Université Paris III – Sorbonne Nouvelle

Institut de Linguistique et Phonétique Générales Appliquées

Mémoire de Master 2 Traitement Automatique des Langues

Parcours Recherche & Développement

Présenté par Trang LAM

Attribution d'auteur des textes courts par utilisation des méthodes d'apprentissage profond

Sous la direction

du professeur Julien LONGHI

Année universitaire 2019 – 2020

Attestation de non-plagiat

Déclaration sur l'honneur

Je soussignée Trang Lam, déclare avoir rédigé ce travail sans aides extérieures ni sources autres que celles qui sont citées. Toutes les utilisations de textes préexistants, publiés ou non, y compris en version électronique, sont signalées comme telles. Ce travail n'a été soumis à aucun autre jury d'examen sous une forme identique ou similaire, que ce soit en France ou à l'étranger, à l'université ou dans une autre institution, par moi-même ou par autrui.

Date

Signature manuscrite de l'étudiante

Table des matières

Résumé

Attribution d'auteur est un sous-domaine de Traitement Automatique de Langues qui utilise les techniques d'apprentissage automatique pour identifier la paternité d'un texte. La plupart des travaux précédents se focalisent sur des textes longs tandis que les tendances actuelles des technologies de l'information encouragent des textes plus courts et informels. Ainsi, il est important de trouver les méthodes qui traitent ce type de textes. Ce mémoire vise à étudier la question d'attribution d'auteur sur des messages courts extraits du service *micro-blogging* Twitter. Nous explorons cette tâche sur la base des réseaux de neurones profonds. Pour ce faire, nous sélectionnons tout d'abord des traits stylométriques (des mots, des n-grammes de caractères et des lemmes) et nous nous en servons ensuite pour construire un système d'attribution d'auteur. La construction de ce système se réalise à l'aide des réseaux de neurones convolutionnels (CNN) et des réseaux récurrents à mémoire court et long terme (LSTM).

1. Introduction

L'avènement de l'internet en général et des médias sociaux en particulier a créé un nouveau moyen de communiquer des informations, de partager des opinions ainsi que de diffuser de la propagande. En offrant de nombreux avantages, selon les dernières statistiques[1], plus de la moitié de la population mondiale utilisent des médias sociaux. Cependant, certains individus en ont profité à des fins illégales, criminelles. Sur ces sites, l'utilisateur peut soit utiliser des pseudonymes, soit ne pas fournir les informations d'identification correctes. De plus, un seul utilisateur peut aussi créer de différents profils. L'anonymat constitue une préoccupation majeure pour les forces de l'ordre pour traquer l'identité de ces utilisateurs. Ainsi, l'identification d'auteur pour des messages anonymes, particulièrement des messages en ligne, s'avère un enjeu pour le domaine du Traitement Automatique des Langues. Cette question devient encore plus difficile quand des messages sont de plus en plus courts. Dans ce travail de recherche, nous nous intéressons à la tâche d'attribution d'auteur sur des textes courts extraits de Twitter[2], appelés *tweets*. Et l'approche sur laquelle nous nous basons pour mener notre étude est l'approche par apprentissage profond dû à son succès sur de nombreuses applications du TAL ces dernières années.

Ce mémoire est structuré comme suit. La première partie présentera des notions clés de la tâche d'attribution d'auteur, dont la stylométrie et l'apprentissage automatique. Des techniques d'apprentissage profond seront aussi explicitées dans cette partie. La deuxième partie sera dédiée à la description de notre méthodologie : le corpus utilisé, la transformation des données, la construction des réseaux et la recherche de performance pour nos réseaux. La troisième partie sera sur l'interprétation des résultats des expérimentations. Enfin, nous présenterons le bilan de notre travail ainsi que les perspectives futures.

[1] https://hootsuite.com
[2] https://twitter.com/

2. Etat de l'art

L'attribution d'auteur est une tâche particulière en catégorisation de textes dont le but est d'identifier la paternité d'un texte (Juola, 2006). Elle peut être vue comme une alliance entre la stylométrie et le TAL, particulièrement les méthodes par apprentissage automatique. Plus concrètement, les méthodes de stylométrie permettent de déduire des critères linguistiques pertinents qui servent de critères aux méthodes d'apprentissage. Ainsi, le processus d'attribution d'auteur se concrétise par deux étapes importantes :

1) L'extraction des caractéristiques quantifiant le style d'auteur (Stamatatos, 2006) qui réfère à des analyses stylométriques.
2) Le choix des méthodes d'apprentissage.

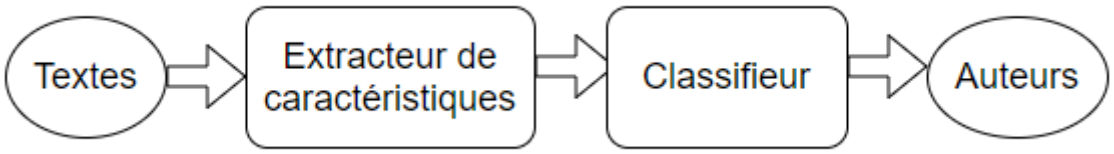

FIGURE 1 : Processus de la tâche d'attribution d'auteur

2.1 Stylométrie

La stylométrie est une branche de la linguistique computationnelle qui étudie le style littéraire à l'aide des méthodes quantitatives. Elle suppose qu'un auteur laisse de façon inconsciente dans son texte des caractéristiques qui peuvent mener à son identification. En d'autres termes, chaque individu possède ses propres habitudes d'écriture, par exemple :

- Certains ont tendance à écrire des phrases courtes et simples alors que d'autres préfèrent des phrases longues et complexes.
- Le vocabulaire utilisé est parfois riche, parfois limité. Par exemple, Ernest Hemingway, un romancier américain qui a remporté le prix Nobel de littérature en 1954, était célèbre pour son écriture minimaliste et simplifiée, dont un emploi faible des mots différents.

- Le choix et la fréquence des signes de ponctuation varient également en fonction d' auteur.

Ce sont tous des marqueurs spéciaux, des indicateurs stylométriques qui contribuent à la caractérisation d'une écriture d'un auteur.

L'analyse stylométrique remonte au 19ème siècle avec les travaux de (Mendenhal, 1887) et est vraiment influencée par les travaux de (Mosteller *et* Wallace,1964). A l'aide de la fréquence des mots courants et, en particulier, des mots fonctionnels (ou mots outils) (*function words*), Mosteller *et* Wallace ont réussi à déterminer les auteurs de *Federalist Papers*. Des mots outils sont des mots qui ne portent pas d'une valeur sémantique forte comme : les mots grammaticaux (les articles, les pronoms, les prépositions, les conjonctions), les auxiliaires ou les verbes supports. Mosteller *et* Wallace ont trouvé que Hamilton n'utilise jamais la préposition *whilst* mais préfère utiliser la préposition *while* (36 fois). A l'inverse, Madison n'utilise jamais la préposition *while* mais utilise 12 fois la préposition *whilst.* En poursuivant cette voie, les travaux de (Holmes, 1994 ; Holmes, 1998) se concentrent plutôt sur la définition des caractéristiques stylométriques supposées invariantes par l'auteur. Près d'un millier de caractéristiques stylométriques à différents niveaux d'analyse (lexical, syntaxique, sémantique, structurel et spécifique aux applications) sont proposés (Rudman,1998). Ces caractéristiques peuvent être : le nombre de mots, de phrases ; la longueur de mots, de phrases ; la fréquence des mots outils, des mots-formes, des n-grammes de mots, des n-grammes de caractères ; la fréquence des parties du discours ; les collocations (la fréquence des bigrammes de parties de discours) ; le nombre de *hapax legomena* (mots apparaissant une seule fois) ; etc. Parmi ces caractéristiques, des mots outils se sont montrés les plus efficaces pour distinguer des auteurs à travers les travaux de (Burrows, 2002 ; Diederich *et* al., 2003 ; Hoover, 2004 ; Zhao *et* Zobel, 2005 ; Savoy, 2013). Dans (Stamatatos, 2009), il souligne que ces mots sont utilisés par les auteurs de manière largement inconsciente et indépendante du sujet de l'étude. Ainsi, ils sont capables de capturer le style des auteurs sur différents sujets. Des caractéristiques syntaxiques comme des parties du discours, des chunks (ou des contituants continus et non-récursifs[3]) constituent aussi des caractéristiques souvent utilisées en attribution d'auteur (Stamatatos *et* al., 2001 ; Gamon, 2004 ; Zhao *et* al., 2006 ; Koppel *et* al., 2007 ; Pacheco *et* al., 2015). En comparant avec des mots outils, des informations syntaxiques sont considérées comme des empreintes digitales (*fingerprint)* encore plus fiables (Stamatatos, 2009). Cependant, la performance de ce type de

[3] **I.Tellier, D. Duchier, I. Eshkol, A. Courmet, M. Martinet**. Apprentissage automatique d'un chunker pour le français. *TALN2012*, Jun 2012, Grenoble, France. pp.431-438.

caractéristiques est liée directement à celle des outils d'étiquetage morpho-syntaxique, de chunking. En effet, si ces outils n'ont pas de bonne qualité, nous obtenons des mauvaises étiquettes de parties du discours, d'où l'impact sur la qualité de l'analyse stylométrique. Un état de l'art complet des caractéristiques stylométriques peut être consulté dans les travaux de (Stamatatos, 2009), de (El Bouanani et Kassou,2014) et de (Lagutina et al., 2019).

En réalité, il n'y a pas de consensus pour dire quelles sont les caractéristiques les plus optimaux. Ces caractéristiques varient d'une étude à l'autre et dépendent aussi de la longueur des textes. Plus le texte est long, plus il est facile de définir, de calculer des caractéristiques stylométriques. En revanche, cela devient plus difficile quand le texte est court et « bruyant ». Il s'agit des textes extraits des médias sociaux comme : des courriels, des blogs, des textes issus des réseaux sociaux (Facebook[4], Twitter), des SMS, des messages instantanés, etc. La difficulté de traiter ce type de textes, notamment les tweets, s'explique par différents facteurs :

- La longueur du texte : ces textes sont plutôt courts. La plupart des recherches précédentes travaillant sur des textes longs supposent qu'il y a un seuil minimum de la longueur du texte sous lequel l'identification d'auteur ne serait plus effective. (Ledger *et* Merriam, 1994) ont estimé un seuil à 500 mots et (Forsyth *et* Holmes, 1996) l'ont fixé à 250 mots.
- L'absence d'une structure précise : A l'inverse des textes littéraires qui ont une syntaxe définie et une structure sémantique, des textes extraits des médias sociaux sont généralement non-structurés et ont leurs propres conventions. Par exemple, dans le cas des tweets, ils sont caractérisés par les hashtags, les mentions, les retweets, les émoticons, les abréviations, etc.
- Le changement de style en fonction du contexte, d'où l'instabilité dans l'usage de vocabulaire.

Parmi les caractéristiques stylométriques mentionnées au-dessus, des n-grammes de mots et des n-grammes de caractères apparaissent comme étant les caractéristiques les plus appropriés pour des textes courts vu qu'ils sont moins liés au type et au sujet de textes. En plus, ils sont capables de capturer les informations lexicales, contextuelles ou thématiques (pour des grandes valeurs de *n*) sans avoir besoin des connaissances préalables de la grammaire d'une langue. En d'autres termes, ils peuvent s'appliquer aux différentes langues naturelles (Stamatatos,2006 ; Grieve, 2007). (Statamatos, 2009) montrent aussi que des n-grammes de caractères ne sont pas trop affectés par des textes « bruyants ». Grâce à sa robustesse, cette

[4] https://www.facebook.com/

approche se trouve dans plusieurs travaux et donne des résultats positifs (Chasaki, 2005 ; Stamatatos, 2009 ; Layton et al.,2010 ; Tanguy et al., 2011 ; Sun *et al.*, 2012 ; Schwartz *et al.*,2013 ; Stamatatos, 2013).

Le choix des caractéristiques stylométriques constitue une étape essentielle dans l'attribution d'auteur. Plus le choix est approprié et robuste, mieux seront les résultats. Dans ce mémoire, nous choisissons trois traits les plus utilisés dans l'attribution d'auteur pour les textes courts qui sont : des mots (ou tokens), des lemmes et des n-grammes de caractères. Une fois les caractéristiques sélectionnées, l'étape suivante consiste à choisir des algorithmes d'apprentissage pour classer les textes.

2.2 Apprentissage automatique

Avant de présenter les différents algorithmes d'apprentissage utilisés pour la tâche d'attribution d'auteur, il est nécessaire de faire une récapitulation de l'histoire ainsi que la définition de l'apprentissage automatique.

2.2.1 Histoire et définition

En 1950, Alan Turing, un mathématicien britannique, s'est posé très tôt la question : « Est-ce que les machines peuvent penser ? ». Pour répondre à cette question, dans son article « Computing Machinery and Intelligence » publié en 1950, il a proposé un test inspiré par le "Jeu de l'imitation" (*The Imitation game*), connu aujourd'hui sous le nom "Test de Turing".

Le principe du test est simple : dans un jeu de questions-réponses, un humain A pose des questions aux deux autres individus : l'un est une machine et l'autre est un humain. Tous les trois se trouvent dans les pièces séparées. Et au bout de quelques minutes d'échanges, si l'humain A qui joue aussi le rôle d'un évaluateur n'est pas capable de discerner l'homme de la machine, la machine est réputée avoir passé le test et peut être considérée comme "intelligente".

Dans le même article, dans la dernière partie intitulée « Les machines qui apprennent » (*Learning Machines*), il a montré que la machine qui est susceptible d'emporter le jeu d'imitation est celle qui est capable **d' « apprendre »** et qui doit soumettre à un processus d'éducation. Depuis lors, l'apprentissage fait en partie intégrante de l'intelligence.

Bien que le concept d'apprentissage ait été introduit par Alan Turing en 1950 mais c'est jusqu'à 1959, l'expression « Apprentissage automatique » (*Machine Learning*) a été utilisée

pour la première fois par Arthur Samuel qui développait un programme de jeu de dames d'auto-apprentissage. Il a décrit *Machine Learning* comme « le domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmé ». Pourtant, cette définition est un peu vague, ancienne et informelle. En 1997, Tom M. Mitchell, dans son livre intitulé *Machine Learning*[5] a fourni une définition, largement citée, qui est plus formelle et moderne : « A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E » (Un programme informatique se dit d'apprendre de l'expérience E par rapport à une classe de tâches T et une mesure de performance P, si sa performance aux tâches dans T, mesurée par P, s'améliore avec l'expérience E). Cette définition suggère l'idée qu'un algorithme d'apprentissage est capable d'inférer un comportement, des règles à partir d'exemples qui lui sont fournis et de les généraliser à des situations nouvelles.

Quelle que soit la définition, le tout est de bien comprendre que **l'apprentissage automatique** concerne l'élaboration d'algorithmes permettant aux machines d'apprendre, de s'améliorer automatiquement à partir d'expériences, de données réelles. Ou plus simplement, il s'agit de « l'art de transformer des exemples en programme »[6].

