République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud Mammeri de Tizi-Ouzou

Faculté de : Génie électrique et d'informatique
Département : Informatique

Mémoire de fin d'études

En vue de l'obtention du diplôme de

Master en Informatique

Spécialité : Systèmes Informatiques

Présenté par :

HOUAZENE Sabrina et MANSOUR Souad

Sujet : Détection des scripts publicitaires à base d'apprentissage automatique profond

Proposé et dirigé par : SADI Samy

Soutenu le 15 juillet 2019 devant le jury composé de:

Mr. HABET Mohammed Saïd	Président du Jury
Mme. BOUGCHICHE Lilia	Membre du Jury
Mr. SADI Samy	Directeur de mémoire

Dédicaces

Je dédie ce modeste travail à tous ceux qui sont chères :

À mes très chères parents source de tendresse et d'amour qui m'ont tout donnés et pour les sacrifices qu'ils ont consenti pour mon instruction, que dieu les gardes et les entoures de sa bénédiction, ainsi qu'à ma grand-mère à qui je souhaite une longue vie

À mes chers frères Mohamed et Amine

À mes chères sœurs Mounira et Meriem

À mes oncles et tantes ainsi qu'à leurs familles

À mes chers amis et camarades et mon binôme Souad

À tous ceux qui ont contribué de prés ou de loin à la réalisation de ce mémoire

Sabrina

Dédicaces

Je dédie ce modeste travail à tous ceux qui sont chères :

À mes très chères parents source de tendresse et d'amour qui m'ont tout donnés et pour les sacrifices qu'ils ont consenti pour mon instruction, que dieu les gardes et les entoures de sa bénédiction

À mes chers frères Tarik et Rafik

À mes chères sœurs Samira et Siham

À mes oncles et tantes ainsi qu'à leurs familles

À ma belle famille ainsi qu'à mon fiancé

À mes chers amis et camarades et mon binôme Sabrina

À tous ceux qui ont contribué de prés ou de loin à la réalisation de ce mémoire

Souad

Remerciements

La réalisation de ce mémoire n'aurait été possible sans la participation de personnes que nous souhaitons remercier.

Nous voudrions tout d'abord adresser toute notre gratitude et remerciement a notre promoteur Monsieur **Samy Sadi**, de nous avoir encadrés et apporté une aide précieuse indispensable à la réussite de ce travail mais également pour sa patience, sa disponibilité et surtout ses conseils, qui ont contribué à alimenter nos réflexions.

Nous remercions également nos membres du jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre modeste travail et de l'enrichir par leurs propositions.

Nous tenons à remercier spécialement tous les enseignants, qui nous ont accompagnés tout au long de notre cursus et pour la qualité de leur travail.

Enfin, nous voudrions exprimer notre reconnaissance envers nos familles et amis qui nous ont apportés leur support moral et intellectuel tout au long de notre cursus. Un grand merci également à nos collègues pour leur bonne humeur et motivation, qui ont rendu meilleure l'ambiance de travail.

Résumé

Avec l'avènement de e-commerce (le commerce en ligne), le marché de la publicité a pris son essor. Ainsi, le nombre de publicités a évolué de plus en plus ce qui présente un impact sur le chargement des sites web. En parallèle, ils provoquent des ralentissements frustrants pour les utilisateurs et dégradent leur expérience de navigation. Dés lors, plusieurs techniques de détection des scripts publicitaires ont été développées afin de faire face à ces désagréments.

Dans le cadre de notre mémoire, en premier lieu, nous avons présenté un état de l'art sur l'apprentissage automatique et les réseaux de neurones artificiels, notamment pour ce qui est de la classification de textes, leurs prétraitements et leurs représentations. En particulier, notre objectif est de détecter les scripts publicitaires dans le contenu web. Pour atteindre cet objectif nous avons proposé trois modèles de réseaux de neurones. Ensuite, nous avons implémenté puis évalué ces trois modèles. Pour ce faire, nous avons commencé par la construction de collections de données nécessaires à l'entraînement et à l'évaluation de nos modèles. Ce qui nous a permis ensuite d'évaluer nos modèles. Ainsi, nous avons obtenu de très bons résultats avec les trois modèles qui peuvent classifier des scripts Web selon les deux classes publicitaires ou non publicitaires de façon très précise.

Mots clés: scripts publicitaires, apprentissage automatique, réseaux de neurones, classification de textes, réseaux de neurones convolutionnels, réseaux de neurones récurrents.

Abstract

With the advent of e-commerce (the online business) the advertising market has taken off. The number of advertisements is evolving each day impacting in the meantime the loading time of websites and bringing other disagreements to users.In particular, they cause frustrating slowdowns for users and degrade their browsing experience.To overcome these, several techniques have been proposed to detect and block advertising scripts in Web content and this work is one of them.

As part of our thesis, we first presented a state-of-the-art of machine learning and artificial neural networks, particularly with regards to text classification, preprocessing and representation. Our goal is to detect advertising scripts in Web content. To achieve this goal, we have proposed three models of neural networks. Then we have implemented and evaluated these three models. To do this, we started by building data collections that are necessary for the training and the evaluation of our models. This allowed us to evaluate our models.As a result, we have obtained very good results with the three models that can classify Web scripts very precisely into one of the two following classes: advertising or non-advertising.

Key-words: advertising scripts, machine learning, neural networks, text classification, convolutional neural networks, recurrent neural networks.

Table des matières

Table des figures

Table des tableaux

Introduction générale

1 Contexte du travail

Ces dernières années, la publicité sur internet a fait ses preuves à travers le monde en tant que moyen de communication commerciale efficace. Son efficacité varie en fonction de la popularité d'Internet lui-même. Mais aujourd'hui, les bannières publicitaires sont de plus en plus décriées par les internautes car leur impact spatial et parfois leur contenu intrusif influence défavorablement l'efficacité des pages web. Grâce aux efforts des développeurs Google et Microsoft, etc. ce phénomène commence à être atténué du fait, notamment, de l'apparition de l'extension AdblockPlus[1] pour plusieurs navigateurs qui permet de filtrer les publicités dans le contenu Web. Il est donc important de développer d'autres outils nous permettant de détecter ce genre de publicités pour s'en prémunir, notamment grâce à l'intelligence artificielle.

L'intelligence artificielle (IA) renvoie à toutes les techniques qui permettent de simuler les processus cognitifs humains. Cette discipline a émergé en 1956 sous l'influence de John McCarthy, Marvin Minsk. Dés lors, les évolutions technologies et les systèmes informatiques, ainsi que l'apprentissage automatique, sont combinés pour le traitement de données, ce qui ouvre une voie à de nouveaux progrès. Les programmes d'intelligence artificielle promettent de prédire ou de détecter ce qui est difficilement décelable à l'homme (détecter des scripts publicitaires, analyse de multiples radiographies en médecine, etc). L'IA se révèle être utile dans toutes les taches intellectuelles. Le but de la recherche en IA est de créer de nouvelles technologies permettant aux ordinateurs et aux machines de fonctionner de manière intelligente et de préférence sans programmation explicite (grâce à l'apprentissage automatique).

2 Problématique

La problématique principale traitée dans ce mémoire est celle de la détection de scripts publicitaires dans le contenu Web. Pour résoudre cette problématique de classification, nous avons opté pour l'utilisation des réseaux de neurones vu l'intérêt grandissant qu'ils portent actuellement à la communauté scientifique. Il en découle plusieurs sous-problématiques que nous décrivons dans ce qui suit.

[1] Adblock Plus est une extension gratuite qui aide à personnaliser l'expérience Web des utilisateurs. Il permet de bloquer les publicités agaçantes, désactiver le suivi et bien plus encore. Il est disponible pour tous les principaux navigateurs de bureau et les appareils mobiles.

Tout d'abord, il faut pouvoir définir des modèles de réseaux de neurones efficaces pouvant opérer sur du contenu textuel. Ces modèles doivent pouvoir être entraînés suffisamment rapidement en utilisant une collection de données d'une taille raisonnable.

De la sous-problématique précédente, découle celle de l'évaluation. En effet, après avoir proposé des modèles pouvant théoriquement classifier des scripts, il faut pouvoir évaluer leurs performances. Il faut alors disposer d'une collection de données pour entraîner les modèles (il s'agit d'apprentissage automatique), puis tester leur performance. Cette problématique est d'autant plus importante du fait qu'il n'existe pas de collections de données toutes faites pouvant être utilisées dans ce sens. Il faut donc d'abord construire ces collections pour pouvoir réaliser l'évaluation des modèles proposés.

3 Contribution

Notre contribution se situe à plusieurs niveaux :

1. Premièrement, nous avons réalisé un état de l'art sur l'apprentissage automatique et les réseaux de neurones artificiels, notamment pour ce qui est de la classification de textes, leur prétraitement et leur représentation.
2. Deuxièmement, nous avons proposé trois modèles pouvant être utilisés pour la classification des scripts publicitaires. Nous avons en particulier, présenté les données en entrée de ces modèles ainsi que les prétraitements nécessaires qu'il faut appliquer à ces données.
3. Enfin, nous avons entraîné, validé et testé les modèles construits sur différentes collections de données que nous avons construites, puis nous avons conclus sur leurs efficacités.

4 Organisation du mémoire

Ce mémoire est organisé en 2 parties : une partie théorique et une partie contenant nos contributions. En tout, le mémoire contient 4 chapitres en plus de cette introduction et la conclusion générale.

La première partie comporte le premier et deuxième chapitre. Dans le premier chapitre, nous présentons des généralités sur l'apprentissage automatique, définitions, types d'apprentissage, ainsi les différents modèles. Ensuite, dans le deuxième chapitre, nous

expliquons les notions de base des réseaux de neurones tels que ses définitions, ses architectures et les différents algorithmes qui sont utilisés dans ce contexte, ainsi que la classification de textes à base des réseaux de neurones, leur prétraitement et leur représentation.

La deuxième partie comporte le troisième et quatrième chapitre. Dans le troisième chapitre, nous présentons trois modèles pour la classification des scripts publicitaires. Ensuite, dans le quatrième chapitre, nous décrivons notre implémentation, les outils et librairies utilisées ainsi que les résultats obtenus.

Chapitre 1: Généralités sur l’apprentissage automatique

1 Introduction

L'intelligence artificielle (IA) est un vaste domaine de recherche qui connaît un essor important aussi bien dans les médias que dans le domaine scientifique. Les avancées technologiques réalisées au niveau du matériel combinées à celles réalisées au niveau des logiciels ont donné à l'IA de nouvelles perspectives notamment à travers l'apprentissage automatique. Ainsi, de grandes entreprises dont Google, Facebook, IBM et Microsoft, ont investi beaucoup d'efforts et d'argent dans la recherche en apprentissage automatique et prévoient d'investir encore davantage dans le futur (1).

L'apprentissage automatique (ou apprentissage machine, *Machine Learning* en anglais) est un sous-domaine de l'IA. Son objectif principal est de comprendre la structure des données et de les intégrer dans des modèles qui peuvent être compris et utilisés par tout le monde.

Dans ce chapitre, nous présentons un ensemble de généralités et de concepts relatifs à l'apprentissage automatique. En premier lieu, nous donnons un bref historique retraçant l'apparition et la généralisation de l'apprentissage automatique. Ensuite, nous donnons quelques définitions de l'apprentissage automatique et nous présentons ses différents types et modèles. Après cela, nous décrivons brièvement les applications d'apprentissage automatique ainsi que ses défis. Avant de conclure, nous décrivons ce qui est l'évaluation d'un modèle d'apprentissage automatique.

2 Historique

Depuis longtemps, l'homme a cherché à recréer l'équivalent de son cerveau. On trouve trace de cette idée depuis longtemps. Les premières études sur les réseaux de neurones artificiels remontent aux années 1940 avec les travaux de Warren McCulloch et Walter Pitts qui ont proposé les premières notions de neurone formel. Puis en 1949, D. Hebb initie, dans son ouvrage "*The Organization of Behavior*", la notion d'apprentissage.

En 1950, Alan Turing mathématicien et théoricien précurseur de l'informatique, lance le concept d'intelligence artificielle. La première utilisation officiel de l'expression intelligence artificielle n'est apparaît qu'en 1955 à la conférence Dartmonth College (2).

La première application concrète des réseaux de neurones artificiels est survenue en 1957 avec l'invention du réseau dit «perceptron» par un dénommé Frank Rosenblatt. Dans la même période, Bernard Widrow et Ted Hoff ont proposé un nouvel algorithme d'apprentissage pour entraîner un réseau adaptatif de neurones linéaires, dont la structure et les capacités sont similaires au perceptron.

Un an plus tard, Arthur Samuel, informaticien américain pionnier dans le secteur de l'intelligence artificielle fut le premier à faire usage du terme apprentissage automatique.

Vers la fin des années 1960, un livre publié par Marvin Minsky et Seymour Papert est venu jeter beaucoup d'ombre sur le domaine des réseaux de neurones. Ces deux auteurs ont démontré les limites théoriques du perceptron, en particulier, l'impossibilité de traiter les problèmes non linéaires avec le modèle développés par Rosenblatt et Widrow-Hoff.

En 1982, Hopfield développe un modèle qui utilise des réseaux totalement connectés basés sur la règle de Hebb pour définir les notions d'attracteurs et de mémoire associative. En 1984 c'est la découverte des cartes de Kohonen avec un algorithme non supervisé basé sur l'auto-organisation et suivi une année plus tard par la machine de Boltzman.

Depuis ce temps, Une révolution survient alors dans le domaine des réseaux de neurones artificiels où bouillonne constamment de nouvelles théories, de nouvelles structures et de nouveaux algorithmes.

Vers les années 1990, le développement de *Deep Blue* rebaptisé *Deeper Blue* : conception d'un système de 256 processeurs fonctionnant en parallèle, chaque processeur peut calculer environ trois millions de coups par seconde.

En 2012, Le laboratoire X de Google développe un algorithme d'apprentissage automatique capable de parcourir des vidéos YouTube de manière autonome pour identifier celles contenant des chats.

En 2016, L'algorithme d'intelligence artificielle de Google bat un joueur professionnel au jeu de société chinois Go, considéré comme le jeu de société le plus complexe au monde et beaucoup plus complexe que les échecs.

De nos jours, l'utilisation de l'apprentissage automatique, notamment les réseaux de neurones dans divers domaines ne cessent de croître. Les applications en sont multiples et variées (3).

3 Définitions de l'apprentissage automatique

L'apprentissage automatique regroupe tellement de concepts différents et variés qu'il est difficile d'en donner une définition unique. En effet, l'apprentissage automatique est pluridisciplinaire et fait intervenir des concepts et des techniques de plusieurs domaines de l'IA incluant : les systèmes experts, représentation des connaissances, simulation du raisonnement humain, résolution de problèmes, réseaux de neurones, etc. Dans ce qui suit, nous donnons les définitions les plus proéminentes :

Définition 1 : L'apprentissage automatique est un domaine de l'intelligence artificielle qui fait référence au développement à l'analyse et à l'implémentation de méthodes automatisables permettant à une machine d'évoluer grâce à un processus d'apprentissage, et ainsi de remplir des tâches qu'il est difficile ou impossible de satisfaire par des moyens algorithmiques plus classiques (4).

Définition 2 : L'apprentissage automatique est une discipline scientifique utilisée en intelligence artificielle. Il s'agit d'algorithmes (procédures traduites en langages informatiques) qui analysent un ensemble de données afin de déduire des règles qui constituent de nouvelles connaissances permettant d'analyser de nouvelles situations (3).

Définition 3 : L'apprentissage automatique (ou apprentissage artificiel) est l'étude des algorithmes qui permettent aux programmes de s'améliorer automatiquement par expérience (5).

4 Types d'apprentissage automatique

Il existe plusieurs types différents d'apprentissage automatique, qui se distinguent essentiellement par leur objectif ; c'est-à-dire la nature de ce qui doit être appris. Parmi les types d'apprentissage automatique les plus largement adoptés sont :

4.1 L'apprentissage supervisé

Cette approche a pour objectif la conception d'un modèle reliant des données d'apprentissage à un ensemble de valeurs de sortie (un comportement) (6). Un superviseur (expert) est employé pour étiqueter correctement des exemples (les données d'apprentissage ou d'entraînement). L'apprenant (ordinateur ou machine) doit alors trouver ou approximer la fonction qui permet d'affecter la bonne étiquette à ces exemples (7).

Dans l'apprentissage supervisé, la construction et la génération d'un modèle passe par deux phases :

4.1.1 Phase d'apprentissage (d'entraînement)

Dans cette phase l'algorithme d'apprentissage reçoit en entré des exemples d'apprentissage (les documents d'entraînement) étiquetés et produit un modèle de prédiction le plus performant possible, c'est-à-dire le modèle qui produit le moins d'erreurs en prédiction comme présente la Figure 1.

La prédiction est une tache qui vise à prédire une ou plusieurs caractéristiques inconnues à partir d'un ensemble de caractéristique connues. Par exemple : prédire la qualité d'un client en fonction de son revenue et de son nombre d'enfant.

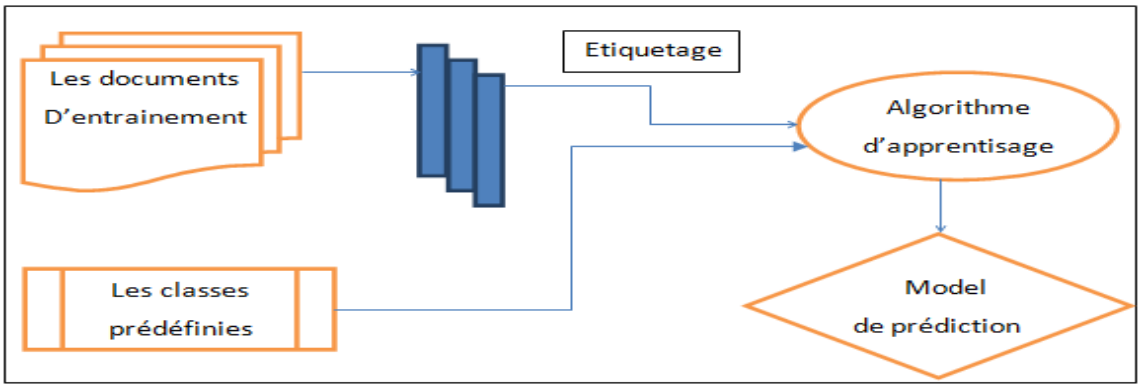


Figure 1 : La phase d'apprentissage

4.1.2 Phase de test

Dans cette phase le modèle obtenu lors de la phase d'apprentissage doit être capable de prédire l'étiquette d'un nouvel exemple en fonction des valeurs d'entrées (8), selon la Figure 2.

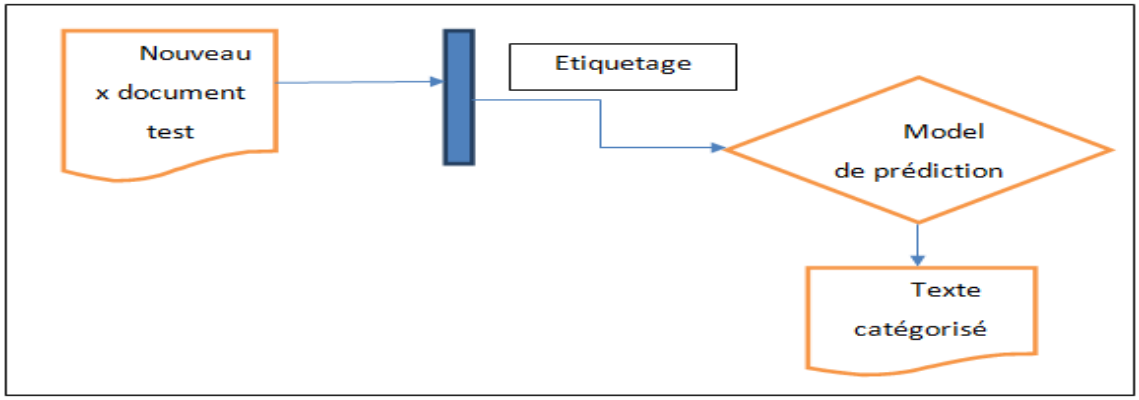


Figure 2 : la phase de test

On distingue deux types de problèmes auxquels l'apprentissage supervisé est appliqué :

- **La classification :** consiste à identifier les classes d'appartenance de nouveaux objets à partir d'exemples antérieurs connus, la variable à prédire peut donc prendre des valeurs discrètes appelées classes (9) (exemple : bon, moyen, mauvais).
- **La régression :** est utilisée lorsqu'il s'agit de prédire une variable continue, qui peut donc prendre un nombre infini de valeurs (exemple : température, gains monétaires, etc). Par exemple, pour une expérience de biochimie, on pourrait vouloir prédire le taux de résistance d'un organisme en fonction des taux de différentes substances qui lui sont administrées (9).

4.2 L'apprentissage non supervisé

Dans l'apprentissage non supervisé, aucun expert n'est disponible. Il vise à concevoir un modèle structurant l'information. La différence ici est que les comportements (ou catégories ou encore les classes) des données d'apprentissage ne sont pas connus, c'est ce que l'on cherche à trouver.

L'apprentissage non supervisé ne nécessite pas l'étiquetage des données de la base d'apprentissage. Son objectif est donc trouver tout seul des points communs parmi ses données d'entrée.

Dans ce contexte, nous utilisons le clustering pour identifier dans un ensemble de données, des sous-groupes présentant des caractéristiques proches. Cette méthode permet par exemple d'identifier des groupes (classes ou clusters) de personnes au sein d'un échantillon.

L'apprentissage non supervisé vise à construire des groupes d'objets similaires à partir d'un ensemble hétérogène d'objets. Chaque cluster issu de ce processus doit vérifier les deux propriétés suivantes :

- La cohésion interne : les objets appartenant à ce cluster sont les plus similaires possibles.
- L'isolation externe : les objets appartenant aux autres clusters sont les plus distincts possibles (10).

4.3 Apprentissage semi-supervisé

Ce type d'apprentissage utilise un ensemble de données étiquetées et non-étiquetées. Il se situe ainsi entre l'apprentissage supervisé qui n'utilise que des données étiquetées et l'apprentissage non-supervisé qui n'utilise que des données non-étiquetées. Ainsi, Il a été démontré que l'utilisation de données non-étiquetées, en combinaison avec des données étiquetées, permet d'améliorer significativement la qualité de l'apprentissage. Un autre intérêt provient du fait que l'étiquetage de données nécessite l'intervention d'un expert (8).

4.4 Apprentissage par renforcement

C'est une approche qui vise à résoudre des problèmes complexes. L'apprentissage par renforcement fait référence à une classe de problèmes d'apprentissage automatique, dont le but est d'apprendre à partir d'expériences, ce qu'il convient de faire en différentes situations, de façon à optimiser une récompense numérique au cours du temps.

L'apprentissage par renforcement a comme objectif d'entraîner un agent à se comporter de façon intelligente dans un environnement donné. La Figure 3 illustre qu'un agent interagit avec l'environnement en choisissant, à chaque temps donné, d'exécuter une action parmi un ensemble d'actions permises. Le comportement intelligent que doit apprendre cet agent est donné implicitement via un signal de renforcement qui, après chaque décision de l'agent, indique s'il a bien ou mal agi (récompense ou pénalité). L'agent doit donc se baser sur ce signal afin d'améliorer son comportement, qui est dicté par sa politique d'actions. À chaque temps donné, l'agent a à sa disposition un ensemble de caractéristiques ou indicateurs d'entrée, décrivant l'environnement. Par exemple, si l'agent correspond à un robot, ces indicateurs pourraient être obtenus à partir de capteurs sensoriels brossant un portrait de l'endroit où il se trouve (11).

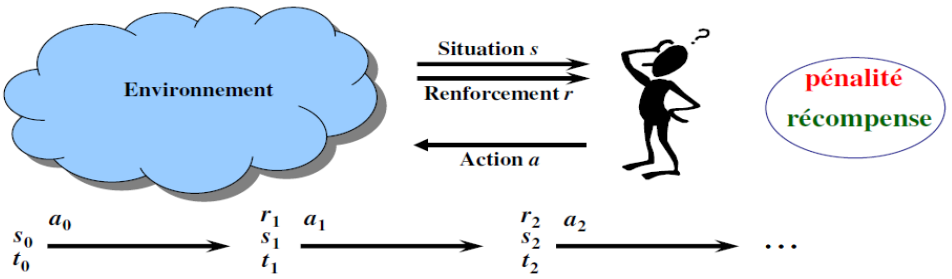


Figure 3 : Le principe de l'apprentissage par renforcement

4.5 Apprentissage par transfert

L'apprentissage par transfert peut être vu comme la capacité d'un système à reconnaître et appliquer des connaissances et des compétences, apprises à partir de tâches antérieures sur de nouvelles tâches ou domaines partageant des similitudes. L'apprentissage se fait donc en fonction d'un apprentissage effectué auparavant (12).

Dans l'apprentissage par transfert, le développeur peut prendre un modèle qui est développé sur un problème donné et le réutiliser sur un nouveau problème, tout en tenant compte des travaux antérieurs pour améliorer la précision du nouveau modèle.

5 Les modèles d'apprentissage automatique

Des diverses méthodes et algorithmes existent pour l'apprentissage automatique dans ses différentes formes pour cela nous présentons quelques-unes.

5.1 Approche basée sur les K plus proches voisins

K plus proches voisins (*K-Nearest Neighbors* ou KNN) est une méthode de classification supervisé. Chaque observation (ou exemple) de l'ensemble d'apprentissage est représentée par un point dans un espace à n dimensions où n est le nombre de variables prédictives. Pour prédire la classe d'une observation, on cherche les k points les plus proches de cet exemple. La classe de la variable cible, est celle qui est la plus représentée parmi les k plus proches voisins.

Il existe des variantes de la méthode où les k observations sont pondérés en fonction de leur distance de l'exemple à classifier. Ainsi, la classe attribuée à un exemple est celle totalisant le point le plus important parmi les K exemples considérés. En d'autres termes, les

observations les plus éloignées de notre exemple sont alors considérées comme moins importantes.

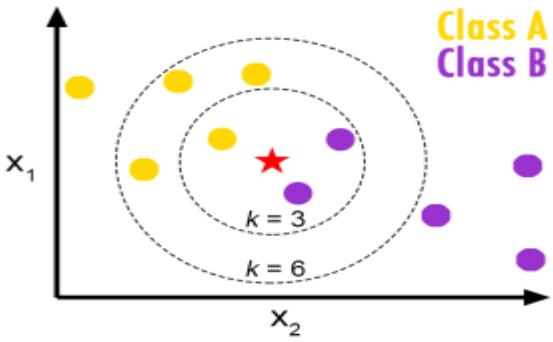


Figure 4 : Exemple d'application de l'algorithme des k plus proche voisins

La figure ci-dessus illustre la méthode des K plus proche voisins pour deux valeurs de K. pour k = 3 la classe majoritaire en partant du point central est la classe B. Cependant, avec une valeur de k = 6, la classe majoritaire devient alors la classe A (13).

Cette méthode est avantageuse, de part sa simplicité de conception. Mais elle est sensible au bruits et moins efficace lorsque le nombre de variables prédictives très grands.

5.2 La méthode k -moyenne

La méthode k-moyenne (*k-means* en anglais) appelée aussi algorithme des nuées dynamiques ou méthode de partitionnement autour des centres mobiles. Ce type d'algorithme est considéré comme un des algorithmes de classification non supervisé le plus utilisé, du fait de sa simplicité de mise en œuvre. Son objectif consiste à construire les k meilleurs centres de classes (clusters ou groupes) de l'ensemble de données d'apprentissage.

Chaque donnée d'apprentissage est représentée par un point dans un espace à n dimensions ou n est le nombre de données. À partir d'un ensemble d'apprentissage de M données $x(1), \dots, x(M)$, cet algorithme va repartir ces données en k classes (clusters) de manière à ce que la distance euclidienne qui sépare les points au centre du cluster auquel ils sont affectés soit minimale. Les étapes de l'algorithme sont :

- Choisir k points qui représentent la position moyenne des clusters.
- Répéter jusqu'à stabilisation des points centraux :

- ✓ Affecter chacun des M points au plus proche des k points centraux.
- ✓ Mettre à jour les points centraux en calculant les centres des k cluster (14).

L'avantage de cette méthode est qu'elle facilite l'implémentation des grands volumes de données. Mais Le choix du paramètre k n'est pas découvert il est choisi par l'utilisateur, aussi la solution dépend des k centre de gravité choisie lors de l'initialisation.

5.3 Les arbres de décision

Les arbres de décision (*decision trees* en anglais) sont des méthodes de l'apprentissage supervisé utilisées principalement dans le cadre de la classification et de la régression. Elles représentent une fonction qui prend en entrée un ensemble d'attributs et retourne une décision (sortie) qui est une valeur unique. Les entrées et les sorties peuvent être discrètes ou continues.

Un arbre de décision est une structure arborescente, prend ses décisions en exécutant une séquence de tests. Chaque nœud interne de l'arbre correspond à un test de la valeur d'un attribut et les branches qui sortent du nœud sont les valeurs possibles de l'attribut. La classe de la variable cible est alors déterminée par la feuille dans laquelle parvient l'observation à l'issue de la séquence de test.

La phase d'apprentissage consiste à trouver la bonne séquence de tests. Pour cela, on doit décider quels sont les bons attributs à garder. Un bon attribut divise les exemples en ensembles homogènes, c'est-à-dire qu'ils ne contiennent que des observations appartenant à la même classe. En revanche, un attribut inutile laissera les exemples avec presque la même proportion de valeurs pour la variable cible.

La Figure 5 montre un arbre de décision pour le jeu de Golf. L'objectif est de déterminer si le joueur est convenable pour le jeu de golf ou non (13).

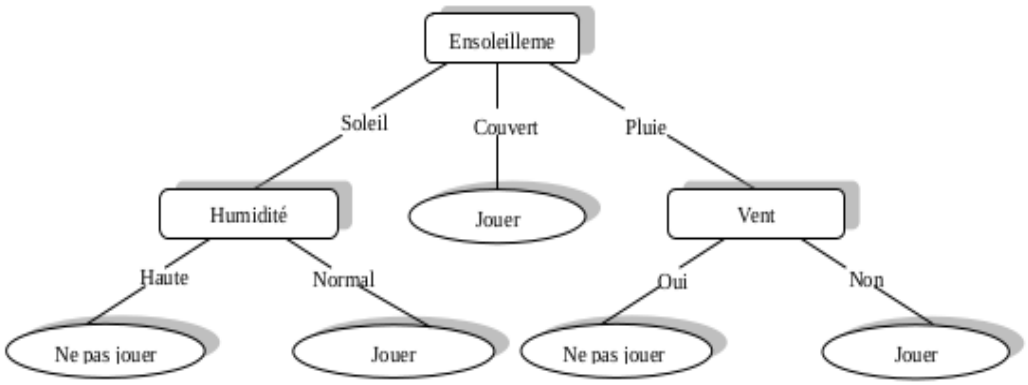


Figure 5 : Exemple d'application de l'algorithme d'arbre de décision

Cette méthode a différents avantages et inconvénients :

Parmi ses avantages :

- Simple à comprendre et à interpréter.
- Capable de traiter à la fois des données quantitatives et qualitatives.
- Performant sur de grands jeux de données.
- Nécessite un minimum de préparation de données. D'autres techniques nécessitent souvent la normalisation des données, des variables indicatrices doivent être créées et des valeurs vides doivent être enlevées.

Et concernant ses inconvénients, nous citons :

- Les arbres de décision peuvent être instables parceque les petites variations dans les données pourraient générer un arbre complètement différent.
- L'apprentissage par arbre de décision peut amener des arbres de décision très complexes, qui généralisent mal l'ensemble d'apprentissage (il s'agit du problème de surapprentissage[1]).

5.4 Classification naïve de Bayes

C'est un modèle d'apprentissage supervisé basé sur le théorème de Bayes. Il utilise les probabilités pour faire une prédiction sur un échantillon de données et décider à quelle classe appartient un nouvel exemple (ou donnée). Ce modèle suppose que l'existence d'une caractéristique pour une classe, est indépendante de l'existence d'autres caractéristiques,

[1] Surapprentissage : c'est lorsque on a un modèle d'apprentissage automatique tellement trop entraîné qu'il arrive à prédire à la perfection les données d'apprentissage mais qui n'arrive pas à généraliser sur les données de test.

raison pour laquelle on utilise l'adjectif naïve. Ainsi, pour déterminer la classe d'un nouvel exemple, la classification naïve de Bayes se contente de calculer la probabilité d'appartenance de l'exemple à chaque classe en se basant sur les exemples antérieurs. Pour ce faire, une supposition d'indépendance entre les caractéristiques définissant un exemple est faite (15).

La probabilité antérieure de chaque classe est calculée en connaissant, les valeurs des caractéristiques (ou attributs) des exemples d'entraînement. Ainsi, la classe avec la plus haute probabilité est assignée à chaque nouvel exemple. L'équation suivante montre la formule Bayésienne naïve qui fait la supposition que les valeurs de caractéristiques sont statistiquement indépendantes dans chaque classe.

$$p(C_i\,|v_1, v_2, \dots, v_n) = \frac{p(C_i)}{p(v_1, v_2, \ \dots v_n)} \prod_{j=1}^{n} p(v_j|C_i) \qquad (1)$$

Le côté gauche de cette équation est la probabilité antérieure de la classe C_i sachant les valeurs de caractéristiques $v_1, v_2, \dots, v_n$. Le dénominateur du côté droit de l'équation est calculé une seul fois pour l'ensemble d'exemples. L'apprentissage avec le classifieur Bayésien naïf est simple et implique à estimer simplement les probabilités dans le côté droit de cette équation à partir de l'ensemble d'entraînement (13).

Cette méthode a différents avantages et inconvénients :

Parmi ses avantages :

- L'algorithme offre de bonne performance.
- Un réseau bayésien est polyvalent, nous pouvons nous servir du même modèle pour évaluer, prévoir, diagnostiquer, ou optimiser des décisions.
- La représentation graphique d'un réseau bayésien est explicite, intuitive et compréhensible par un non spécialiste, ce qui facilite à la fois la validation du modèle, ses évolutions éventuelles et surtout son utilisation.
- Les réseaux bayésiens donnent la possibilité de rassembler et de fusionner des connaissances de diverses natures dans un même modèle.

Et concernant ses inconvénients, nous citons :

- La généralité du formalisme des réseaux bayésiens aussi bien en termes de représentation que d'utilisation les rend difficiles à manipuler à partir d'une certaine taille.
- La prédiction devient erronée si l'hypothèse d'indépendance conditionnelle est invalide.

5.5 Les machines à vecteurs de support

Les machines à vecteurs de support ou séparateurs à vaste marge (en anglais *Support Vector Machine*, SVM) sont des méthodes d'apprentissage supervisé destinées à résoudre des problèmes de classification et de régression et pouvant gérer des variables quantitatives et qualitatives.

Les SVMs opèrent sur des exemples (points de données) plongés dans un hyperespace, le but est de les catégoriser en différentes classes de telle sorte que la distance entre les différentes classes de données et la frontière qui les sépare soit maximale. Cet algorithme vise donc à créer une frontière de décision (ou de séparation) entre deux classes par exemple, permettant la prédiction d'étiquettes à partir d'un ou plusieurs vecteurs de caractéristiques (les axes de cet hyperespace). Cette frontière de séparation, appelée hyperplan, est orientée de manière à être aussi éloignée que possible des points de données les plus proches de chacune des classes. Ces points les plus proches sont appelés vecteurs de support. Dans les SVM, la frontière de séparation est choisie comme celle qui maximise la marge. La marge est considérée comme étant une distance entre l'hyperplan optimal et les vecteurs de support.

Un exemple d'un hyperplan est illustré dans la figure ci-dessous. Nous pouvons constater qu'il peut y avoir plusieurs hyperplans qui peuvent séparer deux classes, mais quant au choix optimal, la solution la plus intéressante peut être obtenue en gagnant la plus grande marge possible (16).

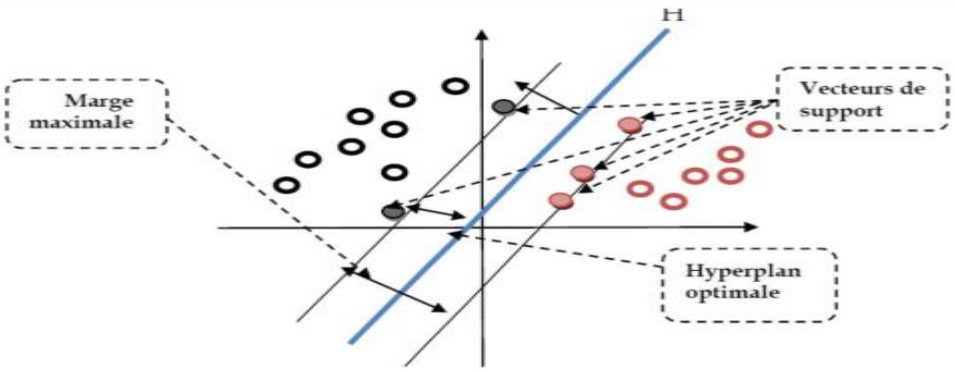


Figure 6 : Exemple d'application des machines à vecteurs support (SVM)

Cette méthode a différents avantages et inconvénients :

Parmi ses avantages :

- Très efficace en dimension élevée.
- Un nombre de paramètres faible à régler.
- Une grande vitesse d'apprentissage.
- Ils sont aussi efficaces dans le cas où la dimension de l'espace est plus grande que le nombre d'échantillons d'apprentissage.
- Cette méthode demande moins de mémoire.

Et concernant ses inconvénients, nous citons :

- Si le nombre d'attributs est beaucoup plus grand que le nombre d'échantillons, les performances seront moins bonnes.
- Comme il s'agit de méthodes de discrimination entre les classes, elles ne fournissent pas des estimations de probabilités.

5.6 Les réseaux de neurones

Les réseaux de neurones artificiels (RNA) ou formels sont des méthodes d'apprentissage automatique inspirées du modèle de neurone biologique, proposé par les biophysiciens Mac Cullogh et Pitts en 1943.

Les RNA sont des modèles de calcul parallèle pour la représentation et le traitement de l'information. Du fait de leur grande similitude avec le cerveau, les réseaux de neurones ont

des capacités semblables à celles de l'être humain et notamment ; l'apprentissage, la classification et la prédiction.

Les Réseaux de Neurones Artificiels constituent à ce jour une méthode de traitement de données bien comprise et bien maîtrisée. De façon formelle, un RNA est une fonction mathématique associant à des entrées, des grandeurs de sortie à l'aide de paramètres ajustables appelés des poids. Grâce au processus d'apprentissage, les RNAs sont des approximateurs universels parcimonieux capables d'estimer un modèle complexe avec une grande précision (17).

Cette méthode a différents avantages et inconvénients :

Les principales qualités et avantages des réseaux de neurones sont :

- La capacité d'adaptabilité et d'auto-organisation.
- La possibilité de résoudre des problèmes non-linéaires avec une bonne approximation.
- La rapidité d'exécution.

Et concernant ses inconvénients, nous citons :

- La difficulté d'interpréter le comportement d'un réseau de neurones est un inconvénient pour la mise au point d'une application.
- Il est souvent impossible d'utiliser les résultats obtenus pour améliorer le comportement d'un réseau de neurones.

Nous présentons plus en détail cette méthode d'apprentissage automatique dans le prochain chapitre.

6 Les applications de l'apprentissage automatique

Avec l'informatisation de la plupart des processus dans les entreprises, de nombreuses données reflétant l'activité de l'entreprise, de ses employés et de ses clients sont générées. Ces données contiennent les règles qui régissent ces processus, on peut donc envisager d'automatiser certaines tâches en utilisant ces données comme exemples d'apprentissage. On trouve donc des applications de l'apprentissage automatique dans quasiment tous les domaines :

- **Robotique** : Les robots dotés d'intelligence artificielle sont maintenant omniprésents, dans des domaines très variés surtout dans les usines. Ils aident, par exemple, dans la production de masse à automatiser des étapes de travail cohérentes.
- **Biens de consommation :** En disséquant les données collectées en magasin sur le comportement des consommateurs, les grandes surfaces peuvent, grâce à l'apprentissage automatique, réorganiser leurs rayons pour booster leurs ventes. L'apprentissage automatique donne aussi l'opportunité aux distributeurs d'optimiser la gestion de leurs stocks.
- **Santé :** L'apprentissage automatique est de plus en plus utilisé dans le secteur de la santé, notamment grâce à l'essor des objets connectés et autres capteurs permettant d'utiliser les données pour accéder aux données de santé d'un patient en temps réel. Cette technologie peut aussi aider les experts médicaux à analyser les données pour identifier des tendances alarmantes afin d'améliorer les diagnostics et les traitements.
- **Télécommunications :** Les entreprises de télécommunications pourront optimiser les investissements très lourds qu'ils doivent réaliser pour entretenir leur infrastructure en analysant les données collectées sur le terrain par des capteurs. L'apprentissage automatique leur permettra également de créer des assistants vocaux intelligents, capables de prendre en charge la majorité des appels qui sont aujourd'hui gérés par les salariés de leurs centres d'appels.
- **Automobile :** Dans le secteur automobile, l'apprentissage automatique combiné à la puissance de calcul informatique qui ne cesse de croître permet d'identifier en temps réel les éventuels obstacles qui se présentent sur une route, une compétence indispensable pour développer des véhicules autonomes. Cette technologie permet également de réaliser de la maintenance prédictive sur les automobiles afin de limiter au maximum les accidents.
- **Marketing** : Les sites web qui recommandent des produits basés sur les précédents achats de l'utilisateur utilisent l'apprentissage automatique pour analyser l'historique d'achat des clients et proposer des produits qui

pourraient les intéresser. La capacité de collecter les données, de les analyser et de les utiliser pour personnaliser l'expérience de shopping représente le futur de la vente au détail (18).

- **La classification de textes :** Grâce à l'apprentissage automatique, la classification de textes devient plus facile à faire. Cette dernières consiste donc à classer automatiquement les documents textuels dans une ou plusieurs catégories prédéfinies.

7 Les défis de l'apprentissage automatique

Les algorithmes et systèmes d'apprentissage automatique ont connu d'importantes avancées ces dernières années grâce à la disponibilité de grands volumes de données et du calcul intensif, sans oublier les avancées intéressantes en optimisation. Ils subsistent toutefois un nombre de limites et défis. Nous classons quelques uns comme suit :

- **Sources de données :** Les défis dans ce domaine sont nombreux : apprendre à partir de données hétérogènes, gérer des informations incertaines, identifier et traiter des événements rares au-delà des approches purement statistiques, et enfin obtenir de bonnes performances d'apprentissage avec peu de données, lorsque des sources de données massives ne sont pas disponibles.
- **Apprentissage sous contraintes** : La protection de la vie privée est la contrainte la plus importante à prendre en compte. Actuellement, plusieurs chercheurs spécialisés dans l'apprentissage automatique ont développé des algorithmes de protection de la vie privée. L'apprentissage automatique est amené à prendre en compte d'autres contraintes externes telles que des données décentralisées ou des limites énergétiques.
- **Processus d'apprentissage avec intervention humaine :** Les défis portent sur la mise en place d'une collaboration naturelle entre les algorithmes d'apprentissage automatique et les utilisateurs, afin d'améliorer le processus d'apprentissage. Pour ce faire, les systèmes d'apprentissage automatique doivent être en mesure de montrer leur état sous une forme compréhensible pour l'homme. De plus, l'utilisateur humain doit pouvoir obtenir des explications de la part du système sur n'importe quel résultat obtenu. Ces

explications pourraient également indiquer des niveaux de confiance, selon le cas.

- **Architectures de calcul :** Les systèmes d'apprentissage automatique modernes nécessitent du calcul intensif et un stockage de données efficace afin de s'adapter à la taille des données et aux dimensions des problèmes. Des algorithmes seront exécutés sur des GPU et d'autres architectures puissantes. De nouvelles recherches doivent s'attacher à améliorer les algorithmes d'apprentissage automatique et les formulations des problèmes afin de tirer le meilleur parti de ces architectures de calcul.

8 Evaluation d'un modèle d'apprentissage automatique

L'évaluation des performances d'un modèle d'apprentissage automatique est une tâche importante qui doit être faite avec soin. Pour ce faire, différentes mesures sont utilisées telles que la précision, le rappel, la F-mesure, l'indice rand, la valeur d'homogénéité, la valeur de complétude et la V-mesure. Ces mesures sont essentiellement utilisées en apprentissage supervisé et non supervisé.

8.1 Evaluation d'un modèle d'apprentissage supervisé

8.1.1 La précision

La précision est définie dans le domaine de l'apprentissage automatique, comme étant la probabilité conditionnelle qu'un exemple choisi aléatoirement soit bien classé par le système. Il s'agit du rapport entre le nombre de bonnes prédictions positives (solutions pertinentes) et le nombre de prédictions positives (vraies et fausses) (19). Elle mesure donc la capacité du système à refuser les solutions non-pertinentes.

La précision se réfère à la capacité du modèle à représenter approximativement le système modélisé. Le terme approximation est également utilisé pour exprimer la précision. Plus l'approximation au modèle est élevée, plus sa précision est augmentée. La proximité représente la similitude entre les réponses du système réel et le modèle (15).

La précision exprime le ratio entre le nombre d'exemples d'entraînements correctement classés dans la classe (C) sur le nombre total d'exemples d'entraînements auxquels la classe (C) est assignée, selon l'équation suivante :

$$Précision(C) = \frac{VP}{VP+FP} \qquad (2)$$

Avec : VP (Vrais positifs), FP (Faux positifs).

8.1.2 Le rappel

Le rappel mesure la largeur de l'apprentissage. Il correspond au rapport entre le nombre de bonnes prédictions positives et le nombre total d'exemples d'entraînement. Il mesure la capacité du système à donner toutes les solutions pertinentes. Par exemple, il peut mesurer le pourcentage de scripts publicitaires que le filtre a bloqué, à savoir son efficacité (19).

Le rappel exprime le ration entre le nombre d'exemples d'entraînements correctement classés dans la classe (C) sur le nombre total d'exemples d'entraînements appartenant à la classe (C), selon l'équation suivante :

$$Rappel(C) = \frac{VP}{VP+FN} \qquad (3)$$

Avec : VP (Vrais positifs), FN (Faux négatifs).

8.1.3 La F-mesure

La F-mesure est définie comme la moyenne harmonique de la précision et du rappel. Il s'agit d'une mesure qui est un compromis entre la précision et le rappel. Une valeur proche de 1 indique que la classification est de très bonne qualité (15). Il mesure la capacité du système à donner toutes les solutions pertinentes et à refuser les autres.

La F-Mesure est un indicateur qui combine le rappel et la précision. Elle est donnée par la formule suivante :

$$F - mesure = \frac{2*précision*rappel}{précision+rappel} \qquad (4)$$

8.2 Evaluation d'un modèle d'apprentissage non-supervisé.

8.2.1 L'indice Rand(RI)

C'est une mesure de similarité permettant de comparer les regroupements. Il exprime la proportion de paires qui sont dans le même cluster ou non. Il est basé sur le dépouillement et la comparaison de paires de points de l'accord et le désaccord entre les deux

regroupements ou deux règles de classement. RI peut être utilisé quand il y a un grand nombre de clusters.

L'indice de Rand a été motivé par des problèmes classiques de classification dans lesquels le résultat d'une méthode de classification doit être comparé à une classification correcte. La mesure de performance la plus courante pour ce problème, calcule la partie des exemples correctement classés (respectivement mal classés) à tous les autres exemples. Ainsi, l'indice de Rand est défini par :

$$R(C, C') = \frac{2(n_{11}+n_{00})}{n(n-1)} \qquad (5)$$

Avec :

- (C, C') : les paires d'échantillons et les paires de comptage qui sont attribuées dans les même ou différents grappes dans les groupements prédits et réels.
- n_{11} : le nombre de couples qui sont dans le même groupe.
- n_{00} : le nombre de couples qui sont dans des groupes différents.
- R varie de 0 (aucune paire classifiée de la même manière sous les deux groupes) à 1 (regroupements identiques). La valeur de R dépend de deux nombres : celui des grappes et des exemples (20).

8.2.2 La valeur d'homogénéité

Pour satisfaire le critère d'homogénéité, une classification doit assigner seulement les exemples appartenant à une même catégorie ou à une même classe. Une communauté est dite homogène, avec un score d'homogénéité égal à 1, si elle ne contient que des exemples issus d'une même classe. On notera que la partition discrète est totalement homogène (21).

8.2.3 La valeur de complétude

Pour satisfaire la notion de complétude, une classification doit affecter tous les éléments appartenant à une même catégorie réelle dans une même classe prédite. Nous noterons que la partition grossière est totalement complète. La complétude étant symétrique à l'homogénéité, sa définition est analogue à celle-ci (21).

8.2.4 La V-mesure

La V-mesure est une mesure basée sur la notion d'entropie et les travaux de la théorie de l'information (Rosenberg et Hirschberg, 2007). Elle est calculée à partir de deux sous-

indices, l'homogénéité et la complétude. La V-mesure est définie comme la moyenne harmonique des indices précédents (21).

9 Conclusion

Nous avons présenté à travers cette première partie des généralités sur l'apprentissage automatique, ses définitions, ses différents types ainsi que quelques algorithmes de l'apprentissage automatique. Nous avons cité quelques domaines d'applications où il peut intervenir.

En particulier, les réseaux de neurones artificiels sont les modèles les plus utilisés. Les RNA sont connus par leurs capacités et leurs rapidités de modélisation pour résoudre des problèmes de non linéarités par apprentissage et par la flexibilité des solutions techniques souhaitées dans plusieurs domaines, notamment dans la classification de textes.

A cet effet, le chapitre suivant sera consacré à la présentation des principes de bases des réseaux de neurones ainsi que la classification de textes. Pour cela, nous décrivons en détails les différentes étapes nécessaires pour notre approche qui est la classification des scripts.

Chapitre 2: Apprentissage à base de réseaux de neurones

1 Introduction

Les réseaux de neurones (RN) est un modèle de l'apprentissage automatique suivant un apprentissage supervisé et non supervisé. Ils sont fabriqués de structures cellulaires artificielles, constituent un modèle permettant d'aborder sous des angles nouveaux les problèmes de perception, de mémoire, d'apprentissage et de raisonnement. C'est une alternative très prometteuse pour contourner certaines limitations des algorithmes classiques. Grâce à leur traitement différent de l'information et à leurs mécanismes inspirés des cellules nerveuses (neurones). Ainsi, ils permettent de solutionner des problèmes jadis qualifiés de complexes.

Les réseaux de neurones connaissent ces dernières années un succès croissant dans divers domaines des sciences de l'ingénieur. Ils sont utilisés dans l'industrie pour le contrôle qualité et le diagnostic de panne, dans la finance pour la prévision et modélisation du marché et l'attribution de crédits, ainsi que dans la télécommunication et l'informatique pour l'analyse du signal et élimination du bruit et pour la reconnaissance de formes (bruits, images, paroles). Ils sont aussi appliqués avec succès à l'apprentissage de tâches de classification notamment à la classification de textes. Il permet de traiter du texte en suivant un ensemble des techniques relatives au traitement de textes.

Dans ce chapitre, nous présentons un ensemble de généralités et de concepts relatifs aux réseaux de neurones. Ensuite, nous verrons que leur fonctionnement dépend largement du choix d'une fonction dite d'activation ainsi que ses différents types. Après cela, nous donnons l'architecture des réseaux de neurones. Puis, et avant de conclure, nous décrivons brièvement quelques algorithmes d'apprentissage.

2 Réseaux de neurones : Généralités

Aujourd'hui, de nombreux termes sont utilisés dans la littérature pour désigner le domaine des réseaux de neurones artificiels, comme connexionnisme ou neuromimétique. En effet, les réseaux de neurones sont des modèles mathématiques imitant la structure et les fonctions des réseaux de neurones biologiques.

Dans ce qui suit, nous présentons ce qu'est un neurone formel. Puis, nous citons les définitions les plus proéminentes ainsi que la structure d'un réseau de neurone. Ensuite, nous donnons le fonctionnement des réseaux de neurones.

2.1 Neurone formel

Il est inspiré du neurone biologique. Par définition, un neurone formel est une fonction algébrique non linéaire, paramétrée et à valeurs bornées. La Figure 7 illustre la structure générale d'un neurone formel, qui est caractérisée par :

- Ses entrées $(X_1, X_2, \ldots, X_i, \ldots, X_n)$ peuvent être collectées depuis les données que l'on souhaite traiter ou les sorties d'autres neurones.
- La somme pondérée de ses entrées définissant le prétraitement (combinaison linéaire) effectué sur les entrées comme le montre l'équation :

$$\sum W_i . X_i + b$$

  Où W_i est le poids synaptique attaché à l'entrée i et le b désigne le seuil d'activation (biais).
- Sa fonction d'activation, ou d'état f_0, définissant l'état interne du neurone en fonction de son entrée totale.
- Une sortie calculant la sortie du neurone en fonction de son état d'activation (22).

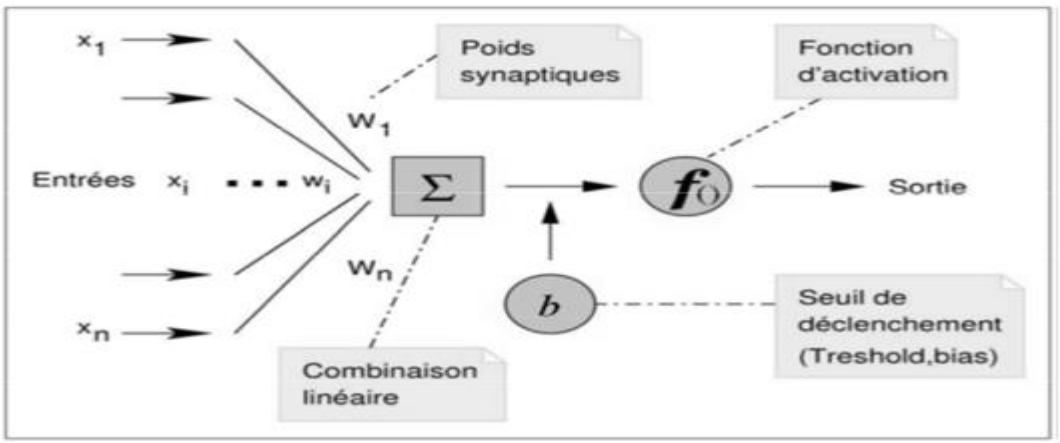


Figure 7 : Structure générale du neurone formel

2.2 Définitions d'un réseau de neurones artificiels

Définition 1 : Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit (23).

Définition 2 : Un réseau de neurones peut être considéré comme un modèle mathématique de traitement réparti, composé de plusieurs éléments de calcul non-linéaires (les neurones), opérant en parallèle et connectés entre eux par des poids (3).

Définition 3 : Un réseau de neurones artificiels est un ensemble de neurones formels (d'unités de calcul simples, de nœuds processeurs) associes en couches (ou sous-groupes) et fonctionnant en parallèle (14).

Dans un réseau, chaque sous-groupe (couche) fait un traitement indépendant des autres et transmet le résultat de son analyse au sous-groupe suivant. L'information donnée au réseau va donc se propager couche par couche, de la couche d'entrée à la couche de sortie, en passant soit par aucune, soit par plusieurs couches intermédiaires (dites couches cachées).

Habituellement (excepté pour les couches d'entrée et de sortie), chaque neurone dans une couche est connecté à tous les neurones de la couche précédente et de la couche suivante. Nous distinguons trois types de couches :

- **Couche d'entrée :** les neurones de cette couche reçoivent les valeurs d'entrées du réseau et les transmettent aux neurones cachés. Chaque neurone reçoit une valeur, il ne fait pas donc de sommation.
- **Couches cachées :** chaque neurone de cette couche reçoit l'information de plusieurs couches précédentes, effectue la sommation pondérée par les poids, puis la transforme selon sa fonction d'activation. Par la suite, il envoie cette réponse aux neurones de la couche suivante.
- **Couche de sortie :** elle joue le même rôle que les couches cachées, la seule différence entre ces deux types de couches est que la sortie des neurones de la couche de sortie n'est liée à aucun autre neurone (3).

2.3 Fonctionnements des réseaux de neurones

Un réseau de neurones combine plusieurs couches de traitement, utilisant des éléments simples fonctionnant en parallèle et inspirés du système nerveux biologique. Il se compose d'une couche d'entrée, d'une ou de plusieurs couches cachées et d'une couche de sortie. Les couches sont interconnectées par des nœuds, ou neurones, chaque couche utilisant la sortie de la couche précédente en guise d'entrée. Il peut apprendre à partir de données. Il peut

ainsi être entraîné sur de nombreux exemples en vue de reconnaître des modèles au niveau de l'image ou de texte par exemple, aussi de classer des données et prédire les cours du marché.

2.3.1 La propagation directe

Le réseau de neurones une fois entraîné, calcule sa sortie en propageant l'information et en effectuant les calculs depuis la couche d'entrée jusqu'à la couche de sortie (*forward-propagation*), en appliquant couche par couche les fonctions des neurones. Pour obtenir une bonne valeur de sortie, il est donc nécessaire de configurer les paramètres des neurones (les poids) pour qu'à chaque instance de nos données, une valeur convenable lui soit associée.

2.3.2 La rétro-propagation

La rétro-propagation (en anglais *backpropagation*) consiste dans un premier temps à circuler vers l'avant les données d'entrées jusqu'à l'obtention d'une entrée calculée par le réseau. Puis dans un second temps, elle compare la sortie calculée à la sortie réelle connue. Les poids sont modifiés de telle sorte qu'à la prochaine itération, l'erreur commise entre la sortie calculée est minimisée. En prenant en considération la présence des couches cachées, l'erreur est rétro-propagée vers l'arrière jusqu'à la couche d'entrée tout en modifiant la pondération des neurones (toujours dans le but de minimiser cette erreur). Le processus est répété sur tous les exemples jusqu'à obtenir une erreur de sortie considérée comme négligeable.

En plus du bon choix de l'architecture d'un réseau de neurones, il effectue un ajustement de paramètre pour chaque neurone jusqu'à obtenir les sorties attendues. Cette partie c'est l'entraînement. Lors de l'entraînement du réseau de neurones nous avons un ensemble de paramètres à entraîner ou à ajuster (par exemple les poids des neurones) suivant un ensemble de données d'entraînement. Quand cet ensemble de données est parcouru complètement une fois, nous appelons cela une *epoch.* Un entraînement valide d'un réseau de neurones nécessite plusieurs *epochs*. Lors de chaque epoch, un taux d'apprentissage (*learning rate*) est fixé et permet de définir avec quelle proportion les poids des neurones sont mis à jour. Celui-ci est choisi de telle sorte à ce qu'il soit grand lors des premières epochs (au début de l'entraînement), puis est progressivement réduit lors des epochs suivantes.

Il existe plusieurs algorithmes d'entraînement qui permettent de mettre à jour les poids associés à chaque neurone au sein du réseau dans le but d'ajuster la réponse du réseau à l'expérience et aux exemples, mais aussi d'ajuster d'autres paramètres comme le taux d'apprentissage. Ces ajustements sont faits de sorte à minimiser l'erreur entre la sortie du réseau et le résultat désiré en se basant sur le calcul d'une valeur de gradient (24). Le gradient est la dérivée partielle de l'erreur par rapport aux poids synaptiques. Il sert à initialiser les poids d'un réseau et de minimiser l'erreur de sortie du réseau par rapport à l'ensemble de ses pondérations. Il permet de trouver les poids qui minimisent le nombre d'erreurs commises sur l'ensemble d'apprentissage.

La qualité d'apprentissage ne dépend pas uniquement de ces algorithmes. Elle dépend aussi de l'architecture du réseau de neurones et des fonctions d'activations choisies (24). Cette fonction d'activation est une fonction appliquée à la somme pondérée de l'entrée de chaque neurone par ses poids.

3 Fonction d'activation

La fonction d'activation (ou fonction de transfert) sert à convertir le résultat de la somme pondérée des entrées d'un neurone en une valeur de sortie. Cette conversion s'effectue par un calcul de l'état du neurone en introduisant une non-linéarité (c'est de prendre toutes les données ou aucune) dans le fonctionnement du neurone.

Le biais joue un rôle de seuil, quand le résultat de la somme pondérée dépasse ce seuil, l'argument de la fonction de transfert devient positif ou nul. Dans le cas contraire, il est considéré négatif. Finalement, si le résultat de la somme pondérée est :

- En dessous du seuil, le neurone est considéré comme non-actif.
- Aux alentours du seuil, le neurone est considéré en phase de transition.
- Au-dessus du seuil, le neurone est considéré comme actif.

Il existe plusieurs types de fonctions de transfert qui peuvent être utilisées dans les RNA, les fonctions d'activation souvent utilisées sont représentées ci-dessous.

3.1 Fonction Linéaire

C'est une fonction simple de la forme :

$$f(x) = ax \text{ Ou } f(x) = x$$

La fonction Linéaire convertit ses entrées en sortie sans une très grande modification ou alors sans aucune modification. Nous restons ici dans une situation de proportionnalité (quand on peut passer d'une série de nombres à une autre, en multipliant ou en divisant par un même nombre) (25). Elle est utilisée en couche de sortie lorsqu'il s'agit d'une régression.

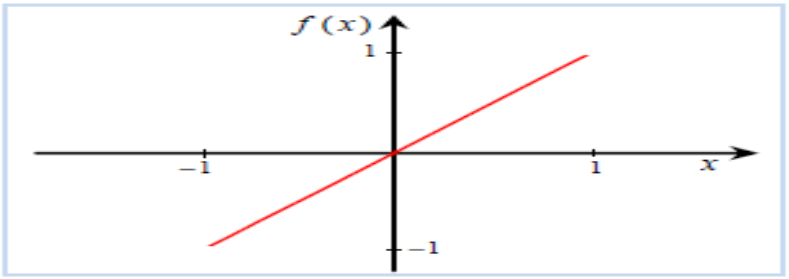


Figure 8 : La fonction Linéaire

3.2 Fonction Sigmoïde

La fonction Sigmoïde est une fonction continue. Elle est utilisée lorsque les valeurs à prédire par le RN sont comprises dans l'intervalle [0, 1]. Son but est d'exprimer sa valeur de sortie sous forme d'une probabilité, si la valeur en entrée est un très grand nombre positif, la fonction convertira cette valeur en une probabilité de 1. A l'inverse, si la valeur en entrée est un très petit nombre négatif, la fonction convertira cette valeur en une probabilité de 0 (24).

La fonction Sigmoïde est définie par :

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

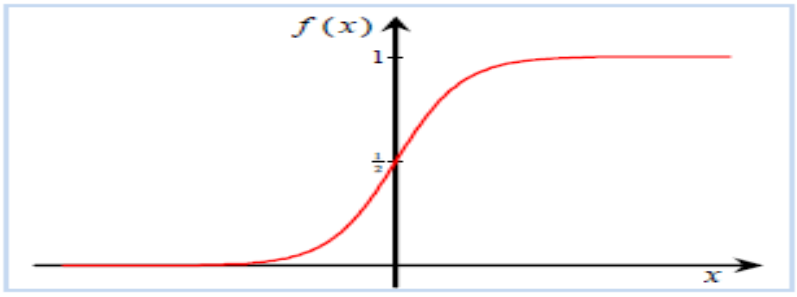


Figure 9 : La fonction Sigmoïde

3.3 Fonction Tangente Hyperbolique

La fonction Tangente Hyperbolique (Tanh) similaire à la fonction Sigmoïde. La différence avec la fonction Sigmoïde est que la fonction Tanh produit un résultat compris entre -1 et 1. Elle produit généralement de meilleurs résultats que la fonction Sigmoïde en raison de sa symétrie, où ses petites entrées négatives tendent vers -1 et ses grandes entrées positives tendent vers 1. Idéale pour les perceptrons multicouches, en particulier, pour les couches cachées (25).

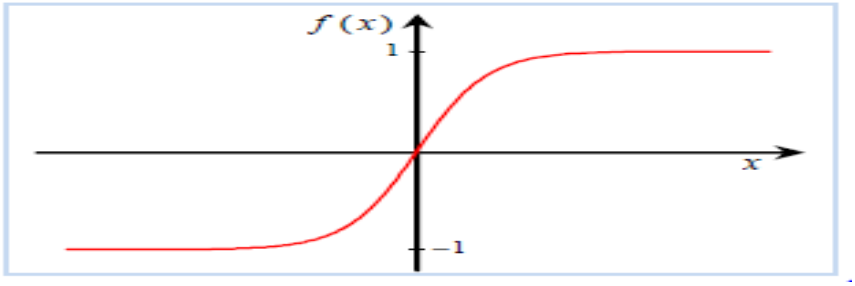


Figure 10 : La fonction Tanh

3.4 Fonction Relu / Leaky Relu

La fonction Relu (Unité de Rectification Linéaire) est crée pour palier au problème de saturation (est la situation où le gradient reste presque nul après chaque période de temps durant le processus d'apprentissage) des deux fonctions (Sigmoïde et Tanh).

Relu désigne la fonction réelle non-linéaire interprétée par la formule :

$$f(x) = \max(0, x).$$

Si l'entrée est négative alors la sortie est 0, si l'entrée est positive alors la sortie est x. Cette fonction d'activation augmente considérablement la convergence du réseau et ne sature pas.

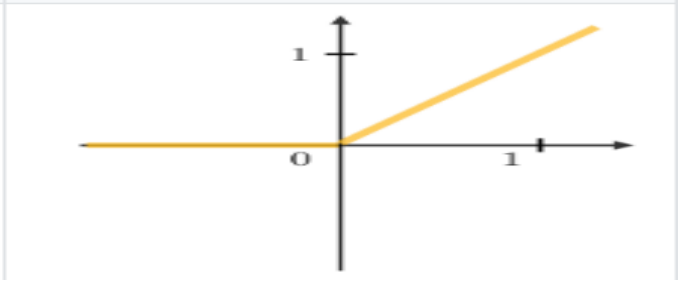


Figure 11 : La fonction Relu

L'inconvénient de la fonction Relu est que si la valeur d'entrée est négative, le neurone reste inactif, ainsi les poids ne sont pas mis à jour et le réseau n'apprend pas en raison des gradients de 0 dans la partie négative.

La fonction Leaky Relu essaye de résoudre le problème de la fonction Relu lorsque l'entrée est négative. Le concept est lorsque l'entrée est négative, il aura une petite pente négative de 0,01 ou plus. Cette fonction élimine le problème d'inactivité de la fonction Relu pour les valeurs négatives (25).

La fonction Leaky relu est donnée par la formule :

$$f(x) = \max(0.01x, x)$$

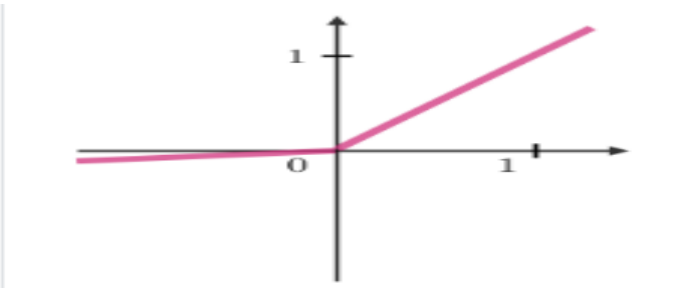


Figure 12 : La fonction Leaky Relu

3.5 Fonction SoftMax

La fonction SoftMax peut gérer plusieurs entrées. Elle attribue des probabilités à chaque entrée. Elle écrase les sorties de chaque unité entre 0 et 1 comme la fonction Sigmoïde. Mais elle divise également chaque sortie de manière à ce que la somme totale des sorties soit

égale à 1. Cette contrainte supplémentaire permet de faire converger l'apprentissage plus rapidement qu'il ne le ferait autrement.

Essentiellement utilisée (mais pas uniquement) pour des tâches de classification. Permet de construire des réseaux de neurones avec plusieurs sorties normalisées ce qui la rend particulièrement adapté à la création de classifications par les réseaux de neurones avec des sorties probabilistes. Autrement dit, elle est souvent utilisée dans la dernière couche d'un réseau de neurones utilisé comme classifieur multi-classes. Elle permet d'attribuer des probabilités à plusieurs classes qui s'additionnent à 1.

La fonction softmax est interprétée par la formule :

$$\sigma(Z)_j = \frac{e^{Z_j}}{\sum_{k=1}^{K} e^{Z_k}}$$

Où z est un vecteur des entrées du calque de sortie (si vous avez 10 unités de sortie, il y a 10 éléments dans z). Et encore une fois, j indexe les unités de sortie, donc j = 1, 2, ..., k.

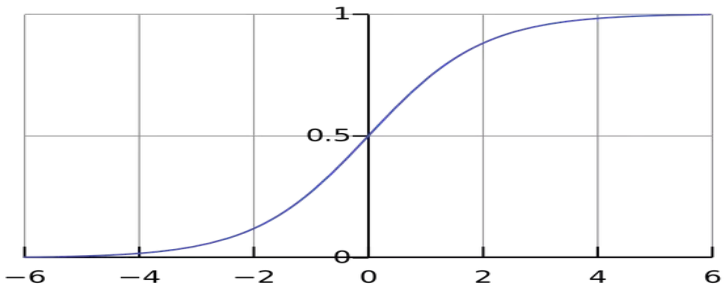


Figure 13 : La fonction SoftMax

4 Architecture de réseaux de neurones

La mise en oeuvre des réseaux de neurones comporte une partie conception, dont l'objectif est de permettre le choix de la meilleure architecture possible. L'architecture d'un RNA consiste à définir les différentes connexions entre les neurones.

L'architecture du RNA joue un rôle important dans le processus d'apprentissage. Chaque architecture à sa propre organisation selon la topologie de connexion des neurones. Nous distinguons trois structures de réseau.

4.1 Réseaux de neurones non bouclés

Un réseau de neurones non bouclé (appelé aussi statique) est représenté graphiquement par un ensemble de neurones connectés entre eux. L'information circule des entrées vers les sorties sans retour en arrière ; c'est-à-dire à partir d'un neurone quelconque, en suivant les connexions, nous ne pouvons pas revenir au neurone de départ. Ce type de réseaux est utilisé pour effectuer des taches d'approximation de fonction non linéaire, de la classification ou de la modélisation de processus statiques non linéaires (26).

4.1.1 Perceptron

Le perceptron simple est le premier réseau de neurones fonctionnel muni d'une méthode d'apprentissage. Il est linéaire et monocouche. Dans sa conception première, illustrée par la Figure 14, il est composé d'une couche de neurones d'entrée appelée (rétine), qui reçoit les informations décrivant l'objet à analyser ou à reconnaître. Tous les neurones de la rétine sont reliés par des liens pondérés à une couche de sortie (cellules de décision) (27).

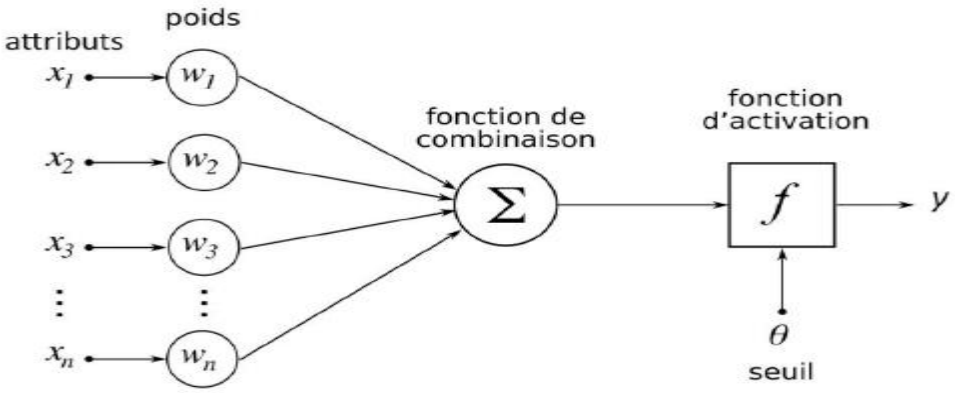


Figure 14 : Schéma de perceptron

4.1.2 Réseau à couches entièrement connectées

C'est le réseau de neurones statique le plus utilisé. Les neurones sont arrangés par couche. Les neurones de la première couche reçoivent des entrée, ils calculent leurs sorties qui sont transmises aux neurones de la seconde couche qui calculent eux même leurs sorties et ainsi de suite de couche en couche jusqu'à celle de sortie. Chaque neurone dans la couche cachée est connecté à tous les neurones de la couche précédente et de la couche suivante, il n'y a pas de connexions entre les neurones d'une même couche. Les Réseaux de neurones à

couches entièrement connectées peuvent opérer sur des données non linéairement séparables (28).

La Figure 15 donne l'exemple d'un réseau contenant trois entrées, deux couches cachées et une couche de sortie. La couche d'entrée représente toujours une couche virtuelle associée aux entrées du réseau, elle ne contient aucun neurone tandis que les couches suivantes représentent des couches effectives de neurones (26).

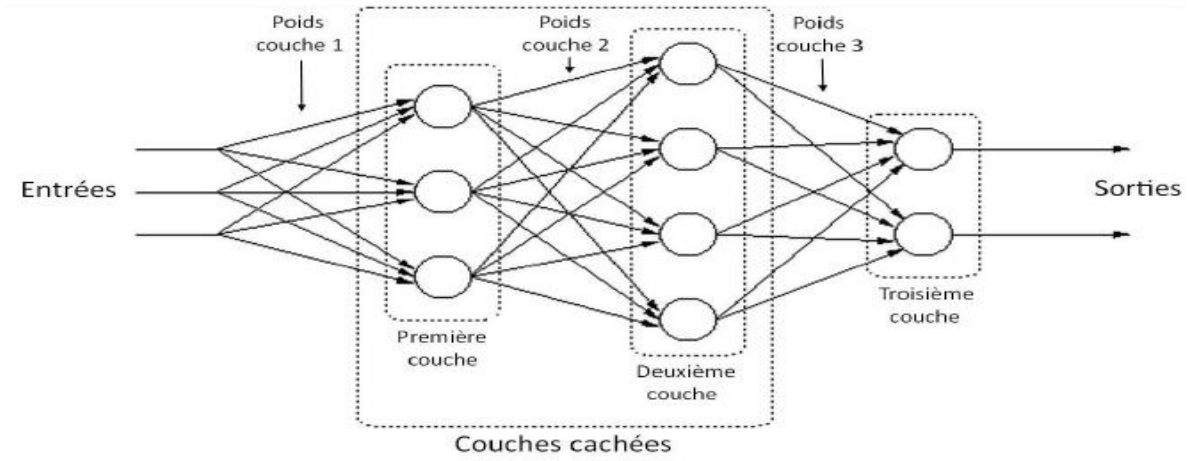


Figure 15 : Schéma de réseau à couches entièrement connectées

4.2 Réseaux récurrents

Les réseaux récurrents (RNN en anglais *Recurrent Neural Network*) se distinguent des réseaux non bouclés par la connexion des sorties de neurones avec leurs entrées. La sortie d'un neurone peut être connectée avec l'entrée du même neurone ou avec celles des autres neurones. L'importance de ces réseaux est qu'ils permettent d'apprendre la dynamique de systèmes, c'est-à-dire qu'ils peuvent imiter le comportement temporel en insérant des délais dans les boucles, reliant l'entrée à la sortie du réseau ou dans des couches internes. La Figure 16 illustre un réseau récurrent dont les sorties sont bouclées avec les entrées de tous les neurones (24).

Ces réseaux sont utilisés pour effectuer des taches de modélisation des systèmes dynamiques, de commande de processus ou de filtrage (28).

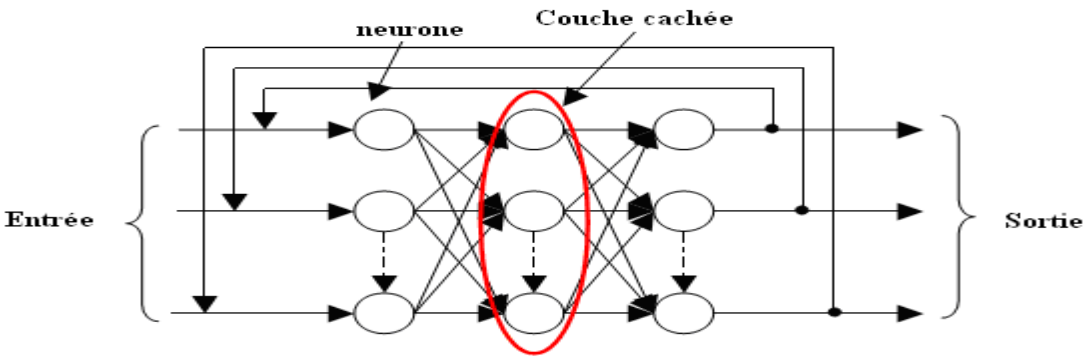


Figure 16 : Schéma de Réseaux récurrents

4.2.1 LSTM

LSTM (*Long Short Term Memory*) est proposée par Hochreiter et Schamid huber. Il apparu comme un modèle efficace et évolutif pour plusieurs problèmes d'apprentissage liés aux données séquentielles.

L'architecture LSTM consiste en un ensemble de sous-réseaux récurrents particuliers (appelés blocs de mémoire) situés au niveau de la couche cachée, et contenant chacun une ou plusieurs cellules de mémoire. Ces blocs de mémoire spéciaux permettent le maintient d'information en mémoire pour de longues périodes de temps.

Une cellule (ou bloc) LSTM comme le montre la Figure 17 est composé d'une mémoire et de trois portes : la porte d'oubli, la porte d'entrée et la porte de sortie. La porte d'oubli (*forget gate*) contrôle quelle est la partie de la cellule précédente qui sera oubliée et permet aussi de réinitialiser l'état de la mémoire au cours de la séquence. La porte d'entrée (*input gate*) doit choisir les informations pertinentes qui seront transmises à la mémoire. La sortie (*output gate*) contrôle quelle partie de l'état de la cellule sera exposée en tant que état caché (29).

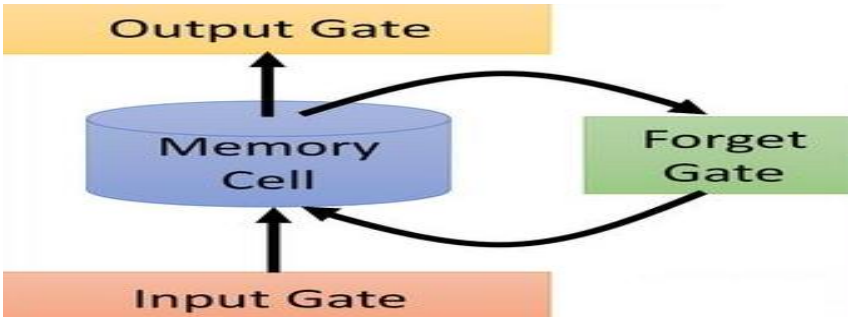


Figure 17 : Schéma d'une cellule LSTM

4.2.2 GRU

GRU (*Gated Recurrent Unit*) est un autre type de réseau récurrent. Ce type de réseau a le même principe que le LSTM mais son architecture est différente. Il est constitué que de deux portes : une porte de réinitialisation, et une porte de mise-à-jour. La porte de réinitialisation (fusion de portes d'entrée et d'oubli) détermine comment combiner la nouvelle entrée avec la mémoire précédente, et la porte de mise à jour définit la quantité de mémoire précédente à conserver.

L'intérêt de GRU comme le montre la Figure 18 par rapport au LSTM est d'être plus léger en termes de calculs pour une performance similaire (30).

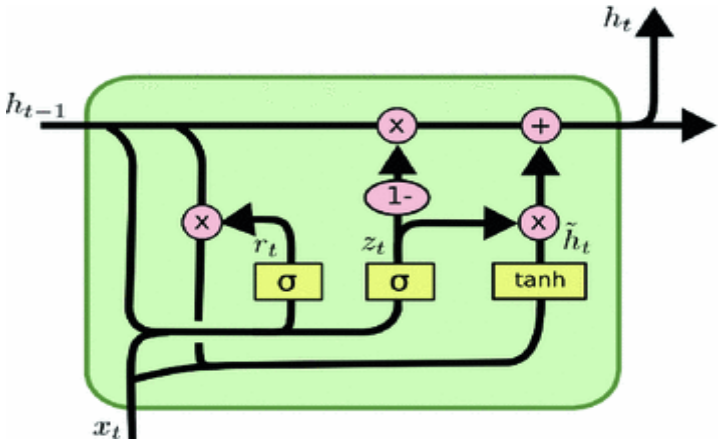


Figure 18 : Schéma de Réseaux GRU

4.3 Réseaux convolutionnels

Les réseaux de neurones convolutionnels (ou réseau de neurones à convolution, ou CNN ou *ConvNet*) sont des réseaux de neurones artificiels acycliques (non bouclés) dans lesquels la connexion entre les neurones est inspirée par le cortex visuel des animaux. Ce modèle de réseaux est proche d'un perceptron multicouches. Il est composé de neurones organisés en couche. Ces neurones ont des poids ou des paramètres, des biais et une fonction d'activation. La principale différence est que les *ConvNet* sont réputés pour leur robustesse aux faibles variations d'entrée et le faible taux de prétraitement nécessaires à leur fonctionnement, et ne requièrent aucun choix d'extracteur de caractéristiques spécifiques. Ce modèle s'est révélé très efficaces dans divers domaines tels que la reconnaissance d'images et de parole ainsi que dans la classification de textes.

Les CNN sont constitués essentiellement de deux types de neurones. Les neurones dits simples, détectent les caractéristiques liées à la forme. Les neurones dits complexes sont sensibles à la forme entière.

Dans les réseaux CNN chaque neurone prend des entrées, effectue une convolution (que nous présentons plus loin dans cette section) et il choisi éventuellement une fonction d'activation (non-linéarité).

Un réseau CNN possède en général plusieurs couches de convolutions et de pooling ainsi que des couches de correction et une couche entièrement connectée les unes à la suite des autres avec des dimensions qui augmentent et se réduisent au fur et à mesure de l'avancée dans le réseau.

Une architecture CNN contient comme le montre la Figure 19 des couches de convolution, de pooling, de correction et une couche entièrement connectée (31).

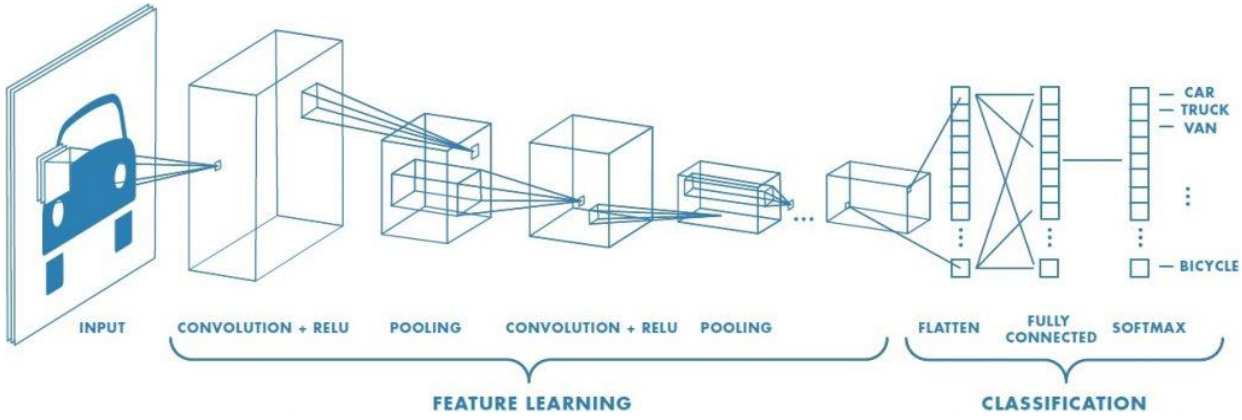


Figure 19 : Schéma de Réseaux Convolutionnels

En effet, avant de connecter à la couche entièrement connectée, les données qui sont en 2 dimensions dont mises à plat et concaténées en un vecteur de caractéristiques, appelé code CNN.

4.3.1 Couche de convolution

Dans la couche de convolution, au lieu de faire un produit scalaire entre les entrées et les poids de chaque neurone, on applique un produit de convolution. Ce produit de convolution sert à extraire des caractères spécifiques des données en entrée. Une donnée est donc passée à travers une succession de filtres, ou noyaux de convolution, créant de

nouvelles données appelées cartes de convolutions. Le produit de convolution est la moyenne pondérée de l'ensemble d'entrée qui est défini par :

$$g(x,y) = \sum_{i,j}^{h_x,h_y} h(i,j)f(x+i,y+j)$$

Où f c'est l'entrée à traiter et h c'est le filtre à appliquer avec taille (h_x, h_y).

Trois paramètres permettent de dimensionner le volume de la couche de convolution : la profondeur, le pas et la marge (31).

- **Profondeur de la couche** : nombre de noyaux de convolution (ou nombre de neurones associés à un même champ récepteur (surface de traitement)).
- **Le pas** : contrôle le chevauchement des champs récepteurs. Plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.
- **La marge** : permet de contrôler la dimension spatiale du volume de sortie. En particulier, il est parfois préférable de conserver la même surface que celle du volume d'entrée (31).

4.3.2 Couche de *pooling*

Après chaque couche de convolution, il peut y avoir une couche de *pooling* (couche d'agrégation en français). Elle est une forme de sous-échantillonnage de l'entrée. Elle a un potentiel d'améliorer les résultats d'un CNN pour certaines applications et elle est souvent utilisée pour la classification de textes. Il est donc fréquent d'insérer périodiquement une couche de *pooling* entre deux couches convolutives successives d'une architecture CNN (31).

L'étape de *pooling* consiste en une sélection de valeur (max, min, moyenne,..) selon des fenêtres coulissantes à la matrice issue de la convolution (le résultat de produit de convolution). Il se traduit mathématiquement par la fonction maximum de plusieurs noyaux de convolution.

4.3.3 Couche de correction (Relu)

Elle permet d'améliorer l'efficacité du traitement en intercalant entre les couches de traitement une couche qui va opérer une fonction mathématique (fonction d'activation

Relu) sur les signaux de sortie. Cette fonction force les neurones à retourner des valeurs positives (31).

4.3.4 Couche entièrement connectée (FC)

Après plusieurs couches de convolution et de pooling, le raisonnement de haut niveau dans le réseau neuronal se fait via des couches entièrement connectées. Les neurones dans une couche entièrement connectée ont des connexions vers toutes les sorties de la couche précédente (31).

4.3.5 Blocs résiduels

Les blocs résiduels sont une organisation particulière des couches précédentes permettant de pallier à certains problèmes d'entraînement pour des RNA profonds (avec un nombre important de couches cachées).

Les entrées de blocs résiduels sont directement fusionnées aux sorties par concaténation. Grâce aux connexions résiduelles, la propagation de l'information devient plus fluide avec une vitesse d'entraînement améliorée.

Ils consistent à utiliser des connexions sautées qui permet de prendre la sortie de la fonction d'activation d'une couche et de la transmettre à une autre couche plus loin dans le réseau comme le montre la Figure 20.

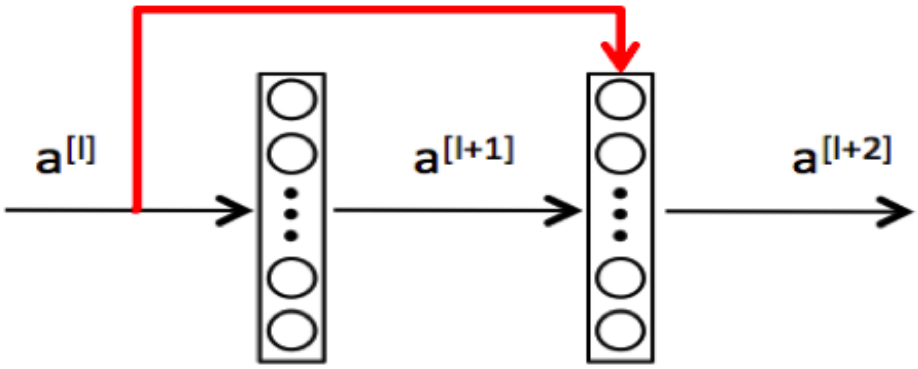


Figure 20 : Schéma d'un bloc résiduel

5 Algorithmes d'apprentissage pour les réseaux de neurones

Toute l'information que peut contenir un réseau neuronal réside dans les poids synaptiques. L'algorithme d'apprentissage est la méthode mathématique qui va modifier les

poids de connexions afin de converger vers une solution qui permettra au réseau d'accomplir la tâche désirée.

Plusieurs algorithmes itératifs peuvent être mis en œuvre. Nous présentons quelques-uns dans ce qui suit.

5.1 Rétropropagation du gradient

La rétropropagation du gradient (*Backpropagation* en anglais) est une méthode adaptée au réseau à couches entièrement connectées. Elle permet de calculer le gradient de l'erreur pour chaque neurone du réseau, de la dernière couche vers la première. Le principe de la rétropropagation peut être décrit en trois étapes fondamentales :

- Acheminement de l'information à travers le réseau.
- Rétropropagation des sensibilités de réseau (propager l'erreur) et calcule du gradient.
- Ajustement des paramètres par la règle du gradient approximé (3).

