République Algérienne Démocratique et Populaire

Ministère d'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed El Bachir El Ibrahimi de Bordj Bou Arréridj

Faculté des Mathématiques et d'Informatique

Département d'informatique

Spécialité : Ingénierie de L'informatique Décisionnelle

Mémoire de fin d'étude

Présenté en vue de l'obtention du diplôme de

Master en Informatique

Thème:

CATEGORISATION DES TEXTES PAR DEEP LEARNING

Présenté par :

- **BEN DIB ASSIA**
- **MEBARKI OUSSAMA**

Encadré par :

Dr. BELAZZOUG MOUHOUB

Promotion : Septembre 2022

Jury de soutenance :

Président : ***BEGHOURA M. AMINE***

Encadreur : ***BELAZOUG MOUHOUB***

Examinateur : ***SAIFI ABDELHAMID***

2022-2023


بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Remerciements

Louange à Allah, le miséricordieux, sans lui rien de tout cela n'aurait pu être, je remercie le bon Dieu qui nous a orienté au chemin de savoir et les portes de la science.

Nous tenons à adresse nos plus vifs Remerciements à : ***Dr. BELAZZOUG MOUHOUB.*** *Pour nous avoir encadré, guidé et prodigué tout l'aide nécessaire pour la réalisation de notre travail.*

Je remercie très vivement le président et les membres de jury qui ont accepté de juger ce travail.

Merci pour tous ceux qui, m'ont aidé de près ou de loin à réaliser ce travail.

Dédicaces

Je dédie ce modeste travail tout d'abord

À mes très chers parents, Pour

Leur patience, leur amour, leur soutien et leurs

Encouragements pendent toutes mes années d'études.

Que Dieu, le tout puissant, vous préserve et vous

Procure santé et longue vie

À mes adorables sœurs et mon cher frère

Merci pour m'avoir toujours supporté Je vous aime beaucoup.

Á toute ma grande famille de loin ou de proche

Á tous mes meilleurs amis

Á tous les gens qui m'ont aidé pour faire ce travail.

Et à tous ceux qui j'ai oublié de mentionner !!!

BEN DIB ASSIA

Dédicaces

Je tiens à remercier en premier lieu mes parents

A mon symbole de sacrifice, écoles de mon enfance qui était mon ombre durant toutes mes

Années d'études, toi mon cher père qui a tant investi pour assurer mon Avenir.

À la source d'amour et tendresse à celle qui a tout donné à toi ma chère maman.

A mes frères Hicham, Hamza, Amine A ma sœur Yasmine

À tous les membres de ma famille

Spécialement A celle avec qui j'ai partagé ce travail au cours de cette année, à toi ASSIA

A tous mes collègues de deuxième année master. Et à tous

Mebarki Oussama

Résumé

Comme la plupart des informations (plus de 80 %) sont stockées sous forme de texte, on pense que le text mining a une valeur commerciale élevée. Les connaissances peuvent être découvertes à partir de nombreuses sources d'information Pourtant, les textes non structurés restent la plus grande source de connaissances facilement accessible.

Classification de texte qui classe les documents selon des catégories prédéfinies. Dans cette thèse, nous essayons d'appliquons l'un des model performant d'apprentissage profond qui est le réseau de neurone (RNA) sur un jeu de donné textuel pour résoudre les problèmes de classification des textes.

Mots clés: text mining, Machine Learning, réseau de neurone, d'apprentissage profond

Abstract

Since most information (over 80%) is stored as text, text mining is believed to have high commercial value. Knowledge can be discovered from many sources of information. Yet, unstructured texts remain the greatest easily accessible source of knowledge.

Text classification that classifies documents according to predefined categories. In this thesis, we try to apply one of the powerful Deep Learning models, which is the neural network (RNA) on a textual data set to solve text classification problems.

Keywords: text mining, machine Learning, neural network, deep Learning

الملخص

نظرًا لأنه يتم تخزين معظم المعلومات (أكثر من 80٪) كنص، يُعتقد أن التنقيب عن النص له قيمة تجارية عالية. يمكن اكتشاف المعرفة من العديد من مصادر المعلومات، ومع ذلك، تظل النصوص غير المهيكلة أكبر مصدر للمعرفة يسهل الوصول إليه.

تصنيف النص الذي يصنف المستندات وفقًا للفئات المحددة مسبقًا. في هذه الأطروحة، نحاول تطبيق أحد نماذج التعلم العميق القوية وهي الشبكة العصبية (RNA) على مجموعة بيانات نصية لحل مشاكل تصنيف النص.

الكلمات المفتاحية: تحليل النصوص، التعلم الآلي، الشبكة العصبية، التعلم العميق

Table des matières

Liste des acronymes

TAL : Traitement Automatique de Langage

IA : Intelligence Artificielle

DL: Deep Learning.

KPPV : Méthode des k plus proches voisins

ML: Machine Learning

RNN: Recurrent Neural Network.

SVM: Support Vector Machine.

KNN: K Nearest Neighbor.

NB: Naïve Bayes.

ANN: Artificial Neural Network.

TC : Text Classification.

RNA : Les réseaux de neurones

CART : Classification and Regression Trees

LSTM : Long short-term memory

CNN : Convolutional Neural Network

IMDB : Internet Movie Database

IA : intelligence artificielle

FS : feature sélection

Liste des figures

Liste des tableaux

INTRODUCTION GENERALE

Introduction générale

La communication est un processus nécessaire pour l'être humain. Cette communication peut être orale (parole) ou écrite. Actuellement, l'information peut avoir comme support trois média de base : le texte, le son et l'image. Notre travail se focalisera sur les textes, ainsi le terme « document » induit directement qu'il s'agit d'un document textuel.

La classification de texte est l'une des tâches les plus courantes du TAL qui permet à un programme de classer des documents en texte libre en fonction de classes prédéfinies. Les cours peuvent être basés sur un sujet, un genre ou un sentiment. L'émergence actuelle de documents numériques volumineux rend la tâche de classification des textes plus cruciale, en particulier pour les entreprises afin de maximiser leur flux de travail ou même leurs profits.

L'apprentissage automatique (Machine Learning en anglais) est le concept selon lequel un programme informatique peut apprendre et s'adapter à de nouvelles données sans intervention humaine. L'apprentissage machine est un domaine de l'intelligence artificielle (IA) qui maintient les algorithmes intégrés d'un ordinateur à jour, indépendamment des changements de l'économie mondiale.

Classification de texte qui classe les documents selon des catégories prédéfinies. Dans cette thèse, nous essayons d'appliquons l'un des model performant d'apprentissage profond qui est le réseau de neurone (RNA) sur un jeu de donné textuel pour résoudre les problèmes de classification des textes .

CHAPITRE I

Classification des textes

I.1. Introduction :

Pendant les dernières années, la production de documents sous forme numérique a explosé, en raison de la disponibilité accrue d'outils matériels et logiciels pour générer des données numériques et pour numériser des données qui avaient été créées sous une forme non numérique. Ce phénomène a également fortement affecté les nouveaux médias numériques tels que l'imagerie, la vidéo, la musique, etc. Cependant, le texte en langage naturel a été le support le plus responsable de cette explosion, en raison de l'omniprésence des outils de traitement de texte et de création de texte.

En conséquence, il existe un besoin accru de solutions matérielles et logicielles pour stocker, organiser et récupérer les grandes quantités de texte numérique qui sont produites.

I.2. Définition de la Catégorisation de textes :

Plusieurs définitions de la C.T ont vus le jour depuis son apparition, nous citons dans ce contexte les définitions suivantes :

- **Définition1**: La CT est une relation bijective qui consiste à "chercher une liaison fonctionnelle entre un ensemble de textes et un ensemble de catégories (étiquettes, classes) ». [**01]**
- **Définition2** : La Catégorisation de Textes (C.T) est le processus qui consiste à assigner une ou plusieurs catégories parmi une liste prédéfinie à un document. L'objectif du processus est d'être capable d'effectuer automatiquement les classes d'un ensemble de nouveaux textes. **[02]**
- **Définition formelle**: Formellement, la catégorisation de texte consiste à associer une valeur booléenne à chaque paire $\mathbf{(d_j, c_i) \in D \times C}$ est l'ensemble des textes et C'est l'ensemble des catégories selon que $d_j \in c_i$, ou non. Le but de la catégorisation de texte est de construire une procédure (modèle, classifieur)

  $\mathbf{\Phi : D \times C \to B}$ qui associe une ou plusieurs étiquettes (catégories) à un document d_j avec la fonction **F : D (C**, la vraie fonction qui retourne pour chaque vecteur dj d'une valeur ci. **[03]**

  F. le processus qui consiste à associer une valeur booléenne à chaque paire (dj, ci)

  D × C, où D est l'ensemble des textes et C'est l'ensemble des catégories.

La valeur V (Vrai) est alors associée au couple (dj, ci) si le texte dj appartient à la classe ci tandis que la valeur F (Faux).

I.3.Applications de la catégorisation de texte

La catégorisation de textes est utilisée dans de nombreuses applications. Parmi ces domaines figurent: l'identification de la langue, la reconnaissance d'écrivains et la catégorisation de documents multimédia, et bien d'autres.

La catégorisation de textes peut être une fin en soi, par exemple lors de l'étiquetage de documents, ou bien représenter une étape dans la représentation et le traitement de l'information contenue dans les textes.

3.1. Catégorisation de textes : une fin en soi:

L'indexation automatique de textes consiste à associer à chaque texte d'une collection un ou plusieurs termes parmi un ensemble prédéfini. L'objectif est de décrire le contenu de ces textes par des mots ou des phrases clés qui font partie d'un ensemble de vocabulaire contrôlé. Dans un tel contexte, si nous regardons ce vocabulaire contrôlé comme des catégories, l'indexation de textes peut être alors vue comme une forme de catégorisation de textes **[04]**

3.2. Catégorisation de textes: un support pour différentes applications

La catégorisation de textes peut être un support pour différentes applications parmi lesquelles le filtrage, consistant à déterminer si un document est pertinent ou non (décision binaire), et le routage, consistant à affecter un document à une ou plusieurs catégories parmi n. Un exemple de l'utilisation de la catégorisation de texte pour le filtrage est la détection de spams (les courriers indésirables) pour ensuite les supprimer **[05].** Un exemple de routage est la diffusion sélective d'information. Lors de la réception d'un document l'outil choisit à quelles personnes le faire parvenir en fonction de leurs centres d'intérêt. Ces centres d'intérêt correspondent à des profils individuels **[06]**. Notons que si la sélection est faite au niveau du producteur de l'information (e.g., l'agence de presse), le système doit « diriger » l'information au consommateur intéressé (e.g., le journal) et on parle alors de routage, tandis que si la sélection est faite au niveau de consommateur de l'information, comme c'est le cas lors de la sélection de l'information pour un même utilisateur, on parle alors de filtrage. **[07]** observe qu'une confusion entre filtrage et routage existe chez certains auteurs.

I.4.Le processus général de catégorisation de textes :

Le processus de catégorisation se déroule de façon générale selon les trois phases suivantes : apprentissage, validation et test (évaluation).

4.1. La phase d'apprentissage: Qui comprend plusieurs étapes et aboutit à un modèle de prédiction:

a) nous disposons d'un ensemble de textes étiquetés (pour chaque texte nous connaissons sa catégorie) ;

b) à partir de ce corpus, nous extrayons les k descripteurs (ou mots, ou termes) (t1;..;tk) les plus pertinents au sens du problème à résoudre ;

c) nous disposons alors d'un tableau « descripteurs × individus », et pour chaque texte nous connaissons la valeur de ses descripteurs et son étiquette ;

d) nous appliquons un algorithme d'apprentissage sur ce tableau afin d'obtenir un modèle de prédiction Φ.

4.2. La phase de validation: La phase de validation a pour objectif d'ajuster automatiquement les paramètres nécessaires aux algorithmes de la phase précédente. En effet, la majorité des algorithmes d'apprentissage (classificateurs) dépend d'un ensemble de paramètres, et ce qui est nécessaire de les fixer correctement.

4.3 Le test: La phase de test permet d'évaluer, de façon définitive, le processus de catégorisation grâce au jeu de test. Le jeu de test, comme le jeu de validation, est un ensemble de couples (document, catégorie) dont l'information associée à la catégorie n'est utilisée que lors de l'évaluation du système. Comme le sens d'un document est une notion subjective, l'évaluation d'un processus de catégorisation s'effectue empiriquement sur un jeu de test. Une fois le jeu de test catégorisé, les affectations fournies par le système sont comparées celles qui sont préalablement définies par un expert. En conclusion, l'objectif d'un processus de catégorisation est donc de maximiser une fonction de score.