2.2.2 Les techniques d'apprentissage

L'apprentissage automatique se compose deux principales phases :

- Phase d'apprentissage (ou d'entraînement) dont le but est de construire un modèle à partir des exemples.
  Plus concrètement, cette phase se déroule de la manière suivante :
  Une fois que les exemples sont fournis, un algorithme d'apprentissage va s'en servir pour « s'entraîner », s'ajuster la valeur de certains paramètres et générer ensuite un modèle.
- Phase de test dont le but est d'évaluer la performance du modèle préalablement entraîné en utilisant des nouvelles données

En fonction de nature des exemples fournis dans la phase d'apprentissage, on peut distinguer deux techniques d'apprentissage :

[5] **T. M. Mitchell**. Machine learning, New York: McGraw Hill, 1997

[6] **I. Tellier**. Introduction à la fouille de textes. Université de Paris 3 - Sorbonne Nouvelle

a) L'apprentissage supervisé :

- Des exemples sont des couples entrées/sorties (x,y). C'est-à-dire, on fournit à l'algorithme des données d'entrées qui sont déjà préalablement étiquetées avec les sorties souhaitées.
- Les algorithmes d'apprentissage supervisé :
  - **SVM (Machine à vecteurs de support)** :

    SVM a été appliqué pour la première fois à la tâche d'attribution d'auteur à travers les travaux de (Vel et al., 2001) pour les courriels. Plusieurs travaux se succèdent et mettent en évidence le succès de cet algorithme dans ce domaine (Fung, 2003 ; Koppel et Schler, 2003; Koppel et al.,2005, 2006, 2007 ; Diederich et al.,2003 ; Abbasi *et* Chen, 2005 ; Zheng et al., 2006 ; Argamon et al.,2008 ; Hedegarrd et Simonsen, 2011 ; Savoy, 2013 ; Mikros et al.,2013 ; Schwartz et al., 2013 ;Koppel et Winter ,2014). La performance de SVM combinée à des n-grammes de caractères est bien détaillée dans (Stamatatos et al., 2013).
  - **Decision Trees (Arbres de décision) :** les travaux de (Koppel et Schler, 2003 ; Abbasi et Chen, 2005 ; Zheng et al.,2006)
  - **Random Forest (Forêts aléatoires)** : les travaux de (Caliskan-Islam et al.,2015 ; Maitra et al.,2015 ; Pacheco et al.,2015)
  - **Naive Bayes** : Mosteller et Wallace, 1964 ; Clement et Sharp, 2003 ; Peng et al.,2004 ; Boutwell, 2011 ; Savoy, 2013 ; Almishari et al.,2014
  - **K-Nearest Neighbor (K-plus proches voisins)**: Kjell et al.,1995; Jockers et Witten,2010; Halvani et al.,2013.
  - **Neural Network (Réseaux de neurones) :** Matthews et Merriam, 1994 ; Kjell, 1995 ; Lowe et Matthews, 1995 ; Tweedie et al., 1996 ; Hoorn et al., 1999 ; Zheng et al.,2006. Au cours des dernières années, l'avènement de l'apprentissage profond avec les différentes architectures profondes de réseaux de neurones ouvre une nouvelle ère dans le domaine de TAL en général et dans celui d'attribution d'auteur en particulier. De plus en plus de travaux s'appuient sur cette approche et donnent des résultats prometteurs : Rhodes, 2015 ; Bagnall *et al.*,2015 ; Ge *et al.*,2016 ; Shrestha *et al* ., 2017; Hitshler J et al., 2017 ; Schaetti, 2018; Gagala,2018. La plupart de ces travaux se trouvent dans la

compétition PAN[7] (Un atelier international s'intéresse aux différents thèmes : la détection du plagiat, l'attribution d'auteur, la vérification d'auteur, le profilage d'auteur et la détection du changement de style).

b) L'apprentissage non-supervisé :

- Contrairement à l'apprentissage supervisé, les exemples dans l'apprentissage non-supervisé sont simplement des données d'entrées {x} sans avoir au préalable les sorties correspondantes. L'apprentissage non supervisé ou le « clustering » consiste à construire des regroupements cohérents (clusters) formés par l'ensemble de données. Dans le domaine d'attribution d'auteur, on parle d'« author clustering ».
- Les algorithmes d'apprentissage non-supervisé sont : K-Means (K-Moyennes), Clustering Hiérarchique avec les travaux de (Iqbal et al., 2010 ; Layton et al., 2010 ; Sittar et al., 2016 ; Sari et al., 2016 ; Rao et al.,2017 ; Gómez-Ardono et al.,2017).

Dans ce mémoire, nous accordons une attention particulière à la classification par apprentissage supervisé en utilisant les réseaux de neurones et plus généralement l'apprentissage profond.

2.3 Apprentissage profond

Au cours des dernières années, l'apprentissage profond connaît un véritable essor et a permis d'améliorer l'état de l'art de nombreuses applications. L'apprentissage profond (*Deep learning)* est basé sur des réseaux de neurones artificiels avec de nombreuses couches cachées. Le terme « profond » fait référence au nombre de couches cachées du réseau. Plus le nombre de couches est élevé, plus le réseau est profond et plus il est capable de résoudre des problèmes complexes. Habituellement, un réseau de neurones dit « profond » lorsqu'il s'est composé d'au moins 3 couches cachées. Bien qu'actuellement à la mode, la théorie derrière cette approche n'est pas récente. En effet, l'idée d'utiliser des réseaux de neurones artificiels, s'inspirant des réseaux de neurones biologiques, à des fins d'apprentissage remonte déjà aux années 1950, avec les travaux de Frank Rosenblatt, à savoir Perceptron.

[7] https://pan.webis.de/

2.3.1 Perceptron

Perceptron se sert d'un algorithme d'apprentissage supervisé conçu pour résoudre des problèmes de classification linéaire.

Cet algorithme a été inventé en basant sur deux notions principales : neurone artificiel (ou neurone formel, les travaux de Warren McCulloh et Walter Pitts en 1943) et la règle de Hebb.

a) Neurone formel :

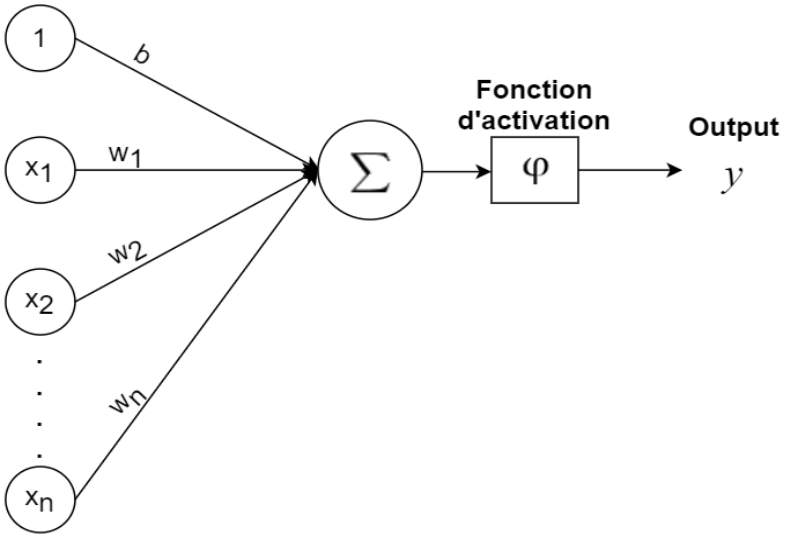

FIGURE 2 : Neurone formel (perceptron)

En entrée, le neurone reçoit des valeurs numériques, notées x_1 à x_n et chacune s'associe à un poids correspondant, noté w_1 à w_n .

On calcule ensuite la somme pondérée des valeurs :

$$\sum_{i=1}^{n}(w_i \times x_i) = w_1x_1 + \ldots + w_n x_n$$

On ajoute puis le biais b qui est le poids d'une entrée constante égale à 1. Il permet de normaliser la fonction d'activation, ajouter la flexibilité au réseau. On obtient donc :

$$\sum_{I=1}^{n}(w_i \times x_i) + b$$

Ce résultat sera transmis puis à une fonction d'activation φ qui va décider si le neurone est activé ou non. Il existe plusieurs types de fonctions d'activations mais ce modèle utilise la fonction de Heaviside qui va produire en sortie une valeur y binaire (0 ou 1).

$$f(x) = \begin{cases} 1, & si\ x \geq 0 \\ 0, & sinon \end{cases}$$

Tout ce processus peut se simplifier par l'équation :

$$y = \varphi(\sum_{i=1}^{n}(w_i \times x_i) + b)$$

b) Règle de Hebb :

Une règle d'apprentissage des réseaux de neurones proposée par le psychologue canadien Donal Hebb en 1949 (Hebb, 1949)[8].

Entre 1970 et 1980, les recherches liées à ce concept ont été mises en sommeil vu ses défauts pointés par Marvin Lee Minsky et Seymour Papert dans leur ouvrage Perceptrons en 1969, dont l'incapacité de résoudre XOR (« OU EXCLUSIF ») à partir d'un seul neurone artificiel. Il est donc nécessaire d'envisager un autre modèle plus performant, le perceptron multicouche.

2.3.2 Perceptron multicouche

En 1986, David Rumelhart et Geoffrey Hinton ont proposé la première génération de réseaux de neurones profonds, sous le nom « Perceptron multicouche ».

Comme indiqué dans son nom, « Perceptron multicouche » est un Perceptron organisé en couches successives. Outre les entrées et la couche de sortie, ce réseau peut contenir une ou plusieurs couches cachées. Chaque couche est constituée d'un ou plusieurs neurones artificiels n'ayant pas de connexion entre eux. Les neurones de chaque couche sont liés à tous les neurones de la couche précédente et tous ceux de la couche suivante. En d'autres termes, les sorties d'une couche serviront d'entrées au calcul de la couche suivante et ainsi de suite. C'est la dernière couche qui produit les sorties d'un perceptron multicouche.

[8] **D. O. Hebb**. The Organization of Behavior, New York, Wiley & Sons, 1949

On parle ici d'un réseau de neurones à propagation avant (*Feed-forward Neural Network*) où l'information ne traverse que dans un sens unique, vers l'avant, des entrées vers la sortie. On le distingue avec les réseaux de neurones dits récurrents (cf. section 2.3.4).

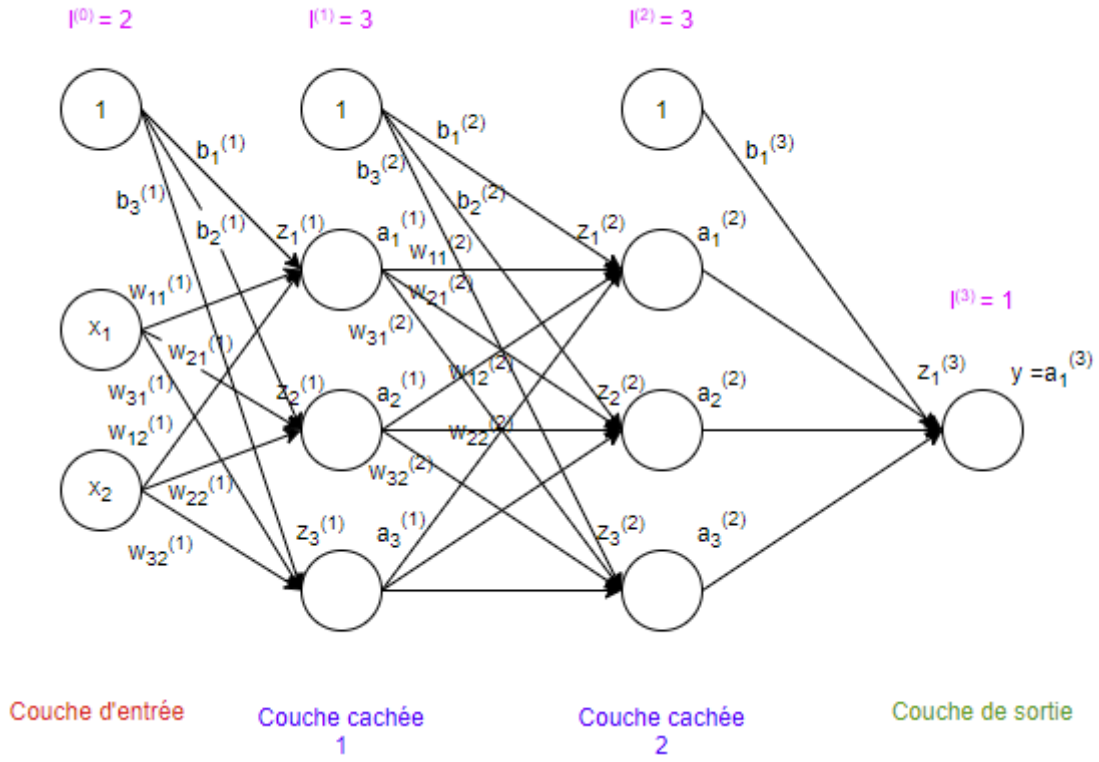

FIGURE 3 : Perceptron multicouche de deux couches cachées

Notations :

- $w_{ij}^{(l)}$ est le poids qui relie le $i^{ième}$ neurone de la $l^{ième}$ couche au $j^{ième}$ neurone de la $(l-1)^{ième}$ couche
- $b_i^{(l)}$ est le biais du $i^{ième}$ neurone de la $l^{ième}$ couche
- $z_i^{(l)}$ est la valeur d'aggrégation du $i^{ième}$ neurone de la $l^{ième}$ couche
- $a_i^{(l)}$ est la valeur d'activation du $i^{ième}$ neurone de la $l^{ième}$ couche

D'une manière succincte, supposons qu'on a un perceptron multicouche de deux couches cachées comme celui de la figure 3. Tout d'abord, chaque neurone de la première couche prend en entrée des valeurs d'entrées associées à leurs poids et produit en sortie une valeur correspondante. Cette valeur se calcule par deux étapes :

$$z_i^{(l)} = \sum_{j=1}^{l^{(l-1)}} a_j^{(l-1)} * w_{ij}^{(l)} + b_i^{(l)}$$

$$a_i^{(l)} = \sigma(z_i^{(l)})$$

où σ est la fonction d'activation.

On obtient donc $l^{(l-1)}$ valeurs en sortie, où $l^{(l-1)}$ est équivalent au nombre de neurones dans cette couche. Ces valeurs sont ensuite utilisées comme les entrées de la couche suivante et ainsi de suite. Finalement, la sortie d'un perceptron multicouche est constituée des valeurs de la couche de sortie après avoir appliquées la fonction d'activation. Le choix de la fonction d'activation de la couche de sortie est déterminé par le problème traité. Les trois problèmes les plus rencontrés sont :

1) Problème de régression standard → la fonction Identité
$$f(x) = x$$
2) Problème de classification binaire → la fonction Sigmoïde
$$f(x) = \frac{1}{1 + e^{-x}}$$
3) Problème de classification mutli-classes → la fonction Softmax.
   Cette fonction prend en entrée un vecteur *z* composé de *k* nombres réels (*k* désigne aussi le nombre de classes) $z = (z_1, z_2, z_3, \ldots. z_k)$ et renvoit en sortie un vecteur $\sigma(z)$ de la même longueur. Les valeurs de ce vecteur correspondent à des probabilités des classes. La somme de ces valeurs doit être égale à 1.
$$\sigma(z)_k = \frac{e^{z_k}}{\sum_{i=1}^{k} e^{z_i}}$$

Apprentissage :

Outre les réseaux multicouches, le regain d'intérêt des réseaux de neurones fut en partie lié à l'introduction d'un algorithme pour entraîner ces réseaux, c'est-à-dire pour ajuster des poids des connexions. Il s'agit de l'algorithme de rétropropagation du gradient (*backpropagation*)[9]. Cet algorithme permet de minimiser l'écart entre la prédiction donnée par le réseau et la sortie

[9] **D. Rumelhart, G. Hinton & R. Williams**. Learning representations by back-propagating errors. *Nature 323*, 533–536 (1986). https://doi.org/10.1038/323533a0

attendue à l'aide de la méthode de descente du gradient. D'une manière succincte, il se déroule en deux étapes :

1. La première est la « propagation avant » (*forward propagation*) lors de laquelle la prédiction est calculée. On calcule ensuite l'erreur de prédiction en utilisant une fonction de coût (ou fonction de perte/d'erreur).
2. Une fois que l'erreur est calculée, on passe à la deuxième étape, la « propagation arrière » (*backward propagation)*. Elle consiste à propager l'erreur de couche en couche vers l'arrière. Enfin, les poids sont mis à jour par l'algorithme de descente du gradient.

Même si le perceptron multicouche nous a montré son efficacité dans la résolution des problèmes de classification, il reste encore des désavantages. Comme nous pouvons remarquer, les couches d'un perceptron multicouche sont entièrement connectées, c'est-à-dire que la valeur d'un neurone de la couche *l* va dépendre des valeurs de tous les neurones de la couche *l-1*. Ainsi si le nombre d'entrées est trop élevé, cela entraîne une augmentation importante du nombre de poids du réseau. Ce problème a abouti à l'invention d'une nouvelle génération de réseaux de neurones, à savoir les réseaux de neurones convolutionnels (*Convolutional Neural Network* ou *CNN*).

2.3.3 Réseaux de neurones convolutionnels (CNN)

Inspiré du comportement du cortex visuel des vertébrés, Yann LeCun, le chercheur français reconnu comme un des inventeurs de l'apprentissage profond, a développé en 1990 le réseau de neurones convolutionnel dédié spécifiquement à la classification d'image de chiffres manuscrits[10]. Mais à l'époque, par manque de dispositifs de calcul puissants et d'accessibilités à des données, cette approche connaît des progrès très limités. C'est jusqu'à des années 2010, grâce à l'évolution technologique, l'arrivée des processeurs sur les cartes graphiques et surtout la disponibilité des données massives (Big Data), elle rencontre un succès particulièrement remarquable et devient désormais une méthode phare de l'intelligence artificielle.

Notamment connus comme les plus performants modèles pour le traitement des images, ces réseaux ne connaissent un essor dans le traitement du langage naturel qu'à partir des travaux de (Kim,2014), (Kalchbrenner et al., 2014), (Zhang et al., 2017), (Conneau et al., 2017) pour la tâche de classification des textes. En TAL, ils sont considérés comme une généralisation des

[10] **Y. LeCun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel & D. Henderson**. Handwritten Digit Recognition with a Back-propagation Network. In *NIPS*, 396-404, 1989.

modèles *n*-grammes. Ainsi, ils permettent de détecter des motifs récurrents dans une phrase, dont les informations relatives à la syntaxe, les cooccurrences, les collocations, les négations. En plus, des n-grammes à distance sont aussi captés grâce à la couche de regroupement (*pooling*) du réseau.

Un CNN appliqué au langage naturel est semblable à celui appliqué à l'image, à part le niveau de l'extraction des caractéristiques par les filtres. En traitement des images, l'extraction se réalise sur des petites régions de l'image, le filtre se déplace donc dans deux directions (horizontalement et verticalement). Alors qu'en traitement du langage naturel, le filtre ne se déplace dans une seule direction (horizontalement).

Dans l'idée de diminuer le nombre de poids à calculer, ces réseaux se décomposent en deux parties : une partie dite convolutive qui sert à extraire des caractéristiques (*features*) des données et une partie qui se sert des couches entièrement connectées sous la forme d'un perceptron multicouche pour la tâche de classification.

La partie convolutive se construit par trois principaux types de couches :

- **La couche d'Embedding (**appelée aussi *lookup table*) sert à transformer chaque texte à *N* mots sous forme d'une matrice de taille *N*M*, où *M* est la dimension des représentations vectorielles de mots. C'est-à-dire, chaque ligne de cette matrice correspond donc à une représentation vectorielle d'un mot, appelé *embedding*. Les mots qui sont sémantiquement et syntaxiquement similaires vont avoir des *embeddings* proches. Par exemple, pour un texte à 50 mots en utilisant un embedding de dimension 100, on obtiendra une matrice de 50*100 comme entrée. La technique de représenter un mot par un vecteur de nombres réels est connue sous le nom « Plongement de mots » (*Word Embedding*). Elle sera détaillée dans la section 2.3.5.
- **La couche de convolution** est une brique de base de ce type de réseau. Elle vise à détecter des caractéristiques locales de l'entrée en faisant glisser les filtres sur la matrice d'embeddings obtenue via la couche d'Embedding.

La sortie de la couche de convolution se calcule par la manière suivante :

Soit x_i le vecteur d'embedding du $i^{ème}$ mot du texte de *n* mots et un filtre *w* de taille *h*d*, où *h* est le nombre de mots couverts par le filtre (les *n-grammes*) et *d* est égale à *M*, la dimension des représentations vectorielles de mots. La caractéristique c_i obtenue par l'opération de convolution entre le filtre *w* et la fenêtre de mots $x_{i:i+h-1}$ est donnée par :

$$c_i = \varphi(x_{i:i+h-1} * w^T + b)$$

b est le biais et φ est une fonction d'activation non linéaire, la fonction la plus utilisée aujourd'hui est la fonction unité de rectification linéaire ReLU (*Rectified Linear Unit*)[11]. Cette fonction permet d'améliorer la convergence de l'apprentissage du réseau et est définie comme :

$$f(x) = \begin{cases} 0, & si\ x < 0 \\ x, & si\ x \geq 0 \end{cases}$$

Le filtre *w* est appliqué à toutes les fenêtres de *h* mots de la matrice d'embeddings pour obtenir un vecteur, appelée une carte de caractéristiques *(feature map*).

$$c = [c_1, c_2, \ldots\ldots, c_{n-h+1}]$$

Afin de pouvoir capturer plusieurs types de motifs particuliers, nous pouvons utiliser k filtres de tailles égales ou différentes. Nous obtenons donc, après cette couche, un ensemble de cartes de caractéristiques (*feature maps*) $c^1, \ldots, c^k$.

- **La couche de regroupement (pooling)** permet de réduire la taille des cartes de caractéristiques tout en s'assurant que les caractéristiques les plus importantes sont gardées. Elle permet donc de réduire le nombre de paramètres et de calculs à effectuer dans le réseau. Deux types de couches de pooling les plus connus sont :
  - Le max pooling qui sélectionne la composante maximale de chacun des vecteurs $c^j, j \in \{1, \ldots\ldots, k\}$
  - L'average pooling qui prend la valeur moyenne de chacun des vecteurs.

A la fin du réseau, les résultats obtenus par cette partie convolutive sont transmis ensuite à des couches entièrement connectées qui permettent de classifier les données en entrée du réseau.

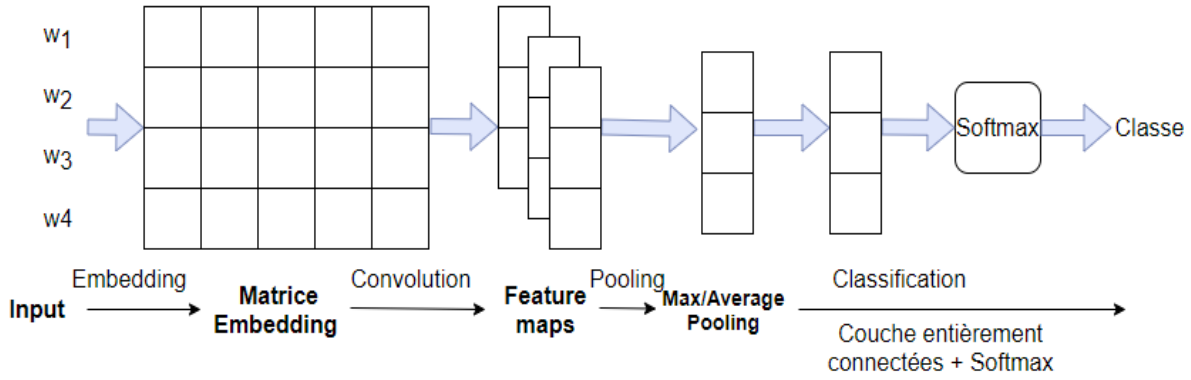

FIGURE 4: Illustration d'un CNN pour la classification des textes

[11] **V. NAIR & G. HINTON.** Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML*, 2010.

2.3.4 Réseaux récurrents à mémoire court et long terme (LSTM)

Les réseaux de neurones récurrents (*Recurrent Neural Networks* ou *RNN*) (Elman, 1990) sont des modèles spécialisés dans le traitement des informations séquentielles. Ainsi, ils sont particulièrement adaptés à des tâches liées au traitement du langage naturel. La particularité de ce type d'architectures réside dans sa capacité à modéliser les dépendances entre les mots. A l'aide de leur mémoire interne, ces réseaux sont capables d'utiliser l'information contextuelle passée lors du traitement de l'information courante dans une séquence. D'une manière succincte, ils vont parcourir une séquence, par exemple de gauche à droite, un mot après l'autre et mettre à jour sa mémoire interne à chaque pas.

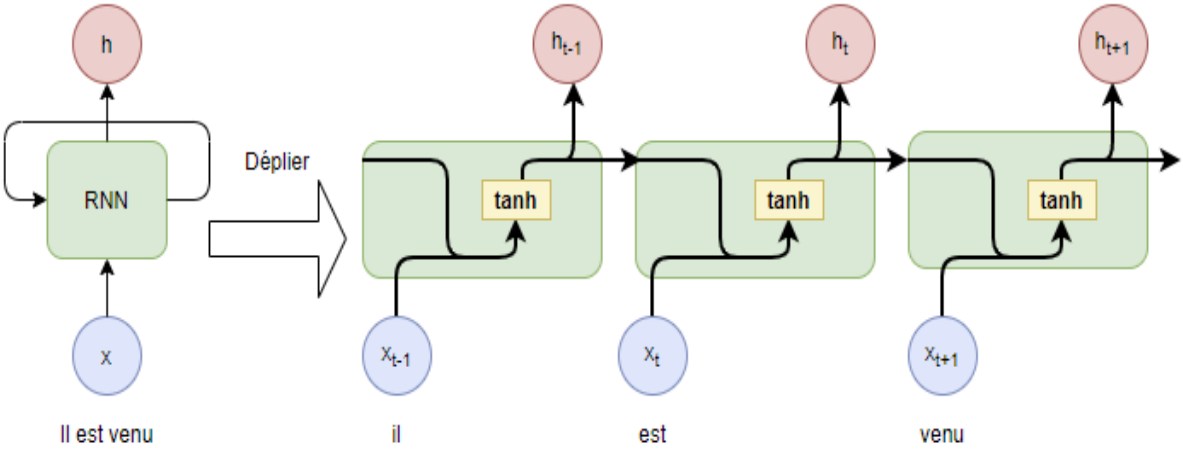

FIGURE 5 : Un réseau de neurones récurrent

Cependant, ce réseau a des difficultés de modéliser des dépendances à longue distance[12]. En 1997, Sepp Horchreiter et Jürgen Schmidhuber ont introduit un nouveau type de RNN. Il s'agit d'un réseau récurrent à mémoire court et long terme (*Long Short Term Memory* ou simplement *LTSM*).

[12] **S. Hochreiter, Y. Bengio, P. Frasconi & J. Schmidhuber.** Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies. In S. *C. Kremer and J. F. Kolen, editors, A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

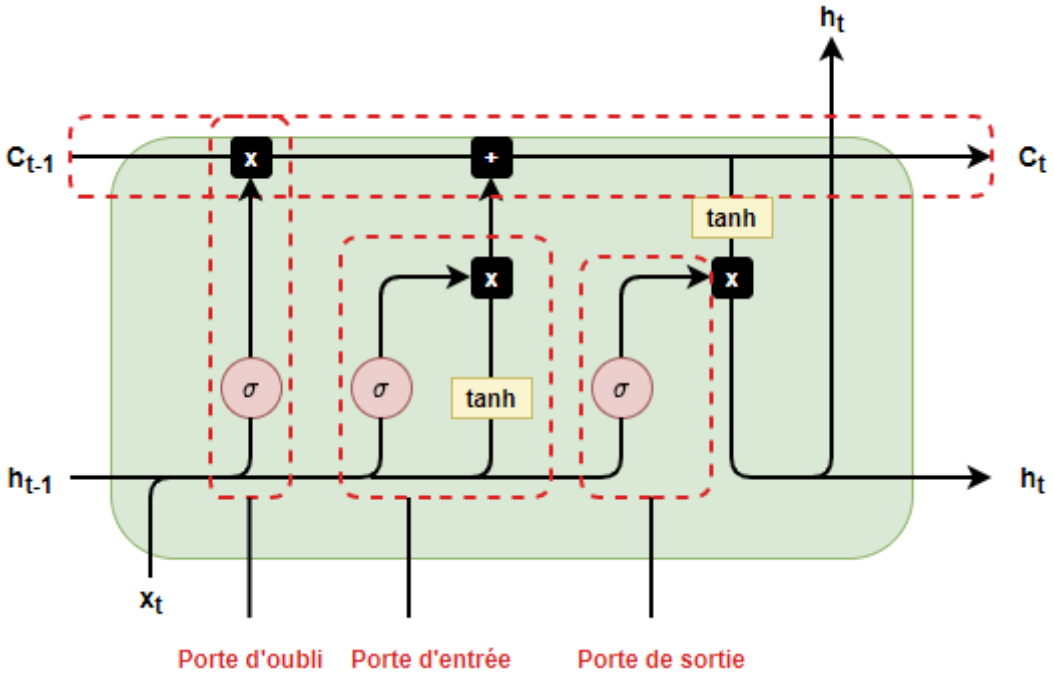

FIGURE 6: Une cellule LSTM

Notations :

- C_{t-1} : l'état interne de la cellule à l'instant *t-1*
- C_t : l'état interne de la cellule à l'instant *t*
- h_{t-1} : l'état caché à l'instant *t-1*
- h_t : l'état caché à l'instant *t* (ou la sortie de la cellule)
- x : la multiplication ; + : l'addition

Ce réseau est capable de mémoriser les informations sur une longue période de temps grâce à son état de la cellule (ou état interne) *C* qui représente la mémoire à long terme du réseau. A chaque pas de temps *t*, le contenu de la cellule *C* est mis à jour en supprimant les informations non pertinentes et en ajoutant les nouvelles informations. Cette opération s'effectue à travers la porte d'oubli et la porte d'entrée.

- **Porte d'oubli** va décider quelles informations de l'état de la cellule *C* doivent être gardées ou oubliées. L'information courante x_t et l'information de l'état caché précédent h_{t-1} va se réunir et passer ensuite à la fonction d'activation Sigmoid. On obtient donc un vecteur f_t de

valeurs entre 0 et 1. Si la valeur est proche de 0, il faut oublier l'information et si elle est proche de 1, il faut mémoriser l'information.

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

- **Porte d'entrée** va choisir quelles nouvelles informations seront ajoutées à la mémoire *C*. Afin de calculer ces nouvelles informations, l'information courante x_t et l'information de l'état caché précédent h_{t-1} sont transmises à deux fonctions d'activation Sigmoid et Tanh. Sigmoid renvoie un vecteur i_t de valeurs entre 0 et 1, de façon similaire à la porte d'oubli.

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i)$$

Alors que Tanh renvoie un vecteur $\tilde{C}_t$ de valeurs entre -1 et 1.

$$\tilde{C}_t = \tanh(W_C . [h_{t-1}, x_t] + b_C)$$

Le produit de deux vecteurs $i_t * \tilde{C}_t$ constitue donc les informations doivent être mises à jour à la mémoire.

Les vecteurs obtenus par les portes d'oubli et d'entrée nous permettent de calculer maintenant l'état de la cellule à l'instant t C_t. Tout d'abord, on multiplie f_t avec l'état précédent de la cellule C_{t-1} pour oublier certaines informations de l'état précédent. Puis on additionne le tout avec $i_t * \tilde{C}_t$ qui correspond à des nouvelles informations.

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

- **Porte de sortie** va déterminer quelle partie de l'état de la cellule sera transmise à la sortie de la cellule, qui est égal à l'état caché h_t, en reposant sur l'état de la cellule C_t.

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

De façon analogue à f_t et i_t, o_t est un vecteur de valeurs entre 0 et 1.

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

Les réseaux LSTM sont très connus aujourd'hui dans le cadre de l'apprentissage des données temporelles ou séquentielles. Ils ont montré leur efficacité dans diverses applications de TAL telles que : l'analyse des sentiments, la modélisation du langage, la reconnaissance des entités nommées, la traduction automatique, l'étiquetage morpho-syntaxique, la reconnaissance de la parole, la synthèse de la parole, etc.

2.3.5 Représentation vectorielle des mots

Pour être exploitables par les algorithmes d'apprentissage, les données textuelles doivent se représenter sous la forme d'un vecteur de nombres réels. La technique de représentation la plus couramment utilisée est « sac de mots » (*bag of words)*. Elle vise à représenter chaque texte sous forme d'un vecteur *v* de taille V qui correspond à la taille du vocabulaire. La valeur de chaque élément v_i du vecteur *v* peut être un simple booléen, le nombre d'occurrences du mot *i* dans le texte ou la pondération TF-IDF (*Term Frequency - Inverse Document Frequency*). Cependant cette représentation soulève quelques problèmes. Le plus critique est qu'elle ne peut pas capturer les corrélations sémantiques ou syntaxiques entre les mots. Ainsi, une nouvelle méthode de représentation est introduite, à savoir la représentation vectorielle des mots ou plongement de mots (*Word Embeddings*). Cette technique repose sur « l'hypothèse de distribution » de (Harris, 1954) qui suppose que les mots apparaissant dans des contextes similaires sont susceptibles d'avoir le même sens. Plus précisément, elle a pour but d'encoder un mot en un vecteur de valeurs réelles qui est capable de représenter des relations, des similarités entre les mots. Ainsi, plus les mots sont sémantiquement ou syntaxiquement proches, plus ils se rapprochent en termes de distance.

FIGURE 7 : A gauche représente les différents vecteurs capturant la relation de genre (masculin-feminin). A droite représente en plus les vecteurs capturant la relation de nombre (singulier-pluriel). Figure extraite de (Mikolov, et al., 2013).

Les *Word Embeddings* étaient d'abord employés dans les travaux de (Bengio et *al.*, 2003) et de (Collobert et Weston,2008) pour les modèles de langage neuronaux. Mais ils ne se sont vraiment généralisés que depuis 2013 avec la sortie de l'algorithme *word2vec*[13] (Mikolov *et al.*,2013). Cet algorithme permet d'entraîner des *Word Embeddings* en s'appuyant sur deux architectures de réseaux de neurones : « Sac-de-mots continus » (*Continuous Bag-Of-Words, CBOW*) et Skip-Gram. *CBOW* permet de prédire un mot en fonction de son contexte, c'est-à-dire les mots qui l'entourent. A l'inverse, *Skip-Gram* consiste à prédire le contexte en fonction du mot.

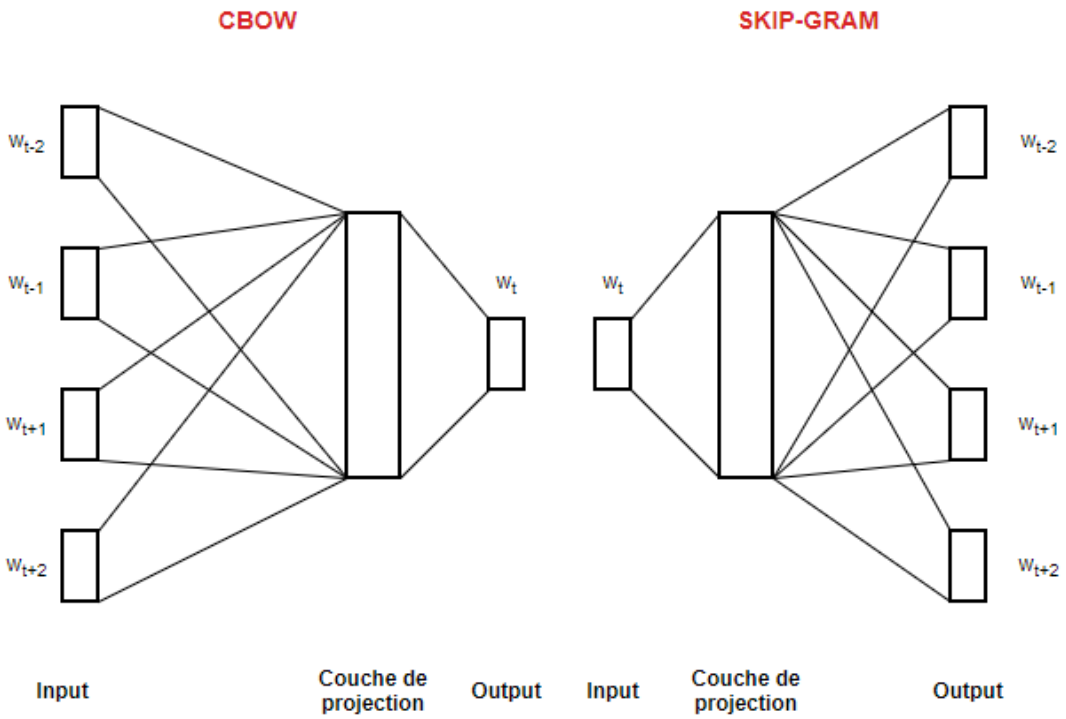

FIGURE 8: L'architecture CBOW et SKIP-GRAM

Bien que *Word2Vec* prouve son efficacité dans plusieurs tâches de TAL, il reste encore des aspects qui ne sont pas pris en compte : l'information portée par le corpus, les mots hors-vocabulaire et l'homonymie. Dans le but de les pallier, d'autres méthodes de plongement sont proposées :

[13] https://code.google.com/p/word2vec/

- GloVe[14] (Pennington et al., 2014) résout le premier problème avec la construction d'une matrice de co-occurrence globale des mots.
- FastText [15](Bojanowski et al., 2016) résout le deuxième problème à l'aide des n-grammes de caractères.
- BERT (Devlin et al., 2018) est capable de produire, pour un mot, des différents *embeddings* en fonction du contexte. En utilisant une partie de l'architecture *Transformer*, un modèle neuronal basé sur le mécanisme d'attention, BERT est considéré aujourd'hui comme un état de l'art en termes de représentation distribuée de mots.

A l'aide des architectures de réseaux de neurones profonds (comme les LSTMs, les CNNs), ces représentations vectorielles compactes de mots peuvent générer des représentations qui sont plus robustes et sont transférables sur plusieurs tâches. On parle dans ce cas des représentations vectorielles de séquences (*sequence* ou *phrase embedding*).

⇨ Nous avons effectué dans cette partie un tour d'horizon de la tâche d'attribution d'auteur. Cette dernière fait intervenir des méthodes relevant à la fois de la stylométrie et de l'apprentissage automatique. Des méthodes stylométriques permettent d'extraire des critères pour servir aux méthodes d'apprentissage supervisé. Les critères exploités dans notre cas sont des mots, des lemmes et des n-grammes de caractères et la méthode d'apprentissage supervisé utilisée est les réseaux de neurones. Nous avons aussi évoqué les concepts de base des réseaux neuronaux vu que la compréhension de ces concepts est importante pour la construction d'un système neuronal.

[14] https://nlp.stanford.edu/projects/glove/
[15] https://fasttext.cc/

3. Méthodologie

Notre méthodologie proposée pour la tâche d'attribution d'auteur est composée en différentes opérations distinctes :

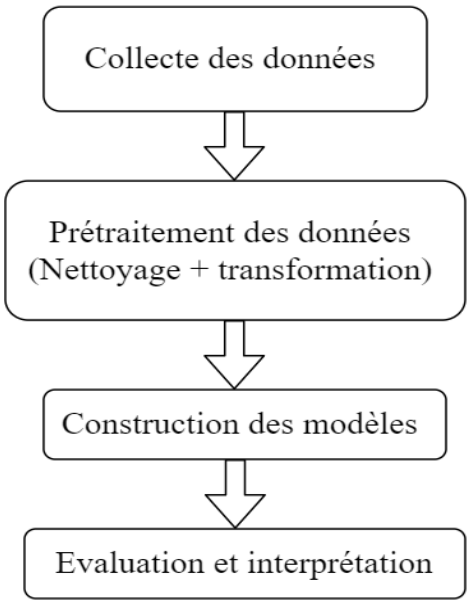

FIGURE 9: Processus de la méthodologie

Avant d'entrer en détail chaque étape, nous présentons des logiciels utilisés pour mener notre étude :

- Google Colab[16] : un outil de Google qui est destiné à la formation et à la recherche dans l'apprentissage automatique. Comme les notebooks Colab exécutent le code sur les serveurs cloud de Google, ils nous permettent d'accéder gratuitement au GPU, qui joue un rôle significatif dans l'implémentation des réseaux de neurones profonds.

- Il existe différentes bibliothèques logicielles (*framework*) d'apprentissage profond telles que : Tensorflow, Pytorch, Theano, Keras, Caffe, etc. Pour notre cas d'étude, nous utilisons Keras [17] (Tensorflow backend). Keras est un framework open-source codé en Python.

[16] https://colab.research.google.com/
[17] https://keras.io/

Développé par François Chollet (Software Engineer de Google) en 2005, il permet de créer facilement et rapidement des réseaux de neurones.

3.1 Corpus

Les données utilisées dans notre protocole proviennent du corpus des tweets de la #présidentielle 2017[18].

Il est vrai que les tweets politiques ne sont pas nécessairement représentatifs les tweets du grand public. Néanmoins, nous constatons que ce type de tweets nous fournissent quand même les caractéristiques appropriées pour que nous puissions voir le potentiel de ce sujet de recherche.

Le corpus #presidentielle 2017 comprend au total 42923 tweets de 11 candidats produits lors des élections de 2017.

Candidats	Nombre de tweets
Benoît Hamon	3815
Emmanuel Macron	3954
François Asselineau	3866
François Fillon	4266
Jacques Cheminade	4516
Jean Lassalle	3664
Jean Luc Mélenchon	5901
Marine Le Pen	4796
N.Dupont Aignan	4026
Nathalie Arthaud	1306
Philippe Poutou	2813

TABLEAU 1: Distribution des tweets par candidat

Les tweets sont généralement assez courts : le tweet le plus court du corpus est constitué seulement 3 caractères le plus long fait 140 caractères et un tweet compte en moyenne 100.2

[18] https://www.ortolang.fr/market/corpora/corpus-presidentielle2017

caractères. Au niveau de mot, le tweet le plus long fait 35 mots et le nombre moyen de mots par tweet est 16.6 mots.

Pour les expériences, l'ensemble du corpus est divisé en trois sous-corpus avec un ratio 60-20-20 :

1) Un corpus d'entraînement qui a pour but d'entraîner le modèle.
2) Un corpus de validation qui sert à estimer le modèle. Il va nous permettre d'ajuster des hyperparamètres pour éviter le phénomène de « surapprentissage » ou de « sous-apprentissage ».

   Le phénomène de « surapprentissage » peut être expliqué lorsqu'un modèle s'adapte bien aux données d'entraînement mais n'est pas capable de généraliser des nouvelles données. Et lorsqu' un modèle s'adapte mal aux données d'entraînement, on parle de « sous-apprentissage ».
3) Un corpus de test qui consiste à évaluer la performance du modèle.

Une fois que le corpus est bien préparé, nous passons à l'étape de prétraitement des données.

3.2 Prétraitement des données

Avant d'être modélisées, les données du corpus passent par une étape de prétraitement. Cette étape joue un rôle considérable à la performance du modèle. Elle comporte des opérations de nettoyage et de transformation des données.

1) Nettoyage des données consiste en :
   - La suppression des liens
   - La suppression des traces des balises HTML telles que : &gt ;, &amp ;, &lt ;…
   - La suppression des caractères de ponctuation.
   - La séparation de la symbole # dans les hashtags. Donc, les hashtags sont considérés comme des mots simples.

Vu que nous voulons garder le plus possible des informations linguistiques et que notre étude porte sur les tweets politiques qui sont plutôt « propres », nous n'y appliquons que les prétraitements dit standards. Pour les autres tweets qui sont plus « bruyants », les autres opérations de nettoyage et de normalisation peuvent être envisagées.

2) Transformation des données consiste à représenter les données sous des formats structurées, compréhensibles par l'algorithme. Pour ce faire, nous sélectionnons tout d'abord des caractéristiques stylométriques pertinents pour les tweets. Une fois le choix de ces caractéristiques est fait, les tweets peuvent être transformés en des formats qui sont exploitables par les systèmes neuronaux.

a) Pour notre étude, nous avons choisi des mots (ou tokens), des n-grammes de caractères et des lemmes comme des caractéristiques stylométriques vu qu'il s'agit des caractéristiques les plus utilisés en attribution d'auteur.

- Au niveau des mots :

Pour simplifier le travail, Keras nous founit un Tokenizer qui sert à segmenter un texte en une série de mots (tokens). Pourtant, il s'agit d'un simple Tokenizer basé sur les espaces. Par conséquent, au lieu de l'utiliser, nous décidons de segmenter les tweets à l'aide de l'outil TreeTagger [19] (Schmid, 1994).

- Au niveau des n-grammes de caractères :

(Shrestha et al., 2017) ont bien montré l'efficacité d'utiliser des n-grammes de caractères comme l'entrée des réseaux de neurones. Dans (Shrestha et al.,2017), ce sont des bigrammes qui donnent les meilleurs résultats. En suivant cette approche, nous segmentons les tweets en des bigrammes de caractères.

- Au niveau des lemmes :

Des lemmes sont aussi les traits souvent utilisés en attribution d'auteur (Savoy, 2012). Bien que ces traits soient considérés plus performants pour les textes longs, le grand corpus, nous supposons qu'ils restent quand même un choix correct pour les textes courts vu qu'il y a peu de redondance de mots dans ce type de texte. Ainsi, nous choisissons aussi l'outil TreeTagger pour lemmatiser le corpus.

Après avoir réalisé la segmentation des tweets, nous utilisons la classe Tokenizer fournie par Keras pour les convertir en des séries de nombre entiers. En considérant que plus les mots sont rares, moins ils sont importants, nous ne conservons que 30000 mots les plus fréquents. Pour les mots qui ne se trouvent pas dans le dictionnaire, ils vont se remplacer par « OOV ».

[19] https://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

Ensuite, nous passons à l'étape de représenter les tweets en des vecteurs exploitables par les réseaux de neurones.

b) Représentation des données :

Les textes en général peuvent être transformés en vecteurs en utilisant, le plus souvent, la pondération tf-idf ou la fréquence. Ces techniques s'adaptent bien aux textes longs vu que ce type de textes est généralement riche en vocabulaire et en contenu. Ainsi, la sémantique de ces textes peut être capturée en s'appuyant sur la fréquence des mots. Cependant, quant aux textes courts, comme les tweets, ces techniques ne fonctionnent pas aussi bien.

Pour notre étude, nous utilisons la technique de représentation *Word Embeddings* (cf. section 2.3.5). Cette représentation sera apprise dans la première couche d'un réseau de neurones profonds, qui est décrit plus en détail dans la section suivante consacrée à la construction des réseaux de neurones.

3.3 Construction des réseaux de neurones

Nous construisons, dans nos expérimentations, deux modèles de réseaux de neurones les plus populaires pour la tâche de classification en général et en particulier celle d'attribution d'auteur : les réseaux de neurones convolutionnels (CNN) et les réseaux à mémoire court et long terme (LSTM).

Pour chacun des modèles, nous présentons les architectures ainsi que les hyperparamètres contribuant à sa phase d'entraînement.

3.3.1 Réseaux de neurones convolutionnels (CNN)

3.3.1.1 Architecture des réseaux

Nous proposons ici deux architectures de CNN, notées respectivement CNN-1, CNN-2. La différence entre ces deux architectures est vraiment légère :

a) La première architecture constitue une architecture dite « traditionnelle » de CNN. Elle est composée d'une couche de convolution suivie d'une couche de max-pooling. La sortie de ce bloc convolutif passe ensuite à une couche entièrement connectée, une couche *Dropout* et une dernière couche entièrement connectée associée à la fonction d'activation *Softmax.*

