/wiki/M%C3%A9dia_social.

Pennington, J., Socher, R., & Manning, C.D. (2014). Glove: Global Vectors for Word Representation. EMNLP.

Petercour, D, (2019). Machine Learning Classification vs Regression.https://dev.to/petercour/machine-learning-classification-vs-regression-1gn . Consulté le 25-04-2021

Petitjean, G. (2006). INTRODUCTION AUX RESEAUX DE NEURONES. https://www.lrde.epita.fr/~sigoure/cours_ReseauxNeurones.pdf.

Purwitasari, D., Fatichah, C., Arieshanti, I., &Hayatin, N. (2015, September). K-medoids algorithm on Indonesian Twitter feeds for clustering trending issue as important terms in news summarization. In *2015 International Conference on Information & Communication Technology and Systems (ICTS)* (pp. 95-98). IEEE

Río, S.D., López, V., Benítez, J.M., & Herrera, F. (2014). On the use of MapReduce for imbalanced big data using Random Forest.

Rouissi, J. (2001). Les effets de réseau en bibliothèques : pour une meilleure prise en compte des coûts et avantages qualitatifs de la coopération. Tiré de http://theses.univ-lyon2.fr/documents/getpart.php?id=lyon2.2001.rouissi_j&part=178747

Saimadhu, P, (2020). Oversampling and Undersampling. https://dataaspirant.com/10-oversampling/?fbclid=IwAR01LcVExR_oBDKY9lZdyPa_akdNpkzdef5nw97PH5PZf1C2yU5UPa24shU . Consulté le 20-07-2021

Salton, G., McGill, M. J. (1983) Introduction to Modern Information Retrieval. In *Encyclopedia of computer science* (pp. 858-863).

Shen, Q.; Wang, Z.; Sun, Y. Sentiment analysis of movie reviews based on CNN-BiLstm. In *International Conference on Intelligence Science*;

Steinberger, J., Ježek, K. (2004). Using Latent Semantic Analysis in Text Summarization and Summary Evaluation. https://www.researchgate.net/publication/220017752_Using_Latent_Semantic_Analysis_in_Text_Summarization_and_Summary_Evaluation

Steinman, A, (2020). Troll sur internet, https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445206-troll-sur-internet-definition-et-conseils-pratiques-pour-bien-reagir/. Consulté le 14-05-2021

Tanguy, L, (1997), Traitement automatique de la langue. https://fr.wikipedia.org/wiki/Traitement_automatique_des_langues

Bigdata, T, (2021), Précision et rappel.
https://fr.wikipedia.org/wiki/Pr%C3%A9cision_et_rappel

Triguero, I., Galar, M., Bustince, H., & Herrera, F. (2017). A first attempt on global evolutionary Undersampling for imbalanced big data. *2017 IEEE Congress on Evolutionary Computation (CEC)*.

valuations, e. (2015). a review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, *5*(2), 1.

Vandeginste, P, (2020). Word Embedding, https://dataanalyticspost.com/Lexique/word-embedding/. Consulté le 15-05-2021

Wang, X.; Jiang, W.; Luo, Z. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics.

Wiliam, S, (2019). Tf-idf from scratch in python on real world dataset, https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089 . Consulté le 05-04-2021

Yenter, A.; Verma, A. Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis. In Proceedings of the 2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), New York, NY, USA, 19–21 October 2017; pp. 540–546.

Yoon, J.; Kim, H. Multi-Channel Lexicon Integrated Cnn-BiLstm Models for Sentiment Analysis. In Proceedings of the 29th Conference on Computational Linguistics and Speech Processing (ROCLING 2017).

Zhao, Y. (2011). R and Data Mining: Examples and Case Studies. Academic Press.

Zhou, P.; Qi, Z.; Zheng, S.; Xu, J.; Bao, H.; Xu, B. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling.