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'obtentiondudiplôme de

Master en Informatique

Spécialité : Systèmes Informatiques

Réalisé par :

AIBOUD Lila et **LASKRI Samia**

Appréciation de la qualité des leads dans le marketing numérique à l'aide de l'apprentissage profond

Proposé et dirigé par : SADI Samy

Soutenu le 11 Novembre 2020 devant lejury composé de:

Dr. REMDANI Mohamed	Président du Jury
Dr. FELALI Idir	Membre du Jury
Dr. SADI Samy	Directeur de mémoire

Dédicaces

Je dédie ce modeste travail :

À ma mère,

Celle qui m'a tout donné amour,sacrifices,

Affection, soutien, je ne saurai tout citer,ni trouver les

Mots pour la remercier

À mon cher frère et sœur, mon père

Pour leur soutien infinietleurs encouragements incessantset leurs précieux conseils tout au longde mes études.

À mes amis (es)

Pour leur présence a tout moment et supports dans les moments difficiles.

Samia

Dédicaces

Je dédie ce modeste travail :

À mes très chers parents,

Qui m'ont inculqué le sens de la responsabilité, du travail etde l'optimisme.

Je ne saurais exprimer la gratitude, le respect et la reconnaissance que j'ai pour eux.

À mes chers frères et sœurs

Pour leur soutien moral et leur précieux conseils tout au longde mes études.

À mes amis (es)

Pour leurs aides et supports dans les moments difficiles.

Leila

Remerciements

Nous remercions *Allah* de nous avoir donné le courage et la force afin d'accomplir ce modeste projet.

Nous remercions notre encadreur, monsieur *Sadi*, pour tout ce qu'il nous a appris dans le monde de l'intelligence artificielle ainsi que toutes ses remarques pertinentes.

Nous remercions l'ensemble des membres du jury, qui nous ont fait l'honneur de bien vouloir étudier avec attention notre travail.

Résumé

Ce mémoire s'inscrit dans le cadre de l'application de l'apprentissage automatique dans le domaine Marketing plus précisément qualifications des leads. Nous avons implémenté les algorithmes et les heuristiques implémentés actuellement. Aussi, nous avons mis en pratique Le traitement des données du langage naturel en utilisant l'apprentissage automatique. Ainsi, nous avons conçu un modèle basé sur les réseaux de neurones artificiels sélectionné pour sa performance. Enfin, nous avons effectué une évaluation de ce modèle lors de laquelle nous avons eu de bon résultats.

Mots clés :intelligence artificielle, apprentissage automatique, apprentissage profond, réseaux de neurones, gestion des leads, marketing

Abstract

The work within this Thesis applies the concepts of Machine Learning in the field of Marketing, specifically, on Lead Qualification, by applying the most heuristic algorithms available. In doing so, we've implemented natural language data processing using ML. We built a model using Artificial Neural Networks that produced favorable results. Finally, we carried out an evaluation of the model and the results it produced which satisfied out criteria.

Key-words: Artificial Intelligence, Machine Learning, Deep Learning, Neural Networks, Lead Management, Marketing

Table des matières

Table des figures

Table des figures

Introduction générale

1 Contexte du travail

Notre travail s'inscrit dans le cadre de l'automatisation d'un service marketing cas de la gestion des relations clients, plus précisément la qualification des leads.

Le Marketing avant l'avenu de l'ère digitale, était un domaine très différent. Dans le 20eme siècle, les entreprises se focalisent principalement sur la diffusion par télévision et radio, bien que beaucoup plus efficace que les alternatives de l'époque (tel que les panneaux d'affichages ou bien les emballages des produits), l'approche était peu optimiser, l'identification des publics visés n'était rien d'autre qu'une approximation, et bien que les budgets de marketing étaient significatives, la recherche consistait à tâtonner jusqu'à trouver une stratégie satisfaisante *(trial and error en anglais*) , ce qui était coûteux et peu efficace.

L'introduction de l'informatique a complètement changé le paysage du Marketing, en donnant accès à de nouveaux outils qui permettent de non seulement identifier le public visé (ceci inclut les centres d'intérêts et les modèles de comportement) mais aussi de l'atteindre instantanément.

Parmi les aspects les plus importants du marketing moderne on retrouve la gestion des relation clients qui consiste en l'acquisition des personnes potentiellement intéressé par les services ou produits fournis par l'entreprise, pour formuler les stratégies Marketing basées sur les données récoltées sur ceci. Contrairement aux anciennes méthodes qui consistait à contacter les clients personnellement (et de façon aléatoire) afin de déterminer qui était un client potentiel, la méthode moderne accomplit le même objectifs en traçant le comportement et les interactions des utilisateurs internet, et d'estimer de façon plus efficace et à plus grande échelle l'intérêt de ces derniers à interagir avec l'entreprise.

Dans le but d'optimiser ce processus essentiel, plusieurs approches ont été développées, afin d'accroître la précision, améliorer l'efficacité et réduire les coûts associés. La meilleure approche actuellement consiste à utiliser les réseaux de neurones.

2 Problématique

La mise en œuvre d'une stratégie marketing efficace est essentiellement basée sur la bonne connaissance du public visé.

Dans ce but les entreprises cherchent toujours à mieux identifier des contacts qui seront intéressé par leurs services ou produit (lead qualifié).

Ce mémoire s'intéresse à l'évaluation de la qualité des leads en utilisant l'apprentissage automatique plus précisément les réseaux de neurones.

En effet nous avons pris l'exemple d'une campagne marketing par mail lancée par une entreprise vers plusieurs personnes dont l'identité est gardée anonyme pour des raisons de confidentialité. À la fin nous allons identifier quel type de personne est plus intéressé en utilisant des modèles de réseaux de neurones et ainsi permettre à l'entreprise d'optimiser sa stratégie marketing.

3 Contribution

Notre projet consiste à réaliser une solution permettant d'automatiser des campagnes digitales personnalisées selon les données de navigation, démographiques et les réactions de chaque prospect, mais aussi des systèmes capables de mesurer l'attractivité du contact en fonction de son comportement. Une étape cruciale qui va permettre ensuite d'attribuer automatiquement des scores à chaque lead selon ses particularités, ses attitudes et son niveau d'intérêt.

Pour ce fait, nousutilisons une collection d'informations récoltée de plusieurs campagnes marketing réalisées par une entreprise réelle. Par la suite, nous analysons et formalisés ces données afin de construire des modèles d'apprentissage automatique (RNA) qui vont estimer la qualité de leads vis-à-vis de différentes campagnes de marketing par email. Ensuite nous entrainons et évaluons ces modèles et au final présentons les résultats obtenus.

4 Organisation du mémoire

En plus de l'introduction générale, ce mémoire est organisé en quatre chapitres et une conclusion générale.

Dans le premier chapitre, nous présentons des généralités sur l'intelligence artificiel et l'apprentissage automatique en se basant sur les types d'apprentissages existants et leurs algorithmes.

Le deuxième chapitre est divisé en 2 parties : dans la première partie nous nous étalons sur le fonctionnement des réseaux de neurones profonds, d'autre part dans la seconde, nous présentons différentes méthodes du traitement automatique du langage naturel.

Dans le troisième chapitre, après avoir fait un état de l'art sur l'automatisation de la qualification des leads, nous présentons nos contributions, en décrivant l'architecture du modèle proposé et ses caractéristiques.

Dans le quatrième chapitre, nous décrivons l'environnement de développement ainsi que les différents outils utilisés, et à la fin nous présentons les résultats obtenus.

Chapitre 1 : Généralités sur l’apprentissage automatique.

1 Introduction

Depuis des années, les marqueteurs mettent l'accent sur une priorité essentielle : nouer, préserver et développer les relations avec leurs clients. Or, dans la réalité, la création d'une expérience client d'exception est précisément ce qui dope les ventes.

La mise en œuvre d'un système d'intelligence artificielle dans les compagnes marketing devient alors plus pertinente pour avoir une conversation engageante avec les consommateurs.

Dans ce chapitre nous nous intéressons au domaine de l'intelligence artificielle en expliquant certains des principaux concepts clés qui le composent, en particulier l'apprentissage automatique nécessaire pour la compréhension de ce mémoire.

Nous commençons par définir l'intelligence artificielle et l'apprentissage automatique. Nous présentons ensuite les étapes de l'apprentissage automatique et ses différents types, les modèles d'apprentissage automatique. Enfin, nous terminons par uneconclusion.

2 Définition de l'intelligence artificielle

L'intelligence artificielle (IA) (en anglais*artificiel intelligence (AI))* consiste à rendre un être artificiel (une machine) intelligent à travers différentes théories et techniques, visant à permettre aux machines d'être capable de penser, raisonner et même d'apprendre comme un humain.

3 Le domaine des Intelligence artificielle

Afin de mieux assimiler les informations qui suivent, il faut savoir que l'intelligence artificielle comprend l'apprentissage automatique (*machine Learning* en anglais) qui lui-même comprend l'apprentissage profond *(Deep Learning* en anglais*)*, ces trois concepts sont étroitement liés. (Voir la figure 1).

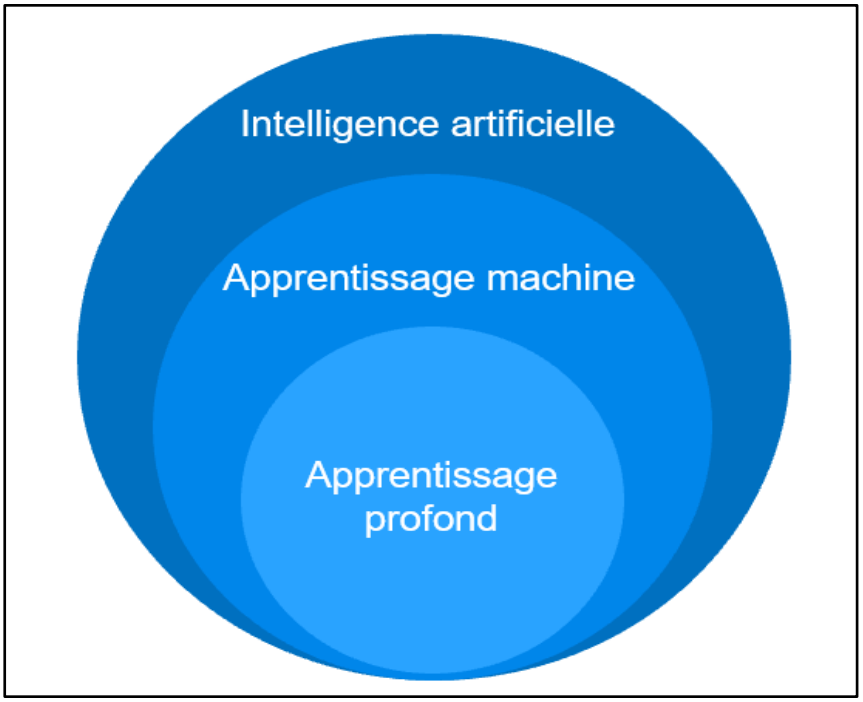

Figure 1 : Les domaines de l'intelligence artificielle

4 Définition de l'apprentissage automatique (AA)

La première définition de L'apprentissage automatique a été introduite par l'informaticien chercheur Arthur Samual en 1958, Il l'a défini comme étant *le domaine d'études qui donne aux ordinateurs la possibilité d'apprendre sans être explicitement programmé.* [1]

Puis elle a été enrichie par le professeur Tom Mitchell en 1998: *« Un programme informatique apprendrait de l'expérience E en ce qui concerne une tâche T et une mesure de performance P, si sa performance sur T, telle que mesurée par P, s'améliore avec l'expérience E ».*[2]

On déduit alors que l'apprentissage automatique correspond au fait d'apprendre à une machine à accumuler de la connaissance à partir d'expériences, afin de résoudre au mieux un problème considéré.

5 Historique

Les premières études sur l'AA remontent à des travaux de statistique dans les années 1920. Mais ce n'est qu'après la seconde guerre mondiale que les premières expériences deviennent possibles.

Le premier cas d'étude était sur les réseaux neuronaux en 1943, lorsque le neurophysiologiste Warren McCulloch et le mathématicien Walter Pitts ont écrit un article sur les neurones et leur fonctionnement. Ils ont décidé de créer un modèle de cela en utilisant un circuit électrique, et donc le réseau neuronal est né.

En 1950, Alan Turing a créé le test de Turing. Pour qu'un ordinateur passe, il doit être capable de convaincre un humain qu'il est un humain et non un ordinateur.

1952 a vu le premier programme informatique qui pouvait apprendre en cours d'exécution. C'était un jeu qui jouait aux dames, créé par Arthur Samuel.

Frank Rosenblatt a conçu le premier réseau de neurones artificiels en 1958, appelé Perceptron.

1967 - L'algorithme du « plus proche voisin » est écrit, permettant aux ordinateurs de commencer à utiliser la reconnaissance de formes très basique.

1981 - Gerald Dejong introduit le concept d'apprentissage basé sur l'explication (EBL), dans lequel un ordinateur analyse les données d'entraînement et crée une règle générale qu'il peut suivre en rejetant les données sans importance.

1982 -John Hopfield a suggéré de créer un réseau qui avait des lignes bidirectionnelles, semblable à la façon dont les neurones fonctionnent réellement.

En 1986, trois chercheurs du département de psychologie de Stanford ont décidé d'étendre un algorithme créé par Widrow et Hoff en 1962. Cela a donc permis de multiples couches à utiliser dans un réseau de neurones, créant ce que l'on appelle des « apprenants lents », qui apprendront sur une longue période de temps.

La fin des années 80 et les années 90 n'ont pas apporté grand-chose sur le terrain. Depuis lors, il y a eu beaucoup plus de progrès dans le domaine, comme en 1998, lorsque la

recherche aux laboratoires AT&T Bell sur la reconnaissance des chiffres a permis une bonne précision dans la détection des codes postaux manuscrits du US Postal Service. [3]

Depuis le début du 21e siècle, de nombreuses entreprises ont réalisé que l'apprentissage automatique augmentera le potentiel de calcul. C'est pourquoi ils y font des recherches plus poussées, afin de garder une longueur d'avance sur la concurrence.

Certains grands projets comprennent :

Google Brain (2012) : Il s'agissait d'un réseau neuronal profond créé par Jeff Dean de Google, qui se concentrait sur la détection de motifs dans les images et les vidéos. Il a pu utiliser les ressources de Google, ce qui le rendait incomparable à des réseaux de neurones beaucoup plus petits. Il a ensuite été utilisé pour détecter des objets dans des vidéos YouTube.[4]

OpenAI (2015) - Il s'agit d'une organisation à but non lucratif créée par Elon Musk et d'autres, pour créer une intelligence artificielle sûre qui peut bénéficier à l'humanité.[5]

Amazon Machine Learning Platform (2015) - Cela fait partie d'Amazon Web Services et démontre comment la plupart des grandes entreprises veulent s'impliquer dans le machine learning. Cette plateforme est utilisée dans divers outils de leurs systèmes internes, des services régulièrement utilisés tels que les recommandations de recherche et Alexa, à des services plus expérimentaux comme Prime Air et Amazon Go.[6]

6 Les étapes de l'apprentissage automatique

Afin de développer et gérer un modèle d'apprentissage automatique adapté à une mise en production, il faut passer par les étapes suivantes :

- Collecte des données
- Prétraitement des données
- Recherche du modèle
- Formation et test du modèle
- Évaluation.

6.1 Collecter des données

La première étape consiste à la collecte de données, celle-ci est très importante car la qualité et la quantité des données collectéesdéterminent la qualité du modèle à venir.

L'ensemble de données (*Data set* en anglais) peut être collecté à partir de diverses sources telles qu'un fichier, une base de données, un capteur et de nombreuses autres sources similaires.

Cependant les données collectées ne peuvent pas être utilisées directement, elles peuvent avoir beaucoup de champs manquants, des valeurs extrêmement grandes ou non organisées,ou bien être bruyantes. Par conséquent, la préparation des données est effectuée.

6.2 Prétraitement des données

Le prétraitement des données est un processus de nettoyage des données brutes en données propres.

Voici quelques-unes des techniques de prétraitement de base:

- Conversion des données : Comme les modèles d'apprentissage automatique ne peuvent gérer que des fonctionnalités numériques, les données catégorielles et ordinales doivent donc être converties en fonctionnalités numériques.
- Ignorer les valeurs manquantes : A la détection des données manquantes, la ligne ou la colonne les contenants peuvent être supprimées selon les besoins. Cette méthode est connue pour être efficace, mais elle ne devrait pas être appliquée excessivement.
- Remplissage des valeurs manquantes : Les données manquantes dans l'ensemble de données peuvent être remplacées manuellement par la valeur moyenne, médiane ou la plus haute fréquence utilisée.
- Apprentissage automatique : La prédiction des données manquantes en utilisant l'apprentissage automatique,en effet nous pouvons prédire quelles données seront présentes à la position vide en utilisant les données existantes.
- Détection des valeurs aberrantes : Certaines données d'erreurs qui pourraient être présentes dans l'ensemble de données s'écartent considérablement des autres observations del'ensemble de données. Par exemple : Poids humain = 800

kg ; en raison d'une faute de frappe sur l'extra 0. Celles-ci doivent être supprimées ou remplacées.

6.3 Recherche du modèle d'apprentissage

Les modèles d'AA sont homogènes aux fonctions qui prédisent une sortie pour une entrée donnée.Il existe plusieurs algorithmes selon le type d'apprentissage.

6.4 Formation et test du modèle

Pour l'entrainement d'un modèle, il faut initialement diviser le modèle en 3 trois sections qui sont : Données de formation«*Training Set*en anglais », Données de validation «*Validation Set* en anglais » et Données de test «*Testing Set* en anglais ».

On entraîne le classificateur à l'aide d'un ensemble de données d'apprentissage,ajuste les paramètres à l'aide d'un ensemble de validation, puis teste les performances du classificateur sur un « ensemble de données de test ». Un point important à noter est que pendant la formation du classificateur, seul l'ensemble d'entrainement et / ou de validation est disponible. L'ensemble de données de test ne doit pas être utilisé pendant l'entrainement du classificateur. L'ensemble de test ne sera disponible que lors du test du classificateur.

6.5 L'évaluation

Elle est une partie intégrante du processus d'élaboration du modèle. Elle aide à trouver le meilleur modèle qui représente les données et à quel point le modèle choisi fonctionnera à l'avenir.

7 Types d'apprentissage automatique

Pour donner à la machine la capacité d'apprendre, on utilise différentes méthodes d'apprentissage automatique :

7.1 Apprentissagesupervisé

Cette approche consiste à faire apprendre une machine à travers des exemples d'entrées qui sont labellisés avec la sortie souhaitée afin que l'algorithme puisse prédire les valeurs des sorties en fonction des entrées.

Plus formellement, étant donné un ensemble de données D, décrit par un ensemble de caractéristiques X, un algorithme d'apprentissage supervisé va trouver une fonction f en entrée x et la variable à prédire y.

La fonction décrivant la relation entre x et ys'appelle un modèle de prédiction.D'une manière générale, la machine peut apprendre une relation $f: x \rightarrow y$ qui relie x à y en ayant analysé des millions d'exemples d'associations $x \rightarrow y$et l'utiliser pour catégoriser les données non triées par la suite.

On distingue deux problèmes d'apprentissage supervisé la classification et la régression que nous décrivons dans ce qui suit :

7.1.1 Classification

La classification consiste à prédire la valeur ou bien la classe d'une variable d'entrée discrète.

Par exemple,la classification peut servir àprédire si un email est un spam ou non selon le nombre de liens présent dans l'email.

7.1.2 Régression

La régression consiste à prédire la valeur ou bien la classe d'une variable d'entrée continue. Par exemple, en utilisant la régression on peut prédirele prix d'un appartement selon sa surface habitable.

7. 2 Apprentissage non supervisé

On parle d'apprentissage non supervisé quand on ne dispose pas d'étiquettes, l'algorithme doit apprendre tout seul à trouver le point commun entre les données d'entrée et les regrouper dans des clusters ou classes.

On prend comme exemple la reconnaissance d'images. L'algorithme va pouvoir regrouper des images de voiture dans un cluster et des immeubles dans un autre cluster ainsi de suite, sans la structure ou le concept de l'objet ne soit prédéfinitqui est dans notre cas une voiture et un immeuble ; il saura seul que le point en commun entre les images de voitures est la roue, et qu'un bâtiment a des forme plus droite.

7. 3 Apprentissage semi-supervisé

L'apprentissage semi-supervisé est une combinaison entre l'apprentissage supervisé (donnée labellisé) et l'apprentissage non supervisé (donnée non labellisé).Cette approche a pour but de prédire les labels des données non labellisé et améliorer les performances.

7.4 Apprentissage par renforcement

L'algorithme va apprendre à se comporter à partir d'un environnement réel ou simulateur, en interagissant avec ce dernier afin de construire son propre data set (données).

7.5 Apprentissage par transfert

L'apprentissage automatique permet de sauvegarder toutes les données et connaissances déjà traité ; de ce fait, on peut résoudre une tâche à partir d'une autre tâche similaire déjà résolu.

8 Les modèles de l'apprentissage automatique

Il existe plusieurs algorithmes d'apprentissage automatique selon la complexité du problème.

8.1 Approche basée sur les K plus proches voisins (KNN)

Le k-NN est le diminutif de k Nearest Neighbors, plus proches voisins en français. Il est utilisé pour les problèmes de régression mais beaucoup plus pour ceux de classification.

C'est une approche très simple et directe. Pour effectuer une prédiction, l'algorithme ne va pas calculer un modèle prédictif à partir d'un ensemble d'entrainement.

Ainsi, pour lui il n'existe pas de phase d'apprentissage à proprement dire, mais il va simplement réaliser le stockage des données d'apprentissage.

Son principe est le suivant : Une donnée de classe inconnue est comparée à toutes les données stockées. La nouvelle donnée sera la classe majoritaire parmi ses K plus proches voisins (elle peut donc être lourde pour des grandes bases de données) au sens d'une distance choisie.

Afin de trouver les K plus proches d'une donnée à classer, on peut utiliser la distance euclidienne. Soient deux données représentées par deux vecteurs xi et xj , la distance entre ces deux données est donnée par : $d(x_i, x_j) = \sqrt{\sum_{k=1}^{d}(x_{ik} - x_{jk})^2}$ [1]

8.2 La méthode k -moyenne (K-MEANS)

K-means est un algorithme non supervisé de classification non hiérarchique qui permet de répartir un ensemble de données caractérisées par des descripteurs en k classes homogènes.

K-means stocke kcentroids [1]qu'il utilise pour définir les classes. Un point est considéré comme étant dans une classe particulière s'il est plus proche du centre de gravité de celle-ci que de tout autre centre de gravité.

K-Means trouve les meilleurs centroïdes en alternant entre l'attribution de points de données aux classes en fonction des centroïdes actuels.

8.3 Les arbres de décision

Les arbres de décision sont une catégorie d'arbres qui emploient une représentation hiérarchique de la structure des données qualitatives sous forme des séquences de décisions (tests) en vue de la prédiction d'un résultat ou d'une classe. Chaque individu (ou observation), qui doit être attribué(e) à une classe, est décrit(e) par un ensemble de variables qui sont testées dans les nœuds de l'arbre. Les tests s'effectuent dans les nœuds internes et les décisions sont prise dans les nœuds feuille.

Exemple,les arbres de décision peuvent répartir une population d'individus en groupes homogènes selon un ensemble de variables descriptives (ex. âge, temps passé sur un site Web, etc.) et en fonction d'un objectif fixé (variable de sortie, par exemple : chiffre d'affaires, probabilité de cliquer sur une publicité, etc.).

8.4 Classification naïve de Bayes

C'est un algorithme qui traite les problèmes de classification. Il est principalement utilisé pour la classification du texte dans des données volumineuses.

1 Les centroids sont des points qui sont le centre d'un cluster.

L'algorithme apprend à calculer la probabilité que l'objet va appartenir à une classe bien particulière ; grâce à ce fait, on peut construire des modèles (programme) rapidement et faire des prédictions rapides.

La classification naïve de bayes considère que les caractéristiques (attribut) d'un objet sont indépendant. Elle est basée sur le théorème de bayes qui ne s'applique que sous cette hypothèse.

Le théorème de bayes qui nous permet de calculer la probabilité conditionnelle d'un événement x sachant que l'événement y s'est réalisé ; il est donné comme suit :

$$p(x/y) = \frac{p(y/x)\mathrm{p(x)}}{p(y)} [2]$$

Dans le cas d'un problème de classification on pose H, tel que H est l'hypothèse qui dit : qu'un vecteurs d'attribut X appartient à une classe C. Et on suppose qu'on cherche à trouver la probabilité que H soit vrai sachant X :

$$p(H/x_1 \dots x_n) = \frac{p(x_1 \dots x_n/H).p(H)}{p(x_1 \dots x_n)} pour\ 1 \leq i \leq\ \ k\ [3]$$

Maintenant supposons qu'on a un vecteur de caractères $X(x_1 \dots x_n)$et on cherche à trouver la probabilité maximale d'appartenance à une classe:

$$p(X) = argmax\ p(x_1 \dots x_n/H).p(H)\ \ [4]$$

Cependant le fait de savoir que les attributs sont indépendants, permet de décomposer la probabilité conditionnelle en un produit de probabilités conditionnelles. Le classifieur devient alors : $classe(x) = argmax\ \prod_{i=1}^{n} p(x_i/H)\,.p(H)[5]$

9 Les machines à vecteurs de support(SVM)

Conçu pour résoudre les problèmes de classification, le SVM est une séparation linéaire entre deux catégories d'exemples qui sont fournies à notre algorithme. Ces exemples sont des données d'entrées représentés par des vecteurs de dimension n.

Le SVM va désigner des vecteurs supports qui vont déterminer un hyperplan pour pouvoir classer de nouvelles données d'entrées (voir la figure 2).

Afin d'avoir un hyperplan optimal, on doit choisir la marge maximale entre les deux catégories d'exemples.

L'avantage de cet algorithme est que même si une donnée d'entrée n'est pas similaire aux exemples, elle sera classée.

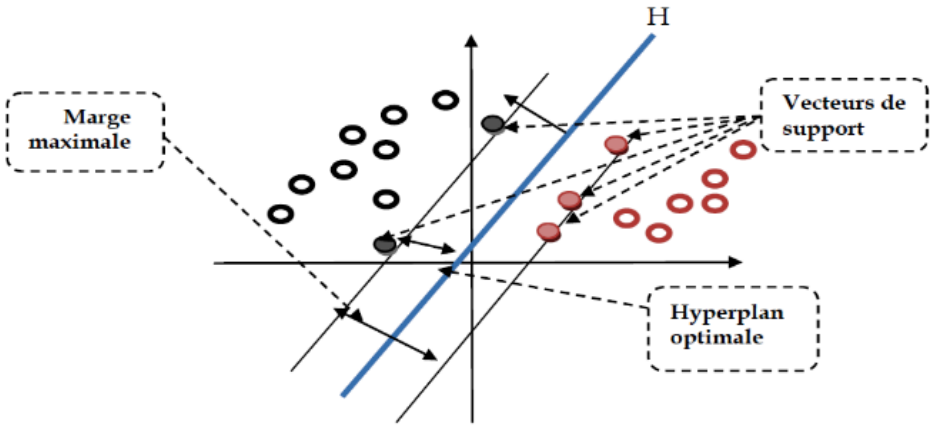

Figure 2: L'hyperplan H optimale, vecteurs supports et marge maximale

10 Les réseaux de neurones

Les réseaux de neurones (RN) sont des systèmes de traitement de l'information dont la structure s'inspire de celle du système nerveux. [7]

C'est 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. Ainsi toute structure hiérarchique de réseaux est évidemment un réseau.

10.1 Structure

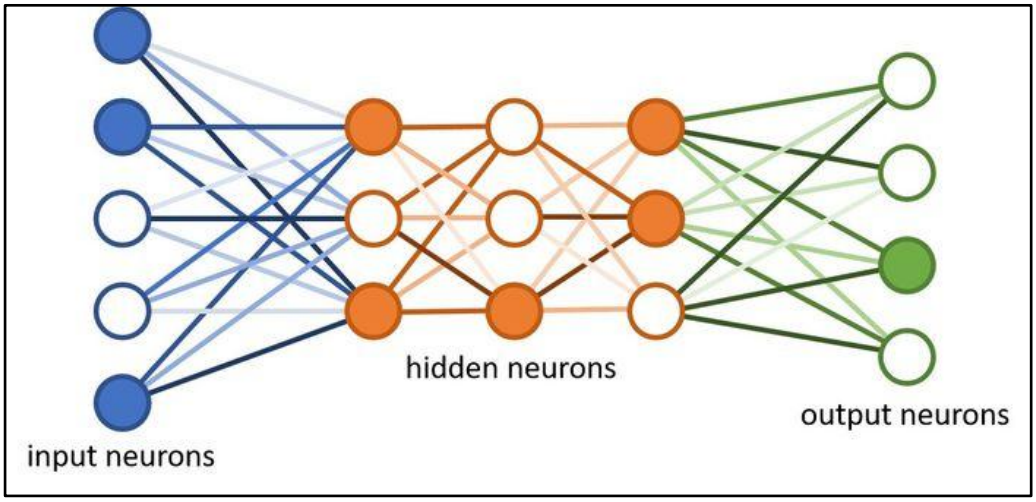

Figure 3 : Structure d'un réseau neuronal artificiel

La figure 3 montre la structure d'un neurone artificiel.

Fondamentalement, chaque neurone d'un réseau de neurones n'est qu'une fonction mathématique. Chaque neurone calcule une somme pondérée de ses entrées (plus le poids d'une entrée est important, plus cette entrée affecte la sortie du neurone). Cette somme pondérée est ensuite injectée dans une fonction non linéaire appelée fonction d'activation (une étape qui permet aux réseaux de neurones de modéliser des phénomènes non linéaires complexes).

10.2 Comportement

Un réseau neuronal typique comprend quelques dizaines voire des millions de neurones artificiels appelés unités disposées en une série de couches, chacune se connectant aux couches de chaque côté.

Certains d'entre eux, appelés unités d'entrée (*Input layers* en anglais), sont conçus pour recevoir diverses formes d'informations du monde extérieur que le réseau tentera de découvrir, de reconnaître ou de traiter d'une autre manière.

D'autres unités sont installées de l'autre côté du réseau et signalent comment il réagit aux informations qu'il a apprises ; ce sont les unités de sortie (*Output layers* an anglais). Entre les unités d'entrées et les unités de sortie se trouvent une ou plusieurs couches d'unités cachées (HIDDEN LAYERS), qui, ensemble, forment la majorité du cerveau artificiel.

La plupart des réseaux de neurones sont entièrement connectés, ce qui signifie que chaque unité cachée et chaque unité de sortie est connectée à chaque unité des couches de chaque côté.

Les connexions entre une unité et une autre est représenté par un nombre appelé poids, qui peut être positif (si une unité en active une autre) ou négatif (si une unité en supprime ou en inhibe une autre). Plus le poids est élevé, plus une unité a d'influence sur une autre. (Cela correspond à la façon dont les cellules cérébrales se déclenchent à travers de minuscules lacunes appelées synapses.

11 Les applications de l'apprentissage automatique

Les cas d'usages de l'AA sont nombreux, afin d'illustrer l'utilité de l'AA nous citons :

- La Fouilles de données (*data mining* en anglais) : utilisé pour l'extraction d'informations. Elle permet d'utiliser toutes les données qui se trouve des bases de données (BDD) hétérogènes, afin de générer des programmes pour l'apprentissage. Par exemple : trouver une prescription pour un patient à travers des fichiers médicaux antérieurs.
- La robotique : Les robots sont maintenant omniprésents, surtout dans les usines. Ils aident, par exemple, dans la production de masse à automatiser des étapes de travail cohérente
- Transcription automatique de la parole (*Chatbot*) : désigne un agent automatisé avec lequel il est possible d'interagir par texte via les principales plateformes de messagerie telles que Facebook, Messenger ou Skype.
- La reconnaissance d'objets ou de personnes dans des images, reconnaissance du langage parlé. Exemple : Logiciel biométrique de reconnaissance de visages et d'empreintes digitales.

12 Conclusion

Dans ce chapitre, nous avons présenté les grandes lignes de l'apprentissage automatique. Nous avons exploré son fonctionnement ainsi que les algorithmes utilisés. Enfin, nous avons exposé d'une part quelques limitations rencontrées en utilisant ces algorithmes et les différents domaines d'applications de l'AA.

Dans le prochain chapitre, nous présentons l'application de l'apprentissage automatique dans notre projet à savoir nos modèles de réseaux de neurones artificiels. En plus, nous analysons leurs architectures ainsi que toutes les configurations ajoutées pour leurs bons fonctionnements.

Chapitre 2 :Les réseaux de neuroneset le traitement automatique du langage naturel

1 Introduction

Tenter de donner un sens au contenu est une nécessité impérieuse de nos jours, c'est pourquoi les scientifiques ont associé l'IA, les statistiques et la linguistique afin de créer un traitement automatique du langage naturel qui est devenu par la suite l'un des principes clés de l'IA et qui s'est développé encore plus avec les nouveaux algorithmes de l'apprentissage profondplus particulièrement les réseaux deneurones.

Dans ce chapitre, nous présentons dans un premier temps des généralités sur le conceptde base des réseaux de neurones. Ensuite on parlera des types d'entraînements ainsi que le principe de l'algorithme du gradient tout en expliquant quelques généralités telles que les fonctions d'activations, les algorithmes d'apprentissage et les différentes architectures de réseaux de neurones. Enfin, nous concluonsen décrivant certains défis et inconvénients à l'utilisation des réseaux de neurones, comme les phénomènes de sur-apprentissage et desous-apprentissage et les réseaux très profonds.

2 Généralités sur les réseaux de neurones

2.1 Neurone formel

Les neurones formelssont à l'origine inspirés du fonctionnement des neurones biologiques[1], afin de trouver une similitude entre eux.

Le neurone formel est un automate qui reçoit des informations en entrée (données), calcule un résultat qu'il transmet ensuite aux neurones suivant par le biais d'une fonction d'activation selon des règles précises.

Les composants élémentaires d'un neurone formel sont :

- Les entrées : notéesX_i, sont des valeurs numériques représentées sous forme de vecteur.
- Poids : noté W_i, associé à chaque entrée, ces poids vont donner une certaine importance à chaque neurone.
- Biais : noté B, représente une unité fictive associée à un poids qui permet de contrôler si un neurone est actif

- Fonction d'activation : noté F, est une équation mathématique qui représente le seuil de [1]stimulation qui, une fois atteint entraîne une réponse du neurone. Plus précisément, cette fonction va permettre de déterminer la valeur de sortie du neurone Y.
- Sortie : notée Y_i, c'est le résultat de l'application de la fonction d'activation à la somme pondéré des entrées d'où l'équation suivante :

$$Y = F\left(\left(\sum_{i=1}^{n} X_i W_i\right) + B\right) [6]$$

n est le nombre d'entrées de neurones.

F est la fonction d'activation.

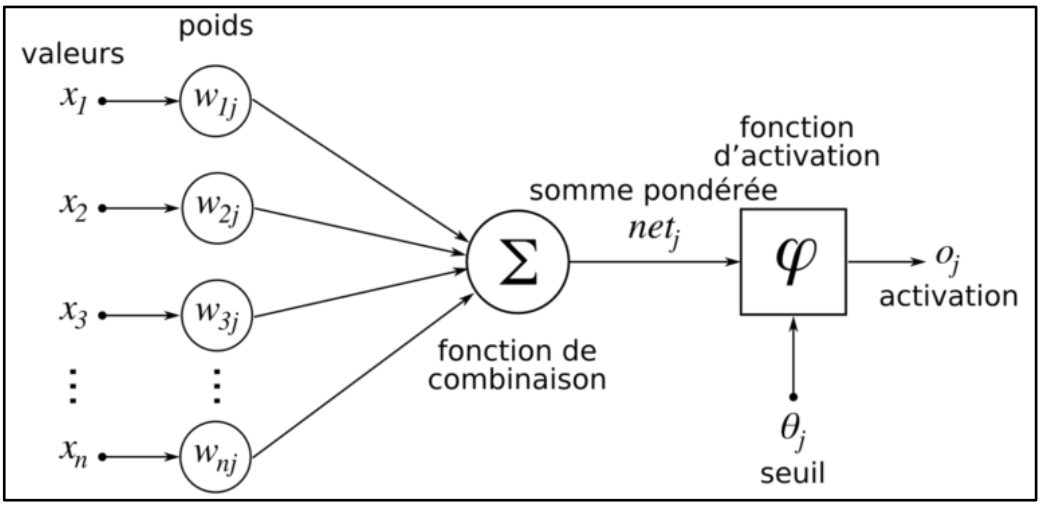

Figure 4 : Structure d'un neurone formel

2.2 Fonctionnement

Chaque neurone reçoit des entrées sous forme vectorielle puis il calcule une somme pondérée de ses entrées pour que le résultat passe ensuite par la fonction d'activation afin de créer une sortie.

Au cours de l'apprentissage, le neurone modifie ses poids grâce à un hyper-paramètre appelé taux d'apprentissage (learning rate) afin d'atteindre la borne minimale lors de l'utilisation du gradient descent et aboutir à la sortie souhaitée.

1 Il s'agit d'une cellule du système nerveux capable de communiquer et de traiter des informations. On le trouve dans le cerveau mais aussi dans la moelle épinière et les nerfs optiques

La modification se fait grâce à un algorithme d'apprentissage qui permet de calculer l'erreur entre la sortie prédite et celle souhaitée. Cet entrainement est itératif, chaque itération est appelée époque (epoch).

3 Types d'entraînements des réseaux de neurones

On distingue deux type d'entraînement selon le problème étudié : récurrent et propagation vers l'avant [8] :

3.1 Entrainement acyclique « Feed-*Forward Propagation* »

Un type d'entraînement réseau simple qui permet la propagation de données dans un seulsens, d'entrée en sortie sans revenir en arrière, Chaque couche cachée accepte les données d'entrée, les traites selon la fonction d'activation et passe à la couche suivante.

3.2 Entrainement cyclique « *Feed-Back* »

Ce type d'entraînement est caractérisé par la possibilité de retour en arrière de l'information. Ce processus itératif sert à calculer les mises à jour des pondérations afin d'améliorer le réseau jusqu'à ce qu'il soit capable d'exécuter la tâche pour laquelle il est traité. Il se compose de deux étapes qui sont représentées par la figure ci-dessous (figure 5) :

3.2.1 Feed-Forward propagation

Elle est composée principalement de trois phases :

- Phase d'initialisation : cette étape consiste à donner des valeurs aléatoires des poids et des bais.
- Phase d'insertion : début de l'apprentissage en insérant les données dans le modèle.
- Calcul de la fonction de coût : lorsqu'on teste notre modèle sur les données, celui-ci nous donne des erreurs de prédiction. L'ensemble de ces erreurs, c'est ce qu'on appelle la Fonction Coût.

3.2.2 Back propagation

Après l'application de l'algorithme du gradient, cette étape permet de retourner l'erreur entre la valeur réelle observée et la valeur nominale demandée dans la couche de sortie en

la propageant vers l'arrière dans l'ordre pour modifier les poids et les valeurs de biais en les calculant avec la formule suivante :

$$\beta_k^{(r+1)} = \beta_k^r - \tau \sum_{i=1}^{n} \frac{\partial \varphi_i}{\partial \beta_k^r} : \ nouveau\ bias\ [7]$$

$$\alpha_k^{(r+1)} = \alpha_{kp}^r - \tau \sum_{i=1}^{n} \frac{\partial \varphi_i}{\partial \alpha_{kp}^r} : \ nouveau\ poids\ [8]$$

Où$\frac{\partial \varphi_i}{\partial \beta_k^r}$est le taux d'apprentissage : dérivée de la fonction de coûts par rapport au bias (poids).

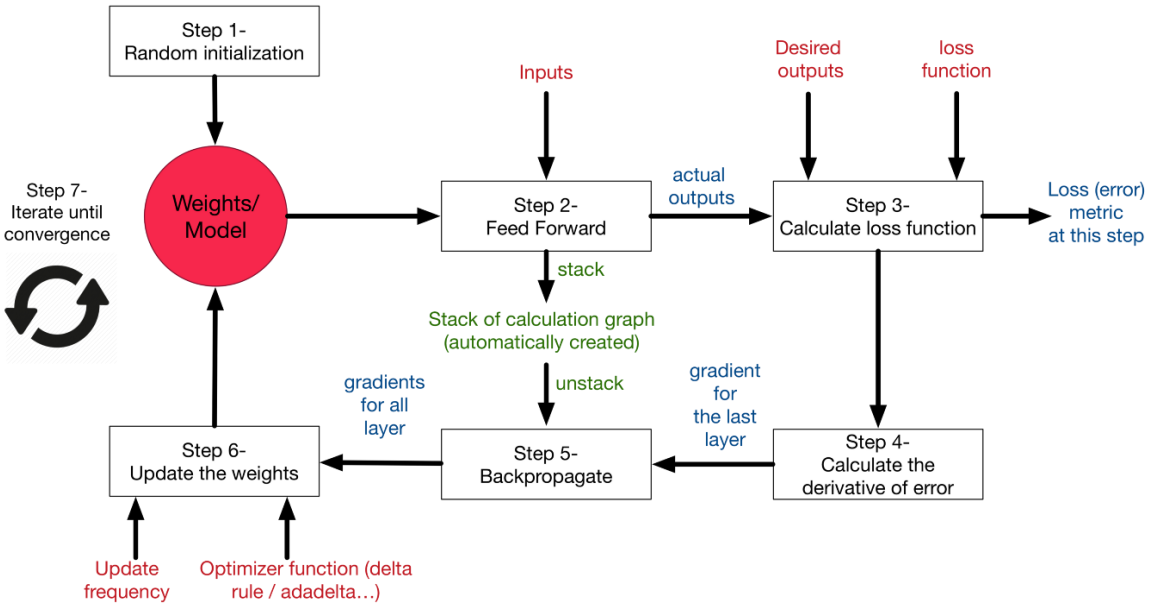

Figure 5 : Entraînement avec rétro-propagation

4 Principe du gradient descent

La descente de gradient (*Gradient Descent* en anglais)[9] est un algorithme d'optimisation qui permet de trouver les paramètres qui minimisent la fonction coût, c'est-à-dire les paramètres qui nous donnent le meilleur modèle.Après avoir sélectionné aléatoirement les paramètres :

1. Calculer la pente (sa dérivée première) de la fonction coût.
2. Évolué d'une certaine distance appelé taux d'apprentissage dans la direction de la pente la plus forte. Cela a pour résultat de modifier les paramètres (poids des neurones et le biais).

3. Utiliser en boucle la descente de gradient pour mettre à jour nos paramètres dans la direction de la Fonction Coût la plus faible.

5 Fonctions d'activations usuelles

La fonction d'activation [10]Est une fonction mathématique appelée aussi fonction de transfert, utilisé par chaque neurone pour le passage de la couche d'entrée vers la couche suivante. Elle décide si un neurone doit être activé ou non en calculant la somme pondérée en ajoutant un biais (variable aléatoire) qui sera modifié au fur àmesure afin d'avoir le résultat souhaité. Le but de la fonction d'activation est d'introduire une non-linéarité dans la sortie d'un neurone afin de la rendre capable d'apprendre et d'effectuer des tâches plus complexes.

La fonction d'activation joue un grand rôle sur la performance du réseau c'est pourquoi il est important de bien choisir le type de fonction d'activation des neurones dans un réseau de neurones.

Dans ce qui suit, nous présentons quelques-unes des fonctions d'activations les plus utilisées dans les réseaux de neurones.

5.1 Fonction linéaire

La fonction linéaire à l'équation similaire à celle d'une ligne droite, c'est-à-dire : $Y = aX$. Peu importe le nombre de couches que nous avons, si toutes sont de nature linéaire, la fonction d'activation finale de la dernière couche n'est rien d'autre qu'une fonction linéaire de l'entrée de la première couche.

Sa courbe est représentée comme suite :

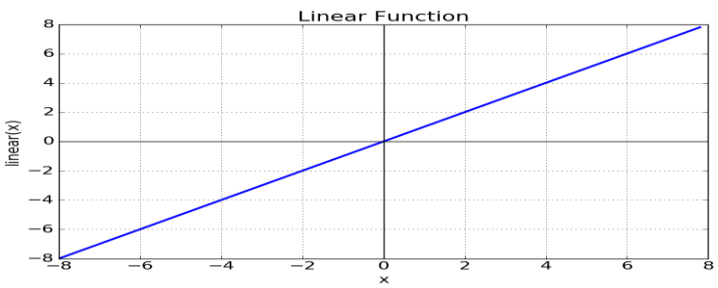

Figure6 : Courbe de la fonction linéaire

5.2 Fonction sigmoïde

C'est l'une des fonctions d'activation non linéaire les plus utilisées, Voici son expression mathématique : $f(x) = \frac{1}{1+e^{-x}}$. La principale raison pour laquelle nous utilisons la fonction sigmoïde est qu'elle donne en sortie des valeurs comprises entre 0 et 1. Par conséquent, il est particulièrement utilisé pour les modèles où nous devons prédire la probabilité en tant que sortie.

C'est une fonction qui est tracée sous forme de graphique en forme de « S » comme on peut le voir dans la figure ci-dessous :

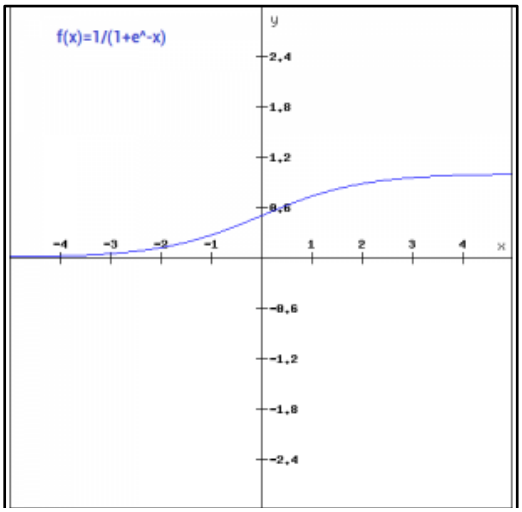

Figure 7 : Courbe de la fonction Sigmoïde

5.3 Fonction Soft max

La fonction Soft max est un type de fonction sigmoïde qui est pratique lorsque nous essayons de gérer des problèmes de classification.

Cette fonction renvoie la probabilité pour un point de données appartenant à chaque classe individuelle.

5.4 Fonction Tanh

C'est très similaire à la fonction sigmoïde. La seule différence est qu'il est symétrique autour de l'origine. La plage de valeurs dans ce cas va de -1 à 1. Ainsi, les entrées des couches suivantes ne seront pas toujours du même signe. La fonction Tanh est définie comme : $f(x) = \tanh(x) = \frac{2}{1+e^{-2x}}$

Sa courbe est représentée comme suite :

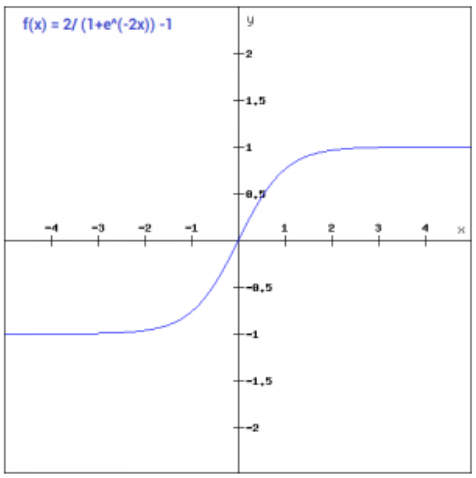

Figure 8 :Courbe de la fonction Tanh

5.5 La fonction Relu

La fonction Relu est une autre fonction d'activation non linéaire qui est défini par $A(x) = \max(0, x)$: Elle donne une sortie x si x est positif et 0 sinon.

Relu signifie *Rectified Linear Unit en anglais.* Le principal avantage de l'utilisation de cette fonction par rapport aux autres fonctions d'activation est qu'elle n'active pas tous les neurones en même temps.

Sa courbe est représentée comme suite :

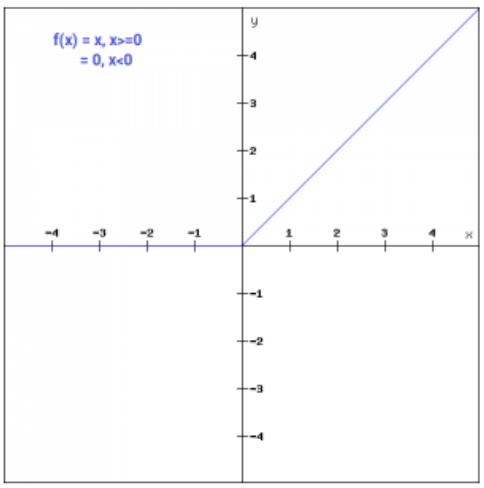

Figure9 : courbe de la fonction Relu

5.6 Leaky Relu

La fonction Leaky Relu n'est rien d'autre qu'une version améliorée de la fonction Relu. En effet, pour la fonction Relu, le gradient est 0 pour x <0, ce qui désactivera les neurones dans cette région. Leaky Relu est défini pour résoudre ce problème. Au lieu de définir la

fonction Relu à 0 pour x < 0, nous la définissons comme une petite composante linéaire de x comme suit :$f(x) = \begin{cases} ax, & x < 0 \\ x, & x \geq 0 \end{cases}$

Sa courbe est représentée comme suite :

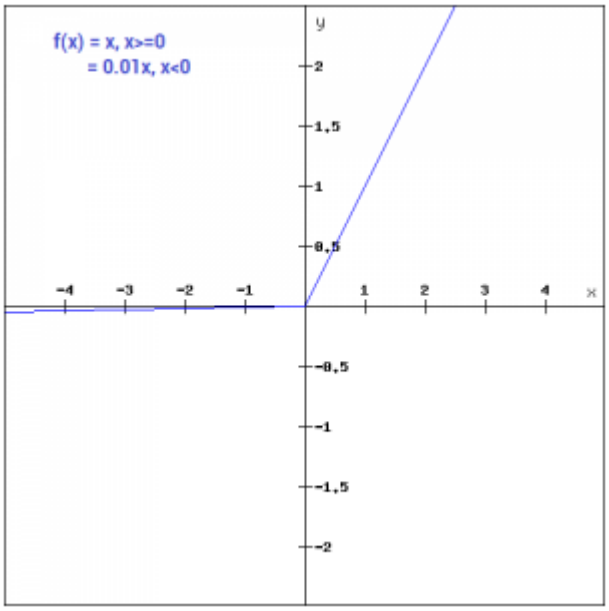

Figure 10 : Courbe de la fonction Leaky Relu

6 Phénomènes de sur-apprentissage (*overfitting*) et de sous-apprentissage (*underfitting*)

Lors du développement des modèles de réseaux de neurones, les programmeurs rencontrent souvent des problèmes de performance liés à leurs conceptions où à d'autres paramétrages.

Nous citons dans ce qui suit les deux problèmes les plus courants qui causent un dysfonctionnement du réseau neuronal artificiel.

6.1 Sur-apprentissage

Le sur-apprentissage [11] se produit lorsqu'un modèle apprend les détails et le bruit dans les données d'apprentissage d'où son impact négatif sur les performances du modèle sur les nouvelles données. Cela signifie que le bruit ou les fluctuations aléatoires des données d'apprentissage sont captés et appris en tant que concepts par le modèle. De ce fait, il réalise de mauvaises prédictions lors des étapes de validation et de test.

6.2 Sous-apprentissage

Le sous-apprentissage[12]se produit lorsqu'on ne dispose pas de suffisamment de données d'entrainement ou que le modèle est mal configuré de telle sorte à ce que, les performances des modèles sont mauvaises, même sur l'ensemble d'entrainement. Par conséquence ce dernier n'a pas bien appris à partie des données qu'on lui a fourniet pas assez puissant pour obtenir des prédictions correctes.

7 Architecture des réseaux de neurones

L'architecture d'un réseau de neurones [15] est l'organisation des neurones entre eux au sein d'un même réseau.

L'architecture d'un réseau de neurones dépend de la tâche à apprendre. On distingue deux grands types d'architectures de réseaux de neurones : les réseaux de neurones non bouclés et les réseaux de neurones bouclés. Il y a également un autre type de réseaux de neurones, dit convolutionnel qui est a priori non bouclé mais son mécanisme de fonctionnement différents de ceux d'un réseau de neurone non bouclé classique.

7.1 Les réseaux de neurones non bouclés

Un réseau de neurones non bouclé est représenté graphiquement par un ensemble de neurones connectés entre eux, l'information circulant des entrées vers les sorties sans retour en arrière. Le graphe d'unréseau non bouclé est acyclique.

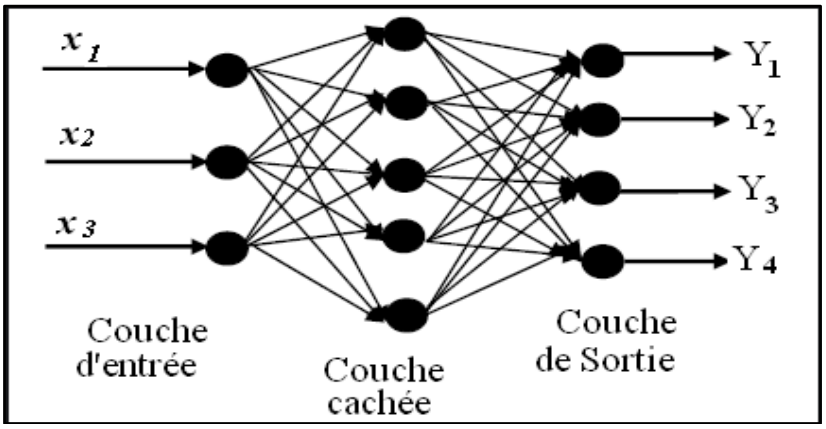

Figure 12 : Structure d'un réseau de neurones non bouclé

7.1.1 Le perceptron

Considérés comme la première génération de réseaux de neurones, les perceptrons ne sont que des modèles de calcul d'un seul neurone. Ils ont été popularisés par Frank Rosenblatt au début des années 1960.

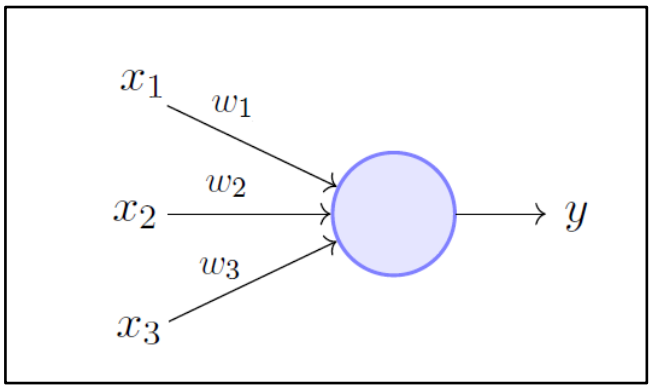

Figure 13 : Modèle de perceptron de Minsky –Papert (1969)

L'architecture d'un perceptron est définie comme suite : les entrées sont envoyées dans le neurone, sont traitées et donnent lieu à une sortie. C'est un réseau dit simple car il ne se compose que de deux couches : une couche d'entrée X_i et une couche de sortie Y ce qui implique une seule matrice de poids W_i.

7.1.2 Couches entièrement connectées

Les neurones sont arrangés par couche. Il n'y a pas de connexion entre neurones d'une même couche, et les connexions ne se font qu'avec les neurones de couches avales. Habituellement, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante et celle-ci seulement.

Ils se composent d'une couche d'entrée, d'une couche de sortie et d'une ou plusieurs couches cachées.

7.2 Réseaux récurrents

Les réseaux de neurones récurrents (RNN)[16] sont un type de réseau de neurones où la sortie de l'étape précédente est envoyée en entrée à l'étape en cours. Dans les réseaux neuronaux traditionnels, toutes les entrées et sorties sont indépendantes les unes des autres, mais dans des cas comme lorsqu'il est nécessaire de prédire le mot suivant d'une phrase, les mots précédents sont requis et donc il faut se souvenir des mots précédents. C'est ainsi que RNN a vu le jour, ce qui a résolu ce problème avec l'aide d'une couche cachée.

Les RNN ont une mémoire qui se souvient de quelques informations sur ce qui a été calculé. Il utilise les mêmes paramètres pour chaque entrée car il effectue la même tâche sur toutes les entrées ou couches cachées pour produire la sortie. Cela réduit la complexité des paramètres, contrairement à d'autres réseaux de neurones.

Parmi les types de réseaux de neurones récurrents, nous citons : les réseaux de mémoire à long terme à court terme (LSTM) [17] et leur variante appelée Gated Recurrent Unit (GRU) [16].

7.2.1 LSTM

Les réseaux de mémoire à long terme à court terme[17] sont un type particulier de RNN, conçus pour éviter le problème de dépendance à long terme.

Leur caractéristique principale c'est de se souvenir des informations pendant de longues périodes. Ils ont été introduits par Hochreiter et Schmidhuber (1997). Ils fonctionnent extrêmement bien sur une grande variété de problèmes et sont maintenant largement utilisés.

Les LSTM sont construit d'une cellule mémoire où les informations sont sauvegardées, et de trois portes pondérées dont les poids sont appris durant l'apprentissage ce qui détermine la bonne connexion entre elles. Ces portes sont : porte d'entrée (Décide de la quantité d'informations provenant de l'entrée actuelle vers l'état de la cellule mémoire), porte de sortie (détermine les conditions pour qu'une valeur en mémoire détermine la valeur de sortie) et la porte d'oubli (détermine les contraintes pour qu'une information reste en mémoire).

7.2.2 GRU

GRU est un mécanisme de déclenchement dans les réseaux de neurones récurrents (RNN) similaire à LSTM mais sans porte de sortie.

Les GRU possèdent deux portes : La porte de mise à jour contrôle les informations qui s'écoulent en mémoire et la porte de réinitialisation contrôle les informations qui sortent de la mémoire. La porte de mise à jour et la porte de réinitialisation sont deux vecteurs qui décident quelles informations seront transmises à la sortie. Ils peuvent être formés pour conserver des informations du passé ou supprimer des informations non pertinentes pour la prédiction.[18]

7.3 Réseaux convolutionnels (CNN)

Les réseaux de neurones convolutifs (CNN)sont des réseaux de neurones multicouches et plus précisément profond, composés de multiples couches. Ils se sont révélés très efficaces dans les tâches impliquant des données étroitement liées, principalement dans le domaine de la vision par ordinateur. Le fonctionnement du cortex visuel des animaux a inspiré les chercheurs pour concevoir un tel réseau.

Un CNN utilise une structure tridimensionnelle dans laquelle les neurones d'une couche ne se connectent pas à tous les neurones de la couche suivante.

Elle est constituée de couches convolutionelles et de couches de pooling.[19]

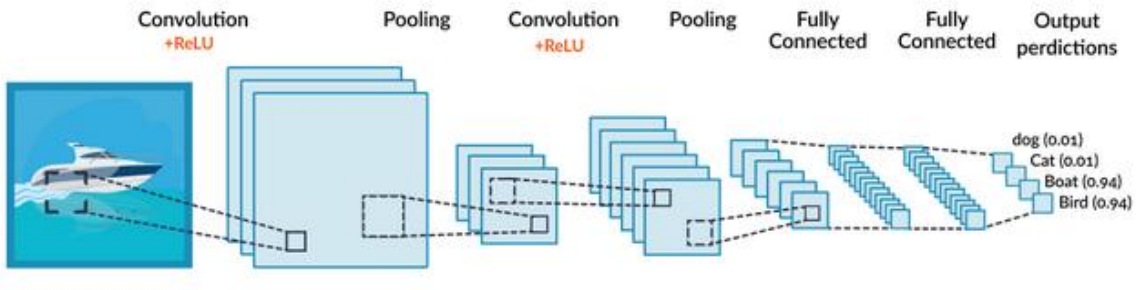

Figure 14 : Architecture convolutionelle (CNN)

Comme la figure ci-dessus le montre, un CNN peut effectuer plusieurs cycles de convolution puis de pooling. Enfin, lorsque les caractéristiques sont au bon niveau de granularité, cela crée un réseau neuronal entièrement connecté qui analyse les probabilités finales et décide à quelle classe l'image appartient. L'étape finale peut également être utilisée pour des tâches plus complexes, telles que la génération d'une légende pour l'image.

7.3.1 Couche de convolution

La couche convolutionnelle (*en anglais convolution layer*) est le bloc de construction de base d'un CNN qui traite les données d'un champ récepteur. C'est la première couche à extraire les caractéristiques des données entrées par exemple des images.

Trois hyper paramètres permettent de dimensionner le volume de la couche de convolution :

Profondeur de la couche : nombre de noyaux de convolution (ou nombre de neurones associés à un même champ récepteur)

Le pas (*stride*) : 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

Marge à zéro (*zeropadding*) : Cette marge permet de contrôler la dimension spatiale du volume de sortie. Il est commun de mettre des zéros à la frontière du volume d'entrée. En particulier, il est parfois souhaitable de conserver la même surface que celle du volume d'entrée.

Couche de pooling : est une nouvelle couche ajoutée après la couche convolutionelle, elle est responsable de la réduction de la taille spatiale de l'entité convolutionnée. Il s'agit de diminuer la puissance de calcul requise pour traiter les données grâce à la réduction de dimensionnalité.

Les types de pooling les plus populaires sont :

1. Le Max-Pooling : Chaque opération de pooling, renvoie la valeur maximale de la surface.

2. L'Average-Pooling : Chaque opération de pooling, renvoie la moyenne de toutes les valeurs de la surface.

7.4 Les réseaux très profonds (*Residual neural network en anglais*)

Lorsque des réseaux plus profonds [20] peuvent commencer à converger, un problème de dégradation a été exposée, avec l'augmentation de la profondeur du réseau, la précision est saturée (ce qui n'est peut-être pas surprenant) puis se dégrade rapidement.

L'utilisation de réseaux plus profonds dégrade les performances du modèle. Microsoft tente de résoudre ce problème en utilisant la structure d'apprentissage résiduel profond.

L'approche consiste à ajouter un raccourci ou une connexion de saut qui permet aux informations de circuler, disons simplement, plus facilement d'une couche à la couche suivante.

On distingue deux types de blocs utilisé dans les réseaux très profonds :

- **Le bloc d'identité (The identity block) :**Le bloc d'identité est le bloc standard utilisé au cas où l'activation d'entrée à la même dimension que l'activation de sortie.
- **Le bloc convolutif (The Convolutional block) :** Il est utilisé lorsque les dimensions d'entrée et de sortie ne correspondent pas.

8 Réseaux de neurones et algorithmes d'apprentissage

La raison d'être de tout algorithme d'apprentissage réside dans sa capacité à apprendre quelque chose des données qui lui sont fournies. Pour que cela soit possible, il y a plusieurs fonctions à mettre en place, plusieurs techniques que l'on peut appeler selon le type de problème et le type de données ainsi que la taille de la base d'apprentissage.

Parmi ces algorithmes d'apprentissage, nous citons ce qui suit :

8.1.1 La rétro propagation du gradient

L'algorithme de rétro propagation est une méthode d'apprentissage supervisé pour les réseaux multicouches à action directe du domaine des réseaux de neurones artificiels.

La rétro-propagation (en anglais back-propagation) vise à minimiser la fonction de coût en ajustant les pondérations et les biais du réseau. Le niveau d'ajustement est déterminé par les gradients de la fonction de coût par rapport à ces paramètres.

La rétro-propagation du gradient consiste donc à effectuer une rétro-propagation. C'est un algorithme itératif qui s'applique autant de fois que nécessaire pour obtenir une meilleure prédiction et ainsi diminuer l'erreur dans le réseau.

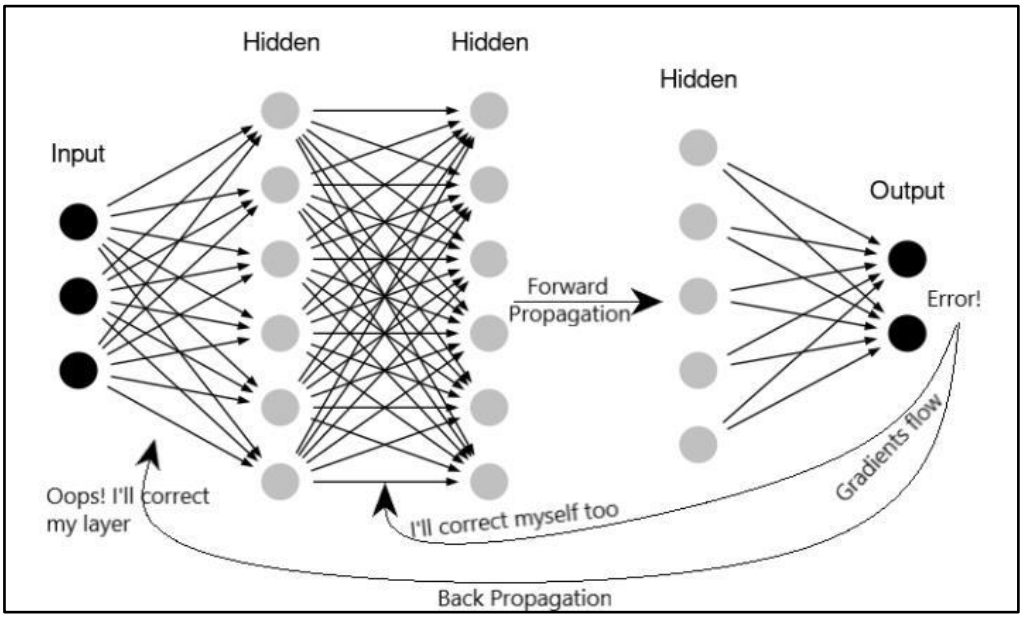

Figure 11 : Principe de la rétro-propagation du gradient

.

8.1.2 Adagrad

AdaGrad *(nom dérivé de : Adaptative Gradient*)[13] est une méthode adaptative conçu pour régler des problèmes où il est impossible de choisir manuellement différents taux d'apprentissage pour chaque paramètre. Cet algorithme modifie le taux d'apprentissage pour chaque paramètre d'entraînement afin d'avoir un processus d'apprentissage avec de bons résultats.

8.1.3 Adam

Adam (*nom dérivé de : Adaptive Moment Estimation*) [14] est un algorithme qui calcule des taux d'apprentissage adaptatifs.

Son principe est simple. Après avoir calculé une moyenne exponentielle pondérée des gradients (ils permettent d'effectuer des descentes de gradients) passés, il calcule une moyenne exponentielle pondérée des carrés des gradients passés. Si le gradient est dans la même direction que ceux précédents, il accélère la vitesse d'apprentissage en mettant à jour les paramètres.

9 Définition du traitement automatique du langage naturel

Le vocable de traitement automatique du langage naturel (TALN), est une discipline qui se trouve à la frontière de l'informatique et de la linguistique, et qui s'inscrit dans le domaine de l'intelligence artificielle aussi appelé avec NLP (*Natural language processing*).

Il représente l'ensemble des recherches et développements visant à modéliser et reproduire, à l'aide de machines, la capacité humaine à produire et à comprendre des énoncés linguistiques dans des buts de communication.

Le NLP peut être divisé en 2 grandes parties, le **NLU** (*Natural Language Understanding*) et le **NLG** (*Natural Language Generation*).

Le premier est toute la partie « compréhension » du texte, prendre un texte en entrée et pouvoir en ressortir des données.

Le second, est générer du texte à partir de données, pouvoir construire des phrases cohérentes de manière automatique.

10 Traitement de texte

10.1 Objectif du traitement automatique du langage nature

Il existe deux sources principales de motivation à l'étude du TALN (traitement automatique du langage naturel): d'une part la volonté de modéliser une compétence fascinante (le langage), afin de tester des hypothèses sur les mécanismes de la communication humaine, ou plus généralement sur la nature de la cognition humaine ; d'autre part le besoin de disposer d'applications capables de traiter efficacement les monceaux d'informations « naturelles » (documents écrits ou sonores) aujourd'hui disponibles sous forme électronique (mails, pages HTML, documents hypermédias, etc.).

Dans le cadre de notre travail, nous avons effectué un prétraitement de texteet une représentation de nos données textuelles, afin de les rendre utilisables comme entrée dans notre réseau de neurones.

Nous décrivons dans ce qui suit les méthodes utilisé dans notre cas.

10.2 Tokénisation

La tokenisation est une technique NLP, qui est une étape pour décrire la division des paragraphes en phrases, ou phrases en mots individuels (divise des chaînes de texte plus longues en petits morceaux, ou jetons). Un traitement ultérieur est généralement effectué après qu'une partie du texte été symbolisé de manière appropriée. La tokenisation est également appelée segmentation de texte ou analyse lexicale.

Les phrases peuvent être divisées en mots individuels et ponctués par un mêmeprocessus. Le plus souvent, cette division entre les espaces blancs.

10.3 Natural Probabilistic language modeling

De façon générale, on appelle langage un ensemble de conventions permettant de conceptualiser la pensée et d'échanger avec d'autres interlocuteurs à l'aide d'un support comme la parole, l'écriture, *etc.*

Construire un modèle de langue statistique consiste à estimer une distribution de probabilité sur des séquences de mots dans une langue particulière. Il permet ainsi d'assigner une probabilité à une séquence de mots.Ces modèles de langue sont statistiques c.-à-d. qu'ils ne sont pas définis par des règles formelles mais sont appris sur base d'un grand nombre de textes appelés aussi corpus.[21]

10.4 Les modèles n-grammes

Ce sont les premiers modèles de langue, ils sont basés sur les chaînes de Markov. La probabilité d'une séquence de mots étant le produit des probabilités de chaque mot sachant les mots précédents, on introduit une simplification en prenant uniquement en compte les n mots précédents. Autrement dit, le modèle permet de prédire les mots suivants étant donné les n mots précédents. [22]

10.4.1 Fonctionnement du modèle n-gramme

Considérons la séquence de mots « m1 m2 m3 m4 », la probabilité d'obtenir cette séquence dans un modèle digramme sera calculée de la façon suivante :

$$p(m_1, m_2, m_3, m_4) = p(m_1) + p(m_2/m_1)\, p(m_3/m_2)\, p(m_4/m_3)$$

Tel que :$p(m_1/m_2) = N(m_1, m_2)/N(m_1)$

10.4.2 Limitations du modèle n-gramme

Le modèle n-gramme ne permet pas de capter les dépendances entre les mots qui sont éloignés dans la séquence.

Lors du traitement d'un texte, un n-gramme qui ne se trouve pas dans le modèle peut apparaître, la probabilité de ce n-gramme est alors de 0. Cependant, il existe des méthodes telles que la méthode de lissage Good-Turing pour corriger ce problème.

10.5 Les modèles de langue basés sur les réseaux de neurones (*Neural Network Language Modeling*)

Comme pour les modèles n-grammes le but du modèle de réseaux de neurones est d'estimer la probabilité d'une séquence de mots.

Ils sont construits sur base des réseaux de neurones récurrents (RNN) et plus particulièrement les RNN à mémoire court terme étendue (LSTM). Ces réseaux sont spécialement utilisés pour le traitement de séquences car ils gardent la mémoire du traitement des mots vus précédemment, on intègre de cette façon le mot à prédire dans son contexte.

10.5.1 Fonctionnement

Dans un réseau de neurones, les séquences de mots sont représentées en vecteurs de mots denses (*Word vector en anglais)* avant d'être injectées en entrée du réseau. Chaque mot est ainsi projeté dans un espace de dimension (*word embedding en anglais*) dans lequel

il est représenté par un vecteur tel que deux mots au concept similaire seront représentés par des vecteurs proches l'un de l'autre dans cet espace. Ces représentations vectorielles peuvent être apprises par le modèle en même temps que la fonction de probabilité ; ils constituent alors des paramètres additionnels du modèle.

10.5.2 Limitations

Les modèles basés sur les réseaux de neurones ont cependant quelques limitations. La première est qu'ils requièrent un très large corpus, le deuxième est le temps de traitement très long. Les informations étant traitées de manière séquentielle dans le réseau, les calculs ne peuvent être parallélisés.

En outre, à cause de ce traitement séquentiel des mots, l'algorithme a plus d'information sur le contexte en fin de séquence car ayant vu plus de mots, qu'en début de séquence.

Et enfin l'historique : information sur le traitement des mots précédents dans la séquence, proche a plus de poids dans le traitement d'un mot que l'historique lointain.

10.6 One hot encoder

Il s'agit de diviser la colonne qui contient des données catégorielles numériques en plusieurs colonnes en fonction du nombre de catégories présentes dans cette colonne. Chaque colonne contient «0 » ou «1 » correspondant à la colonne dans laquelle elle a été placée.

Elle retourne en sortie une liste des entiers dans [1, n]. Chaque entier code unmot.

11 Conclusion

Dans ce chapitre, nous avons présenté les grandes lignes de l'apprentissage profond. Nous avons exploré son application sur le traitement du langage et décortiqué son fonctionnement ainsi que les algorithmes utilisés. Enfin, nous avons exposé d'une part quelques limitations rencontrés en utilisant ces algorithmes.

Dans le chapitre suivant, nous présentons l'application de l'apprentissage automatique dans notre projet à savoir nos modèles de réseaux de neurones artificiels. En plus, nous analysons leurs architectures ainsi que toutes les configurations ajoutées pour leurs bons fonctionnements.

Chapitre 3 : Architecture proposée pour la qualification de leads

Chapitre 3 : Architecture proposée pour la qualification de leads

1 Introduction

Le point fort des stratégies marketing d'aujourd'hui est l'information, qui est devenue très accessible et volumineuse. La bonne utilisation des informations concernant les clients ou les prospects garantit l'acquisition et la fidélité de ces derniers.

Grâce aux nouvelles technologies notamment l'AA, cette tâche est devenue plus facile.

Ce travail s'intéresse à l'automatisation de la gestion des informations concernant les clients en utilisant cette technologie.

Après avoir défini les concepts théoriques de base sur l'intelligence artificielle, l'apprentissage automatique et le traitement automatique du langage naturel, nous passons dans ce chapitre à la deuxième partie de notre travail qui consiste à présenter progressivement notre problématique ainsi que la conception et au final l'architecture des modèles proposées.

2 Analyse de la problématique : Qualification des leads

Dans La nouvelle ère du marketing, un terme devenu récurrent est celui du Lead, il s'agit d'un prospect qui a laissé ses coordonnées, ou bien d'un client potentiel pour l'entreprise. La bonne gestion de celui-ci permet de bien formuler la stratégie marketing.

D'une manière générale, le lead management se déroule en 4 étapes :

- La génération de leads : désigne l'ensemble des actions permettant de créer des contacts commerciaux plus ou moins qualifiés désignés le plus souvent sous le terme de leads.
- La qualification des leads : une fois les leads générés, il va falloir les travailler pour qu'ils se transforment en leads qualifiés. La qualification se distingue par une interaction faite par le prospect.
- Le lead nurturing : cette technique consiste à faire progresser le lead dans son parcours d'achat jusqu'à qu'il soit assez mature pour acheter.
- La conversion des leads : Cette étape consiste en la transformation des leads en clients acquis, c'est-à-dire que le prospect qualifié a effectué un achat.

Dans notre cas, on s'intéresse à l'automatisation de l'étape la plus cruciale ; la qualification des leads, qui s'agit de l'identification des véritables intéressés au milieu d'une foule de curieux, permettant ainsi le gain en coûts et en efficacité.

Pour cela nous proposons une solution se basant sur l'apprentissage automatique.

3 Etat de l'art : Apprentissage automatique et qualification des leads

Auparavant, la qualification de leads reposait sur l'achat de listes de noms et le démarchage téléphonique de personnes par des commerciaux, mais avec le développement technologique et la simplification de l'accès à l'information, celle-ci s'est automatisée et aujourd'hui gérée par des algorithmes d'AA.

En effet, en utilisant ce dernier, un modèle de prédiction précis est créé, il permettra de marquer des prospects et de cibler les offres sur les bons clients là où elles seront les plus efficaces. Il s'agit d'un score prédictif des leads.

La course aux brevets d'IA et d'apprentissage automatique devient de plus en plus compétitive. Selon le groupe Boston Consulting, 85% des dirigeants pensent que l'IA permettra à leurs entreprises d'obtenir ou de maintenir un avantage concurrentiel. [1]

Nous avons sélectionné la solution Microsoft parmi une offre diversifiée de logiciels qui traitent de la qualification des leads, car cette dernière est pionnière parmi la compétition[2] et que leur implémentation a été conçue à partir de présupposé en adéquation avec notre approche méthodologique et théorique.

La qualification des leads, telle qu'elle est implémentée dans BEAM de Microsoft, repose sur le principe de détection de contextes et d'intention, un score est accordé au lead en se basant sur ces deux critères. La détection de contexte consiste à déterminer le sujet du message analysé, et ainsi sa pertinence vis-à-vis de la chaîne de vente. La détection d'intention quant à elle, consiste à une analyse syntaxique du texte et de détecter des termes et phrases clés, clarifier au préalable, pour déterminer l'intention qu'a le lead à faire une transaction. [23]

L'architecture de Beam se présente ainsi : (voir la figure)

1 The Boston Consulting Group, "Is Your Business Ready for Artificial Intelligence, Sept. 2017.
2 Selon : salesforces.com

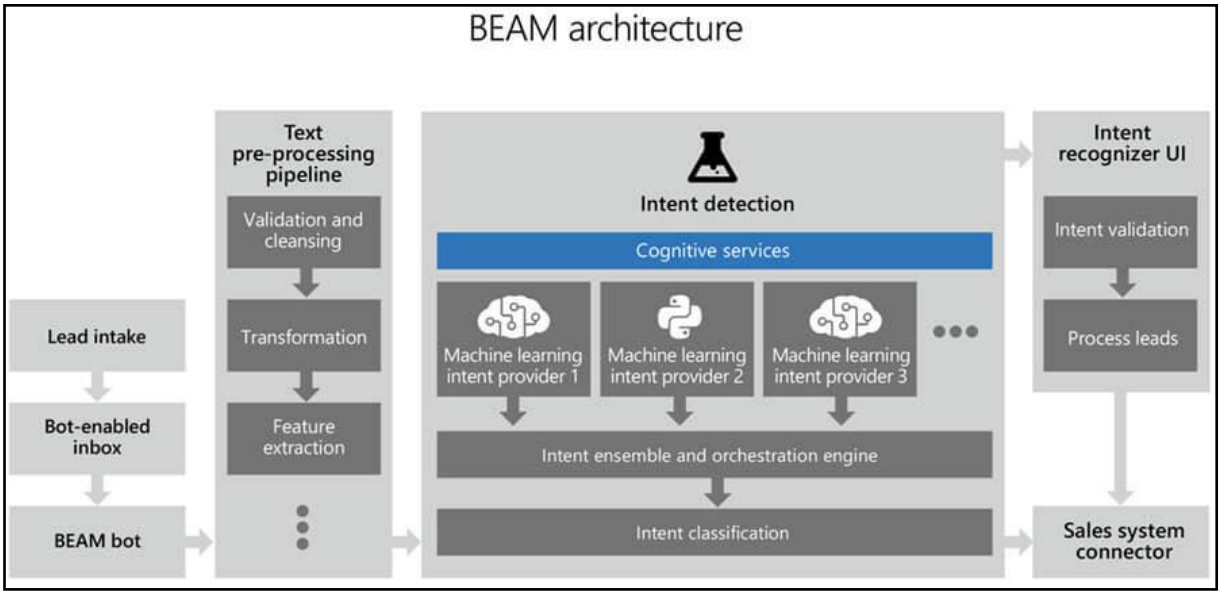

Figure 15 : Le modèle d'architecture de BEAM

D'abord le prétraitement, cette partie consiste à convertir les mails en chaîne de caractères puis les nettoyer en utilisant le Bing Spell Check de Microsoft Cognitives Services.[3] [23]

Ensuite, les données prétraitées seront envoyées en input a un modèle de traitement du langage naturel appelé LUIS[4], qui permettra d'effectuer la détection de contexte et d'intention. [23]

Une fois les données traitées par LUIS, elles seront insérées dans 3 modèles reposant sur des concepts différents : La classification naïve de bayésienne (Multinomial NB), les machines à vecteurs de support (SVM.SVC) et les classificateurs linéaires avec l'algorithme de descente stochastique (SGD Classifier). Chacun des modèles donne des résultats qui sont pesés équitablement, et contribuent à la détermination de l'intention. [23]

Afin de bien évaluer les différents résultats, un méta-algorithme est utilisé pour déterminer les résultats supportés par le plus de modèles, dans ce cas la méthode utilisée est l'Ensemble Majority_MaxOrder [5]. [23]

3 L'API Bing Spell Check permet d'effectuer une grammaire contextuelle et une vérification orthographique du texte.

4 Language Understanding (LUIS) est un service d'IA conversationnel basé sur le cloud qui applique une intelligence d'apprentissage automatique personnalisée au texte conversationnel en langage naturel d'un utilisateur pour prédire la signification globale et extraire des informations pertinentes et détaillées.

Dans le but de toujours améliorer la qualité des prédictions, les modèles utilisés sont entraînés mensuellement en utilisant les résultats du mois précédent, ceci permet une mise à jour constante. [23]

Dans notre travail, nous apportons une autre vision à ce processus en utilisant un algorithme d'apprentissage supervisé basé sur les réseaux de neurones profonds.

4 Analyse des choix d'apprentissage

Avant d'avoir réalisé les spécifications fonctionnelles de notre application, différents besoins ont orienté nos choix lors de la conception de notre modèle (présenté dans le titre suivant).

Ces derniers ont été conçus en se basant sur un apprentissage supervisé. Par la suite, nous étions confrontés à choisir l'algorithme adéquat pour notre solution pour cela nous avons choisi d'utiliser les réseaux de neurones artificiels (RNA).

Les avantages de l'utilisation des RNA sont leur nature non paramétrique et leur adaptation à des données de différents types. Cependant, une application réussie des RNA dépend de la détermination de sa structure optimale, et de la phase de prétraitement des données.

La construction d'un RNA passe par la détermination de son architecture (nombre de couches et nombre de neurones artificiels dans chaque couche) et le choix des fonctions d'activation associées aux couches.

Dans ce travail nous avons réalisé et tester plusieurs architectures, parmi elles (la plus prometteuse) celleque nous présentons dans ce chapitre.

Nous avons choisi une architecture hybride d'un modèle fonctionnel qui inclut deux couches d'entrées :

5Majority_MaxOrder prend en compte toutes les entrées du modèle et choisit l'intention qui a reçu le vote majoritaire de tous les modèles. En cas de conflit (deux intentions ont un nombre égal de voix), celle avec le score de probabilité le plus élevé est choisie. Si plusieurs ont le même score de probabilité, l'ordre des modèles détermine lequel est choisi.

- La première couche reçoit les données catégoriques qui doivent être transformés en données numériques pour cela nous avons utiliséle One Hot Encoding expliqué précédemment. (Voir figure 16)

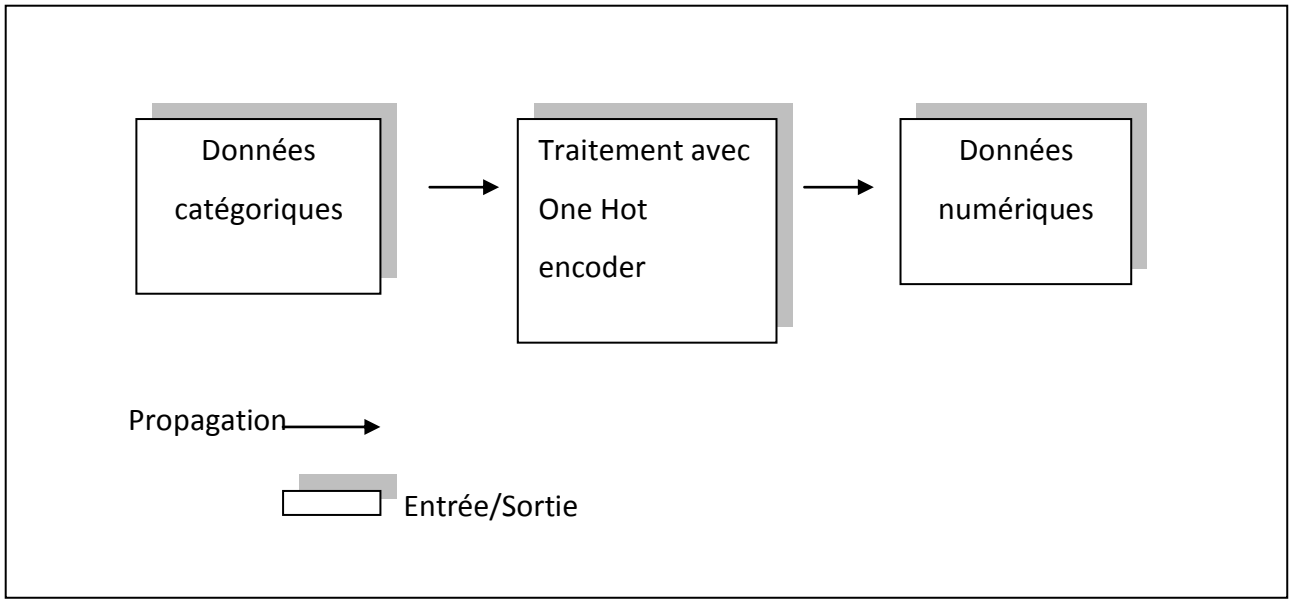

Figure 16 : Architecture de traitement des données catégoriques

- La deuxième couche qui est une couche NNLM (expliqué précédemment - chapitre 2) reçoit les chaînes de caractère en variables d'entrées et les traite.

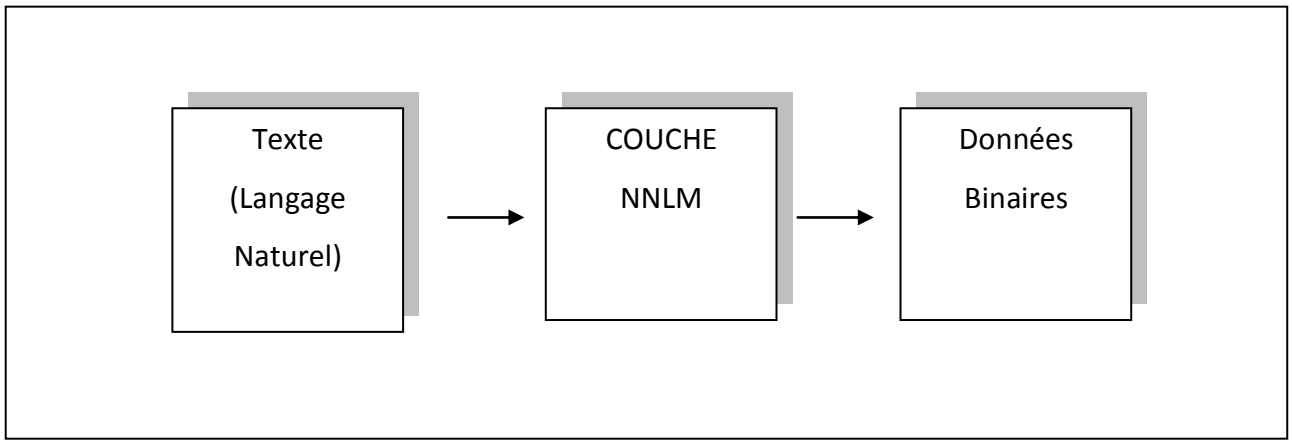

Figure 17 : Architecture de traitement des données avec NNLM

Notre choix s'est porté sur cette approche, vu son efficacité dans notre cas pour le traitement du langage naturel, sa rapidité et ses résultats prometteurs par rapport aux

alternatives que nous avons expérimentés notamment Glove [6] et one hot encoder (Expliqué précédemment dans le chapitre 2) qui se sont avérés comparativement lent et leurs résultats insatisfaisants.

Ces deux couches d'entrées reçoivent les données et les transforment en appliquant la fonction d'activation relu étant la fonction d'activation non linéaire la plus utilisée en apprentissage supervisé et réseaux de neurones multicouches

Ensuite, la sortie de ces deux dernières est l'entrée d'un troisième réseau neuronal qui va concaténer les connaissances acquises précédemment. (Voir la figure 18)

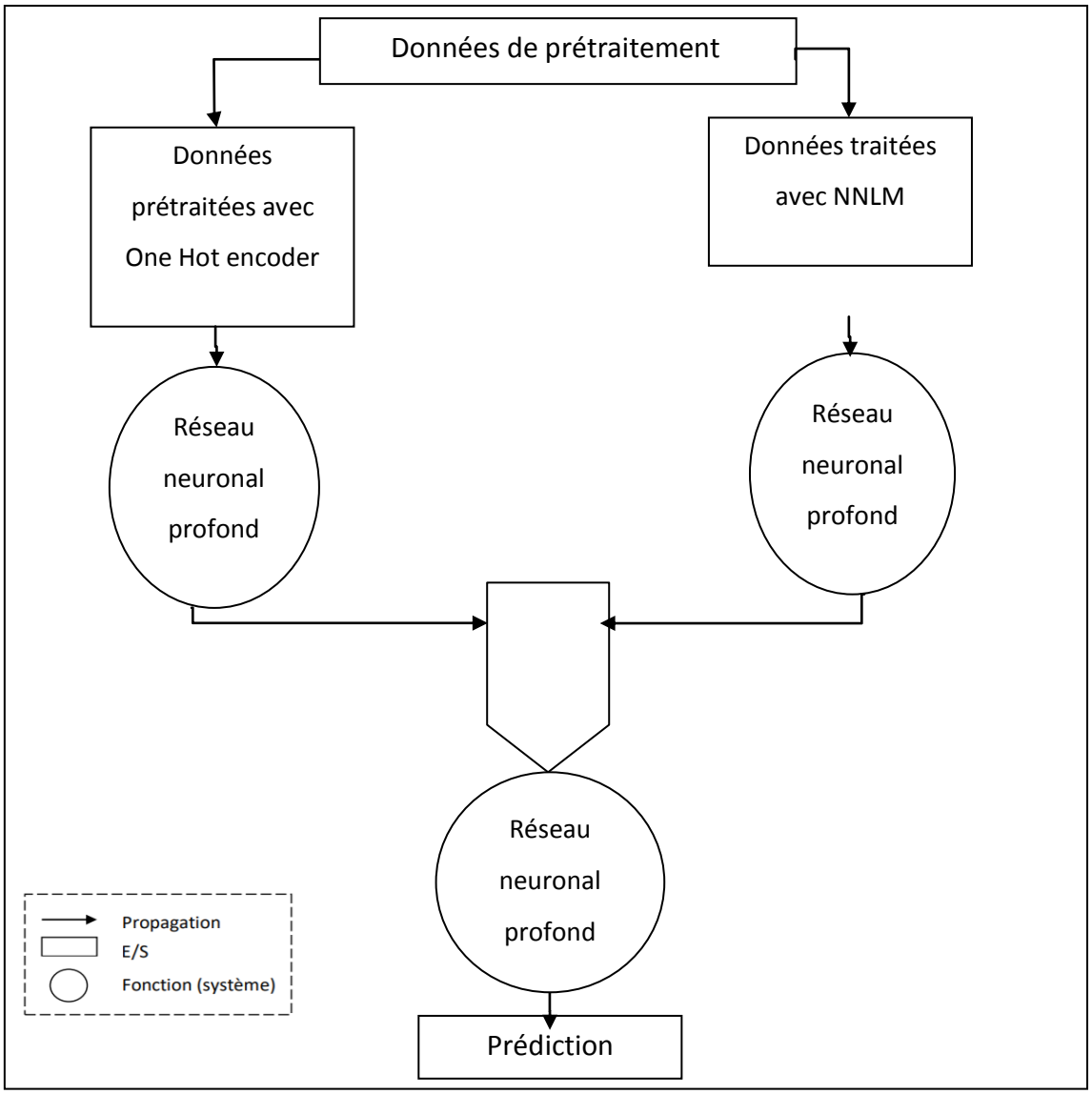

Figure 18 : Diagramme de flux de données

Ce réseau possède 3 couches cachées dont le nombre de neurones est 64, 32 et 20 respectivement. Celle-ci est la combinaison standard dans le domaine professionnel car le

6 GloVe est un algorithme d'apprentissage non supervisé permettant d'obtenir des représentations vectorielles des mots.

fait d'augmenter le nombre de neurones ralenti l'entraînement en outre ce choix est inspiré de la formule de calcul de nombre de neurones dans l'équation 1.1.

À noter que nous sommes restreints à utiliser un nombre moyen de couches cachées (Maximum trois couches successives), qu'elles soient convolutifs, complètement connectées ou résiduelles pour des contraintes matérielles de l'environnement d'entraînement de nos modèles (Voir chapitre 4 environnement de développement).

Au final le choix de la fonction d'activation globale s'est porté sur la fonction sigmoïdale. Elle est la fonction idéale pour notre cas, car elle fournit un résultat appartenant à l'intervalle0 et 1 et par conséquent est utilisée pour les modèles de prédiction de probabilité en tant que sortie.

5 Modèle proposé

Nos propositions ont été faites en étroite relation avec l'implémentation. Après avoir entraîné et testé plusieurs modèles, nous avons sélectionné et proposé ce modèle qui semble être le meilleur de par ses statistiques d'entraînement et ses performances lors des tests(voir chapitre 4 : Résultats obtenus).

Au départ nous avons essayé de travailler avec un modèle séquentiel de *Keras*. Ce modèle doit savoir le type d'entrée à attendre mais comme dans notre cas on avait des formes d'entrées différentes cela n'a pas pu fonctionner.

La différence majeure entre tous les modèles réside dans le nombre des différentes couches utilisées ainsi que le nombre de neurones de chaque couche. Il existe aussi une différence dans les valeurs d'autres hyper-paramètres comme le nombre de batch size, le taux d'apprentissage, la fonction de cout, la fonction d'activation (voir chapitre 4 : Entrainement).

Concernant le nombre de neurones des couches cachées choisi nous avons utilisé une formule appliquée parfois dans le domaine professionnel et souvent dans les petits projets d'apprentissage automatique comme le nôtre. L'équation nous a été proposée par le Dr Andrew NG (chercheur en AA et professeur à l'université de Stanford) et elle est définit comme suit : $Nnc = \alpha\sqrt{Nne * Nns}$ (équation 1.1) [24] [25]

Nnc, Nne, Nns respectivement le nombre de neurones cachés, d'entré, de sortie de la couche courante. α représente un nombre réel arbitraire tel que $1{,}5 \ll \alpha \ll 10$.

5.1 Modèle RNA

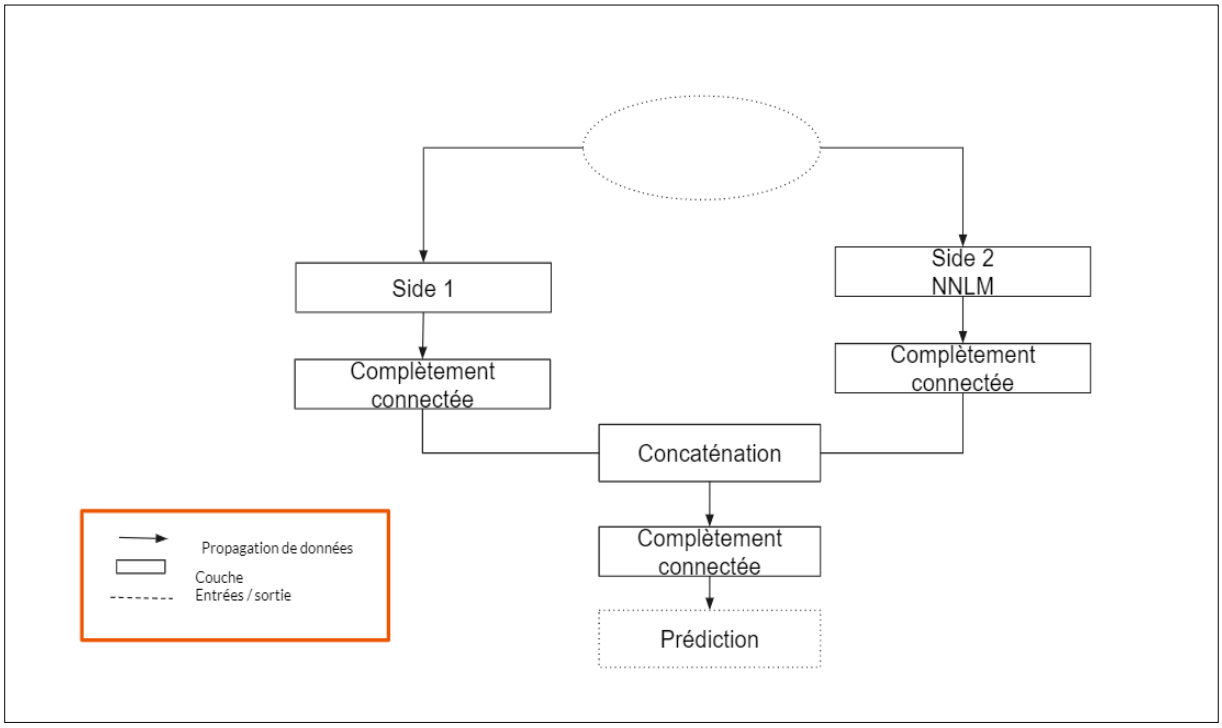

Figure 19 : Architecture RNA

Initialement nous avons un ensemble de données composées de différents types dans un tensor, après cela nous avons décidé de décomposer ce tensor en deux types avec des fonctions lambda.

Pour les données de type catégorique nous avons conçu pour leur apprentissage un réseau neuronal profond.

D'autre part les données de type langage naturel, nous les avons propagées dans une couche spécifique NNLM, puis propagées dans un autre réseau neuronal profond.

Les sortie de ces dernières couches ont été directement liéesafin de constituer une base de connaissance commune entre les différentes architectures, nous avons décidé de concaténer leurs sorties et les propagées dans un dernier réseau neuronal profond constituer de 3 couches cachées se terminant avec une couche de sortie ayant comme fonction d'activation la fonction sigmoïde choisi de par ces capacités à fournir des résultats variant entre 0 et 1 qui correspond au pourcentage de validité des prédictions réalisées à savoir le pourcentage qu'un lead soit intéressé par un produit.

6 Conclusion

Dans ce chapitre, nous avons définit la problématique et analysé cette dernière. Par la suite, nous avons décrit les architectures que nous avons réalisées tout en démontrant leurs objectifs.