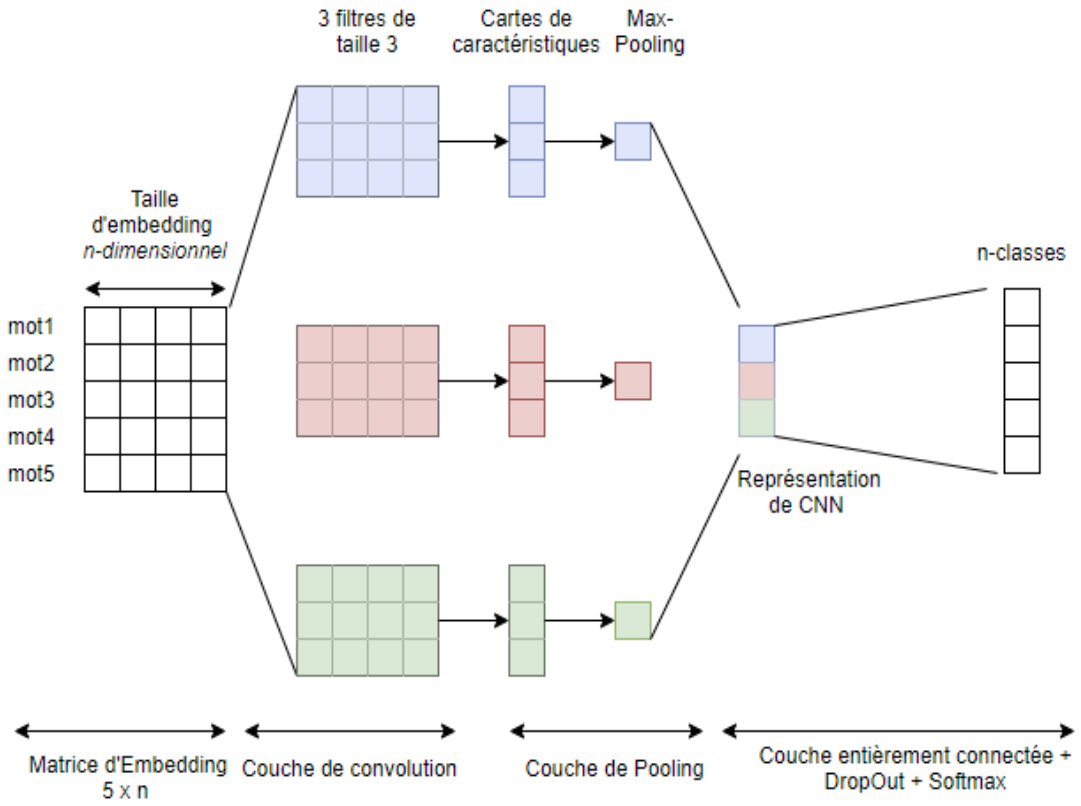

FIGURE 10: Illustration du modèle CNN-1 au niveau des mots

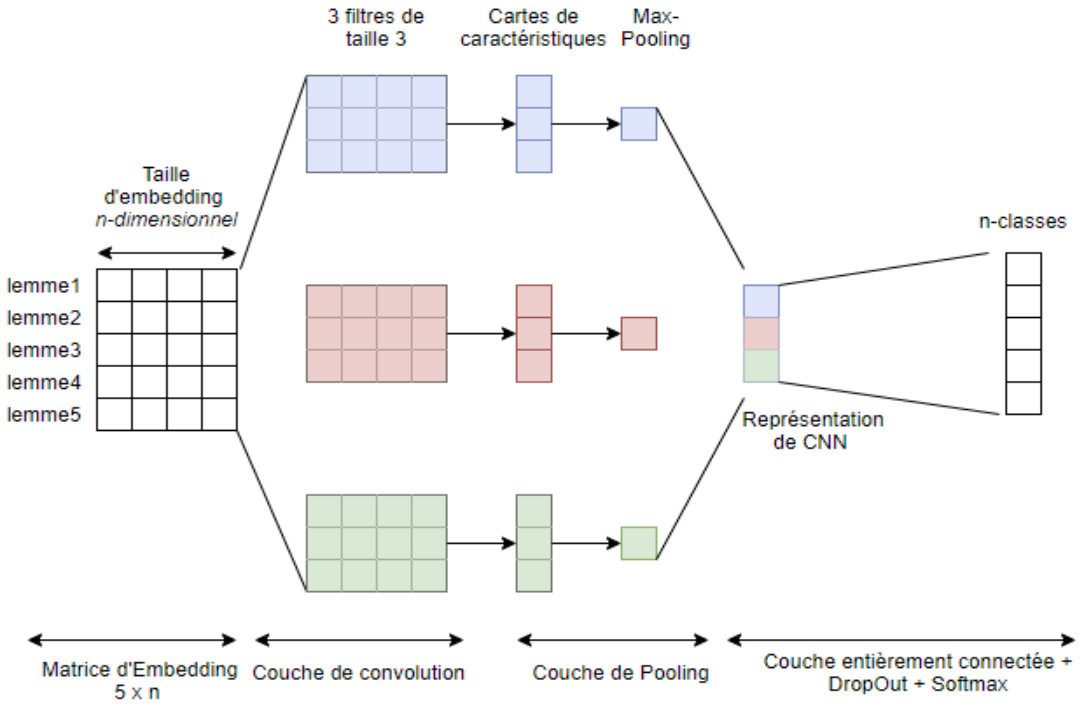

FIGURE 11: Illustration du modèle CNN-1 au niveau des lemmes

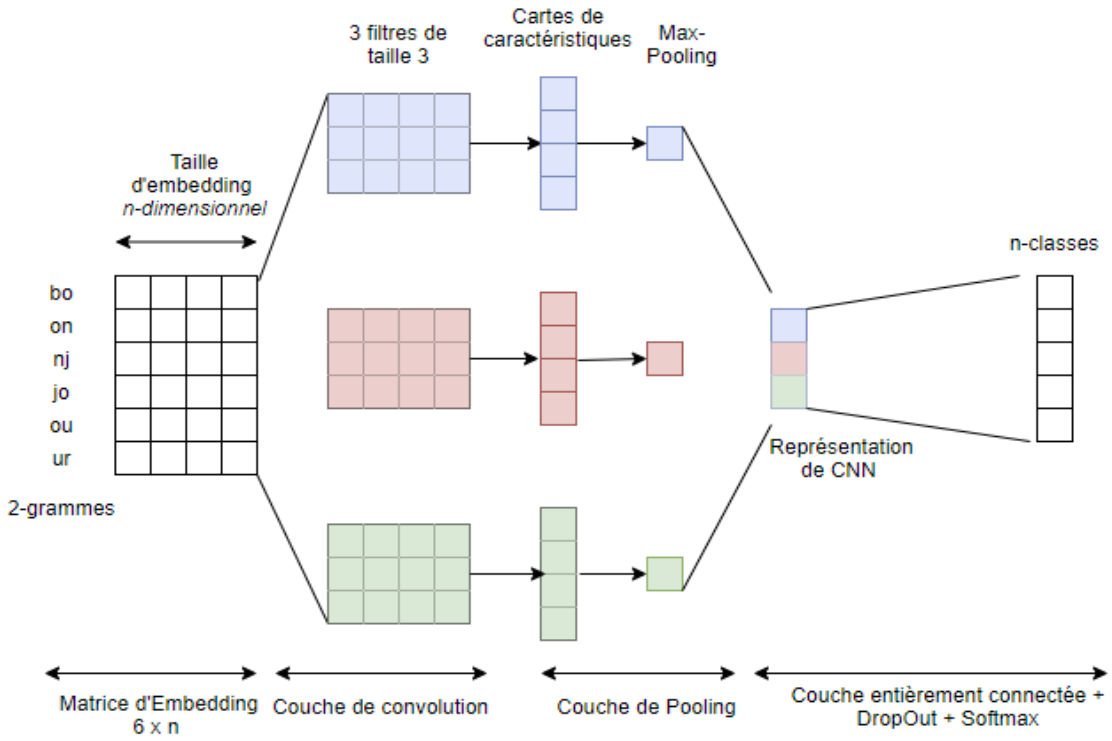

FIGURE 12: Illustration du modèle CNN-1 au niveau des bigrammes

b) La deuxième architecture s'inspire des travaux de (Kim,2014). Au lieu d'utiliser une seule couche de convolution, Kim utilise 3 couches de convolutions mises en parallèle avec des filtres de taille 3,4 et 5 (100 filtres pour chaque taille). Le reste du réseau reste inchangé en comparant au premier.

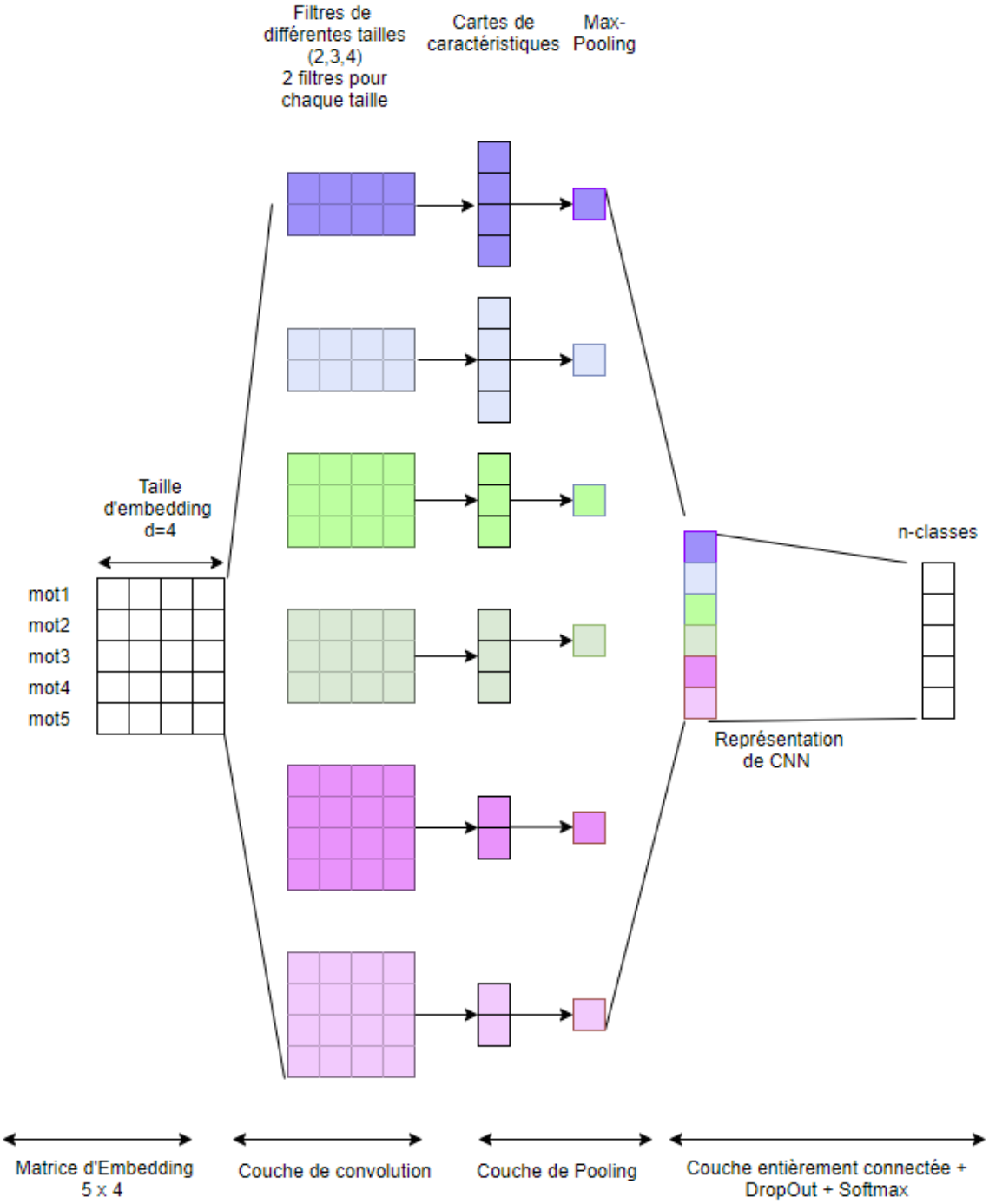

Figure 13: Illustration du modèle CNN-2 avec les filtres de taille (2,3,4), dont 2 filtres pour chaque taille.

Quelle que soit l'architecture, le principe de fonctionnement est le même :

- Le réseau prend en entrée une martrice Embedding représentant la phrase. Cette matrice « convolue » avec les filtres et produit donc pour chaque filtre, une carte de caractéristique (*feature map*). Ensuite, une opération de pooling s'applique à chaque *feature map* pour retirer seulement sa composition maximale. Toutes ces valeurs sont concaténées pour former un vecteur de taille *x* qui correspond au nombre total de filtres choisis. Ce vecteur peut être vue comme une représentation de la phrase dans son ensemble. Finalement, ce vecteur passe à des couches entièrement connectées pour produire le résultat.

Une fois l'architecture définie, nous passons ensuite à l'étape de choisir des hyperparamètres.

3.3.1.2 Hyperparamètres des réseaux

Les hyperparamètres sont des paramètres dont la valeur doit être définie avant le processus d'entraînement du modèle. Choisir des hyperparamètres ainsi que définir leurs valeurs constitue une étape importante parce qu'elle a un impact significatif sur les résultats. Il existe plusieurs hyperparamètres à configurer mais nous détaillons seulement ceux utilisés au cours de nos expérimentations. Deux types d'hyperparamètres sont distingués : hyperparamètres du modèle et hyperparamètres d'entraînement.

a) Hyperparamètres du modèle comprend des hyperparamètres relatifs à l'architecture du modèle.

Différents hyperparamètres sont définis pour chaque couche du modèle :

- Couche d'Embedding :

  Ses hyperparamètres obligatoires sont la taille de vocabulaire et la taille d'Embedding, c'est-à-dire la dimension des représentations vectorielles de mots.

  En plus, au lieu d'initialiser cette couche par des valeurs aléatoires, nous l'initialisons par un modèle *Word Embeddings* pré-entraîné et ces embeddings seront par la suite mis à jour lors de l'entraînement du modèle. Dans notre cas d'étude, nous avons recours à des *Word Embeddings* pré-entraînés de FastText[20]. Comme la dimension de ces *Word Embeddings* est fixée à 300, la dimension d'Embedding doit être aussi fixée à 300.

[20] https://fasttext.cc/docs/en/crawl-vectors.html

- Couche de Convolution :

  Dans cette couche, le nombre de filtres et la taille de filtre sont des hyperparamètres obligatoires. Dans le but de pallier le phénomène de « surapprentissage », nous définissons en plus un hyperparamètre de régularisation l_2 qui consiste à rajouter une pénalité à la fonction de coût.

- Couche entièrement connectée : le nombre de neurones.

- Couche Dropout : la probabilité de Dropout.

  Comme la régularisation l_2, Dropout est aussi une technique permettant de prévenir le surapprentissage. Développée par (Srivastava et al.,2014), cette technique consiste à abandonner, « désactiver » aléatoirement à chaque itération un certain pourcentage des neurones d'une couche.

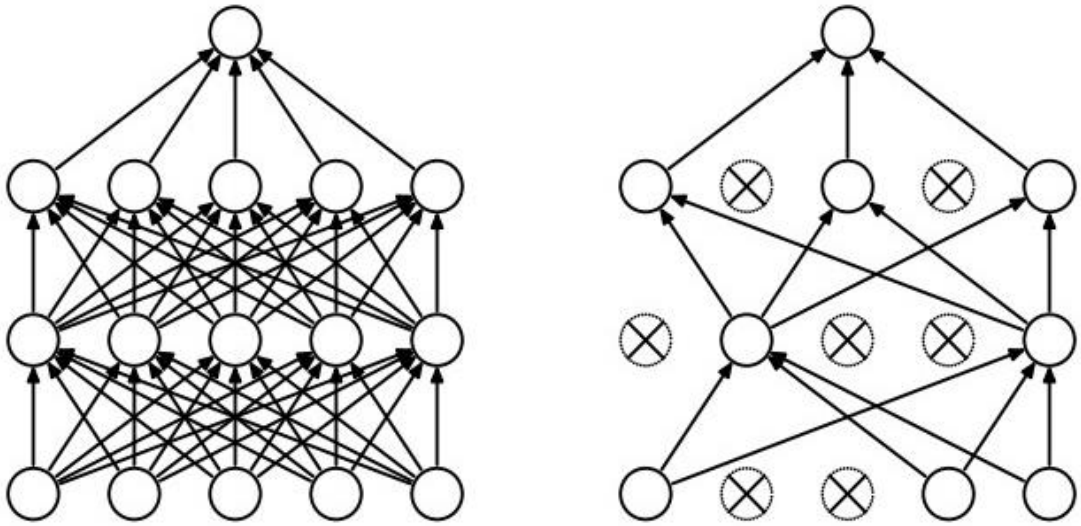

FIGURE 14 : Illustration de la technique Dropout. A droite le réseau entièrement connecté. A gauche, certains neurones du réseau ont été aléatoirement désactivés. Figure extraite de (Srivastava et al., 2014)

b) Hyperparamètres d'entraînement comprend des hyperparamètres relatifs à l'entraînement du modèle :

- Le taux d'apprentissage (*Learning Rate)* détermine la vitesse de convergence de l'algorithme. Si sa valeur est trop petite, la convergence sera lente. Au contraire si on lui donne une valeur plus élevée pour accélérer l'apprentissage, il est probable que l'algorithme ne peut pas converger.
- Le nombre d'époques correspond le nombre de fois que toutes les données d'entraînement sont passées dans le réseau pendant l'entraînement.
- La taille du batch (*Batch size*) : Dans le cas où la taille de données d'entraînement est trop importante, il est impossible de les entrer une seule fois dans une époque. Ainsi, il est nécessaire de les partitionner en des batchs. Le nombre de données dans un batch correspond à la taille du batch. La taille optimale dépend de plusieurs facteurs, dont la capacité de l'ordinateur.
- Afin de réduire le « sur-apprentissage », nous avons eu recours une technique très utilisée, à savoir « l'arrêt prématuré » (*Early Stopping*) (figure 15). Cette technique va permet d'arrêter l'apprentissage du réseau dès que l'erreur de validation (*loss validation*) commence à augmenter (ou lorsque le réseau perd en performance de généralisation). Nous définissons une patience de 3, c'est-à-dire que l'apprentissage est arrêté lorsqu'il n'y a pas d'amélioration de l'erreur de validation au bout de 3 époques.

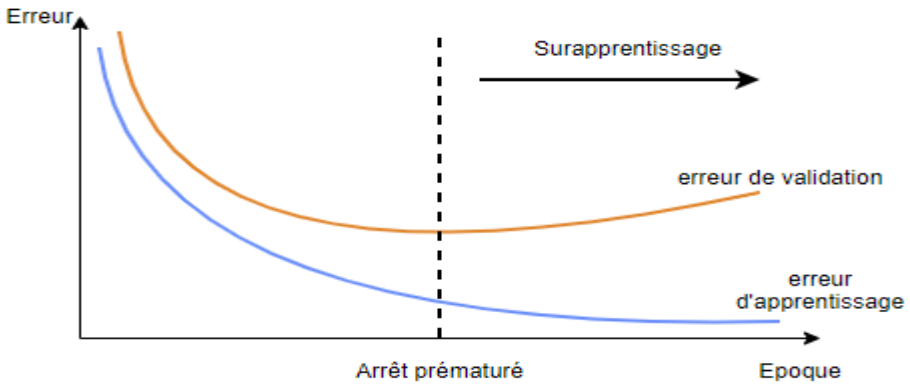

FIGURE 15 : Arrêt prématuré de l'apprentissage

Avant de passer l'étape d'entraînement, nous récapitulons les hyperparamètres de nos réseaux CNN :

Hyperparamètres du modèle	
Couche d'Embedding	Taille de vocabulaire et taille d'embedding
Couche de Convolution	Taille de filtres + nombre de filtres + régularisation L2
Couche entièrement connectée	Nombre de neurones
Couche Dropout	Probabilité de Dropout

Hyperparamètres d'entraînement
Taux d'apprentissage
Le nombre d'époques
La taille du batch
L'arrêt prématuré

TABLEAU 2 : Récapitulatif des hyperparamètres du réseau CNN

3.3.1.3 Entraînement des réseaux

Dans nos expériences, l'apprentissage du modèle se fait par Adam, un des algorithmes les plus utilisés pour l'optimisation par descente du gradient. Il existe d'autres d'optimiseurs[21] avec différents mécanismes de fonctionnement, tels que : Adagrad, RSMProp , Adadelta, etc,. Nous choisissons ensuite la fonction d'entropie croisée catégorielle (*categorical cross-entropy*) comme fonction de coût parce qu'elle adapte au problème de classification multi-classe.

Nous testons les différentes valeurs pour chaque hyperparamètre :

	CNN-1	**CNN-2**
Nombre de filtres	50-300 avec un pas à 50	50-200 avec un pas à 50
Taille de filtres	1-5	(1,2,3), (2,3,4) et (3,4,5)
L_2 **(valeur lambda)**	0 ; 0,1 ; 0,01 ; 0,001 ; 0,0001 ; 0,00001 et 0,000001	0 ; 0,1 ; 0,01 ; 0,001 ; 0,0001 ; 0,00001 et 0,000001
Nombre de neurones (Couche entièrement connectée)	64-256 avec un pas à 32	64-256 avec un pas à 32
Dropout	0,2-0,5	0,2-0,5

[21] https://keras.io/api/optimizers/

Taux d'apprentissage	0,0001 ; 0,0005 ; 0,001 ; 0,005 ; 0,01	0,0001 ; 0,0005 ; 0,001 ; 0,005 ; 0,01

TABLEAU 3: Les différentes valeurs des hyperparamètres de CNN

Les valeurs optimales de ces hyperparamètres sont trouvées à l'aide d'une méthode d'optimisation Hyberband[22]. La taille de batch est obtenue après bon nombre d'essais. Les configurations finales de nos modèles sont définies comme suit :

	CNN-1	**CNN-2**
Couche d'Embedding	Word Embedding pré-entraîné	Word Embedding pré-entraîné
Nombre de filtres	300	100 filtres pour chaque taille
Taille de filtres	3	(2,3,4)
L_2 **(valeur lambda)**	0,00001	None
Nombre de neurones (Couche entièrement connectée)	128	128
Dropout	0,5	0,5
Taux d'apprentissage	0,0005	0,001
Nombre d'époques	20	20
Arrêt prématuré	Patience = 3	Patience = 3
Taille de batch	2000	2000

TABLEAU 4 : La configuration finale du modèle CNN

[22] **L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh & A. Talwalkar**. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. In *JMLR*, 18(185), 1−52, 2018.