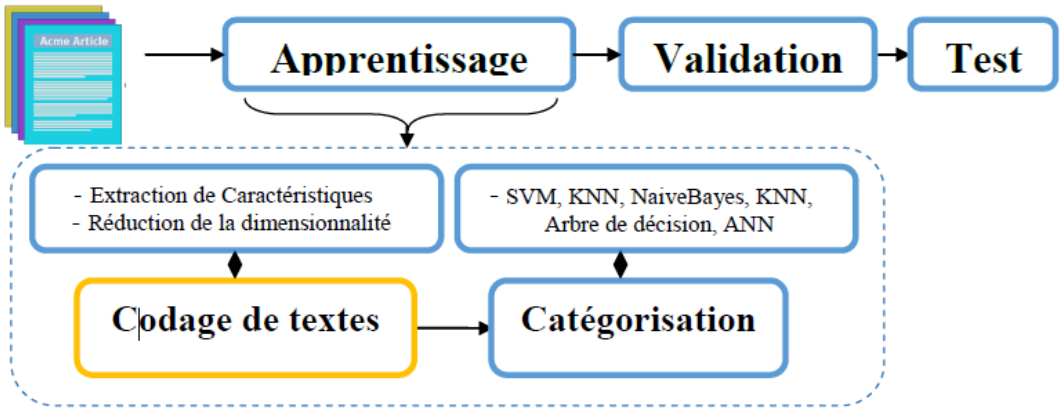


Figure 1. Le processus de Catégorisation de textes

I.5. Problèmes de la catégorisation de textes :

Dans ce qui suit nous allons signaler les dix principales difficultés qui s'opposent à la catégorisation de textes :

5.1. Redondance (Synonymie) :

La redondance et la synonymie permettent d'exprimer le même concept par des expressions différentes, plusieurs façons d'exprimer la même chose. Cette difficulté est liée à la nature des documents traités exprimés en langage naturel contrairement aux données numériques **[08]**

5.2- Polysémie (Ambiguïté) :

A la différence des données numériques, les données textuelles sont sémantiquement riches, du fait qui sont conçues et raisonnées par la pensée humaine. Contrairement des langages informatiques, le langage naturel, autorise des violations des règles grammaticales engendrant plusieurs interprétations d'un même propos. Un mot possède, dans différents cas, plus d'un sens et plusieurs définitions lui sont associées.

5.3. Grandes dimensions :

Le modèle vectoriel proposé par **[09]** consiste en la représentation de chaque texte par un vecteur dont les composants sont les termes constituant le vocabulaire du corpus. Or, pour un corpus de taille raisonnable, le tableau « termes × textes » peut avoir des centaines de milliers de lignes (textes) et des milliers de colonnes (termes) ; mais ce tableau est souvent creux, c'est-à-dire que la majorité de ses cellules sont vides. Ceci affecte le processus

d'apprentissage en rendant certains algorithmes inopérants et en augmentant le risque de sur-apprentissage.

5.4. Complexité de l'algorithme :

Dans la catégorisation de textes, la grande dimensionnalité peut réduire l'efficacité des algorithmes d'apprentissage. En effet, la plupart des algorithmes d'apprentissage sophistiqués, comme les arbres de décision, le k-PPV, et là, sont sensibles au |T |, le nombres des variables utilisées pour coder les textes, car |T | est un paramètre de la complexité de l'algorithme. C'est pourquoi une méthode de réduction de dimension doit être utilisée avant d'estimer les paramètres d'un classifieur**. [10]**

5.4.1. Sur-apprentissage :

Le sur-apprentissage s'explique par le fait que le modèle de prédiction n'arrive pas à bien classer les nouveaux textes, pourtant il l'a bien fait dans la phase d'apprentissage en classant correctement les textes de la base d'apprentissage.

Pour limiter le sur-apprentissage, on doit sélectionner des termes pour réduire la dimensionnalité. D'après les expériences antérieures, le nombre de termes doit être limité par rapport au nombre de textes de la base d'apprentissage**. [11]**

5.4.2. Subjectivité de la décision :

Contrairement, à d'autres situations où l'appartenance à une classe est objective, l'attribution d'une catégorie à un texte est subjective

En effet, la catégorie est attribuée en fonction du contenu sémantique de ce texte, qui est une notion subjective, et dépend du jugement d'un expert**.**

I.6. Classification Supervisée Vs Classification non Supervisée :

La principale différence entre la Classification Supervisée et Classification non Supervisée est que la classification est une approche d'apprentissage supervisé dans laquelle une étiquette spécifique est fournie à la machine pour classer les nouvelles observations. Ici, la machine a besoin de tests et d'une formation appropriée pour la vérification de l'étiquette. Ainsi, la classification est un processus plus complexe que le regroupement. D'autre part, le clustering est une approche d'apprentissage non supervisé où le regroupement est effectué sur

la base de similitudes. Ici, la machine apprend à partir des données existantes et n'a besoin d'aucune formation.

I.7. Introduction à L'Apprentissage Automatique (*en anglais : Machine Learning*) :

La machine Learning (ML) est une forme d'intelligence artificielle (IA) qui est axée sur la création de systèmes qui apprennent, ou améliorent leurs performances, en fonction des données qu'ils traitent.

I.7.1 Les types d'apprentissage Automatique (ML)

7.1.1-classification supervisée :

La classification dite supervisée lorsque les données qui entrent dans le processus sont déjà catégorisées et que les algorithmes doivent s'en servir pour prédire un résultat.

Plusieurs techniques sont utilisées. On peut citer Naïve bayes, Machine a vecteur de support, K voisins Proches, Arbre de Décision …

7.1.2-classification non-supervisée :

L'apprentissage non-supervisé est un processus de division d'un ensemble de données en un ensemble de sous-classes significatives, appelées clusters. Le clustering est identique à la classification dans laquelle les données sont regroupées. Cependant, contrairement à la classification, les groupes ne sont pas prédéfinis. Au lieu de cela, le regroupement est réalisé en déterminant les similitudes entre les données en fonction des caractéristiques trouvées dans les données réelles. Les groupes sont appelés Clusters.

I.7.2. Les Algorithmes d'Apprentissage automatique (ML) :

Dans ce qui suit une liste des différents algorithmes d'apprentissage modèles :

- L'algorithme des K plus proches voisins (ou K-NN).
- Les arbres de décision.
- Machines à support de vecteurs (ou SVM)
- Les réseaux de neurones (RNA).
- L'algorithme de Naïve Bayes.

7.2.1. Algorithme des k-voisins les plus proches KNN :

1. **Définition :**

L'algorithme des k plus proches voisins s'écrit en abrégé k-NN ou KNN, de l'anglais k-nearest neighbors. Est un algorithme d'apprentissage supervisé. Ses performances la situent parmi les meilleures méthodes de catégorisation. L'idée de K-plus proches voisins est de représenter chaque texte dans un espace vectoriel, dont chacun des axes représente un élément textuel. **[12]**

2. **Principes de fonctionnement :** L'algorithme de catégorisation de K-plus proches voisins pris de **[01],** est le suivant :

Paramètre : le nombre K de voisins

Contexte : un échantillon de L textes classés en C = c1, c2, ..., cn classes

Début

Pour chaque texte T faire

Transformer le texte T en vecteur T = (x1, x2, ..., xm),

Déterminer les K plus proches textes du texte T selon

Une métrique de distance,

Combiner les classes de ces K exemples en une classe C

Fin pour

Fin

Sortie : le texte T associé à la classe C.

La distance entre un texte et ses voisins se fait via une métrique de distance. Cette métrique peut être comme suit :

- **Mesure Cosinus**:

La mesure qui consiste à calculer le produit scalaire entre deux vecteurs a et b, que nous divisons par le produit de la norme de ces deux vecteurs. La formule de la mesure Cosinus est alors la suivante :

$$\textbf{Cosinus (a, b)} = \frac{\sum (a \times b)}{\sqrt{\sum a^2 \times \sum b^2}}$$

D'autres mesures ont été proposées dans la littérature, parmi lesquelles on peut citer les mesures de Jaccard et Dice.

- **Mesure de Jaccard**

La formule de la mesure de Jaccard est alors la suivante :

$$\textbf{J (a, b)} = \frac{\sum (a \times b)}{\sum a^2 + \sum b^2 - \sum ab}$$

- **Mesure de Dice**

La formule de la mesure de Dice est alors la suivante :

$$\textbf{D (a, b)} = 2 \times \frac{\sum (a \times b)}{\sum (a^2 + b^2)}$$

7.2.2. Les arbres de décision :

1. Définition :

C'est une des méthodes d'apprentissage supervisé les plus populaires pour les problèmes de classification de données. Ils emploient une représentation hiérarchique de la structure des données sous forme des séquences de décisions (tests) en vue de la prédiction d'un résultat ou d'une classe. Chaque individu (ou observation), qui doit être attribué(e) à une classe, est décrit(e) par un ensemble de variables qui sont testées dans les nœuds de l'arbre. Les tests s'effectuent dans les nœuds internes et les décisions sont prise dans les nœuds feuille.

En théorie des graphes : un arbre est un graphe non orienté, acyclique et connexe. L'ensemble des nœuds se divise en trois catégories :

- Nœud *racine* (l'accès à l'arbre se fait par ce nœud),
- Nœuds *internes* : les nœuds qui ont des descendants (ou *enfants*), qui sont à leur tour des nœuds,

- Nœuds *terminaux* (ou *feuilles*) : nœuds qui n'ont pas de descendant**.**

2. L'algorithme général de création d'un arbre de décision :

1. Déterminer la meilleure caractéristique dans l'ensemble de données d'entrainement.
2. Diviser les données d'entrainement en sous-ensembles contenant les valeurs possibles de la meilleure caractéristique.
3. Générez de manière récursive de nouveaux arbres de décision en utilisant les sous-ensembles de données créés.
4. Lorsqu'on ne peut plus classifier les données, on s'arrête.

Il existe d'autres algorithmes automatiques pour construire les arbres de décision :

- **ID3** (Iterative Dichotomiser 3): développé en 1986 par Ross Quinlan **[13]** Il peut être appliqué seulement sur les caractéristiques nominales. Il est utilisé pour le classement.
- **C4.5**: une extension de ID3 par Ross Quinlan. Il peut être appliqué sur tous les types de caractéristiques. Il est utilisé pour le classement.
- **C5.0**: une extension commerciale de C4.5, toujours par Ross Quinlan.
- **CART** (Classification and Regression Trees): comme C4.5 mais utilise d'autres métriques. Aussi, l'algorithme supporte la régression**.**

7.2.3. Machines à support de vecteurs (ou SVM) :

Les SVMs sont une famille d'algorithmes d'apprentissage automatique qui permettent de résoudre des problèmes tant de classification que de régression ou de détection d'anomalie. Ils sont connus pour leurs solides garanties théoriques, leur grande flexibilité ainsi que leur simplicité d'utilisation même sans grande connaissance de data mining.

Quelques notions de base des SVM :

- **Hyperplan :** est un séparateur d'objets des classes. De cette notion, nous pouvons dire qu'il est évident de trouver une mainte d'hyperplans mais la propriété délicate des SVM est d'avoir l'hyperplan dont la distance minimale aux exemples d'apprentissage est maximale, cet hyperplan est appelé L'hyperplan optimal, et la distance appelée marge.

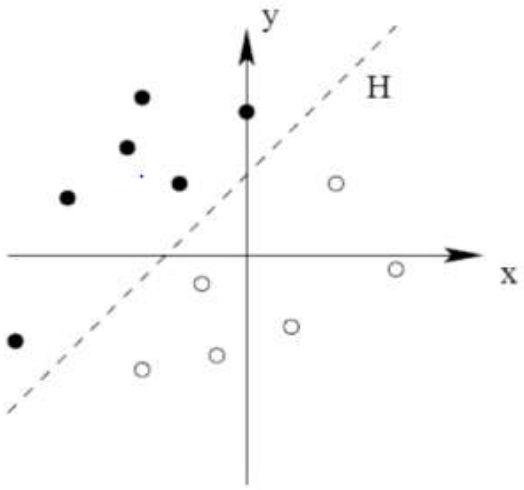


Figure.2 : la séparation du l'hyper plan par les SVM

- **Vecteurs Support :** ce sont les points qui déterminent l'hyperplan tels qu'ils soient les plus proches de ce dernier.

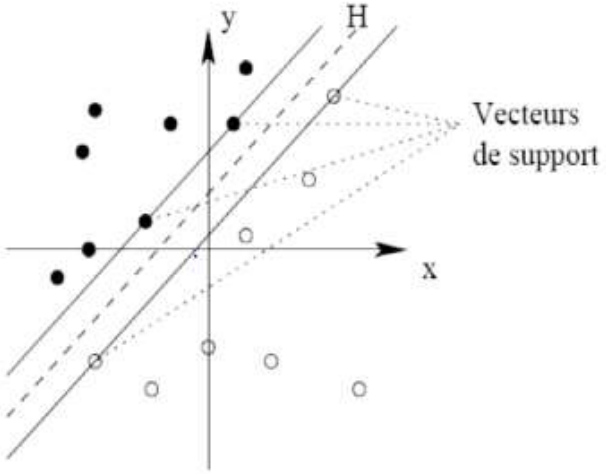


Figure.3 : Les vecteurs de support

7.2.4. Naïve Bayes :

La méthode de classification naïve bayésienne est un algorithme qui permet de classifier un ensemble d'observations selon des règles déterminées par l'algorithme lui-même.

Cet outil de classification doit dans un premier temps être entrainé sur un jeu de données d'apprentissage qui montre la classe attendue en fonction des entrées. Pendant la

phase d'apprentissage, l'algorithme élabore ses règles de classification sur ce jeu de donnée, pour les appliquer dans un second temps à la classification d'un jeu de données de prédiction. Le classificateur bayésien naïf implique que les classes du jeu de données d'apprentissage soient connues et fournit, d'où le caractère supervisé de l'outil.

La prédiction de la classification naïve bayésienne peut aboutir à un cas d'égalité où plusieurs classes obtiennent une même probabilité P(y). Deux approches sont proposées pour gérer ces cas :

- **Choix aléatoire** : choisi une classe de manière aléatoire dans l'ensemble des classes présentant la même probabilité P(y).
- **Plus petit indice** : choisi la première classe rencontrée dans l'ensemble des classes présentant la même probabilité P(y).