L'algorithme du gradient a pour but de converger de manière itérative vers une configuration optimisée des poids synaptiques qui permet d'adapter les poids des connexions du réseau de manière à minimiser la somme des erreurs sur tous les neurones de sortie afin d'obtenir une sortie voulue lors de l'entraînement. Cet état peut être un minimum local de la fonction à optimiser et idéalement, un minimum global de cette fonction (dite fonction de coût ou fonction d'erreur qui sert à calculer la moyenne des erreurs obtenues pour chacun des exemples de la base d'apprentissage et cette moyenne est minimal lorsque les poids et les biais sont ajustées).

La fonction de coût est non linéaire au regard des poids synaptiques. Elle dispose également d'une borne inférieure et moyennant quelques précautions lors de l'apprentissage, les procédures d'optimisation finissent par aboutir à une configuration stable au sein du réseau de neurones (22).

5.2 Adagrad

AdaGrad est un algorithme qui vise à résoudre le problème de l'adaptation du taux d'apprentissage aux différents paramètres à optimiser. Il s'ajuste automatiquement en

fonction de l'éparsité des paramètres (poids du réseau). Adagrad abaisse progressivement le taux d'apprentissage mais pas de la même manière pour tous les paramètres (32).

En d'autres termes, adagrad est un algorithme d'optimisation qui ne fait qu'adapter le taux d'apprentissage aux paramètres (poids du réseau), effectuant de plus grandes mises à jour pour les caractéristiques moins fréquentes et de plus petites mises à jour pour les caractéristiques plus fréquentes (13).

5.3 Adam

Adam (*Adaptive Moment Estimation*) est l'une des méthodes d'optimisation basée sur les gradients. Son principe est le même que pour Adagrad. Elle est simple à mettre en œuvre et efficace sur le plan informatique. En outre, elle requiert également peu de mémoire, est invariante au redimensionnement diagonal des gradients (Adam est invariant pour multiplier le gradient par une matrice diagonale avec uniquement des facteurs positifs) et convient bien aux problèmes importants en termes de données ou de paramètres (poids du réseau) (33).

C'est une méthode de taux d'apprentissage adaptative. Cela signifie qu'elle calcule des taux d'apprentissage individuels pour différents paramètres (poids du réseau). Son nom est dérivé de l'estimation adaptative du moment du gradient. C'est pourquoi Adam utilise des estimations du moment du premier ordre (la moyenne de gradient) et du moment du second ordre du gradient (le gradient carré) pour adapter le taux d'apprentissage à chaque poids du réseau neuronal. Elle fonctionne comme suit :

- premièrement, elle calcule la moyenne pondérée de manière exponentielle des gradients passés ;
- deuxièmement, elle calcule la moyenne pondérée de manière exponentielle des carrés de gradients passés ;
- troisièmement, les paramètres sont mis à jour en utilisant les moyennes calculées ;

5.4 Nastrov-Adam

Nesterov-Adam(NAdam) est une méthode de descente de gradient très adaptée à la classification. Elle intègre la dynamique de la méthode de *Nesterov momentum* (elle accélère le processus de calcul du gradient) dans la méthode d'optimisation d'Adam (qui utilise une

combinaison de vitesse de modification des paramètres et de taux d'apprentissage adaptatif pour améliorer les calculs de gradient). Cela permet de fournir une méthode convergente efficace et rapide pour la descente de gradient (34).

6 Apprentissage automatique et classification de textes

Les classifications classiques (manuelles) d'une grande quantité des documents textuelles s'avèrent très coûteuses en temps et en personnel. Cette opération peut être perçue comme subjective puisque basée sur l'interprétation du document. Pour cette raison, beaucoup de travaux ont été consacré à la classification automatique de textes.

Actuellement, la classification de textes est devenue un domaine de recherche très actif et l'automatisation de cette opération est devenue un enjeu pour la communauté scientifique. Les travaux évoluent considérablement et beaucoup de logiciels de classification des textes ont fait leur apparition pour certaines tâches.

Le processus de catégorisation de textes intègre la construction d'un modèle de prédiction qui, en entrée, reçoit un texte et en sortie, lui associe une ou plusieurs étiquettes.

Le déroulement d'une classification de textes suit quatre étapes principales résumées par la Figure 21.

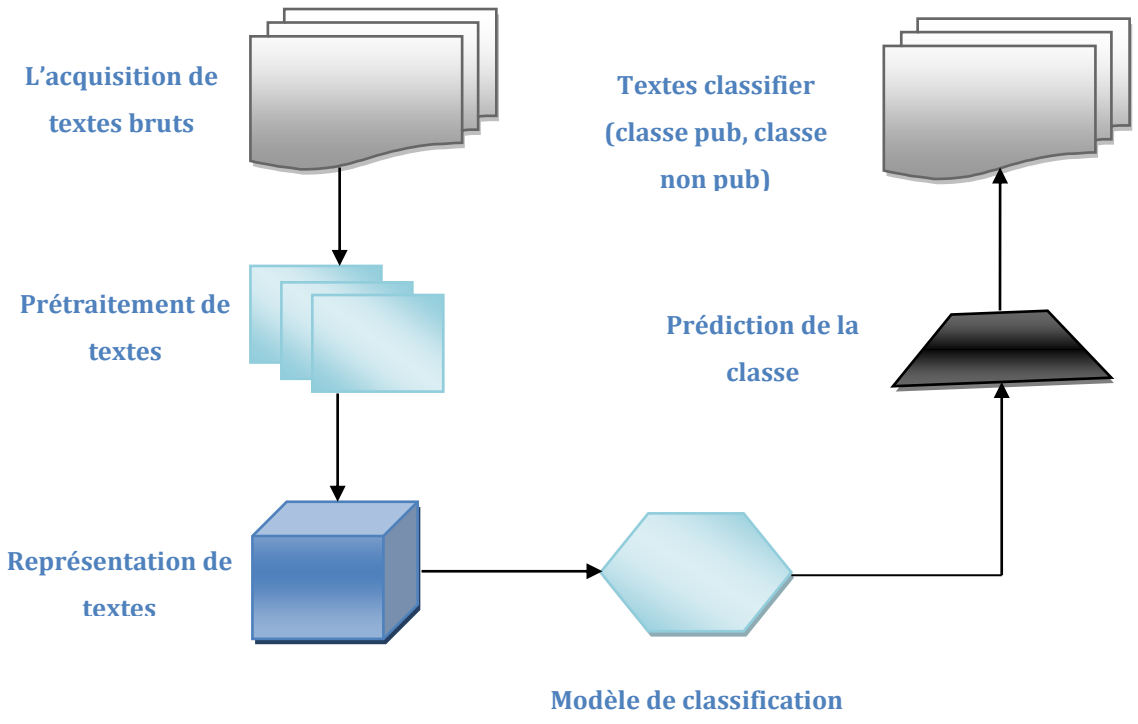


Figure 21 : Processus de catégorisation de textes

6.1 L'acquisition de textes bruts

Cette étape consiste à récupérer du texte brut (35) (ne contient aucune structure). Il existe plusieurs manières de récupérer du texte, soit depuis une base de données, ou bien comme dans notre projet, nous scrapons des pages web pour récupérer les liens et le contenu de leurs scripts.

6.2 Prétraitement de textes

Cette étape consiste à formaliser les textes afin qu'ils soient compréhensibles par la machine et utilisables par les algorithmes d'apprentissage. Elle a pour objectif de collecter les caractéristiques (descripteur) du texte (Une caractéristique est un mot, un groupe de mots ou encore un concept), et de les représenter sous forme de vecteur qui définissent le poids de chaque caractéristique.

Il existe plusieurs techniques d'extraction des caractéristiques à partir du texte brut. Suivant le mode de représentation des textes choisi, certaines techniques sont dispensables.

6.2.1 Nettoyage de mots

Cette technique permet de nettoyer le texte où nous éliminons les ponctuations (reconnaissance des fins de phrase ou de paragraphe), ensuite nous découpons les séquences de caractères en fonction de la présence ou l'absence de caractères de séparation (de type espace, tabulation ou retour chariot), puis supprimer les mots les plus fréquents (https/http, js, function, etc). Eventuellement, nous pouvons unifier les écritures en lettres majuscules ou en lettres minuscules avant ou après les opérations déjà indiquées.

6.2.2 Suppression des mots vides

Les mots vides sont des mots qui n'apportent pas de valeur informative pour la compréhension du sens d'un texte. Nous disons qu'ils ne sont pas significatifs. Par exemple, les déterminants tels que "le", "la", "du".

6.2.3 La tokenisation

Cette technique a pour objectif la segmentation d'un texte en unités atomiques. Nous appelons ces unités les "tokens". Généralement nous divisons le texte en mots ou phrases. Par exemple la tokenisation en mots d'un lien "https://www.google-analytics.com/analytics.js" après nettoyage, donne les tokens suivants : "https", "www",

"google", "analytics", "com","js". Elle permet aussi la transformation de l'ensemble des tokens en vecteur.

6.3 Représentation de textes

Après avoir vu les différentes techniques de prétraitements, nous allons maintenant étudier comment extraire l'information du texte pour le traitement ultérieur par des modèles d'apprentissage automatique. En d'autres termes, nous cherchons une représentation de langages pour un modèle qui vise à exploiter des données textuelles.

Cette étape consiste à présenter le texte comme un vecteur numérique interprétable par les méthodes de classification. Pour la représentation des documents textuels, plusieurs méthodes sont utilisées, nous présentons quelques-unes dans ce qui suit.

6.3.1 Représentation en sac de mots (*Bag of Words*)

C'est la représentation la plus couramment utilisée dans laquelle les textes sont transformés simplement en vecteurs dont chaque composante représente un mot.

Comme son nom l'indique, cette représentation vise à représenter le texte avec une liste de mots simples contenus dans ce dernier. Suivant les différentes techniques de prétraitement, certains mots comme les mots vides ne font pas partis du sac. A chaque mot est associée une mesure. Cette mesure peut être une valeur binaire (positive = 1 ; négative = 0), une fréquence d'apparition dans le document ou encore le nombre d'occurrences dans le texte (36).

Avec cette approche, les documents sont représentés par des vecteurs de dimension égale à la taille du vocabulaire[1], qui est en générale assez grand.

6.3.2 Représentation par prolongement de mots (*Word Embedding*)

La représentation par plongement de mots ou encore dense des mots sont les représentations les plus utilisées actuellement dans les dernières méthodes de traitement du langage.

Les *word embeddings* sont représentés par des vecteurs denses, un vecteur représentant la projection du mot dans un espace vectoriel continu. Ces derniers sont conçus pour capturer ce que nous appelons des similarités entre les mots du vocabulaire :

[1] Ceci est le nombre de mots significatifs ou fréquents pour chaque ensemble de documents

Les mots qui apparaissent dans des contextes similaires seront proches les uns des autres dans l'espace de projection (37). L'idée est de regrouper les mots qui partagent les mêmes propriétés sémantiques ("chien chat vache", "manger dévorer") ou syntaxiques ("jours chapeaux voitures", "vidés portés dansés"). En effet, le degré de similarité entre deux mots A et B, dépend du degré de correspondance entre leurs propriétés.

Il existe plusieurs méthodes d'apprentissage de l'intégration de mots à partir d'un texte, exemple : *Word2Vec*, *Glove*, *wordNet,* etc.

6.3.3 Représentation avec les n-grammes

De nombreux travaux ont montré l'efficacité des n-grammes comme méthode de représentation des textes pour leur classification : recherche d'une partition en groupes homogènes, ou pour leur catégorisation : attribution d'un texte à une, ou plusieurs, catégorie(s) parmi une liste prédéterminée. Un n-gramme est une séquence de n caractères consécutifs. Pour un document quelconque, l'ensemble des n-grammes (en général n prend les valeurs 2 ou 3) qu'on peut générer est le résultat qu'on obtient en déplaçant une fenêtre de n cases sur le corps de texte (38).

6.4 Choix de classifieur

Cette dernière étape consiste à faire le traitement de textes, que ce soit par l'algorithme d'apprentissage ou celui de classification. La catégorisation de textes comporte un choix de technique d'apprentissage (ou classifieur) disponibles. Il existe plusieurs méthodes d'apprentissage et parmi les plus souvent utilisées figurent les classifieurs naïve bayes, les machines à vecteurs de support et les classifieurs à base réseaux de neurones.

6.4.1 Les classifieurs naïf bayes

Les classifieurs naïf Bayésien (39) est une méthode appartenant à la catégorie des méthodes de classification Bayésienne probabiliste fondée sur le théorème de Bayes. L'objectif de cette méthode est le calcul des probabilités à posteriori qu'un document appartienne à chacune des catégories de l'analyse. La classe ayant la plus forte probabilité est sélectionnée comme label du document.

6.4.2 Les machines à vecteurs de support

Les machines à vecteurs de support est une méthode d'apprentissage supervisée qui est très utilisée pour la classification de textes. L'objectif du SVM (40) est la recherche d'un

hyperplan de marge maximale séparant les données d'apprentissage en deux classes d'analyse.

6.4.3 Les classifieurs à base de réseaux de neurones

L'apprentissage neuronal est appliqué avec succès dans les tâches de classification de textes. Le RN (41) est un réseau d'unités construit à partir des documents d'apprentissage, où les unités d'entrées représentent les pondérations des termes, les unités de sortie représentent les catégories, et les arcs reliant les unités représentent les relations d'indépendances. L'activation des unités d'entrées est propagée à travers le réseau et la valeur de l'unité de sortie détermine la classe du document.

6.5 Corpus de Textes

Afin de tester l'ensemble de ces algorithmes et techniques, il est nécessaire de disposer de matières premières : des textes labellisés. Obtenir un corpus de textes labellisés peut parfois être une étape compliquée car ces derniers sont coûteux à produire. En effet, chaque texte doit être labellisé manuellement pour permettre à la machine de retrouver le modèle. De plus, afin de comparer les résultats des différentes méthodes et techniques, il est nécessaire de disposer de corpus libres (42).

6.6 Difficultés particulière de la classification de textes

Beaucoup de difficultés peuvent s'opposer au processus de classification de textes. Des problèmes connus dans la discipline liés à l'apprentissage automatique comme la subjectivité de la décision prise par les experts, etc. mais aussi des problèmes particuliers liés à la nature des données traitées à savoir des données textuelles comme la polysémie, la redondance, Les variations morphologiques ou même L'homographie, etc. Nous citons les trois principales Dans ce qui suit.

6.6.1 Polysémie

Un mot possède, dans différents cas, plus d'un sens et plusieurs définitions lui sont associées. Par conséquent, à cause de la polysémie, les mots seuls sont parfois de mauvais descripteurs ; exemple le mot livre peut désigner une unité monétaire, ou un bouquin (43).

6.6.2 Les mots composés

La non prise en charge des mots composés comme : Arc-en-ciel, peut-être, sauve-qui-peut, etc. Dont le nombre est très important dans toutes les langues, et traiter le mot Arc-

en-ciel par exemple en étant 3 termes séparés réduit considérablement la performance d'un système de classification néanmoins l'utilisation de la technique des n-grammes pour le codage des textes atténue considérablement ce problème des mots composés (44).

6.6.3 Redondance

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 (44).

6.6.4 Subjectivité de la décision

Parmi les problèmes classiques usuels dans le domaine de l'apprentissage automatique c'est la subjectivité de la décision prise par les experts qui décident de la classe à laquelle le texte va être attribué (44).

7 Conclusion

Nous avons présenté à travers ce deuxième chapitre un ensemble de généralités et de concepts relatifs aux réseaux de neurones, ses fonctions d'activation, ainsi que ses différentes architectures et quelques algorithmes d'apprentissage. Ensuite, nous avons présenté un état de l'art sur la classification automatique de textes.

Ces algorithmes peuvent être utilisés pour la reconnaissance d'image et de parole. Mais dans ce mémoire, nous nous intéressons à la classification de textes, plus précisément de scripts publicitaires, à travers les réseaux de neurones.

C'est un axe de recherche important que nous introduisons dans le chapitre suivant, tout en proposant des modèles que nous entraînons par la suite sur un ensemble d'entrée (il s'agit des liens des scripts ainsi que leurs contenus qui est du code JavaScript).

Chapitre 3 : Modèles proposés

1 Introduction

La classification de textes (ou la catégorisation de textes) est une tâche originalement associée à la recherche d'information. Son objectif est de classer de manière automatisée des documents en se basant sur le contenu de texte et certaines métadonnées (lien, date de publication, nombre d'utilisation dans le site web, auteur du site, etc). Elle connaît un fort regain d'intérêt, dû essentiellement à la forte croissance des documents numériques disponibles et à la nécessité de les organiser de façon rapide. C'est une discipline qui a connu des progrès considérables avec l'apparition d'algorithmes d'apprentissage automatique.

Dans ce mémoire, nous abordons le problème de détection des scripts publicitaires comme un problème de catégorisation de textes à deux catégories : la catégorie des scripts publicitaires pour les publicités, et la catégorie des scripts non publicitaires pour d'autres scripts. Il est donc important de bien comprendre ce qu'est la classification de textes avant de traiter le problème de détection des scripts publicitaires.

Dans ce chapitre, nous exposons quelques techniques de prétraitements et de représentations pour la classification des scripts publicitaires. Ensuite, nous proposons trois modèles à base de réseaux de neurones. Enfin, nous terminons par une conclusion.

2 Motivations et objectifs

Notre objectif est de classifier les scripts Web en scripts publicitaires et non publicitaires. Il est donc important de bien comprendre ce qu'est un script dans le contexte du Web. Il s'agit en fait, de fichiers écrits en JavaScript.

JavaScript est un langage de script interprété basé sur la norme ECMAScript[1]. Il s'insère dans le code HTML/XML d'une page Web. Pour appeler du code JavaScript depuis le document HTML, un élément de type <*script*> est utilisé.

Les scripts publicitaires sont souvent considérés comme étant indésirables par certains utilisateurs. En effet, ils peuvent impacter négativement leur expérience Web. Ceci peut se manifester à travers :

[1] ECMAScript est un ensemble de normes de programmation recommandées pour les langages de script et standardisées par Ecma International dans le cadre de la spécification ECMA-262. Ces normes sont principalement appliquées par les interpréteurs JavaScript.

- des pop-ups (les petites fenêtres qui s'ouvrent de manière intempestive) exemple : demander l'autorisation des cookies ;
- des redirections : par exemple, lorsque nous sélectionnons un élément d'un menu déroulant nous serons redirigés vers d'autres sites Web que celui voulu ;
- des annonces vidéos ;
- ralentissement du chargement de la page et de la navigation.

Pour détecter et éliminer ces scripts non désirables, il faut pouvoir les classifier en tant que tels. Pour se faire, nous devons d'abord réaliser le prétraitement de textes bruts. En effet, les scripts sont écrits en langage JavaScript. Puis leurs donner une bonne représentation pour qu'ils deviennent exploitables par les modèles. Enfin, nous comptons à construire des modèles à base de réseaux de neurones pour faire la classification.

3 Prétraitement des scripts publicitaires

3.1 Caractéristiques

La classification des scripts peut faire usage de différentes caractéristiques. Certaines de ces caractéristiques peuvent être bénéfiques pour la classification. D'autres, peuvent être moins (ou pas) bénéfiques. Parmi les caractéristiques que nous pouvons envisager d'inclure pour la classification nous citons :

- Le lien : il s'agit de l'URL du script. Il est toujours précédé par le mot https/http. Chaque script a une extension js, car ils contiennent du code JavaScript.
- Le contenu : il s'agit du code JavaScript correspondant aux fonctionnalités réalisées par le script.
- Nombre d'utilisation dans le site web : il s'agit de nombre de fois que le script est utilisé dans le site.
- Auteur du site : il s'agit de nom de réalisateur de chaque script.
- Dernière mise à jour : la majorité des sites web mentionne la date de leur dernière mise à jour ainsi que celle effectuée sur leurs scripts.
- Date de réalisation de scripts : c'est la date quand les scripts ont été publiés.

Dans ce mémoire, nous avons choisi deux principales caractéristiques qui sont le lien du script et son contenu. Pour effectuer la tache de prétraitement, nous avons opté pour le lien et le contenu car ces derniers contiennent des filtres qui permettent de savoir s'il s'agit d'un script publicitaire ou non. Par exemple pour le lien suivant :
« https://pagead2.googlesyndication.com/pagead/js/adsbygoogle.js », nous avons le mot « pagead » et pour le contenu « googletag.impl.pubads.callbackProxy1 », nous avons le mot « pubads » » qui peuvent être ajoutés manuellement dans un filtre pour déterminer que ce lien et ce contenu contiennent de la publicité.

3.2 Le contenu

Le contenu de lien du script s'agit du code JavaScript. Nous proposons d'effectuer à ce dernier les taches suivantes.

3.2.1 Prétraitement

Nous proposons de nettoyer le texte de touts les mots inutiles. Par exemple nous proposons d'éliminer la ponctuation, de supprimer les mots répétés dans le contenu de chaque script, comme « *var, function, script*, etc ». Ensuite, nous appliquons une tokenisation sur chaque mot dans le contenu du script.

3.2.2 Représentation

Un document textuel sous sa forme brute est difficilement exploitable par un système de classification automatique. Pour pallier à ce problème, il est nécessaire d'ajouter une étape préalable d'indexation qui va lier le texte à une représentation exploitable normalisée appelée modèle vectoriel. Dans notre travail, nous proposons d'utiliser le *word embedding* qui est la projection d'un vecteur (souvent large), représentant un mot dans un nouvel espace de taille contrôlé qui modélise les relations entre les mots (37).

3.3 Le lien

Il est caractérisé par le protocole http, le nom du domaine et le numéro de port. Un lien publicitaire possède des filtres (sous forme d'expressions régulières) qui permettent de les détecter. Pour cela, nous proposons quelques techniques de prétraitement et de représentation pour le lien.

3.3.1 Prétraitement

Pour ce faire, nous proposons de nettoyer le lien de touts les mots inutiles, exemples : éliminer la ponctuation, la suppression des mots répétés dans le lien de script, par exemple « *https/http, js, json, www*, etc ». Ensuite, nous appliquons une tokenisation sur chaque lien de script.

3.3.2 Représentation

Nous proposons la même représentation que celle utilisée pour le contenu qui est le *word embedding*.