3.3.2 Réseaux récurrents à mémoire court et long terme (LSTM)

3.3.2.1 Architecture des réseaux

Nous testons deux types de réseaux LSTM : un réseau unidirectionnel et un réseau bidirectionnel

a) Réseau unidirectionnel :

L'entrée du réseau est une séquence de mots {mot_1,…..,mot_n} où chaque mot est représenté sous forme de vecteur. Le réseau LSTM va parcourir cette séquence, de gauche à droite, un mot après l'autre. Plus concrètement, à chaque pas de temps *t*, le $t^{ème}$ cellule combine l'entrée courante w_t ainsi que la sortie du pas précédent h_{t-1} pour produire une sortie h_t. Cette sortie contient donc l'information du mot courant ainsi que celle des mots précédents. De cette manière, la sortie du dernier mot de la séquence va contenir toutes les informations de la séquence. Cette sortie va passer ensuite à des couches entièrement connectées pour obtenir le résultat final.

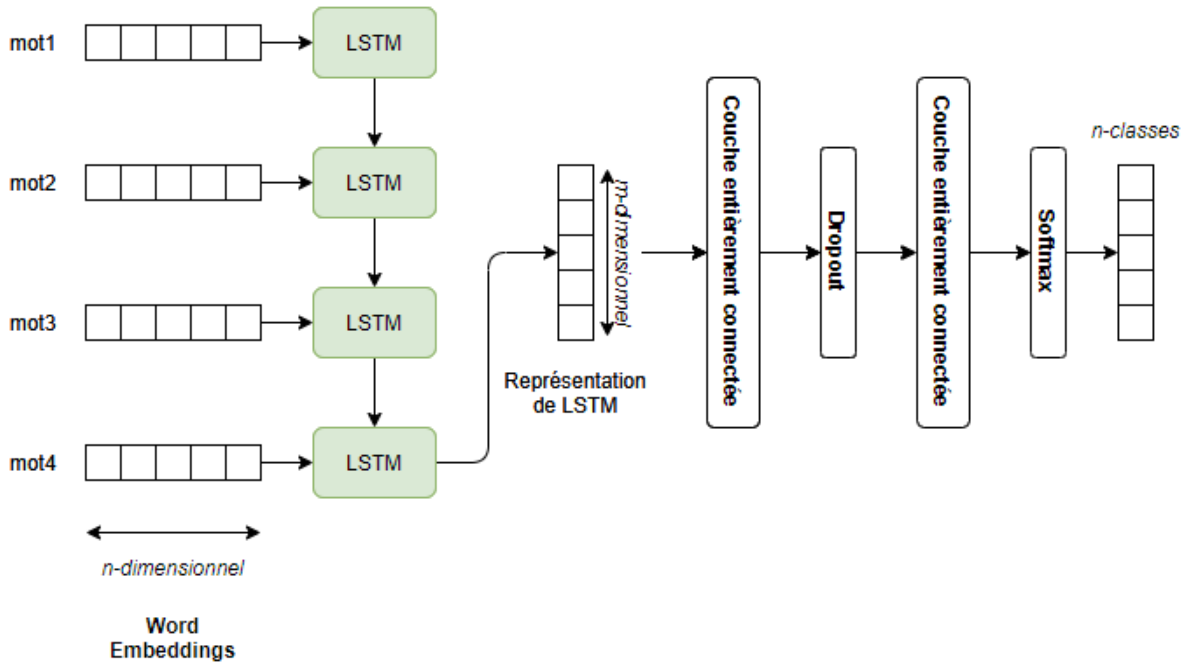

FIGURE 16: Illustration du modèle LSTM-1 pour la tâche de classification

En supposant que le dernier vecteur puisse oublier les informations pertinentes, particulièrement ceux qui sont anciennes, nous définissons une variante de réseau LSTM, notée LSTM-2. Au lieu de prendre seulement le vecteur du dernier état caché (*last hidden state*), nous prenons tous les vecteurs des états cachés et les faisons entrer dans la couche Global Max Pooling.

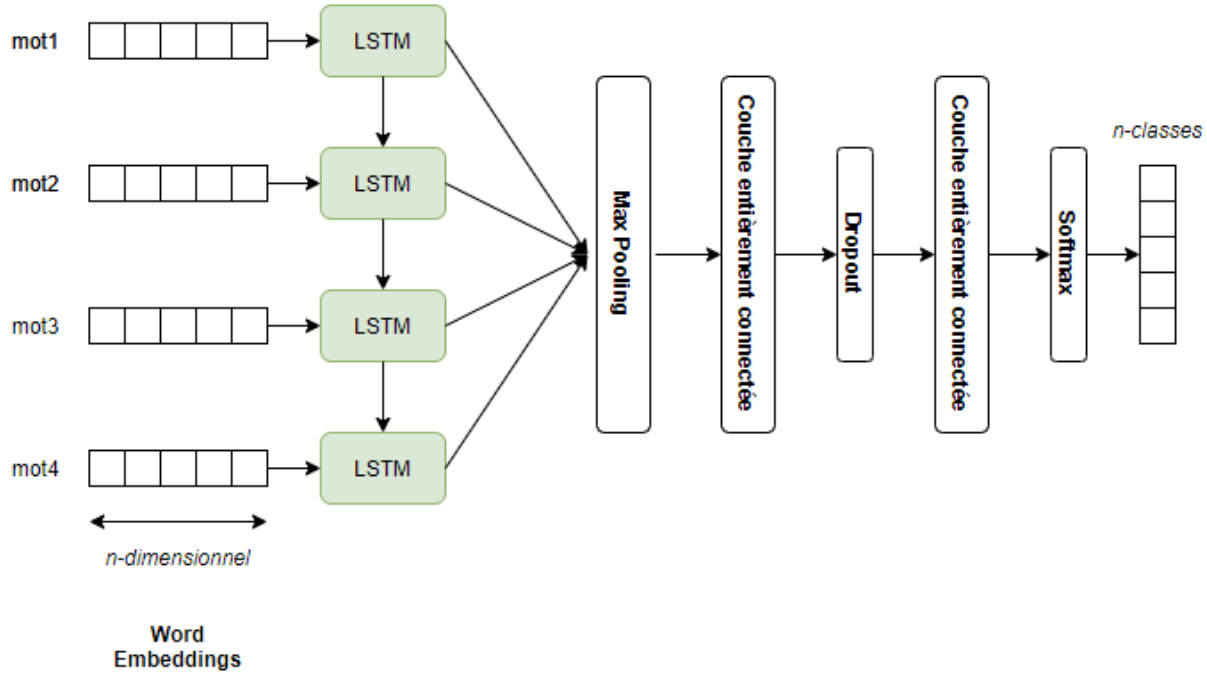

FIGURE 17 : Illustration du modèle LSTM-2

b) Réseau LSTM bidirectionnel :

Dans le cas d'un réseau LSTM unidirectionnel, il prend en compte seulement le contexte précédent lors du traitement de l'élément courant dans une séquence. Cependant, dans certains problèmes, il est utile de prendre en compte aussi le contexte futur. Prenons par exemple le cas de prédiction du mot *« est »* dans la séquence « *il est parti »*. Le contexte précédent *« il »* aide à prédire le mot suivant qui est plutôt *« est »* que d'autres formes de conjugaison *suis* ou *sommes*. En plus, le contexte futur « *parti »* peut aussi contribuer à la prédiction du mot *« est »*. Il permet de choisir le verbe « *être »* plutôt que le verbe *« avoir »*. Et le LSTM bidirectionnel est capable de prendre en compte à la fois le contexte passé et le contexte futur. A l'instar du LSTM unidirectionnel, un LSTM bidirectionnel a aussi deux manières de sorties : soit il retourne seulement le vecteur final, noté BiLSTM-1 ; soit il retourne tous les vecteurs intermédiaires, noté BiLSTM-2.

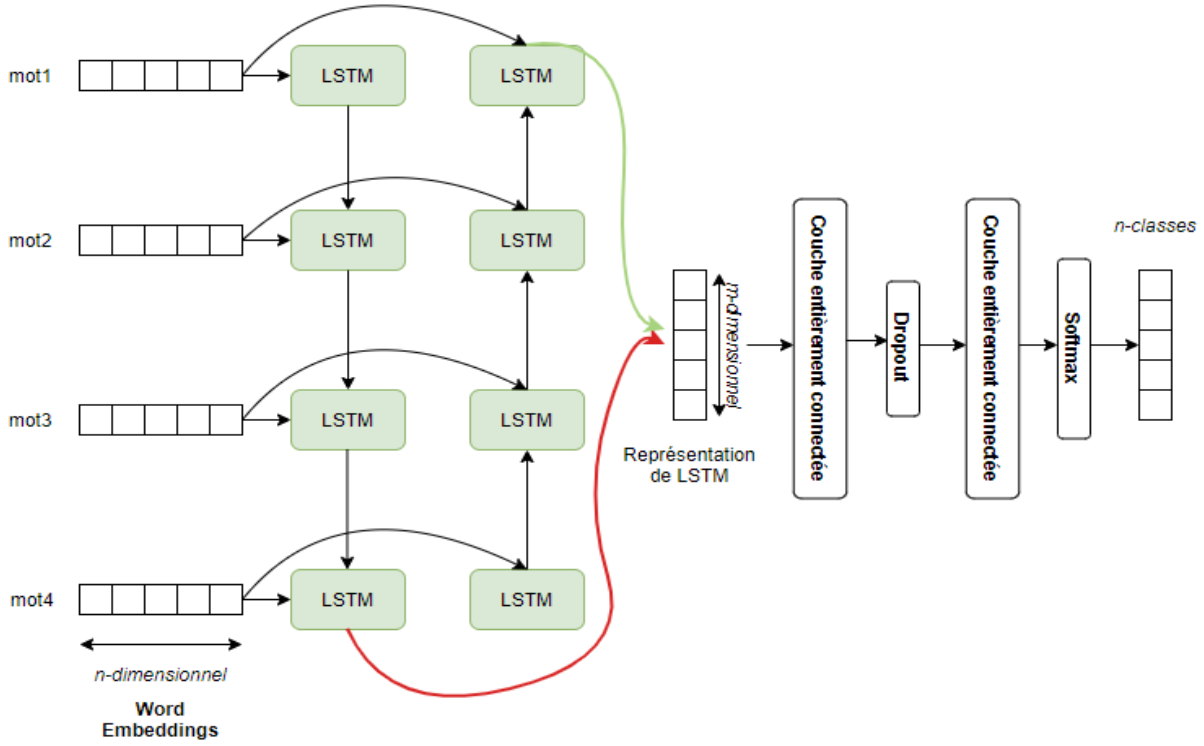

FIGURE 18 : Illustration du modèle BiLSTM-1

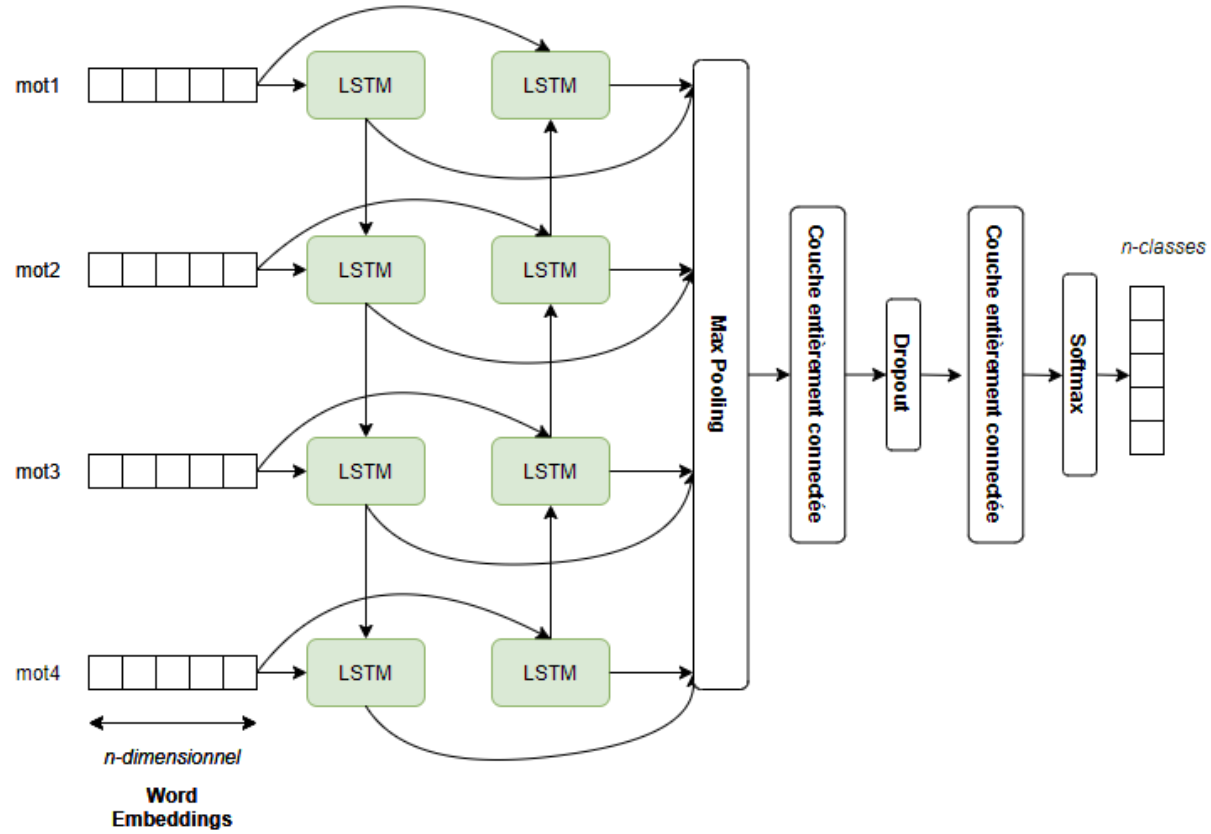

FIGURE 19 : Illustration du modèle BiLSTM-2

3.3.2.2 Hyperparamètres des réseaux

Avant de passer à la phase d'entraînement, il faut définir préalablement les hyperparamètres des réseaux. Comme les CNNs, les LSTMs ont aussi deux types d'hyperparamètres.

a) Hyperparamètres du modèle :

Comme illustrées les Figures 16, 17, 18, 19, le réseau LSTM est composé aussi d'une couche d'Embedding, d'une couche Dropout et des couches entièrement connectées. Par conséquent, leurs hyperparamètres sont les mêmes que ceux définis dans le réseau CNN.

Quant à la couche LSTM, l'hyperparamètre est le nombre de neurones, ou simplement la taille du vecteur en sortie.

b) Hyperparamètres de l'entraînement : le taux d'apprentissage, le nombre d'époques et la taille du batch.

Hyperparamètres du modèle	
Couche d'Embedding	Taille de vocabulaire et taille d'embedding
Couche de LSTM	Taille du vecteur en sortie
Couche entièrement connectée	Nombre de neurones
Couche Dropout	Probabilité de Dropout

Hyperparamètres d'entraînement
Taux d'apprentissage
Le nombre d'époques
La taille du batch
L'arrêt prématuré

TABLEAU 5: Récapitulatif des hyperparamètres du réseau LSTM

3.3.2.3 Entraînement des réseaux

L'apprentissage des modèles LSTMs se fait aussi par l'optimiseur Adam et la fonction de coût est la fonction d'entropie croisée catégorielle (*categorical cross-entropy*).

Afin de garantir une comparaison équitable entre les modèles CNNs et les modèles LSTMs, nous fixons préalablement le nombre d'époques à 20 et la taille de batch à 2000. Ensuite, nous testons les différentes valeurs pour les hyperparamètres qui restent :

	LSTM	Bi-LSTM
Couche LSTM	50-200 avec un pas à 50	50,100
Nombre de neurones (Couche entièrement connectée)	64-256 avec un pas à 32	64-256 avec un pas à 32
Dropout	0,2-0,5	0,2-0,5
Taux d'apprentissage	0,0001 ; 0,0005 ; 0,001 ; 0,005 ; 0,01	0,0001 ; 0,0005 ; 0,001 ; 0,005 ; 0,01

TABLEAU 6: Les différentes valeurs des hyperparamètres de LSTM

Les valeurs optimales de ces hyperparamètres sont aussi trouvées à l'aide d'une méthode d'optimisation Hypberband. Les configurations finales de nos modèles sont définies comme suit :

	LSTM	Bi-LSTM
Couche d'Embedding	Word Embedding pré-entraîné	Word Embedding pré-entraîné
Couche LSTM	150	200
Nombre de neurones (Couche entièrement connectée)	96	64
Dropout	0,5	0,5
Taux d'apprentissage	0,01	0,01
Nombre d'époques	20	20
Arrêt prématuré	Patience = 3	Patience = 3
Taille de batch	2000	2000

TABLEAU 7: La configuration finale du modèle LSTM

4. Résultats et Discussion

Afin d'évaluer la performance de nos modèles, nous les appliquons au jeu de données de test qui contient 8456 tweets.