Avantages de Naïve Bayes :

- C’est relativement simple à comprendre et à construire
- Il est facile à former, même avec un petit jeu de données
- C’est rapide !
- Il n’est pas sensible aux caractéristiques non pertinentes

7.2.5. Les Réseaux de neurones :

Les réseaux de neurones (Artificial Neural Network) sont généralement optimisés par des méthodes d'apprentissage de type statistique grâce à leur capacité de classification et de généralisation, tels que la classification automatique de codes postaux ou la prise de décision concernant un achat boursier**.**

Les réseaux de neurones artificiels sont conçus de la même manière que le cerveau humain, avec des nœuds de neurones interconnectés à la manière d'une toile. Les neurones sont des milliards de cellules qui composent le cerveau humain. Chaque neurone est constitué d'un corps cellulaire qui traite les informations en les acheminant vers et depuis le cerveau (entrées et sorties) **[14]**. L'idée principale de ces réseaux est (dans une certaine mesure) inspirée par la façon dont le système neuronal biologique fonctionne, pour traiter les données et les informations afin d'apprendre et de créer des connaissances. L'élément clé de cette idée est de créer de nouvelles structures pour le système de traitement de l'information. L'architecture du réseau de neurones artificiels est illustrée à la figure 4. **[15]**

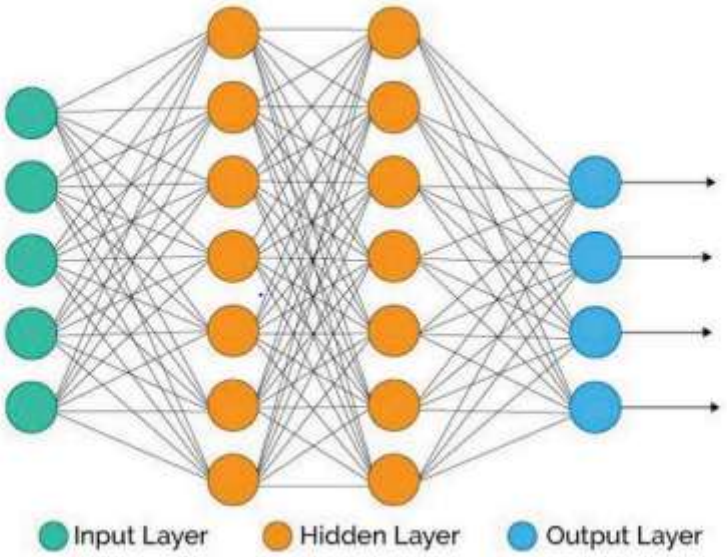


Figure 4: Artificial neural network architecture (Bre, Gimenez, and Fachinotti 2018).

7.2.5.1. Comment fonctionne le réseau de neurones artificiels ?

Un réseau de neurones repose sur un grand nombre de processeurs opérant en parallèle et organisés en tiers. Le premier tiers reçoit les entrées d'informations brutes, un peu comme les nerfs optiques de l'être humain lorsqu'il traite des signaux visuels.

Par la suite, chaque tiers reçoit les sorties d'informations du tiers précédent.

7.2.5.2. Les types des réseaux neuronaux :

Les architectures de réseaux neuronaux peuvent être divisées en 4 grandes familles :

- Réseaux de neurones Feed fowarded
- Réseaux de neurones récurrent (RNN)
- Réseaux de neurones à résonance
- Réseaux de neurones auto-organisés

7.2.5.3. Les domaines d'application des réseaux neuronaux :

- La reconnaissance d'image
- Les classifications de textes ou d'images
- Identification d'objets

- Prédiction de données
- Filtrage d'un set de données

I.8.Conclusion :

Dans ce chapitre, nous avons défini dans le premier partie la catégorisation des textes et le processus général, les problèmes qui s'y rattachent. Nous avons également introduit l'Apprentissage automatique (ML) et ses types ainsi que les algorithmes de classification les plus fréquemment utilisés de manière brève et précise. Ensuite dans la deuxième partie, nous avons présenté les réseaux de neurones, comment fonctionne et les domaines d'application.

CHAPITRE II

Architecture des réseaux de neurones

II.1. Introduction :

Dans cette thèse, les méthodes d'apprentissage à base de réseaux de neurones ont constitué la brique de base et font donc l'objet de ce chapitre introductif. Les réseaux de neurones ont aujourd'hui montré de véritables aptitudes dans de nombreuses tâches comme la classification d'images ou de textes, la prédiction de cours de bourse, etc. Intuitivement les réseaux de neurones sont construits par mimétisme avec les fonctions cérébrales du vivant (voir figure 7). Cette métaphore du vivant ne prétend pas reproduire le fonctionnement biologique du neurone mais plutôt en être une inspiration. On peut citer Yann Lecun à ce sujet

« C'est un abus de langage que de parler de neurones ! De la même façon qu'on parle d'aile pour un avion mais aussi pour un oiseau, le neurone artificiel est un modèle extrêmement simplifié de la réalité biologique. ».

Nous allons maintenant décrire plus en détail les réseaux de neurones mis en œuvre dans cette thèse à savoir le neurone formel (section 1), les perceptrons (section 2 et 3), les réseaux récurrents (section 4), les réseaux LSTM (section 5), les réseaux de convolution (section 6), les auto-encodeurs (section 7), le mécanisme d'attention (section 8) ainsi que l'architecture transformer (section 9).

II.2. Intelligence artificielle:

L'intelligence artificielle désigne « l'ensemble des théories et des techniques mises en œuvre en vue de réaliser des machines capables de simuler l'intelligence d'êtres vivants ». Autrement dit, l'IA est une grande famille qui réunit une grande variété de disciplines (informatique, mathématiques, neurosciences, etc.) et de technologies (matérielles ou logicielles). C'est pourquoi le terme est utilisé dans beaucoup de contextes très différents les uns des autres. Ses frontières étant par nature assez floues, il laisse la porte ouverte à une utilisation parfois galvaudée. Une chose est certaine, même si elle attire tous les regards aujourd'hui, l'intelligence artificielle n'a rien d'une nouveauté. Au début des années 50 déjà, le mathématicien Alan Turing s'interrogeait sur la capacité d'une machine à penser et créait le célèbre test qui porte son nom. Ce n'est toutefois que plusieurs décennies plus tard que la puissance de calcul informatique est devenue suffisante pour concrétiser des scénarios jusqu'alors seulement imaginés.

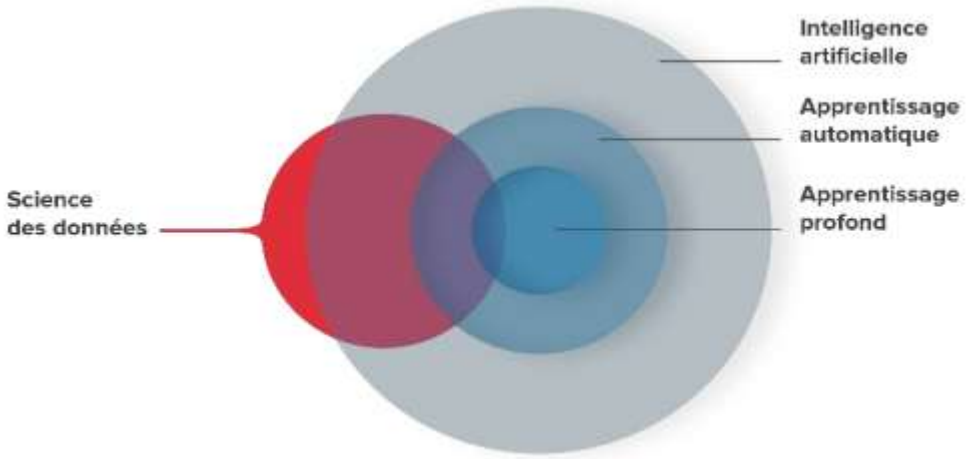


Figure 5: Place de L'IA

Figure 6 : Les differents domaines de L'IA

II.3.Apprentissage profond (Deep Learning)

Le DL, ou apprentissage profond, est quant à lui un sous-domaine du ML. Les deux approches ont beaucoup en commun et c'est pourquoi elles sont souvent confondues. Fondamentalement, ce qui va les distinguer l'une de l'autre est la complexité de l'analyse et la quantité de données exploitées. L'apprentissage profond est en réalité un apprentissage automatique s'appuyant sur un réseau de neurones qui vont affiner l'analyse grâce à un traitement par couches. Chaque couche de neurones artificiels va recevoir les résultats de la couche qui le précède, avant de procéder à ses propres calculs, de manière à pousser toujours plus loin le niveau de compréhension. En résumé, plus le nombre d'étapes de calcul sera important, plus le système sera capable de répondre à des questions complexes,

comme la reconnaissance d'un visage par exemple. C'est cette succession d'itérations qui « creusent » la donnée qui a donné le terme apprentissage « profond ».

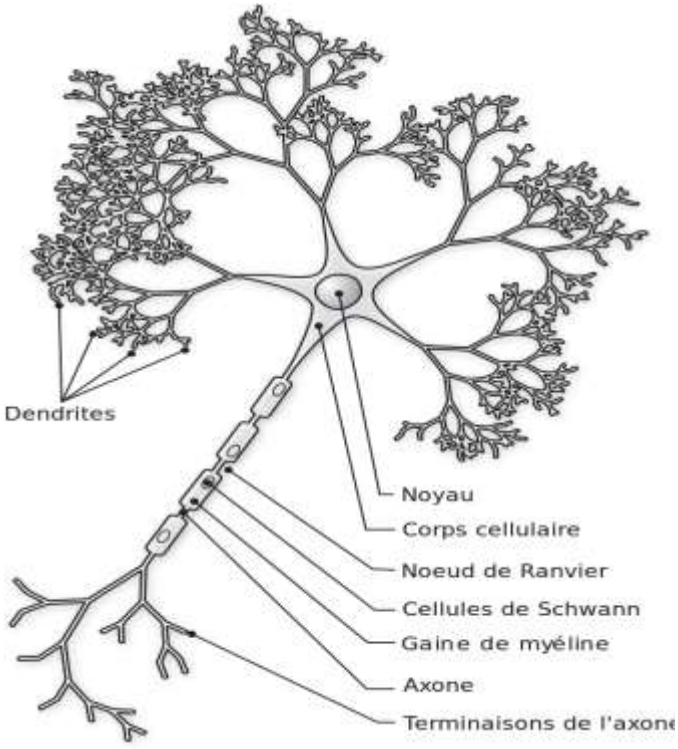
II.4. Le neurone formel

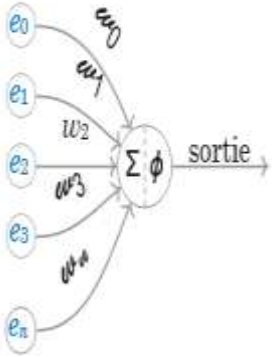
Le premier neurone formel (voir figure 7) a été proposé par les neurologues Warren McCulloch et Walter Pitts (What the frog's eye tells the frog's brain) **[16].** Le neurone formel est construit comme un système composé d'une fonction de transfert qui module ses sorties en fonction de ses entrées selon des règles précises, ce qui lui procure l'aptitude à résoudre des problèmes simples. À chaque entrée est associé un poids synaptique. Il calcule tout d'abord la somme pondérée de ses entrées. Le résultat est ensuite transformé par une fonction d'activation non linéaire, la fonction de Heaviside ou fonction marche.

Équation : Neurone formel étape intermédiaire

$$z = \sum_{j=1}^{n} w_j \cdot x_j$$

Avec x = (x1, x2, ..., xn) vecteur d'entrée, w = (w1, w2, ..., wn) vecteur des poids synaptiques, et w0 une constante représentant un biais. Le biais procurant une certaine malléabilité de la





frontière de décision du neurone. [2]

Figure 8 :Le neurone formel

Équation : Neurone formel expression de la sortie

$$\textbf{sortie} = \boldsymbol{\phi}(\mathbf{z}) = \boldsymbol{\phi}\left(\sum_{\mathbf{j=1}}^{\mathbf{n}} \boldsymbol{\omega}_{\mathbf{j}} \cdot \mathbf{x}_{\mathbf{j}} + \boldsymbol{\omega}_{\mathbf{0}}\right) \qquad \textbf{(figure 8)}$$

La fonction d'activation du neurone peut prendre différentes formes suivant l'application.

$$\text{sigmoïde: } \phi(z) = \frac{1}{1 + e^{-z}}$$

$$\text{Tangente Hyperbolique : } \phi(z) = \frac{1 - e^{-z}}{1 + e^{-z}}$$

$$\text{ReLu (Rectified Linear unit) : } \phi = \max(0, z)$$

Le physiologiste canadien Donald Hebb fournit une méthode pour modifier les poids synaptiques appelée règle de Hebb. Il a imaginé un mécanisme qui permet de modifier la valeur des coefficients synaptiques en fonction de l'activité des entités connectées ensembles. Cette méthode de modification poids du neurone est encore aujourd'hui utilisée couramment**.[17]**

II.5. Le perceptron

Le neurone formel est associé à d'autres neurones afin de constituer un ré- seau de neurones et ainsi augmenter la complexité de résolution. Le perceptron, représenté sur la (figure 9) a été introduit par Frank Rosenblatt en 1957. Il s'agit du premier système automatique capable d'apprendre de ces expériences grâce à une implémentation du concept d'induction. Le perceptron est un réseau de type feedforward, c'est-à-dire que l'information ne se propage que dans un sens, de la couche d'entrée vers la couche de sortie**.[18]**

Malheureusement, le perceptron ne permet pas de traiter les problèmes non linéaires, ce qui constitue une limitation importante.

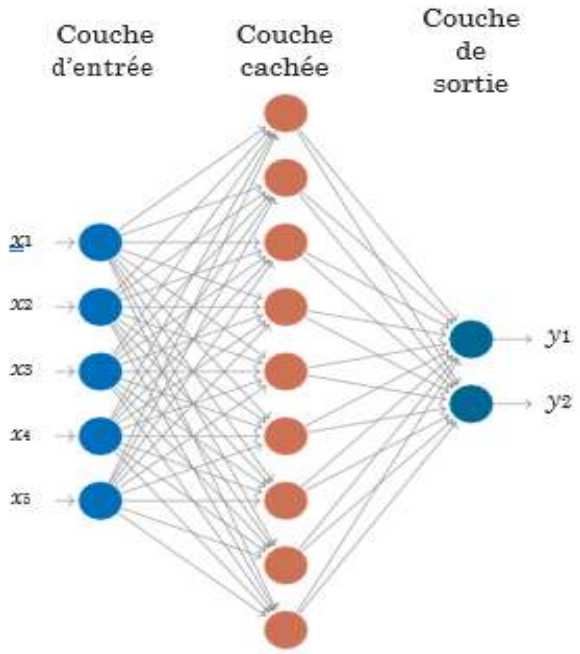


Figure 9: Le perceptron

- ## Le perceptron multi-couches:

Dans les années 80, l'apparition de la rétro-propagation du gradient de l'erreur permet la création du perceptron multi-couches. Les réseaux de neurones multi-couches ont une topologie en trois parties : une première couche connectée à l'extérieur du réseau, une ou plusieurs couches cachées connectées séquentiel- lement à partir de la couche d'entrée et ensuite une couche de sortie. On parle de réseau profond lorsque les couches cachées sont nombreuses.

Soit x = (x1, x2, ..., xn) le vecteur d'entrée d'un perceptron à L couches, expri- mons h1 le vecteur de sortie de la première couche.

Équation : Perceptron expression de la sortie de la couche d'entrée

$$h1 = \phi(W0,1.x + b1)$$

avec W0,1 la matrice des poids de la couche d'entrée en regard aux vecteurs d'entrée, b1 le biais et ϕ la fonction d'activation.

Nous pouvons généraliser cette écriture aux couches suivantes:

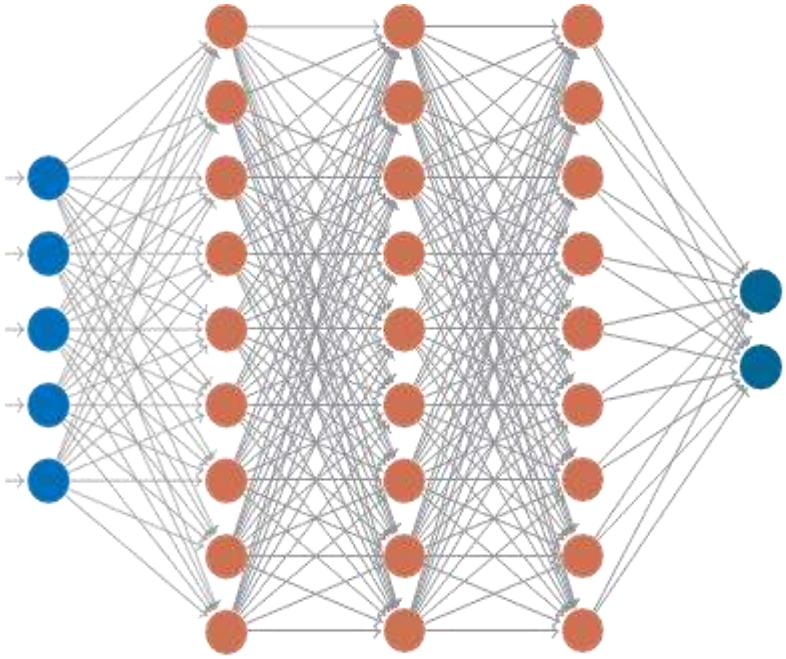


Figure 10 : Le perceptron multi-couches

Équation : Perceptron expression de la sortie de la couche cachée

$$hl+1 = \phi(Wl,l+1.hl + bl+1) \quad \text{(figure 10)}$$

La sortie du réseau sera alors calculable comme suit:

Équation : Perceptron expression de la sortie du perceptron

$$y = hL = \sigma(WL-1,L.hL-1 + bL)$$

avec σ la fonction d'activation de la couche de sortie, et y = (y1, y2, ..., ym) le vecteur de sortie du réseau de neurones pour une tache de classification mono- étiquette de m classes C = (c1, c2, ..., cm).

La fonction d'activation de la couche de sortie est différente de celle des couches cachées. En effet, le rôle de chaque couche est différent ainsi que son implémentions. La dernière couche pour une tâche de classification permettra la production des probabilités des classes pour les échantillons mis en entrée.

Généralement, pour une classification mono-étiquette, la fonction d'activation de la couche de sortie du réseau est de type Softmax. Cependant, ce n'est pas obligatoire.

Équation : Fonction de coût : Softmax

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{m} e^{z_i}}$$

Avec: zi = W.hi−1 + b.

y le vecteur de sortie du réseau est alors une variable aléatoire représentant la probabilité du vecteur x d'appartenir à la classe ci que l'on note P (y = i|x). La fonction Softmax, par sa construction ainsi que la normalisation qu'elle conduit, permet d'obtenir une somme des termes égale à un.

Équation : Les neurones expriment une probabilité

$$\sum_{i\epsilon 1,2,\ldots\ldots,m} p(y = i|x) = 1$$

Avec m le nombre de neurones de la couche de sortie L.

Pratiquement, la prédiction de la classe pour l'échantillon x est obtenue par la valeur maximale parmi les éléments du vecteur y.

Équation : Estimation de l'étiquette

Argmax (Sof tmax(zL))

i∈1,2,...,m

II.6. Les réseaux récurrents

Les (RNN) Réseau de Neurones Récurrents sont des réseaux de neurones dans lesquels l'information peut se propager d'avant en arrière, y compris des couches profondes aux couches plus en amont. En cela, ils sont plus proches du vrai fonctionnement du système nerveux des humains, qui n'est pas à sens unique. Ces réseaux ont des connexions récurrentes dans le sens où ils conservent des informations en mémoire. Ils peuvent prendre en compte à un instant t un certain nombre d'états passés. On peut faire une représentation de la succession des états, les RNN présentent un caractère dynamique du fait que leurs poids dépendent non seulement des entrées apprises, mais éga- lement des sorties précédentes. Pour ces raisons, les RNN sont particulièrement adaptés aux applications impliquant le contexte, la relation d'ordre des données et plus particulièrement au traitement de séquences temporelles telles que l'apprentissage et la génération de signaux,

c'est-à-dire lorsque les données forment une séquence et ne sont pas indépendantes dans leurs successions. D'un point de vue théorique, les RNN ont un potentiel beaucoup plus important que les ré- seaux neuronaux conventionnels. Ils sont «Turing-complete », c'est-à-dire qu'ils permettent théoriquement de simuler l'ensemble des algorithmes calculables. Une amélioration importante des réseaux RNN a été apportée avec le concept de bidirectionnalité intégrant une passe « avant » (forward) de la séquence des données dans un sens suivie d'une passe « arrière » (backward) des données de la séquence en sens opposé. Le transfert bidirectionnel d'informations rend leur conception assez compliquée.

Équation : RNN et bidirectionnalité

$$(\text{forward})\ hf = \phi(W_f \cdot hf + W \cdot xt + b)$$

t h t−1 x h

$$(\text{backward})\ hb = \phi(Wb.hb + Wb.xt + bb)$$

t h t−1 x h

Équation : Sortie d'un RNN bidirectionnel

$$y(t) = \sigma(W_f \cdot hf + Wb.hb + by)$$

o étant la fonction d'activation de la couche de sortie et by les biais.

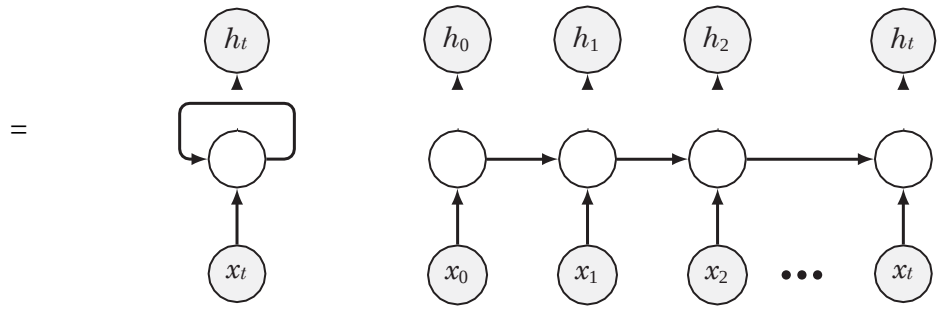


Figure 11 : Réseau de neurones récurrent déplié

Une des limitations importantes des RNN est leur incapacité à capturer les dépendances à long terme. Ceci est dû à la difficulté de la diffusion de la rétroprogation du gradient de l'erreur à travers la boucle de rétroaction.

II.7. Les réseaux LSTM

Les réseaux RNN sont peu efficaces pour les applications impliquant de longues différences de temps, typiquement la classification des séquences vidéo. Leur « mémoire à court terme » n'est pas suffisante. En effet, les RNN classiques (simples réseaux de neurones récurrents ou RNN vanilla) ne sont pas capables de mémoriser ce que le passé exprime à long terme et ils oublient après une cinquantaine d'itérations. À la fin des années 90, afin de résoudre ces problèmes, des méthodes efficaces ont été développées comme les réseaux LSTM. Ces réseaux à grande «mémoire à court terme» ont notamment révolutionné la reconnaissance de la voix par les machines (reconnaissance vocale) ou la compréhension et la génération de textes.

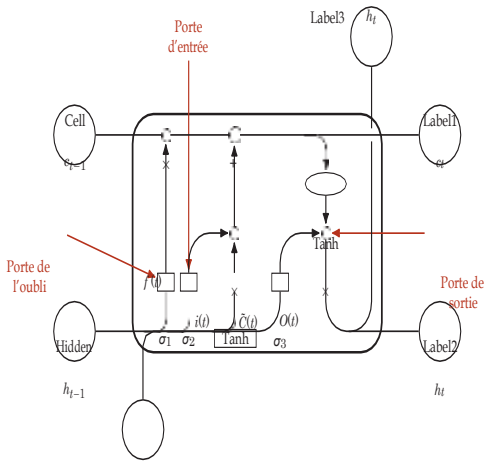


Figure 12 : Architecture d'une couche LSTM

Un réseau LSTM comporte une cellule de mémoire, typiquement une couche de neurones, ainsi que trois portes : une porte d'entrée, une porte de sortie et une porte de l'oubli. Ces trois portes vont permettre de moduler le flux d'informations à l'entrée, à la sortie et à mémoriser de manière analogique grâce à une fonction d'activation de type sigmoïde. La (figure 12) est une représentation dépliée du fonctionnement d'un réseau LSTM avec ses trois principales composantes.

Un réseau LSTM est basé sur une architecture qui lui permet d'oublier les informations inutiles. La couche sigmoïde 1 prend l'entrée x(t) et h(t−1) et décide quelles parties de l'information de la précédente sortie doivent être supprimées

(en sortant un 0). Cette porte est appelée « oubli » . La sortie de cette porte est:

$f(t) * C(t-1)$.

Équation : LSTM porte de l'oublie

f (t) σ (W .[h[t 1, x] b]) 0 → se débarrasser du contenu ct

1 → conserver le contenu de ct

L'étape suivante permet de décider puis de stocker les informations de la nouvelle entrée x(t) dans l'état de cellule. Une couche sigmoïde décide laquelle des nouvelles informations doit être mise à jour ou ignorée. Une couche tanh crée un vecteur de toutes les valeurs possibles à partir de la nouvelle entrée. Ces deux sont multipliées pour mettre à jour le nouvel état de cellule. Cette nouvelle mémoire est ensuite ajoutée à l'ancienne mémoire C(t−1) pour donner C(t).

Équation : LSTM cellule de mémoire

i(t) σ (W .[h[t 1, x] b]) 0 → pas de mise a` jour

1 → mise a` jour

$$\tilde{C}(t) = \tanh(Wc.[h[t-1, xt] + bc])$$

$$C(t) = C(t-1) + i(t) * \tilde{C}(t)$$

Enfin, il faut décider ce qui va être produit. Une couche sigmoïde décide quelles parties de l'état de la cellule vont être générées. Ensuite, l'état de la cellule à travers un tanh est fixé générant toutes les valeurs possibles et il est multiplié par la sortie de la porte sigmoïde, de sorte que la sortie produise uniquement les parties décidées. Notre modèle n'apprend pas la réponse de la dépendance immédiate, mais plutôt de la dépendance à long terme.

Équation : LSTM porte de sortie

O(t) (W .[h[t 1, x] b]) 0 → pas de sortie

1 → retourne l'intégralité de la cellule

Nous venons de voir qu'il y a une grande différence entre l'architecture d'un RNN et d'un LSTM. Dans un LSTM, le modèle apprend quelles informations stocker dans la

mémoire à long terme et lesquelles oublier. L'entraînement des réseaux LSTM est relativement rapide, quelques « époques » 3, et la remontée du gradient de l'erreur est fonctionnelle, cependant la mise en parallèle des calculs n'est pas possible du fait de la prise en compte de la relation d'ordre de l'information Ces approches ont beaucoup été utilisées pour le traitement automatique du langage, l'analyse d'images **[19]** ou encore le diagnostique médical.

II.9.Les réseaux de convolution

Les réseaux de neurones CNN sont construits pour une analyse fine des don- nées exprimées sous la forme d'une grille, typiquement les pixels d'une image. Ils ont été proposés par Krizhevsky et al, 2012 ont montré leur efficacité sur l'analyse de textes ou le traitement d'images.

Un réseau CNN est construit par la succession d'une couche de convolution et d'une couche d'aggrégation de l'information (Pooling) et se termine par une couche de neurones totalement connectés. Sur (la figure 13), on peut retrouver l'architecture d'un réseau CNN pour une tâche de classification de textes.

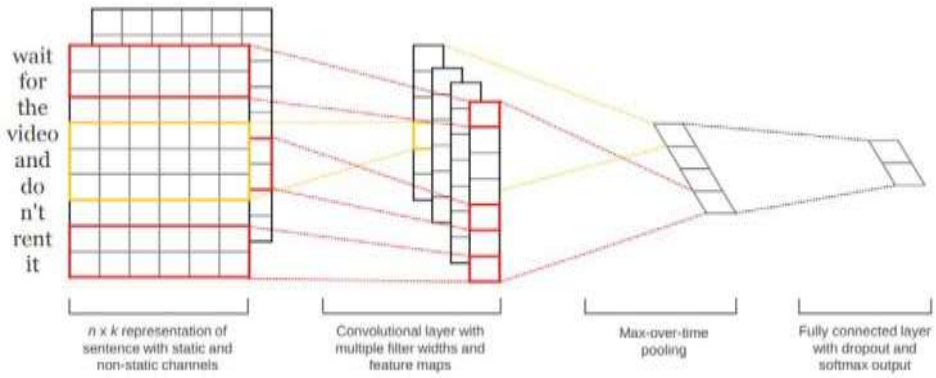


FIGURE 13 :Représentation de la structure d'un réseau CNN pour une analyse de phrases

Une convolution de matrice revient à multiplier deux matrices de dimension incompatible. L'opération de convolution consiste à appliquer un filtre ou « kernel » sur la grille ou matrice de données. Ce filtre est une matrice de dimension inférieure à la matrice de données. On positionne le centre du noyau sur l'élément de la matrice que l'on souhaite modifier et on applique un simple produit scalaire entre le recouvrement par le kernel sur la matrice et le kernel lui-même. Puis, on déplace le noyau d'un pas sur l'élément suivant et ainsi de suite. On peut retrouver une représentation de cette heuristique dans la figure 14.

Cela permet de faire émerger pour chaque filtre choisi, une analyse de caractéristique particulière. Il est courant d'appliquer un grand nombre de filtres sur un réseau de convolution ce qui produit une carte de caractéristiques. Il est évident que cette opération génère une explosion de la quantité de valeurs obtenues. On associe alors un sous-échantillonnage de manière à réduire la quantité d'informations.

Le Pooling est une méthode permettant de prendre une large grille d'informations et d'en réduire la taille tout en préservant les informations les plus importantes. L'algorithme consiste, à faire glisser une petite fenêtre pas à pas sur toutes les parties de la grille et à prendre la valeur maximum, la moyenne ou le minimum de cette fenêtre à chaque pas. La grille obtenue est par conséquent de dimension réduite par rapport à la grille initiale. Le résultat de cette opération permet au réseau de trouver si une information judicieuse est présente dans la grille, sans se soucier de l'endroit précis où cette information se trouve.

Le paramétrage d'un réseau CNN n'est pas une tâche aisée. Ainsi, il faut décider des multiples caractéristiques à prendre en compte de manière empirique : Combien de filtres faut-il utiliser ? Quelles dimensions pour ces filtres ? Pour chaque couche de Pooling, quelle taille de fenêtre doit-on choisir ? Quel pas ? Pour chaque couche entièrement connectée supplémentaire, combien de neurones cachés doit-on définir ?

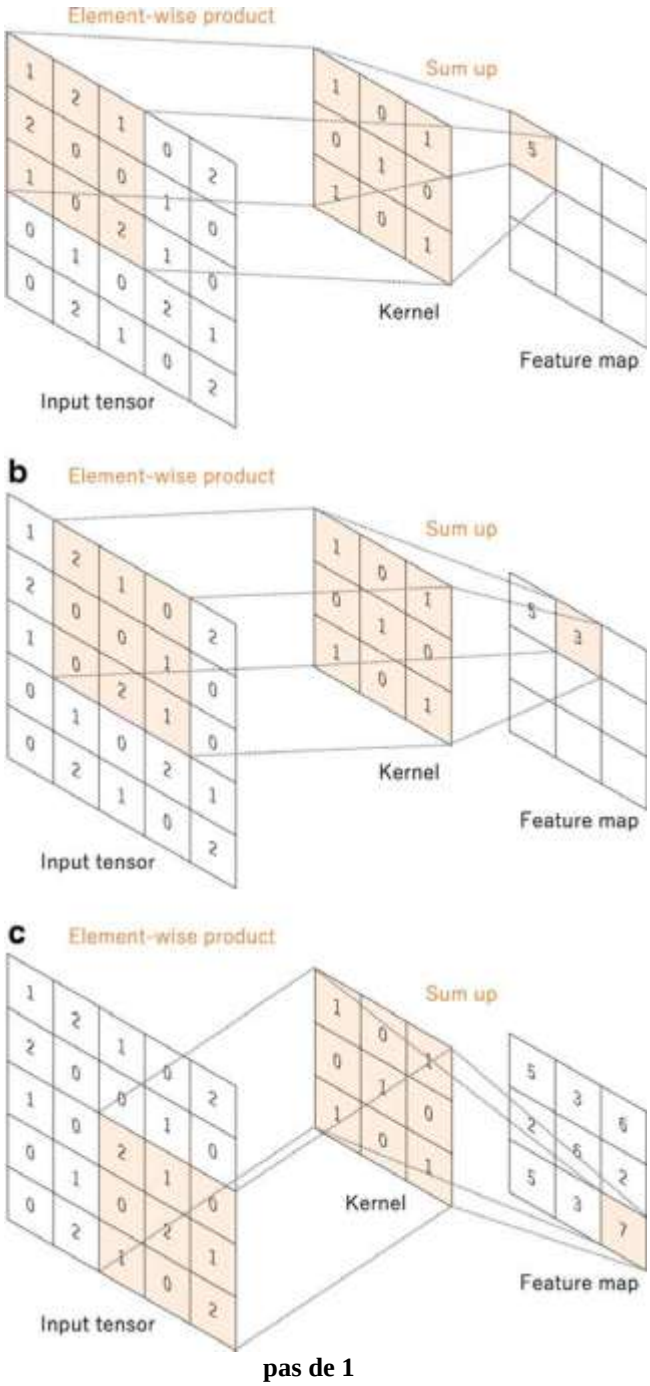
Cependant, un avantage important des réseaux CNN est la facilité de mise en parallèle des calculs. Aussi, ces réseaux ont été fortement utilisés dans les tâches de traitement automatique du langage en apprentissage profond **[20].**

II.10. Les auto-encodeurs

Les auto-encodeurs cherchent par leur structure à être capables de recons- truire à l'identique une entrée vectorielle. L'intérêt de ces réseaux est de tirer partie de l'information apprise pour résoudre cette tâche en apparence triviale. Une fois l'auto-encodeur entraîné, il pourra être utilisé pour diverses tâches comme la réduction de dimension [Anam and Al-Jumaily, 2015; Wang et al., 2016] ou encore la traduction de texte [Zhang et al., 2016a]. Les auto-encodeurs pour la phase d'entraînement utilisent en sortie la même information qu'en en- trée. En cela, on parle d'apprentissage non supervisé car ils ne demandent au- cune autre information additionnelle que les vecteurs d'entrée. Comme on peut le voir sur la figure 15 la couche d'entrée possède le même nombre de neu-rones que la couche de sortie. L'architecture neuronale se situant entre ces deux couches,

peut-être de différents types : multiples couches totalement connectées, couche CNN, couche LSTM, etc. Sur la figure 15, c'est une simple couche de neurones.

La structure d'entrée sert à projeter les vecteurs d'entrée dans l'espace intermédiaire latent.



pas de 1

Figure 14 : Représentation du calcul de convolution d'une grille 5 par 5 avec un noyau en 3 par 3 et un

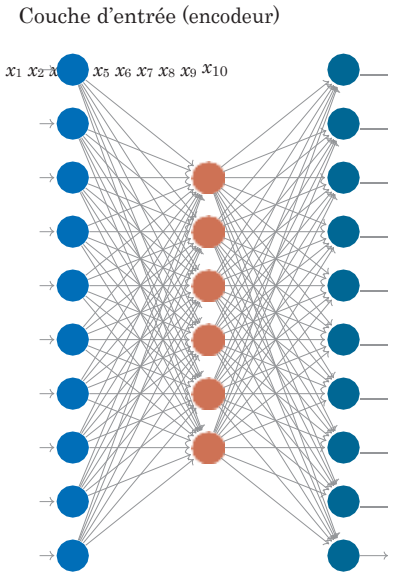


Figure 15 :Architecture d'un auto-encodeur

Équation : Auto-encodeur : espace latent

$$hl = \phi enc(Wenc.X + benc)$$

Avec ϕenc la fonction d'activation de l'encodeur, Wenc la matrice de poids de l'encodeur et benc vecteur de biais.

Le décodeur est ensuite utilisé pour reconstruire le vecteur de sortie le plus identiquement possible à celui fourni en entrée.

Équation : Auto-encodeur : sortie du décodeur

$$y = \phi dec(Wdec.hl + bdec)$$

Avec ϕdec la fonction d'activation du décodeur, Wdec la matrice de poids du décodeur et bdec le vecteur de biais.

Généralement, l'erreur moyenne quadratique est utilisée comme fonction de coût.

Équation : Auto-encodeur : fonction de coût

$$E(X,Y) = \frac{1}{N}\sum_{i=1}^{N} \|X_i - Y_i\|^2$$

Où N représente le nombre d'échantillons.

II.11: Le mécanisme d'attention

En psychologie, l'attention est le processus cognitif de concentration sélectif sur un ou plusieurs objets tout en ignorant les autres.

Le mécanisme d'attention est apparu en vue de faire progresser les systèmes de traduction automatique basés sur des auto-encodeurs dans le TALN **[21].** Plus tard, ce mécanisme, ou ses variantes, ont été utilisés dans d'autres applications, notamment la vision par ordinateur, le traitement de la parole, etc.

Dans les applications de traduction automatique, il est courant d'utiliser des auto encodeurs sur la base d'une architecture LSTM. Chaque fois que le modèle proposé génère une phrase, il recherche un ensemble de positions dans les couches cachées de l'encodeur où les informations les plus pertinentes sont disponibles. Cette idée est appelée «Attention».

Le LSTM bidirectionnel utilisé dans la figure 16 génère une séquence d'annotations (h_1, h_2, ..., h_{Tx}) pour chaque phrase d'entrée. Tous les vecteurs h_1, h_2, ..., etc., utilisés dans le processus sont la concaténation des états cachés de sortie des LSTM avant et arrière dans l'encodeur.

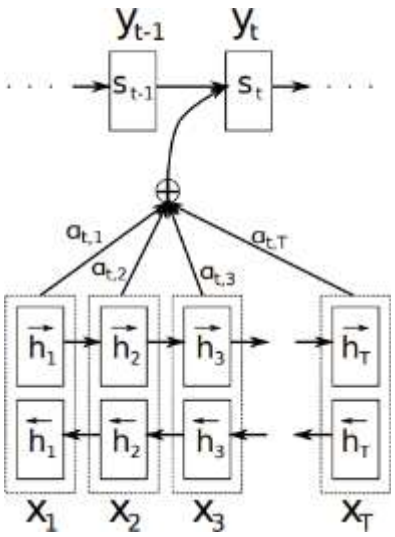


Figure 16 :LSTM

Équation : Attention encodeur

$$h_j = [-h\rightarrow T\ ,\ \leftarrow h-T\ ]$$

Tous les vecteurs h1, h2, h3, ..., hTx sont des représentations du nombre Tx de mots dans la phrase d'entrée. Dans le modèle de l'encodeur et du décodeur simple, seul le

dernier état du LSTM de l'encodeur a été utilisé (hTx dans ce cas) comme vecteur de contexte.

Le vecteur de contexte ci pour le mot de sortie yi est généré en utilisant la somme pondérée des annotations.

Équation : Attention vecteur de contexte

$$c_i = \sum_{j=1}^{T_x} a_{i.j} h_j$$

Les poids αij sont calculés par une fonction Softmax.

Équation : Attention Softmax

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{K=1}^{T_x} e_{ik}} \; e_{ij} = a(S_{i-1} - h_j)$$

ei j est le score de sortie d'un réseau neuronal décrit par la fonction a qui tente de capturer l'alignement entre l'entrée en j et la sortie en i.

Il en résulte que α est un vecteur. Ses éléments sont les poids correspondant à chaque mot dans la phrase d'entrée en vis à vis de leur importance dans le contexte.

L'attention permet de regarder la totalité d'une phrase, pour établir des liens entre un mot particulier et son contexte. Ceci est très différent des RNN à courte mémoire, focalisés en amont, et également très différent des réseaux convolutifs focalisés sur la proximité de l'information analysée.

Actuellement, le mécanisme d'attention est très largement utilisé dans un certain nombre d'applications comme: les soins, la santé, la reconnaissance vocale, les systèmes de recommandation ainsi que les voitures autonomes.

II.12. L'architecture de réseau neurone Transformer

L'architecture de réseau neuronal Transformer proposée par Vaswani et al. [2017a] a marqué l'une des percées majeures ces dernières années dans le domaine de l'analyse de textes en apprentissage profond. Les couches d'attention de l'architecture Transformer alignent les mots d'une séquence avec d'autres mots de la séquence, proposant ainsi une représentation de la séquence. Il est non seulement plus efficace en termes de

représentation, mais aussi plus efficace en termes de prédiction par rapport aux architectures de la littérature de type RNN et CNN.