4 Modèles proposés

Dans cette section, nous proposons des modèles (architectures) à base de réseaux de neurones appliqués à la classification de textes (le lien et le contenu des scripts). Dans ce mémoire, nous proposons trois modèles d'un réseau de neurones qui sont nommés respectivement modèle 1, modèle 2, modèle 3. Nous avons choisi comme base pour le modèle 1, l'architecture CNN car cette dernière est simple à mettre en œuvre et aussi la plus répondue pour les problèmes de classifications de textes. A propos du modèle 2, nous avons choisi comme base l'architecture LSTM car elle possède une mémoire qui maintient les informations pour de longues périodes de temps. Pour le modèle 3, nous avons choisi comme base l'architecture GRU car cette dernière effectue moins de calculs et un gain en temps d'exécution.

4.1 Modèle 1

Nous proposons pour ce premier modèle une architecture CNN décrite dans la Figure 22. Elle est composée de trois couches de convolution, trois couches de maxpooling et deux couches entièrement connectées (FC). Chacune des couches précédentes a une fonction d'activation Relu ou SoftMax). Pour entraîner ce modèle, nous proposons un optimiseur Adam.

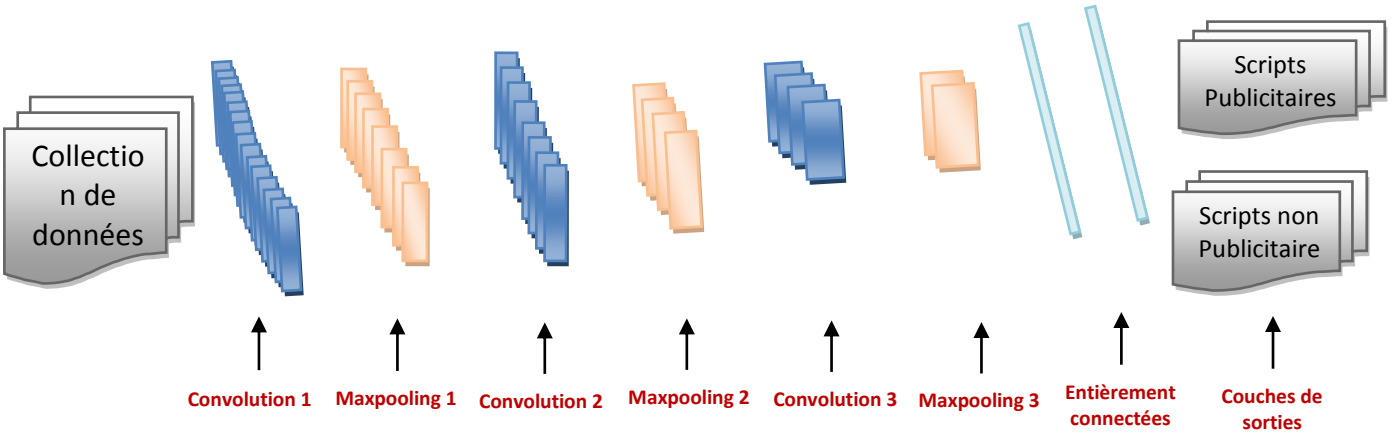


Figure 23 : Architecture de modèle 1

4.2 Modèle 2

Nous proposons pour ce deuxième modèle une architecture LSTM décrite dans la Figure 24. Elle est composée d'une couche LSTM, trois couches de time_distibuted et une couche entièrement connectée. Chacune des couches précédentes a une fonction d'activation (Relu ou SoftMax). Pour entraîner ce modèle, nous proposons un optimiseur Adam.

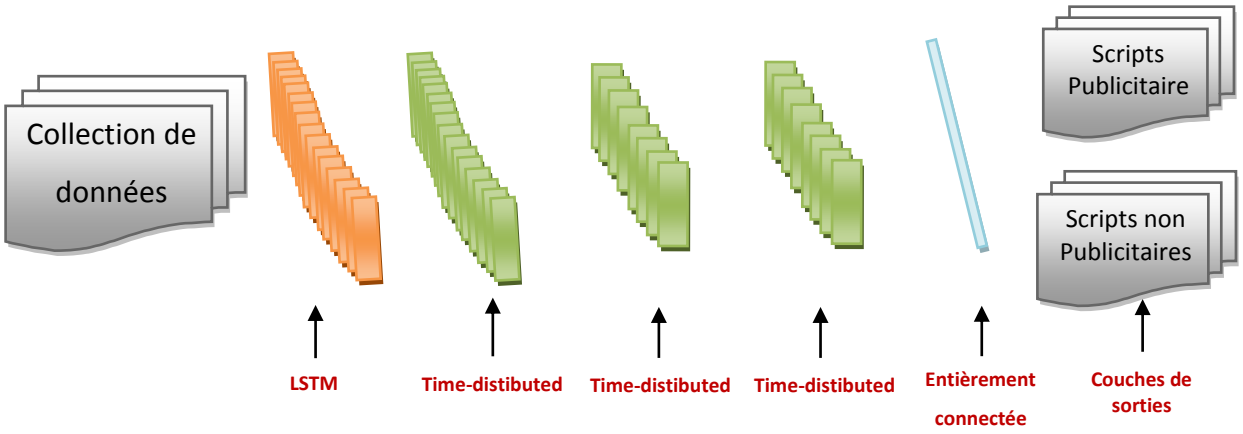


Figure 24 : Architecture de modèle 2

4.3 Modèle 3

Nous proposons pour ce troisième modèle une architecture GRU décrite dans la Figure 25. Elle est composée d'une couche GRU, trois couches de time_distibuted, une couche entièrement connectée. Chacune des couches précédentes a une fonction d'activation (Relu ou SoftMax). Par soucis de cohérence, nous proposons pour l'entraînement de ce modèle, le même nombre et type d'optimiseur que celui de LSTM.

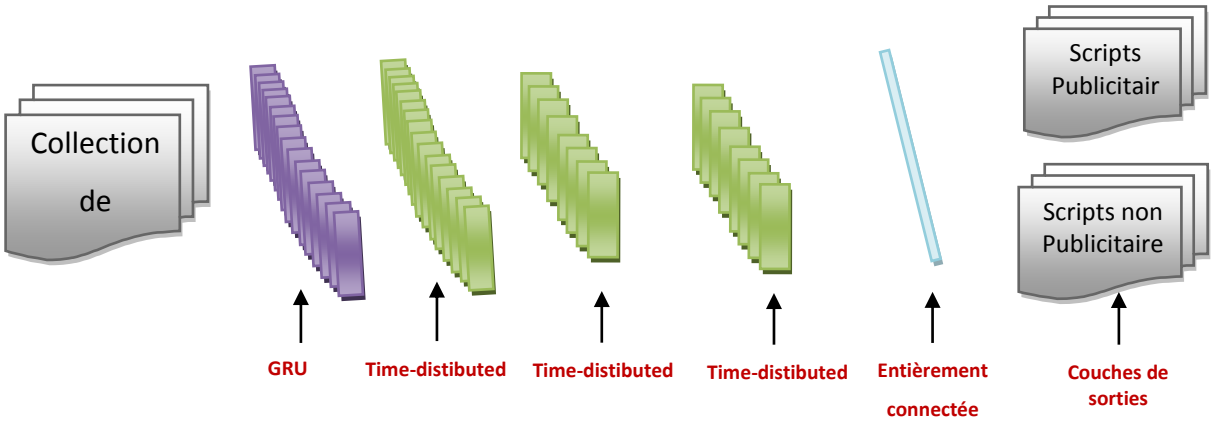


Figure 25 : Architecture de modèle 3

5 Conclusion

Nous avons présenté à travers ce troisième chapitre les techniques de prétraitements et de représentations que nous appliquons aux scripts publicitaires, Ainsi que les différents modèles proposés.

Pour arriver à une bonne classification des scripts, nous avons proposé plusieurs techniques de prétraitements tels que le nettoyage et la tokenisation de textes, et une technique de représentation telle que la représentation par le *word embedding*. Enfin, nous proposons trois architectures pour entraîner et tester l'ensemble de collection de données.

Le chapitre suivant est consacré à l'implémentation des trois modèles pour classifier les scripts en publicitaires ou non publicitaires.

Chapitre 4: Implémentation

1 Introduction

Nous présentons dans ce chapitre la partie implémentation de notre travail, dont l'aboutissement est un code réalisant l'entraînement et l'évaluation des trois modèles de réseaux de neurones que nous avons proposés dans le chapitre précédent.

L'implémentation est réalisée en deux parties. Dans la première partie, nous construisons nos collections d'entraînement, de validation et de test. Nous avons construit trois ensembles de collections de données : le premier contenant uniquement les liens des scripts, le deuxième contenant le contenu des scripts, et le dernier contenant aussi bien le contenu des scripts que leur lien. Nous effectuons ensuite, un prétraitement sur ces collections et nous donnons aux données une représentation pouvant être utilisée dans nos modèles. Enfin, dans la seconde partie, nous entraînons puis évaluons, sur la base des collections précédentes, nos trois modèles que nous avons décris dans le chapitre précédent. A cet effet, nous avons obtenu de très bons résultats avec les trois modèles, en particulier, avec le modèle 3.

Dans ce chapitre, nous définissons les différents outils et bibliothèques nécessaires pour notre implémentation ainsi que la configuration utilisée. Ensuite, nous présentons les Jeux de données et la construction des collections d'entraînement, de validation et de test, ainsi que leurs prétraitements et leurs représentations. Enfin, et avant de conclure, nous discutons les résultats obtenus par les trois modèles après l'entraînement.

2 Outils et environnement de développement

Dans ce mémoire, nous avons opté pour le langage de programmation python et l'IDE Pycharm. Pour ce qui concerne le traitement de données, nous avons utilisé les bibliothèques Numpy, Matplotlib, qui sont des bibliothèques de calculs matriciels et de visualisation de données. Pour l'apprentissage automatique nous avons opté pour l'environnement de travail Keras. Nous définissons ces différents bibliothèques et outils dans ce que suit.

2.1 Logiciels

2.1.1 Pycharm

Pycharm (45) est une boîte à outils IDE essentiellement conçue pour développer des programmes et construire des logiciels en python. Développé par la société Tchèque JetBrains, il fournit une boîte à outils d'assistance et des fonctionnalités intégrées telles que la plupart des IDE comme eclipse.

Pycharm est multiplateforme et fonctionne sous Windows, Mac OS et Linux. Il est disponible en deux éditions : professionnelle publiée sous une licence propriétaire, communautaire publiée sous une licence apache.

2.2 Bibliothèques logicielles

Nous avons utilisé les bibliothèques logicielles suivantes.

2.2.1 UrlLib

Cette bibliothèque fournit une interface de haut niveau pour la récupération de données sur le Web à partir d'une URL (*Universal Resource Locator*). Elle permet d'effectuer facilement des opérations simples (ouvrir une URL comme un fichier, faire des requêtes HTTP). Les principaux protocoles utilisés par cette bibliothèque sont : HTTP, HTTPS, FTP, Gopher, File (système de fichiers).

Urllib (46) dispose de plusieurs fonctions. en particulier, urlopen() est une fonction similaire à la fonction open(), mais accepte les URL au lieu des noms de fichiers. Nous avons utilisé cette dernière Pour récupérer le contenu de pages Web (les liens des scripts) à partir de leur URL.

2.2.2 Beautiful Soup 4

Beautiful Soup (47) est une bibliothèque qui facilite la collecte d'informations dans les pages Web. Elle utilise un analyseur XML ou HTML enfichable pour analyser un document dans une représentation arborescente. Cette bibliothèque fournit des méthodes et des idiomes Pythonique qui facilitent la navigation, la recherche et la modification de l'arbre d'analyse. Nous avons utilisées cette bibliothèque pour extraire les liens des scripts d'une page web.

2.2.3 La bibliothèque RE

Elle fournit des opérations de correspondance d'expression régulière. Une expression régulière ou rationnelle (48) (*regular expression* ou RE) C'est un puissant outil qui permet de vérifier si le contenu d'une variable a la forme de ce que l'on attend. Elle permette aussi de modifier ou de supprimer tous les éléments indésirables dans une variable. Les fonctions de cette bibliothèque permettent de vérifier si une chaîne particulière correspond à une expression rationnelle donnée. Nous avons utilisé cette bibliothèque pour récupérer les liens des scripts (commençant par HTTP ou HTTPS).

2.2.4 Csv-Python

Le format CSV (49) (*Comma Separated Values*) est le format d'importation et d'exportation utilisé par les feuilles de calcul et les bases de données.

Csv est une bibliothèque qui implémente des classes pour lire et écrire des données tabulaires au format CSV. Elle permet d'écrire les données dans le format Excel ou lire les données de fichier généré par Excel.

2.2.5 Adblockparser

C'est une bibliothèque qui permet de travailler avec les règles de filtrage *easylist*[1] de Adblock Plus (50). Nous l'avons utilisé pour construire notre collection de données. D'abord, nous analysons les filtres de *easylist* avec *AdblockRules*. Ensuite, nous avons fait la correspondance entre les liens des scripts avec la liste des filtres grâce à la fonction *shouldblock*, afin de classer les liens des scripts en publicitaires et non-publicitaires.

2.2.6 Requests

C'est une bibliothèque python permettant d'utiliser le protocole http de façon simple. Elle permet, par exemple, de récupérer des données d'une page web à travers un proxy, de spécifier les propres mécanismes d'authentification pour les utilisateurs et enfin de vérifier les certificats SSL sur les requêtes HTTPS, etc (51). Nous avons utilisé cette bibliothèque pour récupérer le contenu de chaque script (publicitaires et non-publicitaires).

2.2.7 Numpy

C'est une bibliothèque de bas niveau écrite en C et Fortran consacrée entièrement au calcul numérique en Python. Elle permet essentiellement de manipuler et de faire des

[1] Easyliste : Est une liste qui contient des expressions régulières pour détecter les scripts publicitaires

opérations sur des tableaux à plusieurs dimensions. Les calculs avec numpy sont particulièrement optimisés car les tableaux sont homogènes (ils ne contiennent que des valeurs d'un même type) et de taille fixée à la création (52). Nous avons utilisé cette bibliothèque pour la représentation finale des scripts sous forme d'indices de mots dans un tableau.

2.2.8 Os

C'est une bibliothèque python qui fournit des fonctions permettant d'interagir avec le système d'exploitation, en particulier, avec le système de fichiers. Elle fait partie des modules utilitaires standard de Python. Elle fournit un moyen portable d'utiliser les fonctionnalités dépendantes du système d'exploitation (53). Nous avons utilisé cette bibliothèque pour parcourir les répertoires contenant nos trois collections une fois celles-ci construites et afin de les utiliser pour l'entraînement et l'évaluation.

2.2.9 Sys

Cette bibliothèque fournit un certain nombre de fonctions et de variables pouvant être utilisées pour manipuler différentes parties de l'environnement d'exécution Python (54). Nous avons utilisé cette bibliothèque pour manipuler les variables d'environnement (arg).

2.2.10 Keras

C'est une bibliothèque de haut niveau permettant de créer et de manipuler des réseaux de neurones. Elle est écrite en Python et est capable d'utiliser TensorFlow ou Theano (des bibliothèques de bas niveau permettant de créer et de manipuler des réseaux de neurones). Son objectif est de permettre des expérimentations rapides (55). Nous avons utilisé cette bibliothèque pour la construction et l'entraînement de nos trois modèles.

2.2.11 Matplotlib

C'est une bibliothèque de traçage Python à deux dimensions qui produit des images de qualité. Elle peut être utilisée dans les scripts Python, les serveurs d'applications Web, etc.

Matplotlib permet de générer des histogrammes, des graphiques à barres, des diagrammes d'erreur, des diagrammes de dispersion, etc, avec seulement quelques lignes de code (56). Nous avons utilisé cette bibliothèque pour tracer des graphes afin de présenter le résultat de notre approche.

2.3 Configuration utilisée

La configuration matérielle utilisée pour l'entraînement et l'évaluation de nos modèles est la suivante :

- Processeur intel Core i5-6200U CPU contenant quatre cœurs à 2.30 GHZ chacun.
- Carte graphique intel HD Graphics 520 (skylake GT2)/AMD Hainan.
- RAM d'une taille de 4 GO.
- Disque dur (HDD) d'une taille de 500 GO.
- Système d'exploitation Linux Ubuntu 64bits version 18.10 avec l'interface GNOME 3.

3 Jeux de données et construction des collections d'entraînement, de validation et de test

Nous avons préparé notre collection (jeux de données) en suivant plusieurs étapes :

- La première étape est l'extraction des liens des scripts avec l'élimination des doublons : Nous avons scrapé[2] des pages web pour récupérer et enregistrer des scripts internes et externes de ces dernières dans un fichier sous format CSV à l'aide d'un *crawler*[3] que nous avons programmé. Pour ce faire, nous avons utilisées les bibliothèques *UrlLib*, *BeautifulSoup* 4 et RE afin de récupérer les liens des scripts pour chaque page web parcourue.
- La deuxième étape consiste à la classification des scripts en publicitaires et non publicitaires : Après avoir récupéré les liens des scripts, nous avons classé la liste des liens en deux classes (publicitaire, non publicitaire). Pour ce faire, nous avons utilisé une liste de filtre (*easyliste*) proposée par adblock à l'aide de la bibliothèque adblockparser.
- La troisième étape est la récupération du contenu des scripts : Pour chaque script, nous avons récupéré son contenu à l'aide de la bibliothèque requests, puis nous l'enregistrons dans un fichier texte.

[2] Extraire des données d'une page Web à l'aide d'un programme automatisé.

[3] Il s'agit d'un programme qui a pour objectif de parcourir les différentes pages Web, afin d'analyser leurs contenus.

Ainsi, nous avons obtenu trois collections comme le montre la figure ci-après : La collection 1 contient uniquement le lien, la collection 2 contient uniquement le contenu, et la collection 3 contient le lien et son contenu. Chaque collection est composée de 1000 scripts (500 publicitaires et 500 non publicitaires), que nous avons séparés en trois parties : 80% pour l'entraînement, 10% pour la validation, 10% pour le test.

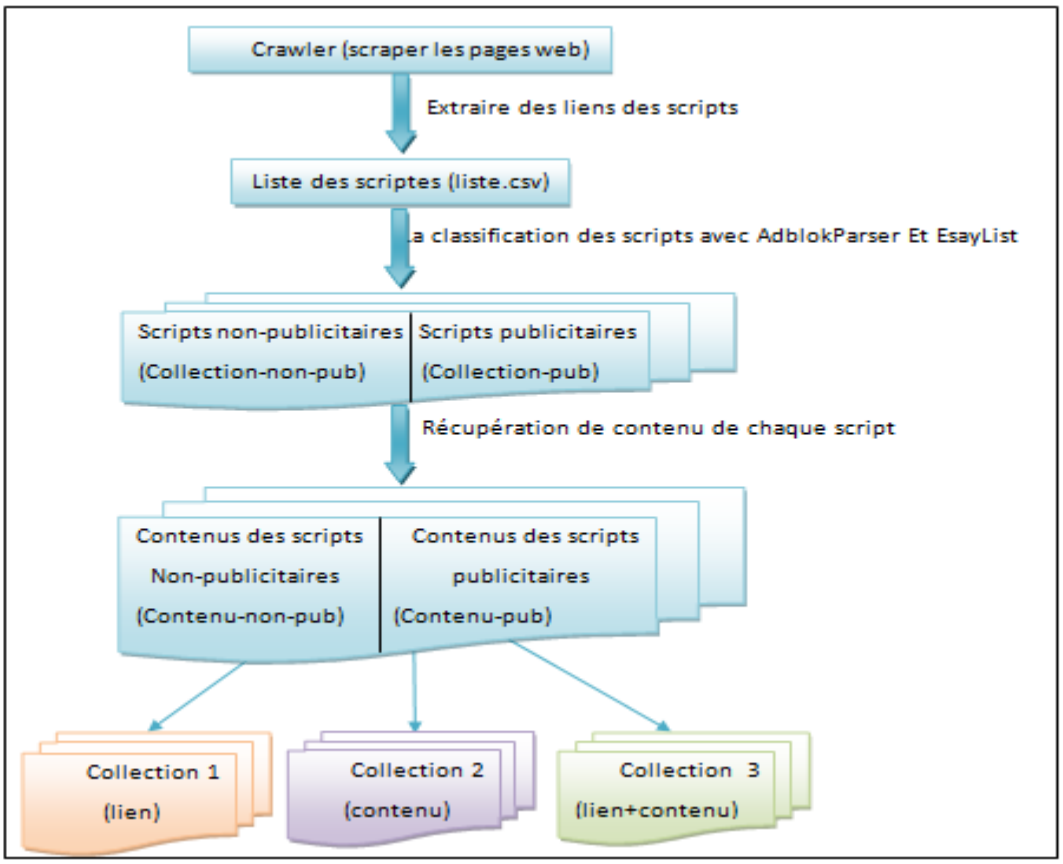


Figure 26 : Processus de construction de la collection d'entraînement, de validation et de test

4 Prétraitement et représentation

Dans cette section, nous avons présenté les différentes techniques que nous avons utilisées pour le prétraitement et la représentation des données (scripts). Pour le prétraitement, nous avons nettoyé les scripts ainsi leurs contenu de tous les mots et caractères inutiles. Ensuite, nous avons appliqué une tokenisation à ces derniers.

Après avoir terminé la tâche de prétraitement, nous avons obtenu des documents textuels (la collection des scripts) étiquetés. Nous avons utilisé Glove pour la représentation des mots qui est une technique de *word embedding*. Glove signifie "vecteurs globaux pour la

représentation de mots". C'est un dictionnaire qui permet d'associer à chaque mot un vecteur. Pour cela, il recueille les caractéristiques de cooccurrence de mots sous forme de matrice pour construire des représentations compactées des scripts. La matrice de cooccurrences M est de dimension V × V avec V qui est le nombre de mots dans notre vocabulaire (la collection des scripts) :

- Collection 1 : contient 1000 liens avec en moyenne 14 mots par lien.
- Collection 2 : contient le contenu de 1000 lien avec en moyenne 300 mots par contenu.
- Collection 3 : contient 1000 liens et leurs contenus avec le nombre de mots est 14 mots pour le lien et 300 mots pour le contenu.