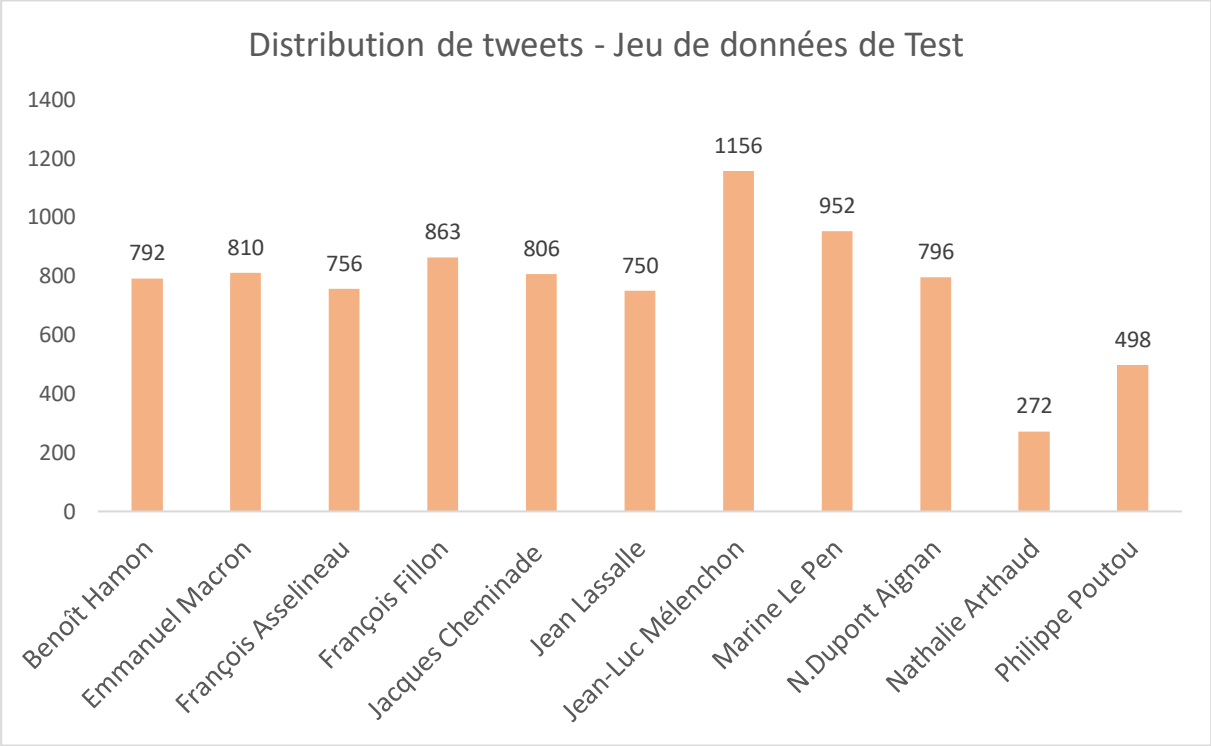

FIGURE 20: Distribution de tweets du jeu de données de Test

Les résultats obtenus sont évalués en termes d'*Exactitude*, *macro_précision*, *macro_rappel* et *macro_fmesure*.

$$Exactitude = \frac{\sum_i nb\ de\ documents\ correctement\ attribués\ à\ la\ classe\ i}{nb\ de\ documents}$$

La Précision mesure le nombre d'éléments correctement étiquetés par le système (Vrais Positifs) par rapport au nombre total d'éléments étiquetés par le système (Vrai Positifs + Faux Positifs). Elle évalue la capacité du système à refuser tous les documents non-pertinents.

$$precision_i = \frac{vrai\_positif_i}{vrai\_positif_i + faux\_positif_i}$$

$$macro\_precision = \frac{1}{N} \sum_{i=1}^{N} precision_i$$

Le Rappel mesure le nombre d'éléments correctement étiquetés par le système (Vrais positifs) par rapport au nombre d'éléments étiquetés dans la référence (Vrais Positifs + Faux Négatifs). Elle évalue la capacité du système à sélectionner tous les documents pertinents.

$$rappel_i = \frac{vrai\_positif_i}{vrai\_positif_i + faux\_négatif_i}$$

$$macro\_rappel = \frac{1}{N} \sum_{i=1}^{N} rappel_i$$

La F-mesure est la moyenne harmonique de la précision et du rappel. Elle donne un score de performance global du système.

$$fmesure_i = \frac{2 * rappel_i * precision_i}{rappel_i + precision_i}$$

$$macro\_fmesure = \frac{1}{N} \sum_{i=1}^{N} fmesure_i$$

Dans le cas de notre étude, nous avons fait beaucoup d'expériences en faisant varier non seulement les modèles mais aussi les types de traits à l'entrée du réseau : mots, n-grammes de caractères et lemmes. Pourtant, les résultats obtenus au niveau des lemmes et les résultats obtenus au niveau des n-grammes de caractères ne sont pas vraiment pertinents. Ainsi, dans cette section, nous nous concentrons seulement sur l'analyse des résultats obtenus au niveau des mots (tokens).

	CNN-1	**CNN-2**	**LSTM-1**	**LSTM-2**	**BiLSTM-1**	**BiLSTM-2**
Tokens	0.81	**0.83**	0.73	0.78	0.77	0.81
N-grammes de caractères	0.75	0.81	0.53	0.78	0.72	0.79
Lemmes	0.69	0.70	0.63	0.67	0.66	0.69

TABLEAU 8: Les résultats obtenus au niveau des mots, des n-grammes de caractères et des lemmes en termes de taux d'exactitude.

Analyse des résultats obtenus au niveau des mots :

	Exactitude	Précision	Rappel	F-Mesure
CNN-1	0.81	0.82	0.80	0.81
CNN-2	**0.83**	0.82	0.82	0.83
LSTM-1	0.73	0.73	0.72	0.72
LSTM-2	0.78	0.77	0.77	0.77
BiLSTM-1	0.77	0.76	0.76	0.76
BiLSTM-2	**0.81**	0.80	0.79	0.80

TABLEAU 9 : Les résultats de six modèles en attribution d'auteur des tweets en termes de taux d'Exactitude, Précision, Rappel et F-mesure au niveau des mots

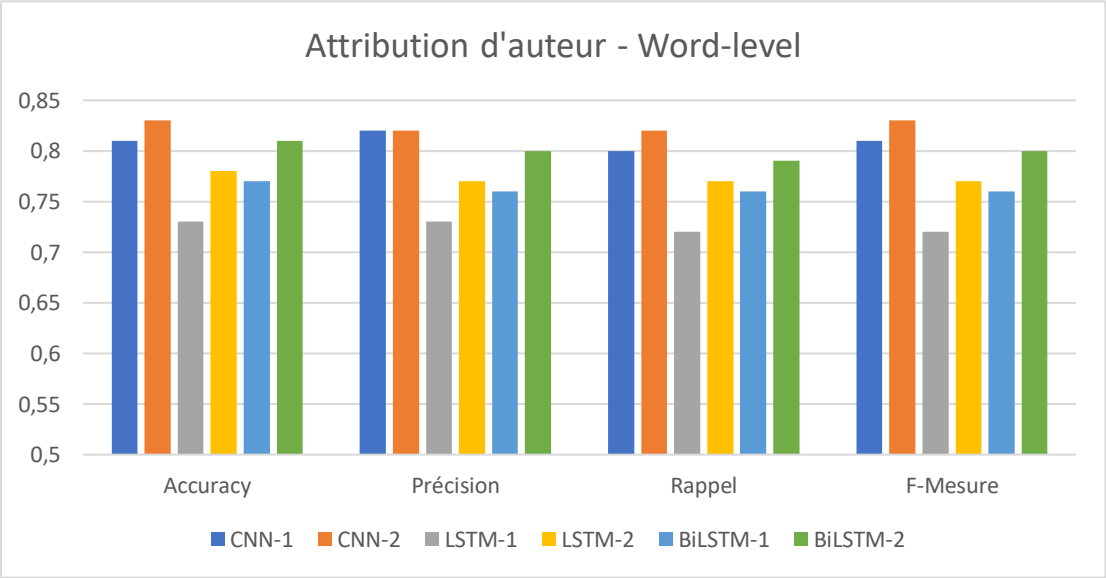

FIGURE 21: Comparaison de six modèles pour l'attribution d'auteur des tweets au niveau des mots.

Le tableau 9 et la figure 21 nous montrent que parmi les modèles proposés, le modèle CNN-2, le modèle CNN avec 3 couches de convolution mises en parallèle, est le plus performant avec un taux d'exactitude de **83%,** une F-mesure de 83%**.** Néanmoins, les résultats obtenus par le modèle CNN-1 sont aussi très remarquables. Étant un simple CNN constitué d'une seule couche de convolution, nous observons tout de même des bons résultats avec un taux d'exactitude de **81%**.

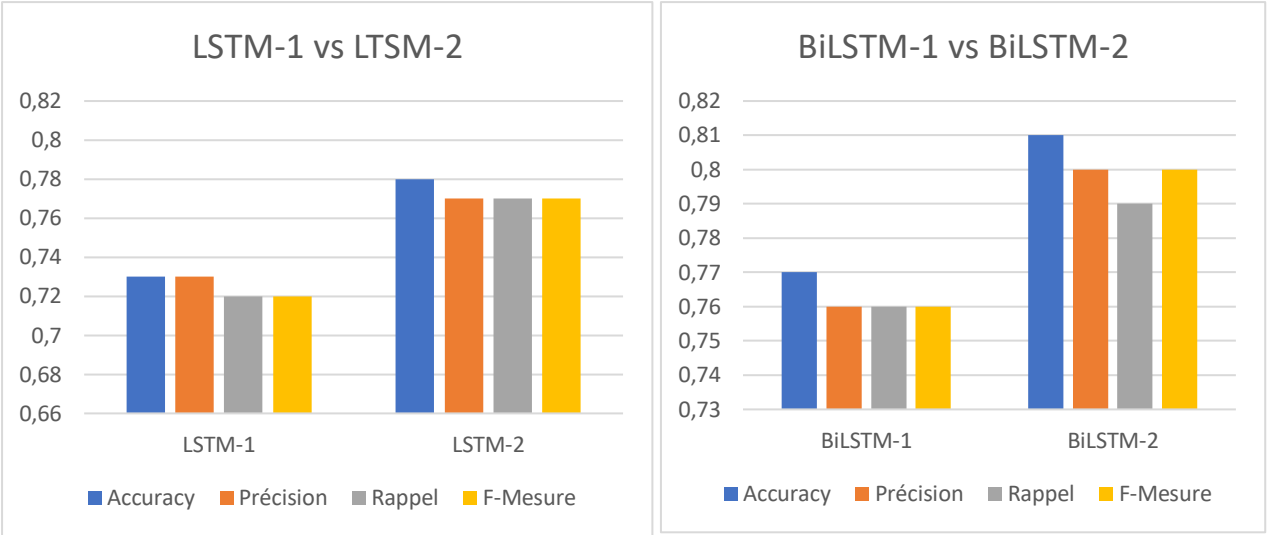

FIGURE 22: Comparaison des quatre architectures du réseau LSTM

Avec une architecture spécifiée au traitement des données séquentielles, au début, nous portions quelques espoirs sur la robustesse des réseaux LSTMs. Néanmoins, les résultats obtenus par ce réseau ne sont pas très satisfaisants. Le taux d'exactitude de LSTM-1, LSTM-2, BiLSTM-1, BiLSTM-2 sont, respectivement de 73%, 78%, 77% et 81%. A partir de ces résultats, nous constatons que l'approche d'utiliser tous les états cachés (LSTM-2 et BiLSTM-2) donne les meilleurs résultats que celle d'utiliser seulement le dernier état caché (LSTM-1 et BiLSTM-1). En plus, les LSTMs bidirectionnels donnent les résultats plus élevés que les LSTMs unidirectionnels.

⇨ Parmi les six modèles, nous choisissons seulement le modèle CNN-2 pour faire une analyse plus détaillée.

Analyse des résultats obtenus de modèle CNN-2 :

La figure 23 montre la F-mesure pour chaque chasse. Ici, nous constatons que le modèle a bien reconnu Jacques Cheminade et Jean-Luc Mélenchon avec les F-mesures de 90%. Viennent ensuite les tweets de François Asselineau (89%), Nicolas Dupont-Aignan (87%), Jean Lassalle (84%) et Marine Le Pen (84%). En revanche, le modèle a des difficultés à classer les tweets de Philippe Poutou, François Fillon, Nathalie Arthaud, Benoît Hamon et Emmanuel Macron avec les F-mesures, respectivement, de 78%, 78%, 77%, 75% et 73%. Les faibles F-mesures de

Philippe Poutou et de Nathalie Arthaud peuvent s'expliquer par le problème de déséquilibre des classes. On parle de ce problème quand les catégories de classification (ou classes) ne sont pas représentées de manière égale dans le jeu de données (cf. TABLEAU 1). En regardant le tableau 1, nous pouvons remarquer qu'il y a un décalage entre le nombre des tweets de ces candidats (Nathalie Arthaud : 1306 tweets et Philippe Poutou : 2813 tweets) avec celui d'autres candidats. Le déséquilibre de classes est un problème qu'on rencontre le plus souvent dans l'apprentissage automatique et cela résulte peut-être d'un biais d'apprentissage.

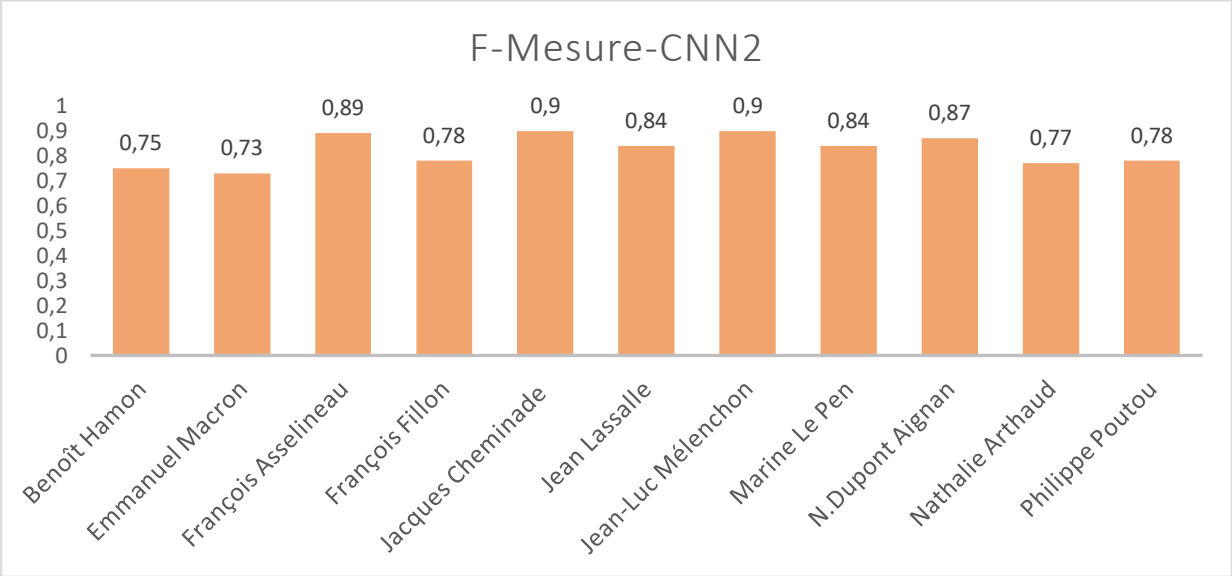

FIGURE 23 : F-Mesure pour chaque classe

Pour avoir une idée générale sur la classification des données du corpus de test, nous les avons projeté dans une espace de deux dimensions à l'aide de t-SNE (*t-Distributed Stochastic Neighbor Embedding)*[23]. Il s'agit d'une technique de réduction de dimension qui permet de projeter des données à haute dimension dans un espace de deux ou trois dimensions.

[23] **L.J.P. van der Maaten & G.E. Hinton**. Visualizing High-Dimensional Data Using t-SNE. In *Journal of Machine Learning Research*, 9(Nov):2579-2605, 2008.