L'attention multi-têtes utilisée ici est un processus essentiellement constitué de plusieurs couches d'attention apprenant conjointement différentes représentations à partir de différentes positions.

Équation : Transformer Attention

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)v$$

II.13. Conclusion

Nous venons dans ce chapitre de faire la description des principales architectures utilisées en apprentissage profond dans le traitement automatique du langage, à savoir les réseaux de convolution, les réseaux LSTM, les auto-encodeurs ainsi que l'architecture Transformer. Toutes ces architectures ont des comportements différents, aussi il est important de savoir si une architecture se détache pour le traitement des données que nous cherchons à classifier quel classifieur.

CHAPITRE III

Partie Réalisation

III.1. Introduction

Tout processus de catégorisation automatique de documents textuels peut se diviser en trois étapes principales qui sont : l'étape de Codage des textes, la classification automatique et finalement l'étape d'évaluation. Les approches modernes utilisent généralement plusieurs tâches pour le codage des documents textuels. Cette dernière comporte deux phases : la première qui est le prétraitement ou le nettoyage de texte. Elle permet aussi à la représentation précise des documents (ex : Tokenisation, filtrage des mots vides (stop words), lemmatisation, stemming, etc.) [50]. Quant à la deuxième phase, il s'agit de la sélection/réduction de l'espace des caractéristiques avec un impact négatif et également minimal sur la précision de la classification.

Les techniques d'apprentissage profond (Deep Learning) sont aujourd'hui très populaires dans le domaine d'apprentissage automatique où il a été prouvé leur performance dans les domaines de traitement d'image et particulièrement dans le traitement du langage naturel (NLP), notamment dans la traduction automatique, la segmentation, la reconnaissance du parole, l'analyse des sentiments et la classification des textes.

Après avoir étudié dans le chapitre 01 un état sur les différentes techniques d'apprentissage profond (Deep Learning). Nous allons présenter dans cette partie un nouveau modèle de LSTM qui est à la fois proposé et adapté pour la classification textuelle. Une Comparaison est introduite tout en présentant les résultats de notre modèle LSTM par rapport aux méthodes d'apprentissage automatique classique à savoir SVM, Naive Bayes, Random Forrest et les Arbres de décision. Ensuite, une autre comparaison entre les déférentes techniques de sélection des caractéristiques notamment Back Propagation et Gain d'Information est présentée en détail dans ce travail. Cette dernière sert à montrer l'utilité de l'algorithme de Back Propagation et notamment son impact que ce soit sur les performances de classification ou sur la sélection des caractéristiques. Cette partie fournit aussi une description sur les Benchmarks utilisées dans cette expérimentation, les outils appliqués et bien évidemment elle décrit les résultats des tests réalisés et leurs discutions.

III.2. Dataset

Les documents de la collection Reuters-21578 sont apparus sur le fil de presse de Reuters en 1987. Les documents ont été assemblés et indexés avec des catégories par le personnel de Reuters Ltd. (Sam Dobbins, Mike Topliss, Steve Weinstein) et Carnegie Group, Inc. (Peggy Andersen, Monica Cellio, Phil Hayes, Laura Knecht, Irene Nirenburg) en 1987.

En 1990, les documents ont été mis à disposition par Reuters et CGI à des fins de recherche au Laboratoire de recherche d'informations (W. Bruce Croft, directeur) du Département d'informatique et des sciences de l'information de l'Université du Massachusetts à Amherst. La mise en forme des documents et la production des fichiers de données associés ont été réalisées en 1990 par David D. Lewis et Stephen Harding de l'Information Retrieval Laboratory.

Un formatage et une production de fichiers de données supplémentaires ont été effectués en 1991 et 1992 par David D. Lewis et Peter Shoemaker au Center for Information and Language Studies de l'Université de Chicago. Cette version des données a été mise à disposition pour FTP anonyme sous le nom de "Reuters-22173, Distribution 1.0" en janvier 1993. De 1993 à 1996, Distribution 1.0 a été hébergé sur une succession de sites FTP gérés par le Center for Intelligent Information Retrieval (W. Bruce Croft, directeur) du Département d'informatique de l'Université du Massachusetts à Amherst.

Lors de la conférence ACM SIGIR '96 en août 1996, un groupe de chercheurs en catégorisation de texte a discuté de la manière dont les résultats publiés sur Reuters-22173 pourraient faire l'objet de plus d'études croisées. Il a été décidé qu'une nouvelle version de la collection devrait être produite avec une mise en forme moins ambiguë et comprenant une documentation énonçant soigneusement les méthodes standard d'utilisation de la collection. L'occasion serait également mise à profit pour corriger une variété d'erreurs typographiques et autres dans la catégorisation et le formatage de la collection.

Steve Finch et David D. Lewis ont fait ce nettoyage de la collection de septembre à novembre 1996, en s'appuyant fortement sur la version balisée SGML de Finch de la collection d'une étude antérieure. L'un des résultats du réexamen de la collection a été la suppression de 595 documents qui étaient des doublons exacts (basés sur l'identité des

horodatages à la seconde près) d'autres documents de la collection. La nouvelle collection ne compte donc que 21 578 documents, et s'appelle donc la collection Reuters-21578. Ce README décrit la version 1.0 de cette nouvelle collection, que nous appelons "Reuters-21578, Distribution 1.0".

Selon le README, si REUTERS TOPICS contient OUI, mais qu'aucun sujet n'est donné dans TOPICS, alors les articles "peuvent raisonnablement être considérés comme des exemples négatifs de l'ensemble des 135 sujets". [5] Ces entrées doivent donc être conservées. Si REUTERS TOPICS contient NON, mais qu'un sujet est donné, il a été ajouté après l'indexation d'origine. Ils doivent donc eux aussi être conservés.

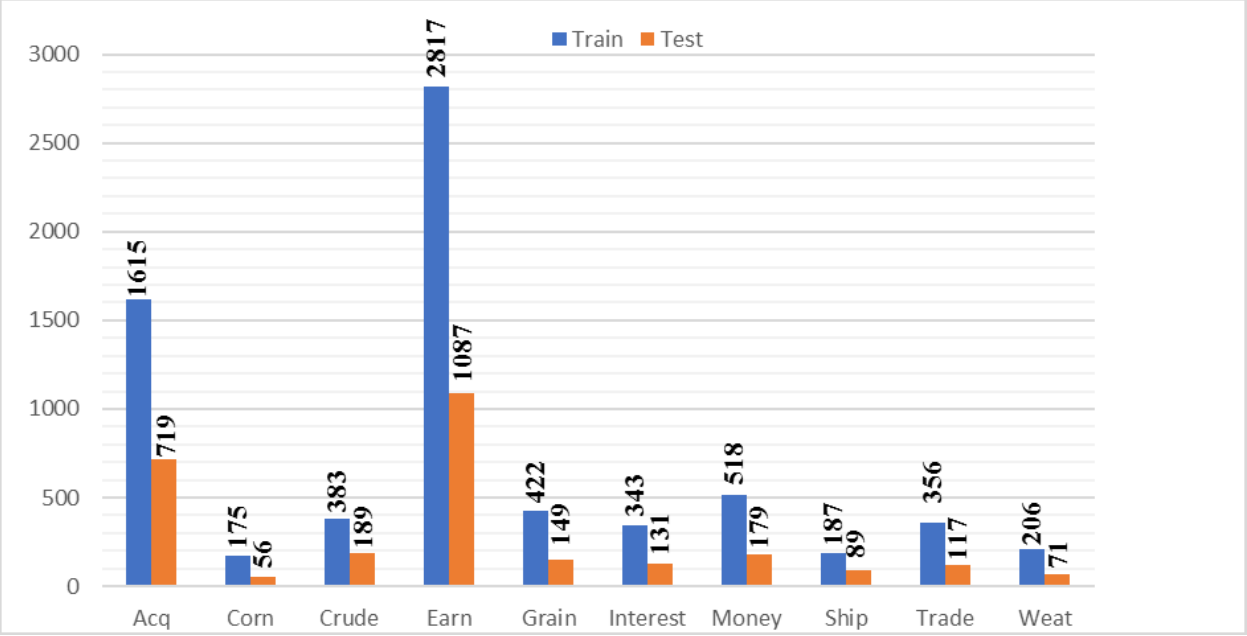


Figure III.1 : diagramme des topic

Cependant, si REUTERS TOPICS contient NON et qu'aucun sujet n'a été donné, il est toujours ambigu de savoir s'ils étaient destinés à être des exemples négatifs ou non. Ainsi, ces entrées doivent être supprimées de l'ensemble de données. J'ai inclus uniquement les entrées qui satisfont à ces conditions en n'incluant que les entrées qui ont des sujets ou indiquent qu'il y a des sujets.

Dans certains cas, il n'y a pas de texte dans le champ BODY, mais l'article est toujours catégorisé. Dans ces cas, on a i utilisé le champ TITRE comme CORPS car c'est tout ce que le catégoriseur devait faire.

III.3. Outils utiliser

a- TensorFlow: est une framework Open Source de programmation développée pour le calcul numérique par l'équipe Google Brain en Novembre 2015 qui l'utilisait initialement en interne. Depuis son release, TensorFlow n'a cessé de gagner en popularité, pour devenir très rapidement l'un des frameworks les plus utilisés pour le Deep Learning et donc basé sur le principe des réseaux de neurones profond. Son nom est notamment inspiré du fait que les opérations courantes sur des réseaux de neurones sont principalement faites via des tables de données multidimensionnelles, appelées Tenseurs (Tensor). Un Tensor à deux dimensions est l'équivalent d'une matrice. Aujourd'hui, les principaux produits de Google sont basés sur TensorFlow: Gmail, Google Photos, Reconnaissance de voix.

b- Keras est une API de réseaux de neurones de haut niveau, écrite en Python et capable de fonctionner sur TensorFlow ou Theano. Il a été développé en mettant l'accent sur l'expérimentation rapide. Il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), et son principal auteur et mainteneur est François Chollet, un ingénieur Google. En 2017, l'équipe TensorFlow de Google a décidé de soutenir Keras dans la bibliothèque principale de TensorFlow. Chollet a expliqué que Keras a été conçue comme une interface plutôt que comme un cadre d'apprentissage de bout en bout. Il présente un ensemble d'abstractions de niveau supérieur et plus intuitif qui facilitent la configuration des réseaux neuronaux indépendamment de la bibliothèque informatique de backend. Microsoft travaille également à ajouter un backend CNTK à Keras aussi.

c- Python est un langage de programmation de haut niveau, interprété (il n'y a pas d'étape de compilation) car, avant de pouvoir les exécuter, un logiciel spécialisé se charge de transformer le code du programme en langage machine, orienté objet avec une sémantique dynamique, multi paradigme et multiplateformes. Le langage python est placé sous une licence libre, et fonctionne sur la plupart des plates-formes informatiques, des Smartphones aux ordinateurs centraux. Il est très sollicité par une large communauté de

développeurs et de programmeurs et Python peut être utilisé pour gérer des données volumineuses et effectuer des calculs complexes. Il existe ce qu'on appelle des bibliothèques (packages) qui aident le développeur à travailler sur des projets particuliers. Plusieurs bibliothèques peuvent ainsi être installées pour, par exemple, développer des interfaces graphiques en Python.

d- Scikit-learn : est une bibliothèque libre Python destiné à l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment dans le monde académique par des instituts français d'enseignement supérieur et de recherche comme Inria et Télécom ParisTech. Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes.

e- Colaboratory : est un outil de recherche open source pour l'enseignement et la recherche en apprentissage machine. Il s'agit d'un environnement pour ordinateur portable Jupyter qui ne nécessite aucune installation ou bien une configuration pour être utilisé, il s'exécute entièrement dans le cloud. Cet environnement supporte Python 3.0 et ses versions supérieures. Il nous permet d'écrire et d'exécuter du code, de sauvegarder et partager, et d'accéder à de puissantes ressources informatiques depuis notre navigateur. Nos expériences ont été réalisées sur une machine virtuelle dans l'environnement Google Colab qui offre une bonne performance.

f- Classificateur LSV

Le classificateur LSV est une machine à vecteurs de support qui traite chaque classe comme si elle était linéairement séparable et tente de tracer cette ligne qui les sépare dans une certaine marge. Le paramètre par défaut du LSV de scikit-learn consiste à traiter plusieurs classes via une stratégie un contre repos ou un contre tous. Vous trouverez plus d'informations dans le Guide de l'utilisateur. Les pondérations des classes ont été définies sur "équilibrées" pour donner à chaque classe une pondération inversement proportionnelle à la fréquence de ses entrées afin que chaque catégorie soit traitée de la même manière par la fonction de régression. Le choix de cette fonction elle-même est riche de possibilités qui seraient explorées dans un format différent.

De plus, les paramètres ont été ajustés via le module GridSearchCV de scikit-learn qui permet d'essayer plusieurs paramètres possibles en combinaison avec le meilleur sélectionné pour l'ajustement final. La validation est effectuée à l'aide d'un schéma de

validation croisée triple sur l'ensemble d'apprentissage afin qu'il n'y ait aucune fuite d'informations sur l'ensemble de test dans le processus d'apprentissage.

Enfin, le choix des paramètres à ajuster a été choisi pour donner une impression de variété dans quelques variables tout en maintenant le temps d'exécution dans des limites raisonnables.