Le prochain chapitre décrit notre implémentation, les expérimentations réalisées sur le modèle précédent ainsi que les résultats obtenus.

Chapitre 3 : Etude de l’architecture proposé

Chapitre 4 : Implémentations et Expérimentations

1 Introduction

Nous avons présenté dans les chapitres précédents les différentes notions de l'apprentissage automatique ainsi que le modèle proposé et son architecture,nous exposons dans cette partie les résultats obtenus et leur implémentation.

Dans ce chapitre, nous abordons d'abord la description des outils logiciels et les différentes bibliothèquesutilisées, ensuite nous décrivons la procédure entreprise pour créer le jeu de donnée efficace pour l'entrainement, évaluation et test du modèle. Enfin, nous exposons les résultats obtenus par l'architecture proposée.

2 Outils et environnement de développement

2.1 Logiciel

2.1.1 Pycharm

Pycharm [1] est un environnement de développement intégré utilisé pour programmer en python. C'est un logiciel multiplateforme qui fonctionne sous Windows, Mac OS X et Linux.

Il permet l'analyse de code et contient un débogueur graphique. Il permet également la gestion des tests unitaires, l'intégration de logiciel de gestion de versions, et supporte le développement web avec Django qui est un Framework Web Python.

2.2 Bibliothèques logicielles

2.2.1 Tensorflow

TensorFlow [2] est une plateforme de bout en bout qui facilite la création et le déploiement de modèles de machine learning.

Elle propose un écosystème complet et flexible d'outils, de bibliothèques et de ressources communautaires permettant aux chercheurs d'avancer dans le domaine du machine learning, et aux développeurs de créer et de déployer facilement des applications qui exploitent cette technologie.

1https://fr.wikipedia.org/wiki/PyCharm
2 https://www.tensorflow.org/

2.2.2 Keras

Keras[32]est une API de haut niveau de TensorFlow permettant de créer et d'entraîner des modèles de Deep learning.

Elle est utilisée dans le cadre du prototypage rapide, de la recherche de pointe et du passage en production. Elle présente trois avantages majeurs : convivialité, Modularité et facilité de composition, Facilité d'extension

2.2.3 Pandas

Pandas [4] est un outil d'analyse et de manipulation de données open source rapide, puissant, flexible et facile à utiliser, construit sur le langage de programmation Python.

2.2.4 Numpy

NumPy [5] a été créé en 2005 par Travis Oliphant. C'est un projet open source et vous pouvez l'utiliser librement.

NumPy est une bibliothèque python utilisée pour travailler avec des tableaux. Il a également des fonctions pour travailler dans le domaine de l'algèbre linéaire, de la transformée de Fourier et des matrices.

2.2.5 Tensorboard

TensorBoard [7] fournit les solutions de visualisation et les outils nécessaires aux tests de machine learning. Il est utilisé pour visualiser les graphiques Tensorflow, tracer des métriques quantitatives sur leur exécution et afficher des données supplémentaires, telles que des images qui les traversent.

2.3 Configuration matérielle utilisée

La configuration du matériel utilisé dans notre implémentation est comme suivie :

- Un PC portable HPElitebook ;
- Processeur i5-8256U CPU 1,6 1,8 GHZ;
- Carte graphique Intel UHD Graphics ;
- RAM de taille 8 GO ;

3 https://keras.io/
4 https://pandas.pydata.org/)
5 https://www.w3schools.com
7 https://www.tensorflow.org

- Disque dur SSD de taille 256 GO ;
- Système d'exploitation Windows 10 Edition Intégrale.

3 Jeu de données

Pour constituer un jeu de données qualitatif, nous avons récoltédes informations à partir d'une plateforme marketing préexistante dont nous gardons l'anonymat.

Cependant, il ne suffit pas d'avoir beaucoup de données. Il est important aussi qu'elles soient de bonne qualité, sans quoi les systèmes d'IA sont des échecs. Pour que les modèles statistiques soient correctement entrainés et qu'ils fournissent les résultats escomptés, les données utilisées doivent être propres, précises, complètes et bien labélisées. La préparation de ces données est donc une étape cruciale.

Dans cet intérêt, nous procédons premièrement au nettoyage des données. En premier, nous utilisons Pandas pour ouvrir la collection de données qui était sous forme CSV[3]. (Voir figure 20)

```
def openDataSet(self):
    """ouverture de la collection avec pandas"""
    print("Ouverture de " + self.collecionPath)
    self.data = pd.read_csv(self.collecionPath)
```

Figure 20 : Ouverture de la collection de données en utilisant pandas

Ensuite, nous utilisons la méthode *Fillna* de *Pandas* afin de remplacer toutes les valeurs NAN (non numérique) par 0. (Voir figure 21)

```
self.data = self.data.fillna(self.data['lead_type'].value_counts().index[0])
self.data = self.data.fillna(self.data['1st_category'].value_counts().index[0])
self.data = self.data.fillna(self.data['categories'].value_counts().index[0])
```

Figure 21 : Remplacement des données Nan

La collection de données contient les objets (*subject*) et contenus (*body*) des emails envoyés à des Leads lors de différentes campagnes de marketing, ainsi que les champs *open* et

3 CSV (Comma-separated values) est un format texte ouvert représentant des données tabulaires sous forme de valeurs séparées par des virgules.

*click*qui indiquent si le Lead a ouvert l'email et / ou a cliqué sur le lien de l'offre dans l'email. Nous avons aussi différentes informations sur les 1361812 leads qui sont :

- User id : Identifiant de l'utilisateur de type numérique.
- Replied : Valeur prenant 1 si l'utilisateur arépondu ou 0 sinon.
- Tracked : Valeur prenant 1 si l'utilisateur a ouvert l'email ou 0 sinon.
- Company : L'entreprise du lead.
- Email : Email du lead.
- Phone : Numéro de téléphone du lead.
- 1st-category : Catégorie du produit principale.
- Catégories : Autres types de catégories proposés par l'entreprise.
- Keyword : Mots clés.
- Mailing address:Address du lead.
- Job_title : Profession du lead.
- Lead_type: Le type du lead:Hot Lead, Cold Lead, Warm Lead.

Dans le cadre de notre projet, avant d'alimenter le modèle avec ces données, ces dernières sont représentées sous forme numérique. Pour cela, nousdevisons ces champs en deux types de données en utilisant des fonctions Lambda (Voir les figures 22, 23).

```
side1 = Lambda(lambda x: x[len(nnlm.trainDataX):])(inp)
```

Figure 22 : Données catégoriques

```
side2 = Lambda(lambda x: x[: len(nnlm.trainDataX)])(inp)
```

Figure 23 : Données en entrée pour NNLM

- Les données traitées comme étant du langage naturel et qui sont introduites comme entrée à la couche NNLM fournit par le modèle Hub de *Keras* : il s'agit des champs objets et emails envoyés.
- Les données catégoriques : il s'agit des champs *lead type*, *1st_category*, *catégories* et *company*.

Enfin les champs *opened*, *clicked* que nous concaténons et qui constituent la sortie Y lors de l'entrainement du modèle.

Pour le prétraitement des données catégoriques nous construisons une méthode nommée FeaturesDataPreprocessing. Cette méthode prend en entrée la collection de données puis, Elle traite le champ *1st category* et *lead type* ainsi que *company* en utilisant *One Hot encoder* (voir la figure 24).

```
def token(self,text):
    test2d = text.values.reshape(1,-1)
    ohe = OneHotEncoder(sparse=False)
    return ohe.fit_transform(test2d)
```

```
    st_cat = self.token(self.data['1st_category'])
    lead = self.token(self.data['lead_type'])
```

Figure 24 : traitement des données en utilisant one hot encoder

Cependant, pour le champ catégories comme chaque colonne est constituée d'une suite de mots séparés par une virgule, des points virgules. Ct, nous utilisons la méthode *Tokenize* de *Keras* car elle fait la séparation des mots, supprime automatiquement la ponctuation et elle donne un résultat moins volumineux par rapport à one hot encoder. (Voir figure 25)

```
def featuresDataPreprocessing(self):
    """pretraitement des entreées"""
    for element in self.data['categories']:
        vocab = element.split(',')
        tokenizer = Tokenizer(num_words=1361812)
        tokenizer.fit_on_texts(vocab)
        categories = tokenizer.texts_to_matrix(vocab)
```

Figure 25 : traitement du champs categories

Le résultat de ces traitements a été redimensionné avant d'être concaténé pour construire une seule entrée des données catégoriques. (Voir la figure 26)

```
company =np.asarray(company).reshape(1361812,1)
lead = np.asarray(lead).reshape(1361812,1)
st_cat = np.asarray(st_cat).reshape(1361812,1)
categories = np.asarray(categories).reshape(1361812,1)
x = np.concatenate((lead,st_cat,categories, company),axis=0)
```

Figure 26 : Concaténation des données catégoriques

Nous avons ainsi, divisé notre jeu de données en trois parties :

• **Données d'entraînement** : qui constituent 80% de notre jeu de données à savoir 1089449 leads. Ces données sont utilisées pour l'entrainement de nos modèles.

• **Données de validation** : qui constituent 10% de notre jeu de données à savoir 136182 leads. Ces données sont utilisées lors de l'apprentissage afin de valider nos modèles et pour éviter le sur-apprentissage.

• **Données de test** : qui constituent les 10% restantes de notre jeu de données à savoir 136182 leads. Ces données ne sont pas utilisées lors de l'entrainement, mais uniquement lors des tests, pour évaluer nos modèles.

A noter aussi que pour diminuer le temps de chargement des données lors de l'entrainement des différents modèles, nous enregistrons les données dans des fichiers numpy. Pour cela, nous utilisons deux méthodes *SaveData* et *LoadData* qui font l'enregistrement des données dans des fichiers numpy et le chargement des données de ces fichiers respectivement. (voir figure27)

```
def saveFeaturesData(self):
    """Sauvegarde des features dans des fichiers numpy"""
    np.save('trainDataX',self.trainDataX)
    np.save('testDataX',self.testDataX)
    np.save('trainDataY',self.trainDataY)
    np.save('testDataY',self.testDataY)
def loadFeaturesData(self):
    """Chargement des Features"""
    self.trainDataX=np.load('trainDataX.npy')
    self.trainDataY=np.load('trainDataY.npy')
    self.testDataX=np.load('testDataX.npy')
    self.testDataY=np.load('testDataY.npy')
```

Figure 27 : Les Méthodes LoadData et SaveData

4 Entrainement

Afin de former la bonne combinaison des hyper-paramètres qui nous conduisent à un modèle optimal, nous avons appliqué quelques approches et heuristiques utilisées par les ingénieurs de l'IA dans le domaine professionnel. Ces approches sont sous forme d'un processus dont les démarches sont les suivantes :

- Augmentation du nombre de données : Fournir plus de données pour le modèle afin d'éviter le sous apprentissage et permettre au modèle d'apprendre sur plus d'exemples et mieux comprendre le problème.

- La modification du taux d'apprentissage (Learning rate) : Il s'agit d'essayer les variantes suivantes :
  - Essayer avec une très grande valeur.
  - Essayer avec une valeur très réduite.
  - Essayer de réduire sa valeur au cours de l'entrainement (voir la figure ci-dessous)

```
reduc = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=2)
```

Figure 29 :Implémentation de ReduceLROnPlateau

  - Rechercher les valeurs communes utilisées par la communauté de l'apprentissage automatique qui se situe dans l'intervalle [$1*10^{-7}$, 1].

- Définir la valeur du prochain taux d'apprentissage selon le résultat du modèle passé,suivant la figure ci-dessous :

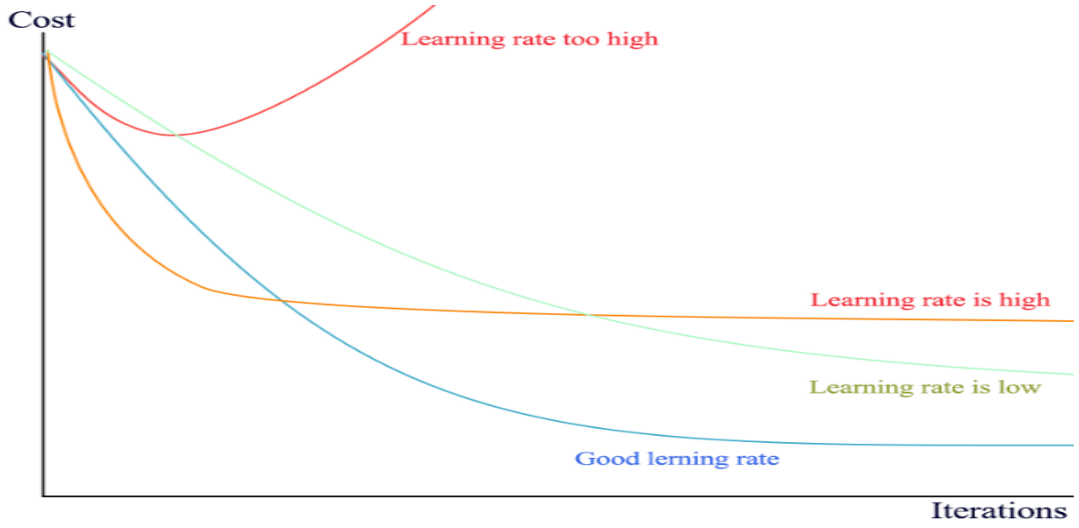

Figure 30 : Le choix du taux d'apprentissage

- **Ajustement de la taille des lots (batch size) :** La valeur du batch size définit le nombre d'échantillon qui seront propagés à travers le réseau.

  Une valeur réduite peut accélérer l'entrainement mais risque d'avoir des oscillations dans le graphe d'erreur. D'autre part, si la valeur est très grande l'entrainement risque d'être long. Nous avons fixé la valeur de 1000 dans le modèle entrainé.

- Eviter de tomber dans le sur-apprentissage en utilisant :
  - **Dropout** : c'est une technique qui est destinée à empêcher le sur-ajustement sur les données d'entrainement en abandonnant des unités dans un réseau de neurones avec une probabilité p>0. Cela force le modèle à éviter de trop s'appuyer sur un ensemble particulier de *features* [26], nous avonstesté des variantes avec un pourcentage de dropout variant entre 20 à 50%.
  - **Arrêt précoce** : Il s'agit d'une méthode de régularisation qu'on a utilisée. Dans notre cas nous avons choisi la valeur 40. (Voir figure ci-dessous)

```
earlystop =keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', patience=40, verbose=1)
```

Figure 31 : Implémentation de l'arrêt précoce

- **Augmentation du nombre d'époque** : l'époque est une itération complète sur les données d'entrainement dans un réseau de neurones, il est conseillé de choisir un nombre très grand dans le cas de l'utilisation d'un arrêt précoce vu qu'on l'atteindra rarement.

Les résultats de l'entrainement et de validation seront présentés dans le 6 eme titre.

L'entrainement du modèle a duré approximativement 48h, c'est pourquoi nous avions sauvegardé le modèle obtenu lors de cette phase d'entrainement dans un fichier h5 en utilisant la bibliothèque h5py de python afin qu'il puisse être réutilisé ultérieurement dans la phase de test.

5 Mesure d'évaluation

Afin de sélectionner des modèles nous pouvons faire recours à plusieurs types de critères d'évaluation, indiquant la performance d'une fonction de prédiction. Ces différentes critères appelés mesures d'évaluation permettent de mieux comparer les modèles entre eux afin de répondre aux objectifs qui lui ont été fixés [27].

Pour ce faire, nous avons utilisé plusieurs mesures d'évaluation : la précision et la valeur de perte.

5.1 Précision (Accuracy)

La précision [27] est le nombre de points de données correctement prédits parmi tous les points de données, c'est la plus utilisée pour juger et évaluer la performance d'un modèle.

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions faites}}$$

5.2 Perte (Loss)

La perte [28] est un nombre qui indique la médiocrité de la prévision du modèle pour un exemple donné. Si la prédiction du modèle est parfaite, la perte est nulle.

5.2.1 Métriques d'évaluation :

Il existe plusieurs métriques qu'on utilise pour la valeur de perte :

- Le carré moyen des erreurs (MSE pour Mean *Square Error*en Anglais)[4] : c'est la moyenne arithmétique des carrés des écarts entre prévisions du modèle et observations.
- L'erreur quadratique moyenne (RMSE)[5] : la racine carrée du précédent.
- L'erreur absolue moyenne (EAM ou MAE pour Mean Absolute Error en Anglais)[6] : moyenne arithmétique des valeurs absolues des écarts.