La valeur $M_{i,j}$ représente le nombre de fois où le mot i et le mot j apparaissent dans un même contexte. Ensuite, une analyse factorielle est appliquée pour réduire les dimensions de M et en extraire l'information utile. Les représentations extraites avec Glove modélisent linéairement des relations sémantiques entre les mots. Le schéma ci-dessous montre le résultat de chacune des étapes de prétraitement et de représentation appliquées sur la collection 3 (collection-lien-contenu).

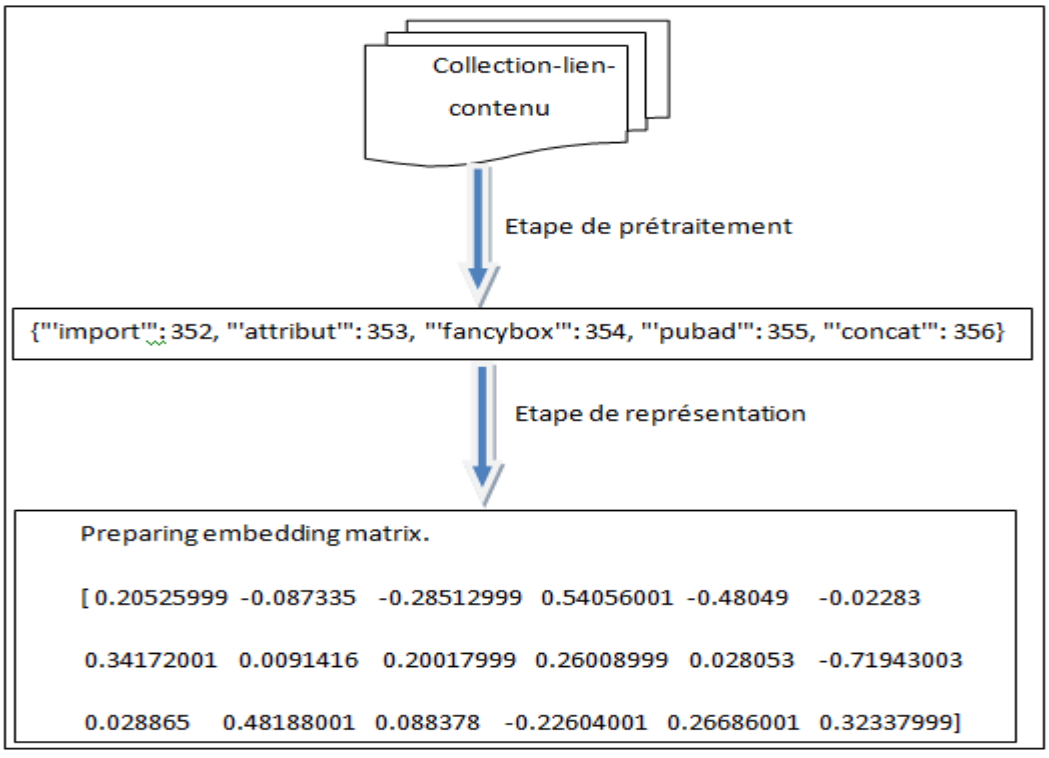


Figure 27 : Exemple de résultats de prétraitement et de représentation de textes

5 Entraînement et résultats

Nous avons entraîné les modèles jusqu'à obtention d'une précision (*accuracy*) suffisante avec l'ensemble d'entraînement. Pour cela, un nombre suffisant d'itération (epoch) à été réalisé sur l'ensemble de données des trois collections (présentées au préalable). Dans chaque modèle, nous avons essayé d'ajuster le nombre de paramètres de façon à ce que le résultat obtenu soit meilleur avec le moins de paramètres.

Au cours de nos expérimentations, nous avons créé trois modèles (modèle 1, modèle 2 et modèle 3) avec différentes architectures. Nous avons choisi l'architecture CNN pour le modèle 1, l'architecture LSTM pour le modèle 2 et enfin l'architecture GRU pour le modèle 3.

Dans ce qui suit, nous présentons les différents résultats obtenus pour chaque architecture des trois modèles.

5.1 Modèle 1

Le premier modèle est construit avec la configuration représentée dans la Figure 28.

- Le texte en entrée est de taille 300.
- La première couche de convolution est composée de 300 filtres de taille 5 et une fonction d'activation Relu.
- La première couche de Maxpooling permet de réduire la taille de textes (obtenus après la première convolution) ainsi le nombre de paramètres (division sur 2).
- La deuxième couche de convolution est composée de 150 filtres et une fonction d'activation Relu.
- La deuxième couche de Maxpooling permet de réduire la taille de textes (obtenus après la deuxième convolution) ainsi que le nombre de paramètres (division sur 2).
- La troisième couche de convolution est composée de 75 filtres, et une fonction d'activation Relu.
- La troisième couche de Maxpooling permet de réduire la taille de textes (obtenus après la troisième convolution) ainsi que le nombre de paramètres (division sur 2).

- La première couche entièrement connectée est composée de 300 neurones où la fonction d'activation utilisée est Relu.
- La deuxième couche entièrement connectée utilise la fonction SoftMax et est composée de deux neurones qui permettent de calculer la distribution de probabilité des 2 classes (classe pub, classe non-pub).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 300, 300)          90300
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 300, 300)          450300
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 150, 300)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 150, 150)          225150
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 75, 150)           0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 75, 75)            56325
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 37, 75)            0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2775)              0
_________________________________________________________________
dense_1 (Dense)              (None, 300)               832800
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 602
=================================================================
Total params: 1,655,477
Trainable params: 1,655,477
Non-trainable params: 0
```

Figure 28 : Configuration du modèle 1

5.1.1 Collection 1

Les résultats obtenus avec la collection 1 sont présentés dans le tableau ci-dessous.

Précision avec l'ensemble d'entraînement	Précision avec l'ensemble de test	Le temps écoulé pour l'entraînement
98 %	88 %	8 min 21 s 16 ms

Tableau 1 : Les résultats du modèle 1 avec la collection 1

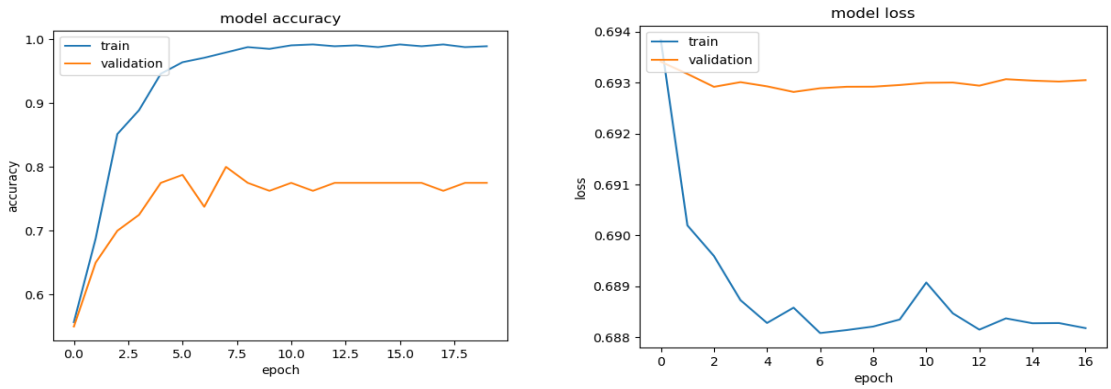


Figure 29 : Précision (accuracy) et erreur (loss) du modèle 1 avec la collection 1

D'après la Figure 29 La précision de l'apprentissage et de la validation augmente avec le nombre d'époques. Ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

5.1.2 Collection 2

Les résultats obtenus avec la collection 2 sont présentés dans le tableau ci-dessous.

Précision avec l'ensemble d'entraînement	Précision avec l'ensemble de test	Le temps écoulé pour l'entraînement
98,6 %	89 %	9 min 14 s 43 ms

Tableau 2 : Les résultats du modèle 1 avec la collection 2

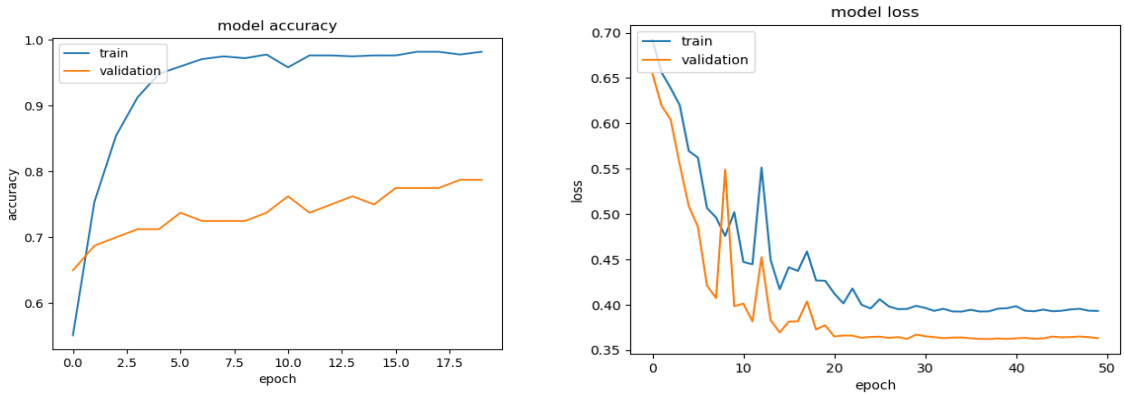


Figure 30 : Précision (accuracy) et erreur (loss) du modèle 1 avec la collection 2

D'après la Figure 30 La précision de l'apprentissage et de la validation augmente avec le nombre d'époques. Ceci reflète qu'à chaque époque le modèle apprenne plus

d'informations. De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

5.1.3 Collection 3

Les résultats obtenus avec la collection 3 sont présentés dans le tableau ci-dessous.

Précision avec l'ensemble d'entraînement	Précision avec l'ensemble de test	Le temps écoulé pour l'entraînement
99,41 %	90 %	10 min 35 s 59 ms

Tableau 3 : Les résultats du modèle 1 avec la collection 3

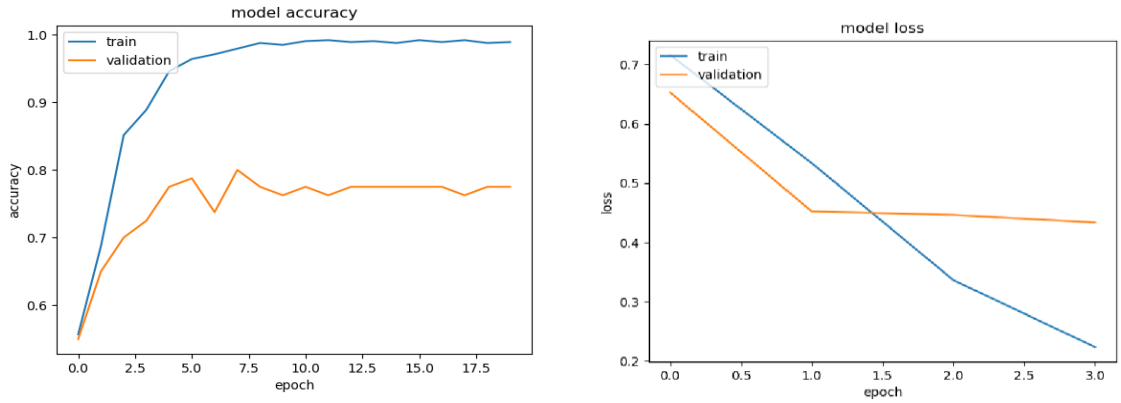


Figure 31 : Précision (accuracy) et erreur (loss) du modèle 1 avec la collection 3

D'après la Figure 31 La précision de l'apprentissage et de la validation augmente avec le nombre d'époques. Ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

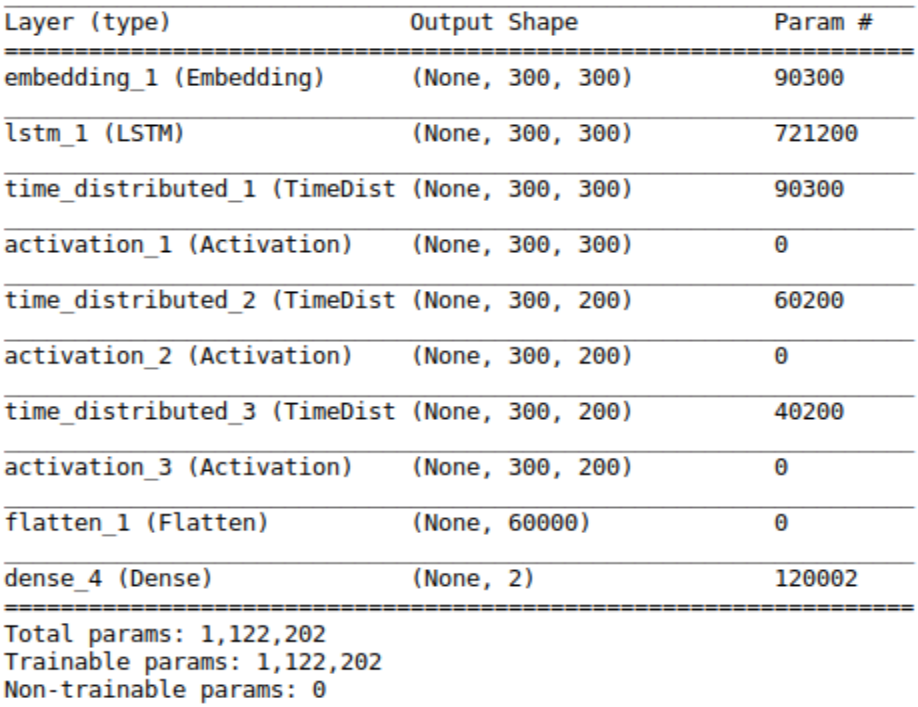
5.2 Modèle 2

Le deuxième modèle est construit avec la configuration représentée dans la Figure 32.

- Le texte en entrée est de taille 300.
- La couche LSTM lit les données d'entrée et génère 300 neurones.
- La première couche de time_distributed 4qui est composée de 300 neurones en entrée avec une tranche de temps (timesteps) égale à 300 et une fonction d'activation Relu.

[4] Il applique une même densité à chaque pas de temps pendant le déroulement de la cellule GRU/LSTM. Il attribut pour chaque couche de réseau les mêmes biais et poids.

- La deuxième et la troisième couche de time_distributed est composée de 200 neurones en entrée avec une tranche de temps (*timesteps*) égale à 300 et une fonction d'activation Relu.
- La couche entièrement connectée utilise la fonction SoftMax et est composée de deux neurones qui permettent de calculer la distribution de probabilité des 2 classes (classe pub, classe non-pub).



Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 300, 300)	90300
lstm_1 (LSTM)	(None, 300, 300)	721200
time_distributed_1 (TimeDist	(None, 300, 300)	90300
activation_1 (Activation)	(None, 300, 300)	0
time_distributed_2 (TimeDist	(None, 300, 200)	60200
activation_2 (Activation)	(None, 300, 200)	0
time_distributed_3 (TimeDist	(None, 300, 200)	40200
activation_3 (Activation)	(None, 300, 200)	0
flatten_1 (Flatten)	(None, 60000)	0
dense_4 (Dense)	(None, 2)	120002

Total params: 1,122,202
Trainable params: 1,122,202
Non-trainable params: 0

Figure 32 : Configuration du modèle 2

5.2.1 Collection 1

Les résultats obtenus avec la collection 1 sont présentés dans le tableau ci-dessous.

Précision avec l'ensemble d'entraînement	Précision avec l'ensemble de test	Le temps écoulé pour l'entraînement
98 %	86 %	9 min 14 s 29 ms

Tableau 4 : Les résultats du modèle 2 avec la collection 1

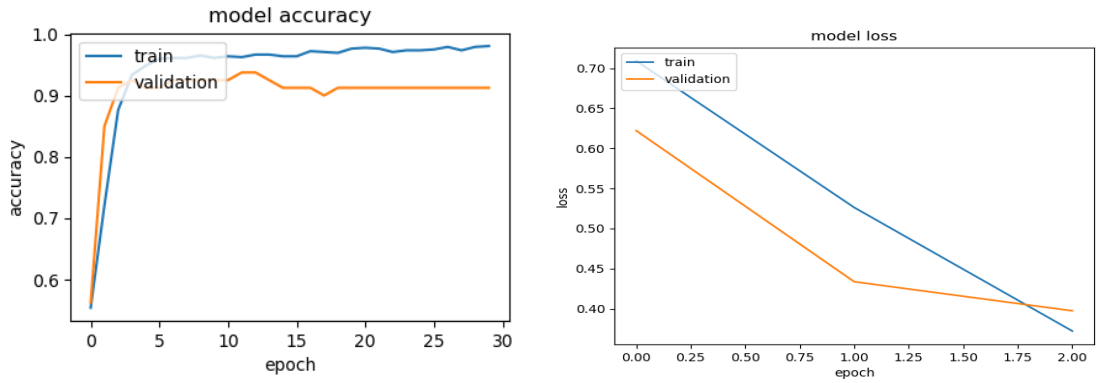


Figure 33 : Précision (accuracy) et erreur (loss) du modèle 2 avec la collection 1

D'après la Figure 33 La précision de l'apprentissage et de la validation augmente avec le nombre d'époques. Ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

5.2.2 Collection 2

Les résultats obtenus avec la collection 2 sont présentés dans le tableau ci-dessous.

Précision avec l'ensemble d'entraînement	Précision avec l'ensemble de test	Le temps écoulé pour l'entraînement
98,9 %	87 %	9 min 20 s 68 ms

Tableau 5 : Les résultats du modèle 2 avec la collection 2

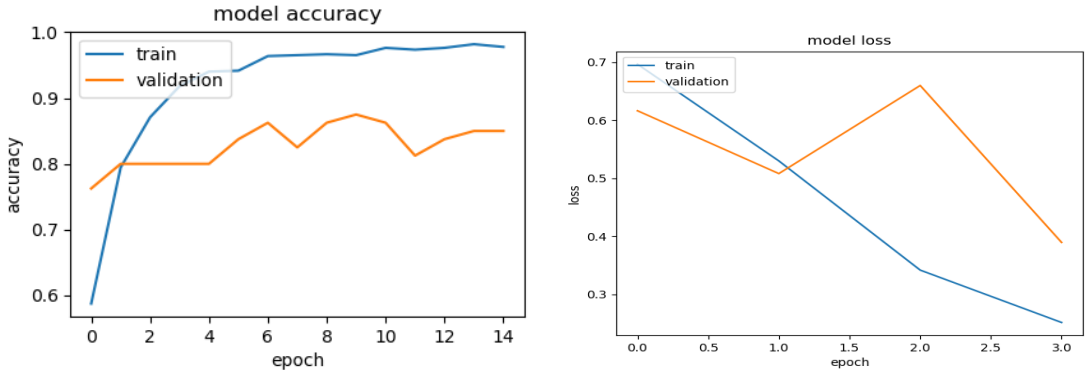


Figure 34 : Précision (accuracy) et erreur (loss) du modèle 2 avec la collection 2

D'après la Figure 34 La précision de l'apprentissage et de la validation augmente avec le nombre d'époques. Ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

5.2.3 Collection 3

Les résultats obtenus avec la collection 3 sont présentés dans le tableau ci-dessous.

Précision avec l'ensemble d'entraînement	Précision avec l'ensemble de test	Le temps écoulé pour l'entraînement
99,44 %	89 %	10 min 00 s 66 ms

Tableau 6 : Les résultats du modèle 2 avec la collection 3

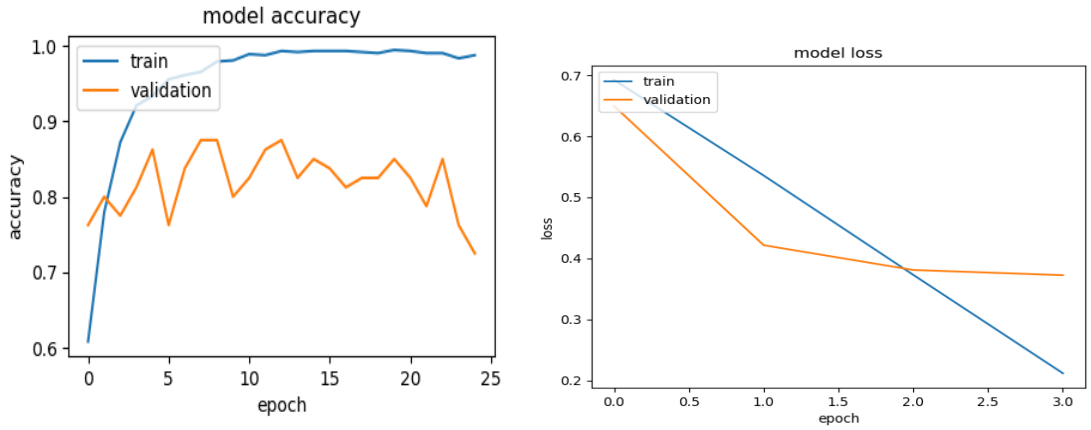


Figure 35 : Précision (accuracy) et erreur (loss) du modèle 2 avec la collection 3

D'après la Figure 35 La précision de l'apprentissage et de la validation augmente avec le nombre d'époques. Ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

5.3 Modèle 3

Le troisième modèle est construit avec la configuration représentée dans la Figure 36.

- Le texte en entrée est de taille 300
- La couche GRU lit les données d'entrée et génère 300 neurones.
- La première couche de *time_distributed* qui est composée de 300 neurones en entrée avec une tranche de temps (timesteps) égale à 300 et une fonction d'activation Relu.
- La deuxième et la troisième couche de time_distributed est composée de 200 neurones en entrée avec une tranche de temps (timesteps) égale à 300 et une fonction d'activation Relu.

- La couche entièrement connectée utilise la fonction SoftMax et est composée de deux neurones qui permettent de calculer la distribution de probabilité des 2 classes (classe pub, classe non-pub).