Rapport de classification :

+----------+

Classification report :

+----------+

	precision	recall	f1-score	support
acq	0.8526	0.9643	0.9050	588
alum	1.0000	0.9231	0.9600	13
bop	0.5789	0.7857	0.6667	14
carcass	0.6667	0.2857	0.4000	7
cocoa	1.0000	0.9412	0.9697	17
coffee	0.8857	1.0000	0.9394	31
copper	0.8667	0.8125	0.8387	16
corn	0.0000	0.0000	0.0000	2
cotton	0.8333	0.7143	0.7692	7
cpi	0.8261	0.9048	0.8636	21
crude	0.8138	0.9077	0.8582	130
dlr	0.4000	0.1667	0.2353	12
earn	0.9881	0.9539	0.9707	955

fuel 0.6667 0.6667 0.6667 3

Comme on pouvait s'y attendre, les classes à faible soutien ont des scores très différents. En effet, il est difficile d'apprendre, du moins avec cohérence, les caractéristiques des classes sous-représentées. Les scores des classes à support plus élevé sont plus intéressants, avec les scores f1 pour 'money-fx' à environ 0,81, 'trade' à environ 0,84, 'acq' à environ 0,90 et 'earn' - avec le support le plus élevé - - à environ 0,97. Ce qui est peut-être plus intéressant, c'est que le modèle a pu prédire la classe créée, "aucune", avec un score f1 d'environ 0,88. C'était intéressant pour moi parce que cela signifie qu'il y a des caractéristiques d'apprentissage des entrées qui indiquent qu'elles devraient sortir du champ d'application du système de classification donné. Si je devais aller plus loin, j'analyserais certainement ces résultats en utilisant PCA et tSNE pour identifier la confusion de chaque classe entre elles. Pour l'instant, un sous-ensemble de la grande matrice de confusion devra faire l'affaire.

Figure III.2. HEATMAPS-PREDICTER LABEL

III.4. Classification binaire par rapport à 'earn'

Pour effectuer la tâche de classification binaire, le dataset doit être modifié afin de créer deux classes. Dans ce cas, les cours étaient « earn » et « not-earn ». on s’attends à ce

que ce soit une tâche difficile étant donné que la variété des entrées "not earn" pourrait aggraver la confusion entre elles et "earn".

Pour cette tâche, j'aurais aimé utiliser à nouveau un SVC, mais cela a pris trop de temps, on a donc décidé d'essayer un autre classificateur : Multinomial Naive Bayes. En fin de compte, on préfèrerait former et comparer plusieurs classificateurs différents

Ci-dessous, les données linguistiques sont transformées de la même manière que ci-dessus en utilisant CountVectorizer afin d'obtenir des vecteurs représentant chaque entrée. Cependant, il n'est pas nécessaire ni même avantageux de convertir les comptages dans ces vecteurs via le Tf idf Transformer pour MNB. Une plage de variables est à nouveau sélectionnée pour donner un temps d'exécution raisonnable mais donner une certaine variation. Dans MNB, les probabilités a priori sont initialisées puis mises à jour en fonction des instances d'entrées observées et des classes auxquelles elles appartiennent.

- **Rapport de classification**

Classification report :

---

	precision	recall	f1-score	support
earn	0.9930	0.8942	0.9410	955
not-earn	0.9584	0.9974	0.9775	2333
accuracy			0.9675	3288
macro avg	0.9757	0.9458	0.9593	3288
weighted avg	0.9685	0.9675	0.9669	3288

Ici, nous pouvons voir que la performance globale pour prédire la classe 'earn' est inférieure à celle de la tâche précédente. Cependant, un problème avec Naive Bayes est qu'il semble bénéficier davantage de plus d'exemples que de classificateurs alternatifs (voir

les exemples de référence dans la documentation). Le classificateur a également donné un score de précision très élevé pour « earn », ce qui signifie que presque tout ce qui est étiqueté « earn » appartenait en fait à cette classe. Cependant, le rappel beaucoup plus faible signifie que plusieurs entrées appartenant à la classe "earn" ont été manquées. Nous pouvons étudier cela plus en utilisant la matrice de confusion ci-dessous.

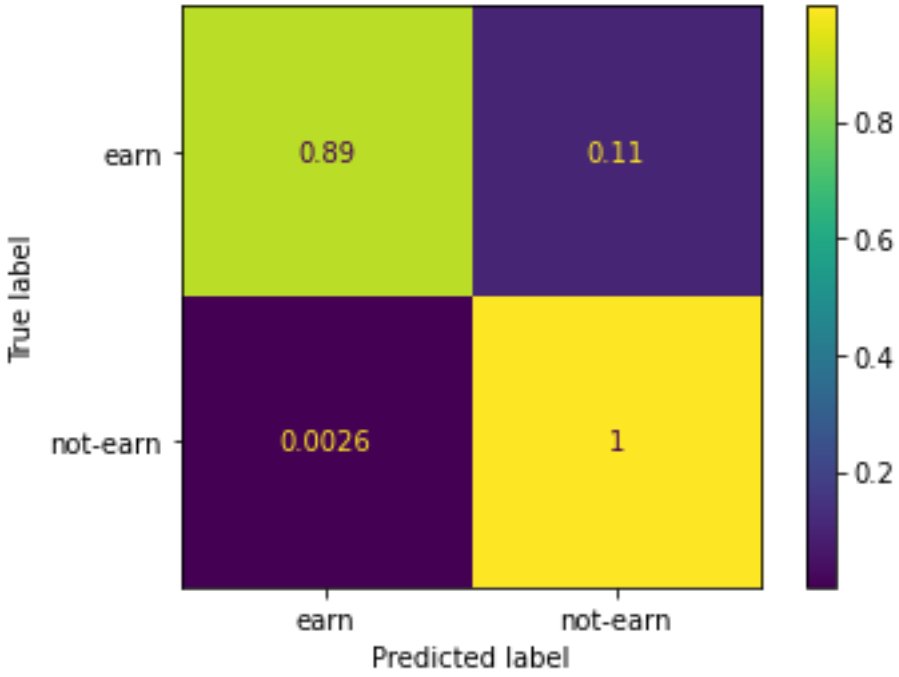


Figure III 3 : MATRICE(EAR-NOTEARN)

Données : Nous avons utilisé les trois ensembles de données utilisés dans : IMDB, Elec et RCV1, comme résumé dans le tableau.

1- IMDB (critiques de films) est livré avec un ensemble non étiqueté. Pour faciliter la comparaison avec les études précédentes, nous avons utilisé une union de cet ensemble et de l'ensemble d'entraînement en tant que données non étiquetées. Elec se compose d'avis Amazon de produits électroniques. Pour les utiliser comme données non étiquetées, nous avons choisi 200 000 avis provenant de la même source de données afin qu'ils soient disjoints des ensembles de formation et de test, et que les produits examinés soient disjoints de l'ensemble de tests. Sur la classification à 55 voies des sujets de deuxième niveau sur RCV1 (actualités), les données non étiquetées ont été choisies pour être disjointes des ensembles de

formation et de test. Sur la catégorisation multi-étiquettes de 103 sujets sur RCV1, puisque la division officielle LYRL04 pour cette tâche divise l'ensemble du corpus en un ensemble de formation et un ensemble de tests, nous avons utilisé l'ensemble des tests comme données non étiquetées (le paramètre d'apprentissage transductif).

Tableau 1 : Ensembles de données. † Le RCV1 multi-étiquettes n'est utilisé que dans le tableau 6

	#train	#test	#non étiqueté	#classe	Sortie
BDIM	25,000	25,000	75K (20M mots)	2	Positif/négatif ***Sentiment***
Elec	25,000	25,000	200K (24M mots)	2	
RCV1	15,564	49,838	669K (183M mots)	55 (simple)	Sujet(s)
	23,149	781,265	781K (214M mots)	103 (multi)†	

Implémentation Nous utilisé les modèles CNN à une couche qui se sont avérés efficaces dans [11] comme base modèle B, nomly, seq-CNN>, sur IMDB/Elec et bow-CNN sur RCV1. Intégration TV formation mini-

Les régions cibles, et p est la sortie du modèle. Les pondérations αi,j ont été fixées pour équilibrer la perte résultant de la présence et de l'absence de mots (ou de concepts dans le cas de la cible partiellement supervisée) et pour accélérer l'entraînement en éliminant certains exemples négatifs, similaires à l'échantillonnage négatif de [19]. Pour expérimenter avec la cible non supervisée, nous avons défini z comme vecteurs d'arc des régions adjacentes à gauche et à droite, tout en ne conservant que les 30 000 mots les plus fréquents avec contrôle du vocabulaire ; sur la classification du sentiment c, les mots de fonction ont été supprimés, et sur la classification du sujet, les nombres et les mots d'arrêt fournis par [16] ont été enlevés. Notez que ces mots ont été retirés (et seulement de) le vocabulaire cible. Pour produire la cible partiellement supervisée, nous avons d'abord entraîné les modèles CNN supervisés avec 1000 neurones et appliqué la couche de convolution entraînée à des données non étiquetées pour générer des vecteurs de dimension 1000 pour chaque région. Le reste de la mise en œuvre suit [11] ; c.-à-d. les modèles supervisés minimisaient la perte carrée avec la régularisation L 2 et l'abandon facultatif [9]; σ et σ(U) étaient le redresseur; la normalisation de la réponse a été effectuée; l'optimisation a été effectuée par SGD.

Sélection du modèle Sur toutes les méthodes testées, le réglage des méta-paramètres a été effectué en testant les modèles sur la partie retenue des données d'entraînement, puis les modèles ont été réentraînés avec les méta-paramètres choisis en utilisant l'ensemble des données d'entraînement.

II.5. Résultats du rendement

Aperçu après avoir confirmé l'efficacité de nos nouveaux modèles par rapport au CNN supervisé, nous rapportons les performances du CNN de [13], qui repose sur des vecteurs de mots pré-entraînés avec un très grand corpus (Tableau 3). Outre la comparaison des performances des approches dans leur ensemble, il est également intéressant de comparer l'utilité de ce qui a été appris à partir de données non étiquetées ; par conséquent, nous montrons comment cela fonctionne si nous intégrons les vecteurs de mots dans notre modèle de base à un chaud CNN . Dans ces expériences, nous testons également des vecteurs de mots entraînés par word2vec [19] sur nos données non étiquetées . Nous comparons ensuite nos modèles avec deux méthodes semi-supervisées standard, la SVM transductive (TSVM) [10] et le co-apprentissage (tableau 3), et avec les meilleurs résultats précédents de la littérature (tableaux 4 à 6). Dans toutes les comparaisons, nos modèles surpassent les autres. En particulier, les intégrations tv de notre région se révèlent plus compactes et plus efficaces que les intégrations de région obtenues par simple manipulation des intégrations de mots, ce qui soutient notre approche de l'utilisation incorporation de région au lieu d'incorporation de mots.

Noms dans le tableau 3

Tableau 2 : Tests d'intégration tv

Noms dans le tableau 3	$\mathbf{r}^{(U)}_{(\mathbf{x})}$: $X1$	$X2$: cible de la formation U
unsup-tv. parsup-tv. unsup3-tv.	Vecteur arc Vecteur arc Bag-of-{1,2,3}-gram vecteur	Vecteur arc Sortie du vecteur arc d'intégration supervisé

Tableau 3 : Taux d'erreur (%)

			BdIM	Elec	RCV1
SVM linéaire de 1 à 3 grammes [11]			10.14	9.16	10.68
TSVM linéaire avec 1-3grammes			9.99	16.41	10.77
[13] Cnn			9.17	8.03	10.44
CNN à un chaud (simple) [11]			8.39	7.64	9.17
Co-formation CNN (simple) à un chaud			(8.06)	(7.63)	(8.73)
Notre CNN	unsup-tv.	100-dim 200-dim	7.12 6.81	6.96 6.69	8.10 7.97
	parsup-tv.	100-dim 200-dim	7.12 7.13	6.58 6.57	8.19 7.99
	unsup3-tv.	100-dim 200-dim	7.05 6.96	6.66 6.84	8.13 8.02
	tous les trois	100×3	**6.51**	**6.27**	**7.71**

À titre de comparaison, tous les modèles CNN a été contraints d'avoir 1000 neurones. Les parenthèses autour des taux d'erreur indiquent que les méta-paramètres de co-apprentissage ont été réglés sur les données de test.

Notre CNN avec TV-embeddings Nous avons testé trois types de tv-embddingcomme résumé dans le tableau

2- La première chose à noter est que tous les nos CNN (Tableau 3, rangée 6 à 12) surpassent leur homologue supervisé à la rangée 4. Cela confirme l'efficacité du cadre que nous proposons. Dans Table 3, pour un rayonnement significatif comparaison tout le Cnn are contraint À avoir exactement Un convolution couche (sauf pour [13]'s CNN) avec 1000 Neurones. Le les plus performants Supervisé Cnn dedans ceux-ci contraintes (ligne 4) are: seq-CNN (région taille 3) sur BDIM et Elec et arc-CNN (région taille 20) sur RCV11. Ils ont également servi de modèles de base (avec la taille de la région paramétrée sur IMDB/Elec). CNN supervisés plus complexes de [11] sera examiné plus tard. Sur la classification des sentiments (IMDB et Elec), la taille de la région choisie par la sélection du modèle pour nos modèles était de 5, plus grande que 3 pour les modèles supervisés. Cnn. Ceci Indique cela non étiqueté données Activé effective utiliser de Grandes Régions quel sont plus prédictifs mais peuvent souffrir de la rareté des données dans la supervision Paramètres.

'unsup3-tv.' (Lignes 10 à 11) utilise un vecteur sac de n-gramme pour représenter initialement chaque région, ce qui permet de rétablir l'ordre des mots partiellement dans la région. Lorsqu'il est utilisé individuellement, unsup3-tv. N'a pas surpassé les autres intégrations de télévision, qui utilisent plutôt l'arc (rangées 6 à 9). Mais nous avons constaté qu'il contribuait à la réduction des erreurs lorsqu'il était combiné avec les autres (non indiqué dans le tableau). Cela implique qu'il a appris des informations prédictives de données non étiquetées que les deux autres intégrations ont manquées. Les meilleures performances (ligne 12) ont été obtenues en utilisant les trois types d'intégrations TV à la fois selon (6). Ce faisant, les taux d'erreur ont été améliorés de près de 1,9 % (IMDB) et de 1,4 % (Elec et RCV1) par rapport à la CNN supervisée (ligne 4), en raison des trois intégrations de télévision avec des forces différentes qui se complètent.