Nous avons la fonction « Mean Square Error » (MSE) comme fonction de coût car c'est la plus utilisée [24]. Elle permet de calculer l'erreur quadratique entre les données prédites et celles attendues. En plus, elle permet de calculer le gradient plus facilement que les autres métriques [24] [25].

6 Résultats obtenus

Nous avons entrainé notre modèle décrit dans la section précédente sur l'ensemble des données d'entrainement,en mêmetemps que cette étape s'effectue l'étape de validation qui utilise l'ensemble de données de validation. Nous testons le modèle en utilisant l'ensemble des données de test.

6.1 Résultats obtenus pour le modèle lors de l'entrainement et de la validation

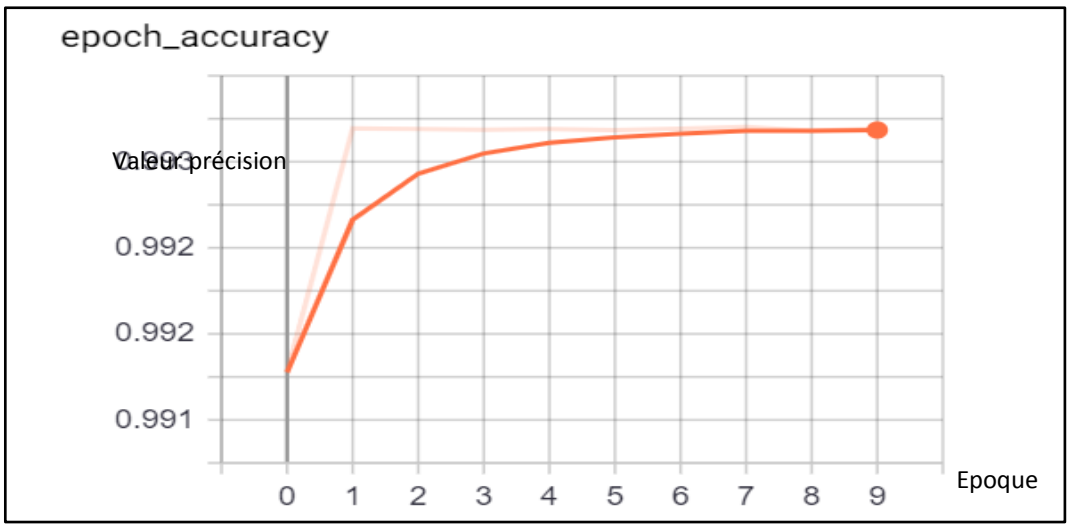

Figure 32 : Evolution de la précision du modèle RNA sur les données d'entrainement

4 http://www.jybaudot.fr/Stats/indicecarts.html
5 http://www.jybaudot.fr/Stats/indicecarts.html
6 http://www.jybaudot.fr/Stats/indicecarts.html

La figure ci-dessus décrit l'évolution de la précision au cours de l'entrainement. Nous constatons que sa valeur est passée de moins de 0.991 à 0.993.

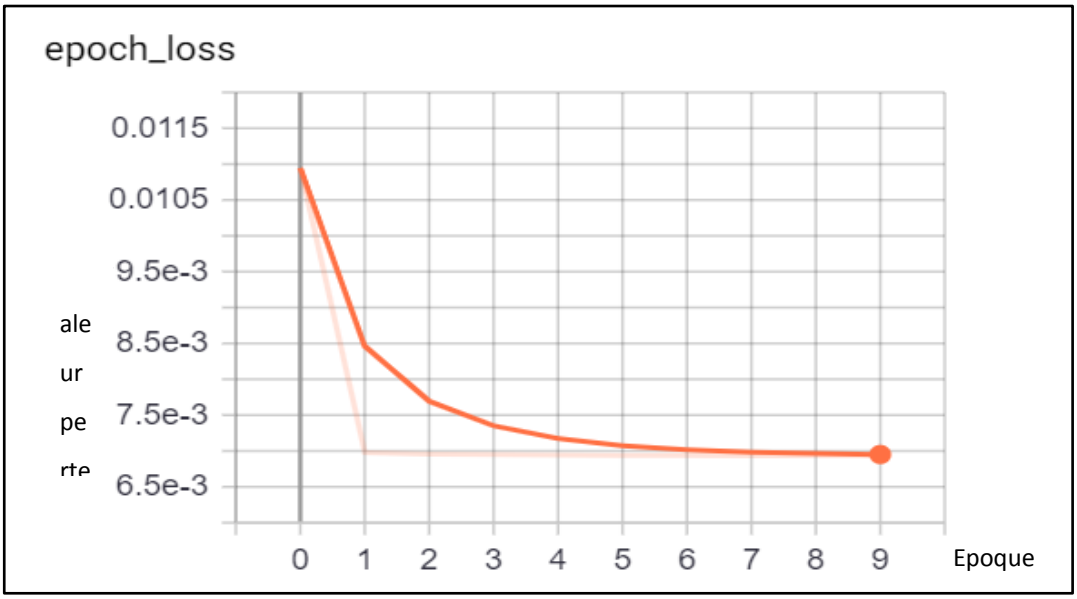

Figure 33 : Evolution de la perte du modèle RNA sur les données d'entrainement

La figure ci-dessus décrit l'évolution de la perte au cours de l'entrainement. Nous constatons que sa valeur est passée de plus de 1,105 $.10^{-2}$à7.10^{-3}

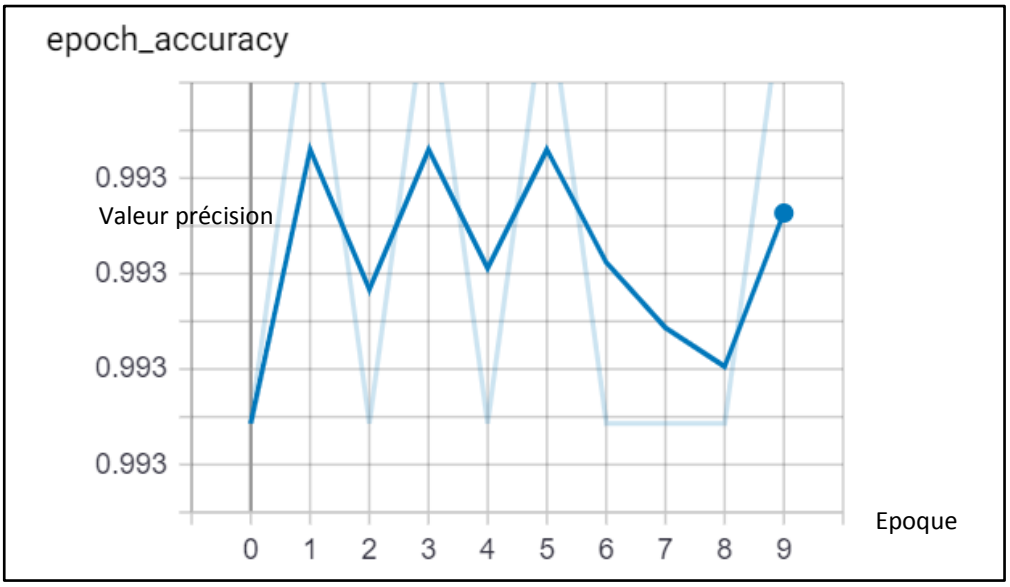

Figure 34 : Evolution de la Précision du modèle RNA avec les données de validation

La figure ci-dessus décrit l'évolution de la précision au cours de la validation. Nous constatons que sa valeur passe à 0,993

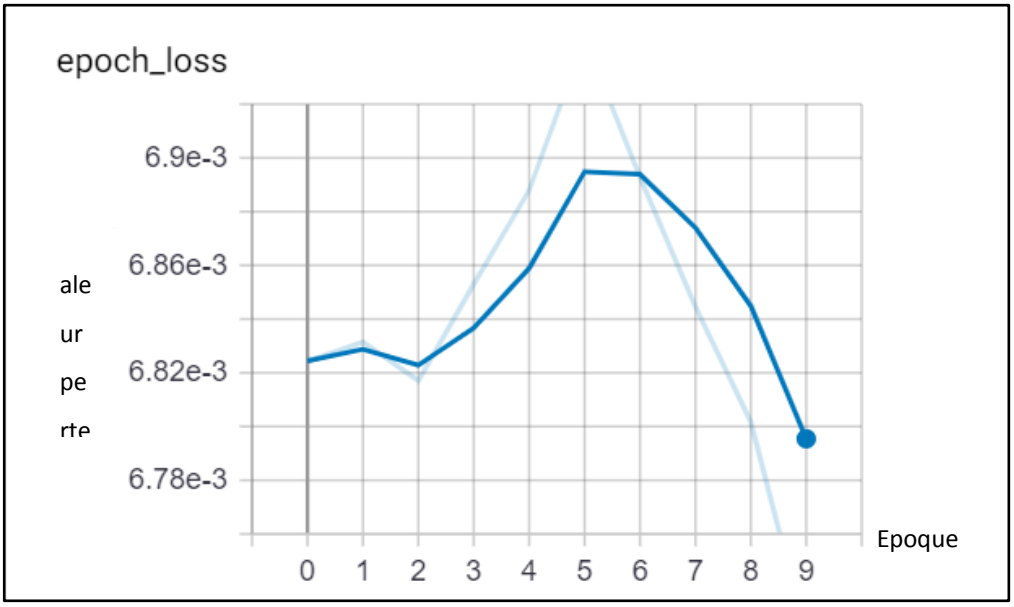

Figure 35 : Evolution de la Perte du modèle RNA avec les données de validation

La figure ci-dessus décrit l'évolution de la perte au cours de la validation. Nous constatons que sa valeur est passée de plus de 6,82 $.10^{-3}$à 6,78.10^{-3}.

6.2 Les résultats obtenus pour le modèle dans la phase de test

La performance du modèle lors de la phase de test est la même avec une valeur qui apprivoise les **0,99** ce qui confirme les bons résultats trouvés lors de la phase d'entrainement (voir figure 32).

D'autre part, suivant la métrique de perte nous remarquons que le modèle effectue aussi de bonne prédiction et qu'il a gardé toute sa performance en en réalisant la même réduction d'erreurs qu'à l'entrainement en l'occurrence **7. 10^{-3}**, ce qui prouve que notre modèle à bien garder sa robustesse acquise lors de l'entrainement et qu'il est prêt pour être utilisé et exploité sur le terrain et dans le monde professionnel.

6.3 Discussion

Dans l'entrainement ainsi que l'évaluation nous cherchons à augmenter au mieux la valeur de la précision de façon à approcher 100% et réduire la perte au plus pour approcher le 0.

Nos résultats obtenus sont acceptables car nous remarquons une augmentation dans la précision lors de test atteint une valeur maximale de 0.993. Ainsi de même pour la perte où nous remarquons une diminution de sa valeur qui atteint une valeur minimale de **7. 10^{-3}**.

En d'autre termes, quand il prédit qu'un lead est qualifié, sa prédiction est juste dans 99 %des cas.

Cependant ces résultats peuvent être optimisés en approfondissant le réseau par l'ajout de couches cachées, les blocs résiduels ou bien par l'augmentation du nombre d'époques ce qui ne nous a pas été possible à cause de problèmes reliés au limitations du matériel utilisé.

7 Conclusion

Nous avons présenté dans ce chapitre les outils et environnement de développement et la configuration utilisés pour notre implémentation ainsi que les ensembles d'apprentissage, de validation et de test que nous avons utilisés. Enfin, nous avons montré l'évaluation du modèle proposé.

En comparant les résultats trouvés on remarque que le nombre d'époque, la taille de la base d'apprentissage et les profondeurs des réseaux sont des facteurs important pour de meilleurs résultats.

Conclusion générale

1 Synthese

La gestion de la relation client est une tâche importante pour toute entreprise pour pouvoir rationaliser son processus de fonctionnement, offrir une meilleure qualité de service et pour améliorer sa rentabilité. Parmi les facettes inhérentes à cette tâche, le marketing est surement l'une des plus importantes car il permet, par exemple, de démarcher de nouveaux clients et de renouveler l'intérêt des anciens clients envers les services ou produits offerts par l'entreprise.

Notre mémoire a pour but d'implémenter un service de marketing dynamique à l'aide de l'apprentissage automatique plus particulièrement les réseaux de neurones, pour cela nous avons commencé par l'apprentissage automatique puis les réseaux de neurones toute en expliquant commet ce fait le traitement automatique du langage naturel. Par la suite nous avons présenté l'étude qu'on a réalisée sur le modèle proposé ainsi que son évaluation.

Les résultats obtenus démontrent que l'approche que nous avons utilisée pour réaliser notre service de marketing dynamique est certainement intéressante mais afin d'avoir des résultats plus satisfaisant et un service de qualité, un travail et des recherches sont nécessaire.

2 Perspectives

Nous envisageons plusieurs suites à nos travaux

- La première perspective serait d'intégrer l'apprentissage par renforcement aux modèles déjà réalisés
- La deuxième perspective serait Améliorer les résultats obtenus en ajoutant des couches cachées au modèle réalisé.
- Une autre perspective consiste à développer un service Web sous *Laravel lume* pour générer des leads suivant un besoin spécifique.

Bibliographie

1. **Samuel, A. L.** *Some studies in machine learning using the game of checkers*. IBM Journal of research and development, 3(3), 210-229, 1959. [En ligne]

2. **Tom M Mitchell et al.***Machine learning*. McGraw-Hill Education, 1st Edition 01 Mars 1997. [En ligne]

3. *Gouvernance de l'intelligence artificielle dans les entreprises*. CIGREF, Septembre2016. [En ligne]

4. "Brain Team", *Google Research*. [En ligne]

5. "OpenAI". *Openai.* [En ligne]

6. "Introducing machine learning". *Amazon, 09* Avril 2015. [En ligne]

7. **Claude Touzet**, *LES RÉSEAUX DE NEURONES ARTIFICIELS, INTRODUCTION AU CONNEXIONNISME*, Collection de l'EERIE, 1992. [En ligne]

8. **Djeriri, Youcef**. *Les réseaux de neurons artificiels.*University of Sidi-Bel-Abbes, Septembre 2017. [En ligne]

9. **Guillaume Saint-Cirgue**, "Apprendre les Bases du Machine Learning". *Machinelearnia*, Juin 2019. [En ligne]

10."Réseaux de neurones". *Stat soft*, 2013. [En ligne]

11. **Brownlee, Jason**. "Overfitting and underfitting with machine learning algorithms", *machinelearningmastery.* 21 Mars 2016. [En ligne]

12. **Brownlee, Jason.** "Overfitting and underfitting with machine learning algorithms", *machinelearningmastery.*21 Mars 2016. [En ligne]

13. "Glossaire du machine learning". *Google developers*, 05 Mars 2020. [En ligne]

14. **Diederik P. Kingma, Jimmy Lei Ba**, *A METHOD FOR STOCHASTIC OPTIMIZATION*, 2015. [En ligne]

15. “The Complete Guide to Artificial Neural Networks: Concepts and Models”. *Missinglink*, 2019. [En ligne]

16. **Aishwarya.**“Introduction to Recurrent Neural Network”. *geeksforgeeks,* 03 Octobre 2018. [En ligne]

17. **Olah, Christopher**. “Understanding LSTM Networks”. *github*, 27 Août 2015. [En ligne]

18. **Taleb, Mohamed.***Mise en oeuvre des réseaux de neurones pour une recommandation basée sur du contenu*, Mémoire de fin d’études UMMTO, 2018/2019.

19. **Benlahmar, El Habib.** “Les réseaux de neurones convolutifs”. *Datasciencetoday*, 12 Octobre 2018. [En ligne]

20. **Raghuram, Nandepu**. “Understanding and implementation of Residual Networks(ResNets)”. *medium*, 30 Octobre 2019. [En ligne]

21. “Microsoft increases sales by using AI for lead qualification”. *microsoft,* 09 Avril 2019. [En ligne]

22. **Masters, Timothy.***Practical Neural Network Recipes in C++*. Morgan Kaufmann, 1993. [En ligne]

23. **Raval, Siraj.**“Loss Functions Explained”.,24 Juillet 2018. [En ligne]

24. **Brownlee, Jason**. “How to Choose Loss Functions When Training Deep Learning Neural Networks”. *Machinelearningmastery*, 30 Janvier 2019. [En ligne]

25. **Ah-Pine, Julien**. *Apprentissage automatique*, Université Lyon 2, 2019. [En ligne]

26**.Afshine Amidi et Shervine Amidi**, « petites astuces d'apprentissage profond », Stanford.edu. [En ligne]

27.“DeepAI: The front page of A.I.”. *Deepai*. [En ligne]

28. “Google Developers”. *Google.* [En ligne]