```
model fitting - Monodirectional GRU
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 300, 300)          90300
_________________________________________________________________
gru_1 (GRU)                  (None, 300, 300)          540900
_________________________________________________________________
time_distributed_1 (TimeDist (None, 300, 300)          90300
_________________________________________________________________
activation_1 (Activation)    (None, 300, 300)          0
_________________________________________________________________
time_distributed_2 (TimeDist (None, 300, 200)          60200
_________________________________________________________________
activation_2 (Activation)    (None, 300, 200)          0
_________________________________________________________________
time_distributed_3 (TimeDist (None, 300, 200)          40200
_________________________________________________________________
activation_3 (Activation)    (None, 300, 200)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 60000)             0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 120002
=================================================================
Total params: 941,902
Trainable params: 941,902
Non-trainable params: 0
```

Figure 36 : Configuration du modèle 3

5.3.1 Collection 1

Les résultats obtenus avec la collection 1 sont présentés dans le tableau ci-dessous.

Précision avec l'ensemble d'entraînement	Précision avec l'ensemble de test	Le temps écoulé pour l'entraînement
98,47 %	87 %	7 min 45 s 28 ms

Tableau 7 : Les résultats du modèle 3 avec la collection 1

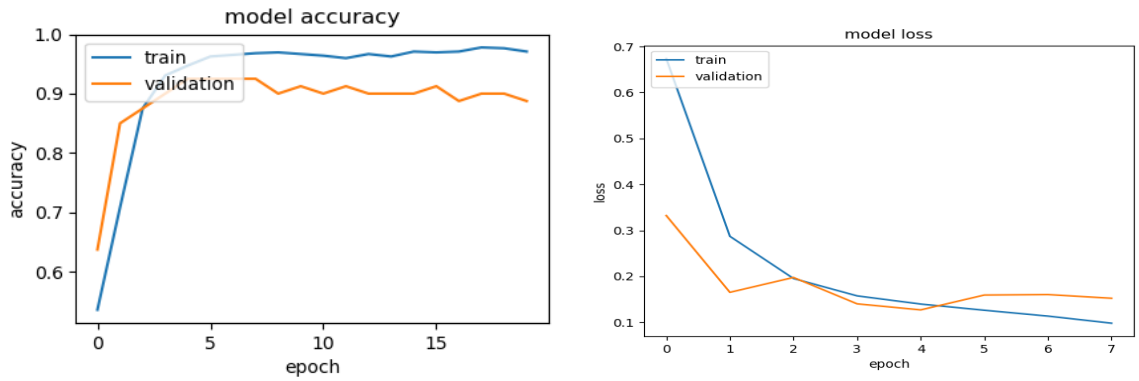


Figure 37 : Précision (accuracy) et erreur (loss) du modèle 3 avec la collection 1

D'après la Figure 37 La précision de l'apprentissage et de la validation augmente avec le nombre d'époques. Ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

5.3.2 Collection 2

Les résultats obtenus avec la collection 2 sont présentés dans le tableau ci-dessous.

Précision avec l'ensemble d'entraînement	Précision avec l'ensemble de test	Le temps écoulé pour l'entraînement
98,7 %	88 %	7 min 35 s 56 ms

Tableau 8 : Les résultats du modèle 3 avec la collection 2

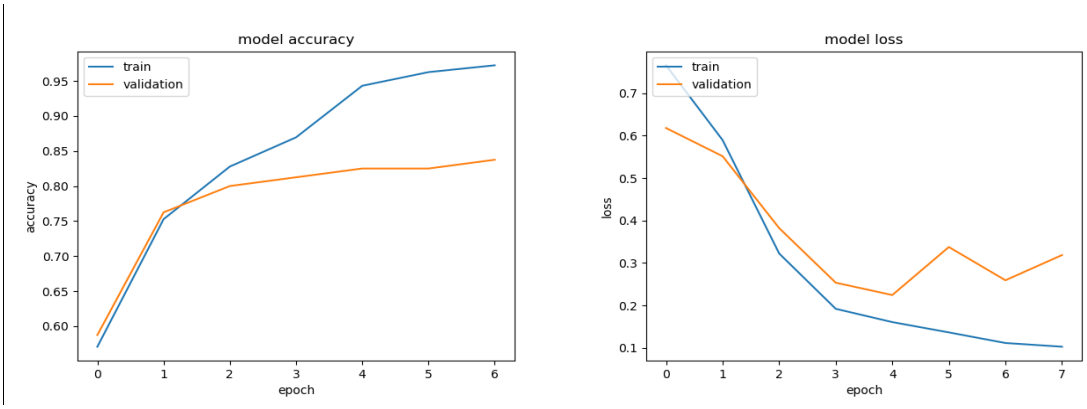


Figure 38 : Précision (accuracy) et erreur (loss) du modèle 3 avec la collection 2

D'après la Figure 38 La précision de l'apprentissage et de la validation augmente avec le nombre d'époques. Ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

5.3.3 Collection 3

Les résultats obtenus avec la collection 3 sont présentés dans le tableau ci-dessous.

Précision avec l'ensemble d'entraînement	Précision avec l'ensemble de test	Le temps écoulé pour l'entraînement
99,6 %	92%	7 min 9s 59 ms

Tableau 9 : Les résultats du modèle 3 avec la collection 3

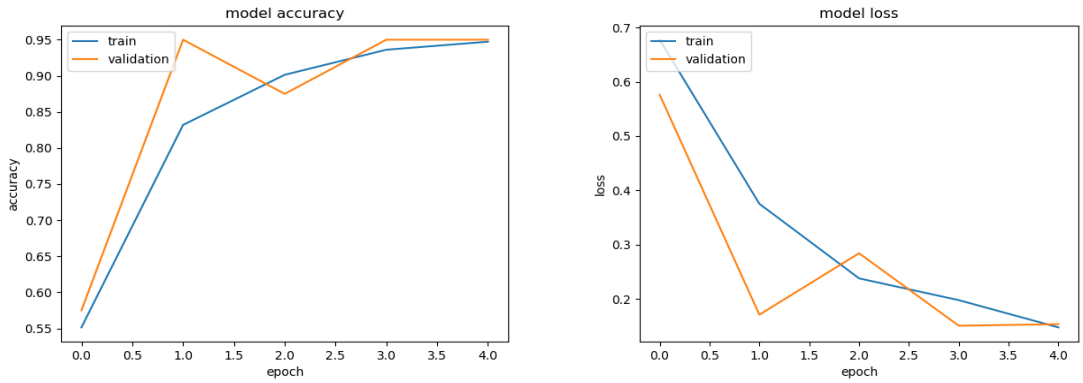


Figure 39 : Précision (accuracy) et erreur (loss) du modèle 3 avec la collection 3

D'après la Figure 39 La précision de l'apprentissage et de la validation augmente avec le nombre d'époques. Ceci reflète qu'à chaque époque le modèle apprenne plus d'informations. De même, l'erreur d'apprentissage et de la validation diminue avec le nombre d'époque.

5.4 Discussion et comparaison des résultats

Le Tableau 10 montre les résultats finaux de l'expérience regroupant les trois modèles (1, 2 et 3). Par soucis de cohérence, les résultats sont présentés de la même manière (même epoch, jeux de données) pour les trois modèles.

Tout d'abord, nous observons que les résultats obtenus sont assez similaires pour les trois modèles (1, 2 et 3). L'ajout de métadonnées (lien de script) influencent peu la classification. Il est important d'ajuster les paramètres afin d'améliorer les résultats de test et de l'entraînement, en particulier, lorsque nous utilisons les réseaux de neurones récurrents. Concernant le temps d'exécution, nous avons constaté que le modèle 3 prend moins de temps pour l'entraînement avec les trois collections.

En conclusion, en fonction du modèle et de la collection, les trois modèles 1, 2 et 3 ont des comportements différents. Le modèle 3 a présenté les meilleurs résultats avec un taux de 99.6 % pour la précision sur l'ensemble d'entraînement et 92 % de Précision sur l'ensemble de test avec la collection 3.

Le modèle	Collection	Nombre d'epochs	Précision finale de l'entraînement	Précision finale de test	Temps écoulé pour l'entraînement
Modèle 1	Collection 1	200	98 %	88 %	8 min 21 s 16 ms
	Collection 2	200	98,6 %	89 %	9 min 14 s 43 ms
	Collection 3	200	99,41 %	90 %	10 min 35 s 59 ms
Modèle 2	Collection 1	200	98 %	86 %	9 min 14 s 29 ms
	Collection 2	200	98,9 %	87 %	9 min 20 s 68 ms
	Collection 3	200	99,44 %	89 %	10 min 00 s 66 ms
Modèle 3	Collection 1	200	98,47 %	87 %	7 min 45 s 28 ms
	Collection 2	200	98,7 %	88 %	7 min 35 s 56 ms
	Collection 3	200	99,6 %	92 %	7 min 9 s 59 ms

Tableau 10 : Comparaison des résultats finaux des trois modèles

6 Conclusion

Dans ce chapitre, nous avons présenté en premier lieu, les différents outils et environnement de développement utilisés pour l'implémentation de nos modèles. Ensuite, nous avons décrit les collections d'entraînement, de validation et de test utilisées. Puis, nous avons présenté la partie implémentation de notre approche, d'où nous avons expliqué en détail les étapes de prétraitement et de représentation. Enfin, nous avons discuté les résultats obtenus après l'entraînement de chaque modèle (CNN, LSTM, GRU) sur nos collections.

L'évaluation de notre approche, réalisée pour les trois modèles proposés ci-dessus sur les trois collections (collection1, collection2, collection3) ont mené aux constatations qui suivent. Tout d'abord, le choix du modèle et de la collection influence sur les résultats de l'entraînement et de test. En effet, les expérimentations ont montré que le modèle 3 (GRU) et la collection 3 sont idéales pour avoir un meilleur résultat d'entraînement et de test (respectivement 99,87%, 92%).

Conclusion générale

1 Synthèse

L'apprentissage automatique par réseaux de neurones est devenu un outil puissant qui a démontré son efficacité à plusieurs reprises. Ce dernier permet d'améliorer les résultats obtenus par d'autres techniques considérées comme étant l'état de l'art dans le domaine de la classification de textes. Mais les techniques de classification des scripts publicitaires à base de l'apprentissage automatique restent loin d'être performantes à 100%. Donc il est très important de faire des progrès dans le domaine de traitement linguistique de textes afin d'arriver à une meilleure représentation textuelle manipulable par les algorithmes de classification, aussi de construire et de modéliser des classifieurs performants.

Les contributions présentées dans ce mémoire sont un bon exemple de ces avancées pour les tâches comme la représentation des données textuelle codées en JS.

A travers ce mémoire, nous avons présenté les différentes notions nécessaires pour la compréhension de notre sujet. Ensuite, nous avons proposé trois modèles pour implémenter notre approche qui consiste à classifier les scripts. Ces derniers reposent sur l'utilisation d'un réseau de neurones convolutif pour le modèle 1, LSTM pour le modèle 2 et GRU pour le modèle 3.

Enfin, nous avons montré l'intérêt de la prise en compte des liens des scripts et leurs contenus. Ainsi, nous avons effectué un bon prétraitement pour ces derniers lors de la classification. De plus, nous avons évalué nos modèles en utilisant des collections de données que nous avons-nous-mêmes construites. Les différentes expérimentations réalisées ont montré l'intérêt des modèles adoptés, avec un résultat de test très élevé dans le modèle 3.

2 Perspectives

Plusieurs travaux futurs peuvent être envisagés à l'issue de ce mémoire. Nous en décrivons quelques-uns dans cette section :

- la première perspective consiste à développer une extension pour les navigateurs à l'image d'AdBlockPlus, permettant de bloquer les publicités non pas à base de filtres, mais à base de l'un des modèles de réseaux de neurones que nous avons construits dans ce mémoire. Une telle extension pourrait

bloquer de nouveaux scripts publicitaires sans édition des filtres et serait donc beaucoup plus performante que les extensions existantes (telles que AdBlockPlus) qui nécessitent des mises à jour manuelles coûteuses.

- La deuxième perspective consiste à construire une collection contenant plus de scripts pour mieux entraîner nos modèles.

Bibliographie

1. **Ezratty, Ollivier.** *les usages de l'intelligence artificeille.* paris : edition 2018, 2018.

2. **Blanchot, Valentin.** Intelligence artificielle. [En ligne] [Citation : 30 04 2019.] http://villemin.gerard.free.fr/Wwwgvmm/Logique/IAHisto.htm.

3. **M.Y.Aammar.** *Mise en ouevre de réseaux de neuronnes pour la modélisation de cinétique reactionnelles en vue de la transposition batch/continu.* Tunis : Université de Tunis, 2007. Thèse de Doctorat.

4. **D.Hammoud.** *Apprentissage Automatique dans un Agent.* constantine : université Mentouri Constantine, 2014. Thèse de Doctorat.

5. **I.Tellie.** *192369936-L-apprentissage-automatique-Preface.* Orléan : Université d'Orléan, 2010.

6. **A.Amraoui.** *Améloiration de la fiabilité du lien sans fil pour un terminnal radio cognitive mobile.* Tlemcen : Université de Tlemcen, 2011. Thèse de Doctorat.

7. **Krief, B. Benmammar & F.** *Resource Management for End-to-End QoS in a Mobile Environment.* Montréal : International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2006), 2006. Comptes rendus de conférence.

8. **L.Saidi.** classification et apprentissage automatique. [En ligne] [Citation : 24 04 2019.] https://www.univ-saida.dz/butec/doc_num.php?explnum_id=155..

9. **M.Laur.** *Anticipation des changements des notes des obligations des portefeuille d'un assureur par méthode de machine learning.* paris : Universtés Dauphine, 2017. Mémoire de fin d'etude.

10. **MA.Allal.** *Utilisation du Deep Learning dans la radio cognitive.* Tlemcen : Université Abou Bakr Belkaid, 2018. rapport de reherche.

11. **H.Larochelle.** *Etude de techniques d'apprentissage non-supervisé pour l'amélioration de l'entraînemment supervisé de modèles connexionnistes.* Montréal : Université de Montréal, 2008. Thèse Doctorat.

12. **L.Cantat, G.Hebrail &.** blog et systemx. [En ligne] 24 04 2019. https://blog.irt-systemx.fr/le-machine-learning-un-domaine-de-l'intelligence-artificielle-en-forte-evolution/..

13. **DY.Moualek.** *Deep Learning pour la classification des images.* Tlemcen : Université Abou Bakr Belkaid– Tlemcen, 2016/2017. mémoire de fin d'etude .

14. **D.kadous.** *Utilisations des réseaux de neuronnes comme outils de datamining.* Tlemcen : Université Abou-Bakr Belkaid de Tlemcen, 2012. thèse doctorat.

15. **F.Beloufa.** *conception d'un classifieur à base des régles floues.* Tlemcen : Université Abou-Bakr Belkaid de Tlemcen, 2016. Thèses Doctorat.

16. **H.Belekasmi.** *reconaissence automatique des expressions faciales par support vector machine.* 2017. mémoire de fin d'étude.

17. **S.Benhammada.** *Etude comparative de methodes de seléction de caractéristiques en apprentissage automatique.Proposition d'une variante.* Mila : universitaire de Mila, 2006/2007. Mémoire de fin d'étude.

18. **Matharel, L.De.** JDN. [En ligne] https://www.journaldunet.com/solutions/reseau-social-d-entreprise/1191979-machine-learning-12-secteurs/.

19. **Mahammed, F.Ait.** *approches d'apprentissage automatique pour la détection d'un spam web.* Montréal : Université Québec , 2018. Mémoire de fin d'etude .

20. **A.Saidi, S.Lakhdari &.** *Etude des techeniques d'apprentissage semi-supervisé par regroupement.* Telemcen : Université Abou Bekr Belkaid, 2017. Mémoire de fin d'étude.

21. **D.Combe.** *détection de communautés dans les réseaux d'information utilisant liens et attributs.* paris : Université Jean Monnet - Saint-Etienne, 2013. thèse doctorat.

22. **Y.Bendaoud.** *Prédiction des résistances mécaniques des bétonsà base des ciments composés en utilisant les réseaux neurones artificiel.* Constantine : Université Constantine1, 2014. Thèse de Magister.

23. **Rosenfeld, J. Anderson.** *Neurocomputing : Foundations of research, MIT Press.* Cambridge : Second printing, 1988. Comptes rendus de conférence.

24. **H.Chaoui.** *conception et comparaison de lois de commande adaptative à base des réseaux de neurones pour une articulaton fléxible avec non-linéarité dure.* québec : l'universités du québec à trois-rivières, 2002. Mémoire de fin d'étude.

25. **SUPINFO.International.University .** Supinfo. [En ligne] 03 avril 2019. https://www.supinfo.com/articles/single/7923-deep-learning-fonctions-activation?fbclid=IwAR02jV_4TqBPhF8GRNmMkF9DBBtzQGRGYOnUHiQ-0E5URHl10NkRm_gjg4Q.

26. **Z.Asradj.** *identification des systèmes non-linéaires par les réseaux de neurones.* bédjaia : Université Abd Erahmane Mira, 2015. Mémoire de fin d'etude .

27. **H.Dendani.** *Modélisation et détection des modes de défaillance dans un système complexe par les méthodes graphiques:Application au réseau électrique Sonelgaz.* Annaba : Université Badji Mokhtar, 2010. Mémoire de fin d'etude .

28. **M.Nouressadat.** *Etude des performances des réseaux de neurones dynamiques à représenter des systèmes réels : une approche dans l'espace d'état.* Setif : Université de setif1, 2009. Mémoire de fin d'etude .

29. **J.Schmidhuber, S.Hochreiter &.** *Neural Computation.* Allemagne : Université technique de Munich, 1997.

30. **Y.Bengio, J.Chung & C.Gulcehre & K.Hyun Cho.** *Calcul neuronal et évolutif.* suisse : Présenté dans l'atelier d'apprentissage approfondi et de représentation de NIPS 2014, 2014. Thèse de Doctorat.

31. **MZ.Mokri.** *classification des images avec les réseaux de neurones convolutionnels.* Tlemcen : Université Abou Bakr Belkaid, 2017. Mémoire de fin d'étude .

32. **Singer., J. Duchi & E. Hazan & Y.Singer.** *Adaptive subgradient methods for online learning and stochastic optimization.* New York : Tong Zhang, 2011. Journal.

33. **Ba, Diederik P. Kingma.** *3rd International Conference for Learning Representations.* San Diego : edition San Diego, 2015. Comptes rendus de conférence .

34. **W.Zhou, A.M Rossetto &.** *Improving Classification with CNNs using Wavelet Pooling with Nesterov-Accelerated Adam, P84-93.* Lowell : Université Massachusetts Lowell, 2019. Comptes rendus de conférence .

35. **M.Sasaki, H.Shinnou &.** *Spam detection using text clustering.* washington : Association for Computing Machinery, 2005. Comptes rendus de conférence .

36. **A.Fusiello, R.Toldo & U.Castellani &.** *A bag of words approach for 3d obeject categorization.* france : Springer, 2009. Comptes rendus de conférence .

37. **R.Vedantam, S.Kottur &.** *Learning visually grounded word embeddings using abstract.* la chine : Dans les actes de CVPR, 2016. Comptes rendus de conférence .

38. **Trenkle, William B Cavnar & John M.** *N-gram-based text categorization.* Michigan : Ann Arbor MI, 1994. Comptes rendus de conférence .

39. **D.Lewis.** *Naive (bayes) at forty: The independence assumption in information retrieval. In European conference on machine learning.* Berlin : Springer, 1998. Comptes rendus de conférence .

40. **A.Gammerman, V.Vovk, and V.Vapnik.** *Learning by transduction In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence.* Californie : Morgan Kaufmann Publishers Inc, 1998. Comptes rendus de conférence .

41. **F.Barigou.** *Contibution à la Catégorisation de Textes et à l'Extraction d'Information.* Oran : Université d'Oran, 2012/2013. Thèse doctorat.

42. **JC.RISCH.** *Enrichissement des Modèles de Classification de Textes Représentés par des Concepts.* Reims : Université De Reims Champagne-Ardenne, 2017. Thèse Doctorat.

43. **H.Hadjira, R.N.El Houda &.** *Amélioration du produit scalaire via les mesures de similarités sémantiques dans le cadre de la catégorisation des textes.* Tlemcen : Université Abou Bakr Belkaid Tlemcen, 2015. Mémoire de fin d'études.

44. **A.Abdelali.** *Classification automatique de textes arabes supervisee par l'ontologie lexicale wordnet.* Msila : Université Mohamed Boudiaf Msila, 2017 /2018. Mémoire de fin d'etude.

45. **P.Gupta.** Quora. [En ligne] [Citation : 11 06 2019.] https://www.quora.com/What-is-PyCharm-used-for.

46. **Python.Software.Foundation.** Python. [En ligne] [Citation : 11 06 2019.] https://docs.python.org/fr/2/library/urllib.html.

47. **Pytho.Software.Foundation.** Beautifulsoup. [En ligne] [Citation : 11 06 2019.] https://pypi.org/project/beautifulsoup4/.

48. **Python.Software.Foundation.** Python. [En ligne] [Citation : 11 06 2019.] https://docs.python.org/fr/2/library/re.html.

49. **Pytho.Software.Foundation.** Python. [En ligne] [Citation : 11 06 2019.] https://docs.python.org/fr/2/library/csv.html.

50. **GitHub.Inc.** GitHub. [En ligne] [Citation : 11 06 2019.] https://github.com/scrapinghub/adblockparser?fbclid=IwAR27XG9483gXvVL-h1xmWdaE4TmIoV9QJ7u1kTR3yxkOgaQLkSkAn8rWqIc.

51. **AK.Reitz.** Requests: HTTP for Humans™. [En ligne] [Citation : 11 06 2019.] https://2.python-requests.org/en/master/.

52. **Y.Copin.** Analyse scientifique avec Python. [En ligne] [Citation : 11 06 2019.] https://informatique-python.readthedocs.io/fr/m2/Cours/science.html?fbclid=IwAR3-OWFUVn7n4G89j80DibL-wL_COYODFTAj7jdDYd7LmIZbd7wkVM_OUsQ.

53. **Python.Software.Foundation.** Python. [En ligne] [Citation : 11 06 2019.] https://docs.python.org/fr/3/library/os.html?fbclid=IwAR3TqI5YXkuLjJtLq0hl2DHPPAdbnpyWSn-6fa7Tz1Rt2ig_BiLOEfQ5jBA.

54. **Tutorials.Teacher.** TutorialsTeacher. [En ligne] [Citation : 11 06 2019.] https://www.tutorialsteacher.com/python/sys-module?fbclid=IwAR2eGW3HGuSUjFhIoADN6QDerQo--aa2EO5crav-8gCnmUO5PlUvnqjvEEs.

55. **Active.Concept.Net.** Actu IA. [En ligne] [Citation : 11 06 2019.] https://www.actuia.com/keras/?fbclid=IwAR0h7BIMZrIB7wuF9G06bMfWCIzuNLVX8eNrPWDWdDxVH2OO-o-USvmZOx0.

56. **J.Hunter.** Matplotlib. [En ligne] [Citation : 11 06 2019.] https://matplotlib.org/.