1- Le taux d'erreur sur RCV1 à la ligne 4 diffère légèrement de [11] car ici nous n'avons pas utilisé la liste des mots d'arrêt.

Sur nos données non étiquetées. Axe x : dimensionnalité de l'entrée supplémentaire pour l'intégration de région supervisée. 'r :' : région, 'w :' : mot.

CNN Il a été démontré dans [13] que CNN qui utilise les vecteurs de mots Google News comme entrée est compétitif sur un certain nombre de tâches de classification de phrases. Ces vecteurs (300 dimensions) ont été formés par les auteurs de word2vec [19] sur un très grand corpus Google Actualités (GN) (100 milliards de mots ; 500 à 5 000 fois plus grand que nos données non étiquetées). [13] a fait valoir que ces vecteurs peuvent être utiles pour diverses tâches, servant d ' « extracteurs de caractéristiques universelles ». Nous avons testé le CNN de [13], qui est équipé de trois couches de convolution de différentes tailles de région (3, 4 et 5) et d'un pool maximal, en utilisant les vecteurs GN comme entrée. Bien que [13] n'ait utilisé que 100 neurones pour chaque couche, nous l'avons changé en 400, 300 et 300 pour correspondre aux autres modèles, qui utilisent 1000 neurones. Nos modèles surpassent nettement ces modèles (tableau 3, ligne 3) avec des différences significativement importantes.

Comparaison des intégrations Outre la comparaison des performances des approches dans leur ensemble, il est également intéressant de comparer l'utilité de ce qui a été appris à partir de données non étiquetées. À cette fin, nous avons expérimenté l'intégration d'un mot incorporé dans nos modèles de base en utilisant deux méthodes ; l'un prend la concaténation, et l'autre prend la moyenne, des vecteurs de mots pour les

mots de la région. Ceux-ci fournissent des informations supplémentaires pour l'intégration supervisée des régions à la place de uf(x) in (5). C'est-à-dire que, par comparaison, nous produisons une intégration de région à partir d'une intégration de mots pour remplacer une intégration de télévision de région. Nous montrons les résultats avec deux types d'intégration de mots : l'incorporation de mots GN ci-dessus (Figure 3) et les intégrations de mots que nous avons entraînées avec le logiciel word2vec sur nos données non étiquetées, c'est-à-dire les mêmes données que celles utilisées pour l'apprentissage de l'intégration TV et toutes les autres (tab 4). Notez que (tab 4) représente les taux d'erreur par rapport à la dimensionnalité de l'entrée supplémentaire produite ; une dimensionnalité plus petite présente l'avantage d'un entraînement / prédiction plus rapide.

En ce qui concerne les résultats, tout d'abord, la région tv-embedding est plus utile pour ces tâches que les mots d'intégration testés puisque les modèles avec un tv-embedding surpassent clairement tous les modèles avec un word embedding. Mot concaténations vectorielles de dimensions beaucoup plus élevéesLité que ceux montrés dans la figure encore sous-performant 100-dim région tv-embedding. Deuxièmement, puisque notre région tv-embedding prend la forme de σ (W rf(x) + b) avec rf(x) étant un vecteur arc, les colonnes de Dans correspondent à des mots, et par conséquent, W rf(x) est te somme de WEst colonnes dont les mots correspondants se trouvent dans le C-ième région. Sur cette base, on peut se demander pourquoi nous ne devrions pas simplement utiliser la somme ou la moyenne des vecteurs de mots obtenus par un outil existant tel que word2vec à la place. Les performances sous-optimales de « dans : moyenne » (Graphique 4) Dit nous cela ceci est à mauvais idée. Nous attribut il A le fait que les intégrations de région apprennent le caractère prédictif de la coprésence et de l'absence de mots dans une région ; l'incorporation d'une région peut être plus expressive que la moyenne des vecteurs de mots. Ainsi, un efficace et compact l'incorporation de région ne peut pas être obtenue trivialement à partir d'un mot encastrement. En particulier, l'efficacité de la combinaison de trois intégrations TV (« r : 3 tv-embed. » dans la figure 4) Est dehors.

De plus, notre mécanisme d'utilisation de l'information provenant de données non étiquetées est plus efficace que le CNN de [13] puisque nos CNN avec GN (Figure 3) surpassent les CNN de [13] avec GN (Tableau 3, ligne 3). En effet, dans notre modèle, les vecteurs à une broche (les caractéristiques d'origine) compensent la perte d'informations potentielle dans l'intégration apprise à partir de données non étiquetées. Ceci, ainsi que

l'intégration de régions par rapport à mots, est une différence majeure entre notre modèle et le modèle de [13].

Méthodes semi-supervisées standard Bon nombre des méthodes semi-supervisées standard ne sont pas appliquées. Plicable à CNN car ils ont besoin de vecteurs d'arc comme entrée. Nous avons testé TSVM avec des vecteurs de sac de 1,2,3 grammes en utilisant SVMlight. TSVM a sous-performé la SVM supervisée2 sur deux des trois ensembles de données

2- Notez que pour des raisons de faisabilité, nous n'avons utilisé que les 30K n-grammes les plus fréquents dans les expériences TSVM, montrant ainsi les résultats SVM également avec un vocabulaire 30K à des fins de comparaison, bien que sur certains ensembles de données, les performances SVM puissent être améliorées par l'utilisation de tous les n-grammes (par exemple, 5 millions de n-grammes sur IMDB) [11]. En effet, le coût de calcul de TSVM (single-core) s'est avéré élevé, prenant plusieurs jours, même avec un vocabulaire 30K.

Tableau 4 : BDIM : taux d'erreur antérieurs (%).

Modèle	micro-F	macro-F	ressource supplémentaire
SVM [16]	81.6	60.7	–
bow-CNN [11]	84.0	64.8	–
bow-CNN avec trois tv-embed.	**85.7**	**67.1**	Données non étiquetées

Tableau 5 : Elec : taux D'erreur précédents (%).

SVM 1-3grammes [11]	8.71	–
dense NN 1-3grammes [11]	8.48	–
NB-LM 1-3grammes [11]	8.11	–
[11]le meilleur CNN de	7.14	–
Notre meilleur	**6.27**	Unlab.data

Tableau 6 : RCV1 micro- et macro-moyenne F sur la tâche multi-étiquettes (103 sujets) avec la division LYRL04.

Modèle	micro-F	macro-F	ressource supplémentaire
SVM [16]	81.6	60.7	–
bow-CNN [11]	84.0	64.8	–
bow-CNN avec trois tv-embed.	**85.7**	**67.1**	Données non étiquetées

(Tableau 3, lignes 1 à 2). Étant donné que la co-formation est un méta-apprenant, elle peut être utilisée avec CNN. La division aléatoire du vocabulaire et la division en first et la dernière moitié de chaque document ont été testées. Pour réduire la charge technique, nous rapportons les meilleures (et irréalistes) performances de co-entraînement obtenues en optimisant les méta-paramètres, y compris quand s'arrêter sur les données de test. Même avec cet avantage injuste pour la co-formation, la co-formation (tableau 3, ligne 5) a

clairement sous-performé nos modèles. Les résultats démontrent la difficulté d'utiliser efficacement des données non étiquetées sur ces tâches, étant donné que la taille des données étiquetées est relativement grande.

Comparaison avec les meilleurs résultats précédents Nous comparons nos modèles avec les meilleurs résultats précédents sur IMDB (tableau 4). Notre meilleur modèle avec trois intégrations TV surpasse les meilleurs résultats précédents de seulement 0,9%. Tous nos modèles avec un seul téléviseur intégré. (Tableau 3, ligne 6 à 11) affichent également de meilleurs résultats que les résultats précédents. Étant donné qu'Elec est un ensemble de données relativement nouveau, nous n'avons connaissance d'aucun résultat semi-supervisé antérieur. Notre performance est meilleure que celle du CNN le mieux supervisé de [11], qui a une architecture réseau complexe de trois paires de convolution-mise en commun en parallèle (tableau 5). Pour comparer avec les résultats du benchmark en [16], nous avons testé notre modèle sur la tâche multi-étiquettes avec le split LYRL04 [16] sur RCV1, dans lequel plus d'une catégorie sur 103 peut être attribuée à each document. Notre modèle surpasse la meilleure SVM de [16] et la meilleure CNN supervisée de [11] (Tableau 6).

III.6. Conclusion

Cet article proposait un nouveau cadre CNN semi-supervisé pour la catégorisation du texte qui apprend les em-beddings des régions de texte avec des données non étiquetées, puis des données étiquetées. Comme nous l'avons vu à la section 1.1, une intégration de région est formée pour apprendre les prédictifs de la coprésence et de l'absence de mots dans une région. En revanche, l'intégration d'un mot est entraînée à ne représenter que des mots individuels isolément. Ainsi, l'intégration d'une région peut être plus expressive que la simple moyenne des vecteurs de mots malgré leur similitude apparente. Notre comparaison des intégrations a confirmé son avantage ; les intégrations tv de notre région, qui sont formées spécifiquement pour la tâche d'intérêt, sont plus efficaces que les intégrations de mots testées. En utilisant nos nouveaux modèles, nous avons pu obtenir des performances plus élevées que les études précédentes sur la classification des sentiments et la classification des sujets.

Conclusion générale

Conclusion et perspective :

La classification de textes s'est imposée ces dernières années comme un domaine de recherche majeur pour les entreprises comme pour les particuliers.

Ce mémoire de fin d'études a comme objectif d'applications des techniques d'apprentissage profond pour résoudre des problèmes de classification automatisée des textes

Cette phase a pour but d'améliorer nos connaissances dans le domaine du Deep Learning et de trouver un moyen de l'utiliser dans la compréhension du sens des textes et classés en catégories

Après analyse des résultats obtenus, dès la première application de la classification de texte, nous l'avons trouvée des résultats acceptable, mais cela n'empêche pas l'existence de certains problèmes : nous n'avons pas pu utiliser l'algorithme d'apprentissage en profondeur (CNN/RNN...) car chaque fois que nous avons essayé, l'application n'a pas fonctionné, aucun résultat, nous avons donc utilisé deux autres algorithmes à la place (SVM - Naïve Bayes)

Nous avons également créé une autre application IMDB (critiques de films) nous avons utilisé deux algorithmes (CNN-RNN simple) les résultats étaient bons l'application peut savoir si les commentaires sont négatifs ou positifs.

Perspective :

A travers ce travail on a pu réunir plusieurs compétences :

1- Des connaissances dans le domaine du Deep Learning (processus, taches et techniques).

2- Les différents traitements de la représentation et codage textuelle (la préparation des textes, la pondération et la réduction de dimensions).

3- Les concepts de bases des différentes techniques d'apprentissage profond pour la classification (CNN, LSTM) en comparaison avec les techniques d'apprentissage automatique classique (Arbre de décision, K plus proche voisin, SVM, Naïve Bayes,).

4- La démarche à suivre pour effectuer une recherche technique (à travers ce modeste travail).

REFERENCES BIBLIOGRAPHIQUES

Références Bibliographiques :

[1] : Radwan JALAM, « Apprentissage automatique et catégorisation de textes multilingues», thèse de doctorat, Université Lumière Lyon 2, France, Juin 2003

[02] : https://www.mcours.net/cours/pdf/leilclic3/leilclic929.

« Cours de classification automatique de textes »: page12 (Brown & Chong, 1998)

[03]: FABRIZIO SEBASTIANI, « Machine Learning in Automated Text Categorization », Conseil recherche National, Italie, Mars 2002.

[04]: Fuhr and Knorz, 1984, Robertson and Harding, 1984, Hayes and Weinstein, 1990, Fuhr et al., 1991, Tzeras and Hartmann, 1993].

[05] : Androutsopoulos et al., 2000, Cohen, 1996].

[06] : Liddy et al, 1994

[07]: Sebastiani, 2002

[08]: (https://www.mcours.net/cours/pdf/leilclic3/leilclic929.pdf)

[09]: Salton and McGill, 1983

[10]:http://theses.univ-lyon2.fr/documents/lyon2/2003/jalam_r/pdf/jalam_r-TH.2.pdf page 15

[11] : J.Clech, D.A.Zighed. « Une technique de réétiquetage dans un contexte de catégorisation de textes » 2004.

[12]: H. B. Barlow, ŖUnsupervised learning, ŗ Neural Comput., vol. 1, no. 3, pp. 295Ŕ311, 1989.

[13] : Ross Quinlan1986

[14] : (Van Gerven et Bohte 2017

[15]: Bre, Facundo, Juan M Gimenez, and Víctor D Fachinotti. 2018. "Prediction of wind pressure coefficients on building surfaces using artificial neural networks." Review of. Energy and Buildings 158:1429-41.

[16] : Levin I, Kromer B, Schoch-Fischer H, Bruns M, Münnich M, Berdau B, Vogel JC, Münnich KO. 1985. 25 years of tropospheric.

[17] : https://www.sciencesetavenir.fr/high-tech/intelligence-artificielle/selon-yann-lecun-lintelligence-artificielle-a-20-ans-pour-faire-ses-preuves_120121

[18]: https://commons.wikimedia.org/wiki/File:Neuron-figure-fr.svg?

[19] Stollenga et al., 2015

[20] : Zhang et al., 2016b; Abulaish and Sah, 2019

[21] . Bahda- nau et al., 2015