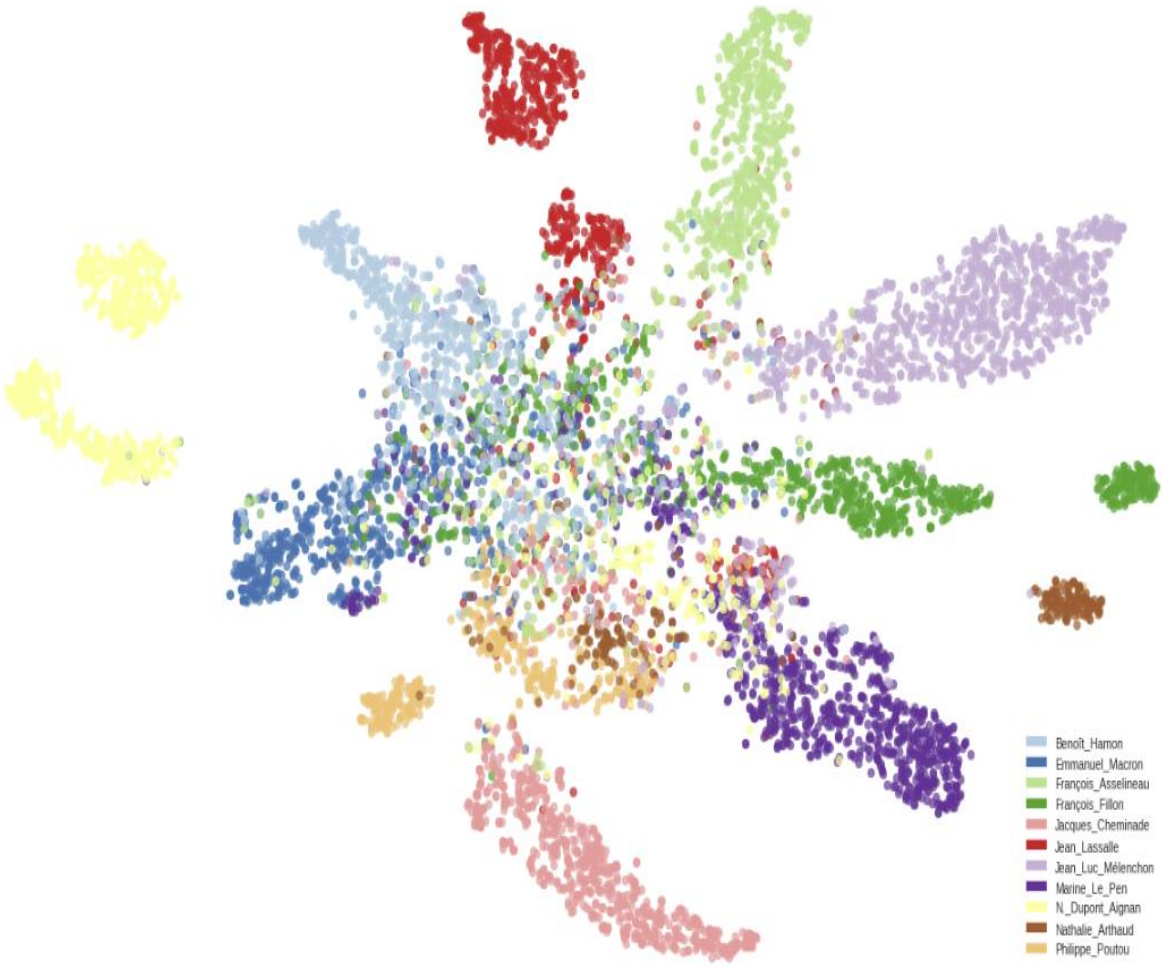

FIGURE 24: Visualisation du regroupement des tweets du corpus de test

Comme le montre la figure 23, la figure 24 nous montre aussi que les tweets de Jacques Cheminade, Jean-Luc Mélenchon, François Asselineau, Nicolas Dupont-Aignan et Jean Lassalle sont bien regroupés. Dans le cas de Nathalie Arthaud, il y a un échantillon de ses tweets sont reconnus, par l'algorithme, proches des tweets de Phillipe Poutou. De même, il y un échantillon des tweets de Nicolas Dupont-Aignan se trouvent proches des tweets de Marine Le Pen. En plus, nous remarquons que le modèle rencontre des difficultés dans la distinction des tweets de Benoît Hamon, Emmanuel Macron et François Fillon. La confusion entre les tweets de Marine Le Pen et ceux de François Fillon est aussi remarquable.

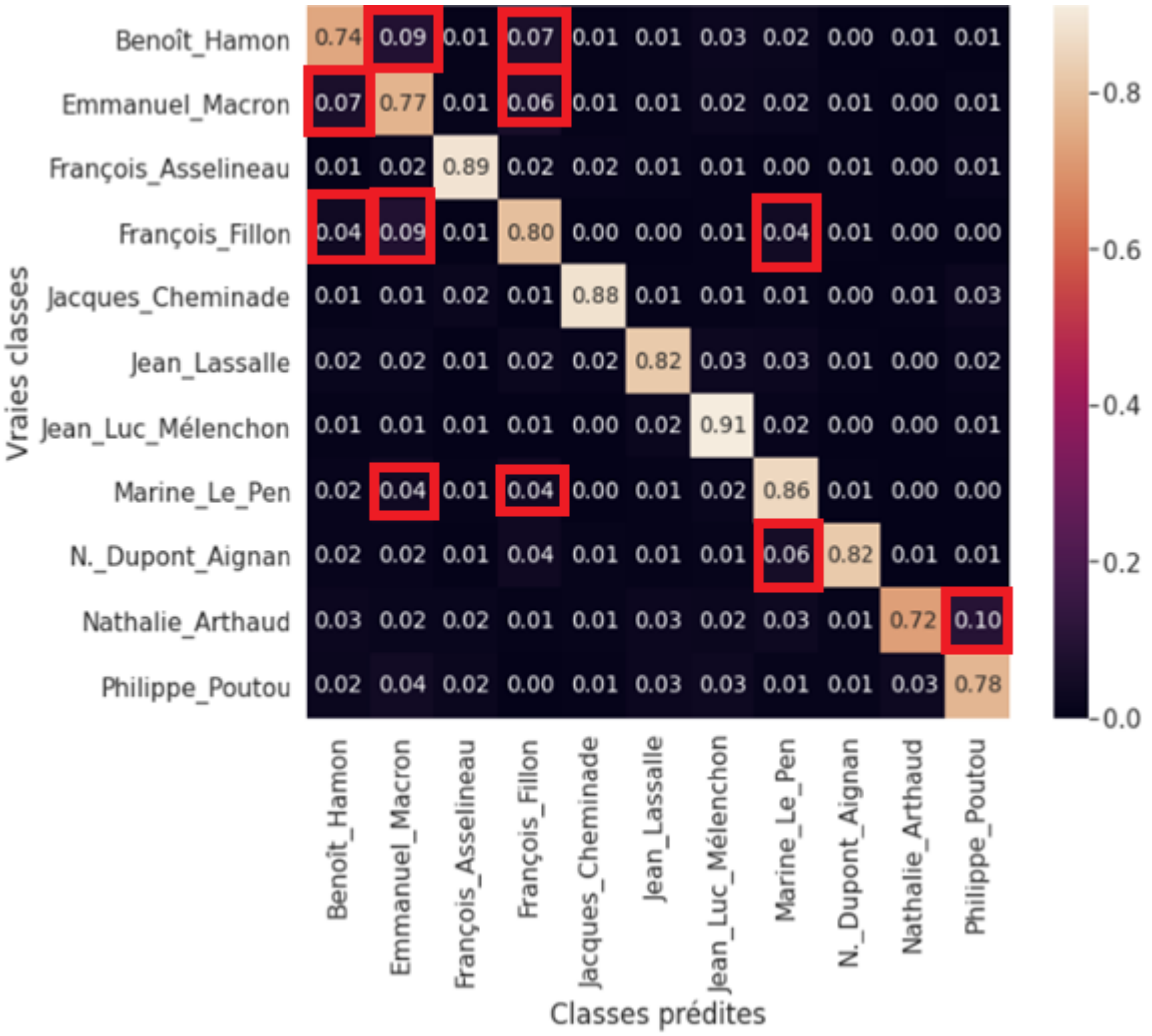

FIGURE 25: La matrice de confusion du modèle CNN-2

Comme les algorithmes d'apprentissage profond sont souvent considérés comme des « boîtes noires », nous ne pouvons pas expliciter leurs mécanismes décisionnels. Pourtant, nous essayons d'expliquer la confusion du modèle dans la classification en se basant sur la mesure de similarité. Plus concrètement, nous supposons que quand le modèle assigne un auteur A à un tweet, cela signifie que l'algorithme reconnaît une similitude entre ce tweet et un tweet de cet auteur dans la base de données d'entraînement. Dans cette idée, nous avons utilisé la similarité cosinus pour trouver, dans le jeu de données d'entraînement, le tweet le plus similaire du tweet en question. A partir de cela, nous pouvons mieux comprendre pourquoi les tweets ne sont pas correctement classifiés. Pour faire une analyse plus détaillée, nous sélectionnons seulement les cas dont le taux d'erreur est important. Si l'on regarde la matrice de confusion

(Figure 25), il s'agit des cas de Benoît Hamon, Emmanuel Macron, François Fillon, Marine Le Pen, Nicolas Dupont-Aignan et Nathalie Arthaud.

- Benoît Hamon – Emmanuel Macron :
  - Il y a 73/792 (soit 9%) des tweets de Benoît Hamon sont attribués à Emmanuel Macron. Nous prenons l'exemple un tweet de Benoît Hamon qui n'est pas correctement classifié « *Je souhaite offrir aux collectivités les moyens de mettre en oeuvre plus librement leurs compétences. #ADF* ». Le modèle prédit que ce tweet appartient à Emmanuel Macron. Et le tweet similaire de ce tweet est un tweet d'Emmanuel Macron « *Je veux donner la possibilité aux élus locaux de mettre en œuvre des réformes de structure. #ADF* ». Dans cet exemple, on voit bien que ces deux tweets partagent quelques similarités comme : *mettre en œuvre,* le hashtag *#ADF* et les deux groupes « *je souhaite offrir* » et « *je veux donner »* peuvent être considérés comme ayant une même signification.
  - En revanche, il y a 56/810 (soit 7%) des tweets d'Emmanuel Macron sont attribués à Benoît Hamon.

    Nous prenons l'exemple un tweet d'Emmanuel Macron « *Je m'engage à mettre en place une procédure simplifiée pour autoriser la production des énergies renouvelables »*, qui est attribué, par l'algorithme, à Benoît Hamon. Le tweet similaire de ce tweet, trouvé à partir de la base de données d'entraînement, est un tweet de Benoît Hamon : « *Je m'engage à inscrire dans la loi l'obligation de consacrer 0,7% PIB à l'aide publique au développement* ». Nous supposons ici que l'algorithme a bien capturé le motif « *je m'engage à* » dans les deux tweets.
- Benoît Hamon – François Fillon :
  - Il y a 53/792 (soit 7%) des tweets de Benoît Hamon sont attribués, par le modèle, à François Fillon.

    Par exemple, un tweet de Benoît Hamon *« Je parle aujourd'hui des 12 millions de personnes en situation de handicap et on m'interroge sur les costumes de M. Fillon* » qui est attribué, par l'algorithme, à François Fillon. Le tweet similaire de ce tweet est un tweet de François Fillon « *10 millions de Français sont concernés par le handicap. Pour construire mon programme, j'ai travaillé avec eux et entendu leurs demandes. »*

    Un autre tweet de Benoît Hamon est : « *J'ai la conviction que la France ne pourra pas répondre aux grands défis sans l'appui et l'intelligence des communes et des maires ».* Le modèle attribue ce tweet à François Fillon en référence à son tweet : *« J'ai l'intime conviction que les outre-mer assurent le rayonnement, la grandeur et la diversité de la*

France. » Nous supposons ici que le modèle a bien reconnu le motif « j'ai la conviction que » dans les deux tweets.

- Au contraire, 35/863 (soit 4%) des tweets de François Fillon sont attribués à Benoît Hamon.

  Par exemple, un tweet de François Fillon qui est assigné à Benoît Hamon « *Hommage aux forces de l'ordre qui donnent leur vie pour protéger les nôtres.* ». Son tweet similaire est celui de Benoît Hamon : « *Mes pensées vont au policier tué, à ses collègues blessés. Soutien total aux forces de l'ordre contre le terrorisme*. ». Nous prenons encore un autre tweet qui est: « *Le deuxième axe fondamental, c'est l'accès aux soins. Je ferai de la lutte contre les déserts médicaux une priorité de mon quinquennat.* ». La machine attribue ce tweet à Benoît Hamon en référence au tweet : « *Je propose une Mission nationale d'accès aux soins, qui disposera d'un budget et répondra au problème des déserts médicaux.* »

- Emmanuel Macron – François Fillon :
  - Il y a 51/810 (soit 6%) des tweets d'Emmanuel Macron sont attribués à François Fillon. Par exemple, un tweet d'Emmanuel Macron « *La France restera engagée contre le terrorisme partout dans le monde. »* et son tweet similaire est de François Fillon: « *Le rétablissement de la sécurité en France passera aussi par une guerre sans merci contre le terrorisme islamique* ». Vu que le tweet d'Emmanuel Macron est considéré proche, par l'algorithme, à un tweet de François Fillon qui se trouve dans les données d'entraînement, l'algorithme le classe dans catégorie François Fillon.
  - A l'inverse, il y a 74/863 (soit 9%) des tweets de François Fillon sont attribués à Emmanuel Macron.

    Par exemple, le tweet : « *Je veux que, d'ici la fin du quinquennat, 100% du territoire soit couvert en Très Haut Débit fixe et mobile.* » et son tweet similaire est un tweet d'Emmanuel Macron : « *Ce projet, c'est le vôtre : la couverture du territoire en très haut débit sera une des priorités de ce quinquennat* ».
- François Fillon – Marine Le Pen :
  - Il y a 38/863 (soit 4%) des tweets de François Fillon qui sont attribués à Marine Le Pen. Un tweet de François Fillon qui est attribué à Marine Le Pen *« J'inclus à cet apprentissage celui du respect, de la discipline et le port d'une tenue uniforme pour symboliser l'égalité. #LeGrandDébat*». Son tweet similaire trouvé à partir du corpus d'entraînement est : « *On doit revenir à une école qui transmet dans la discipline, avec*

le retour de l'autorité et du respect du maître." #2017LeDébat ». Nous supposons que l'algorithme a bien capturé la similarité entre « apprentissage » et « école », les mots « respect » et « discipline ».

- En revanche, 35/952 (soit 4%) des tweets de Marine Le Pen sont attribués à François Fillon.

  Un tweet de Marine Le Pen qui est attribué à François Fillon « *Des générations d'hommes et de femmes ont travaillé et souffert, parfois sont morts, pour nous transmettre un pays libre* ». Son tweet similaire est : « *Nos militaires sont des femmes et des hommes exceptionnels. Ce sont des héros d'une dignité rare qui méritent notre respect absolu. »*.

- Marine Le Pen – Emmanuel Macron :
  - Il y a 35/952 (soit 4%) des tweets de Marine Le Pen sont attribués à Emmanuel Macron. L'exemple d'un tweet de Marine Le Pen est : *« La fermeture de #Fessenheim est une décision idéologique faite sous la pression des écologistes. »*. Ce tweet est considéré « proche » à un tweet d'Emmanuel Macron : *« La fermeture de Fessenheim est une décision responsable que je maintiendrai »*.
- Nathalie Arthaud – Philippe Poutou
  - Il y a 28/272 (soit 10%) des tweets de Nathalie Arthaud sont attribués à Philippe Poutou. Nous prenons par exemple un tweet de Nathalie Arthaud : « *Vive la lutte des ouvriers de #renault à Bursa en turquie !* ». Ce tweet est considéré similaire d'un tweet de Philippe Poutou : « *#Turquie : L'usine de #Renault Bursa en grève après le licenciement de 10 travailleurs* »
- Un autre cas qui doit être pris en compte est celui de Nicolas Dupont-Aignan. 44/796 (soit 6%) ses tweets sont attribués à Marine Le Pen. Par exemple, le tweet : « *Grippe aviaire : 10 000 canards et faisans vont être abattus à Arthies. J'apporte mon soutien total aux éleveurs touchés par ce drame.* » est considéré proche d'un tweet de Marine Le Pen : « *La solidarité nationale doit s'appliquer à nos éleveurs durement touchés par la catastrophe sanitaire qu'est la grippe aviaire* ».

→ L'observation de ces quelques analyses nous permet donc de confirmer que l'algorithme a tendance à attribuer à un tweet un auteur A qui correspond à l'auteur du tweet le plus proche. Cependant, il reste encore des problèmes dans l'interprétation des erreurs du l'algorithme. Il existe des tweets que l'algorithme considère qu'ils sont proches, similaires mais cette similarité

n’a aucune signification pour l’humain. En effet, vu que les réseaux de neurones sont vus comme des « boîtes noires », il est difficile de visualiser, d’expliquer comment ils font les prédictions, les classifications. Il y a plusieurs d’études travaillant sur ce sujet mais elles se focalisent plutôt sur le traitement des images. Ainsi, pour un travail plus loin, nous envisageons de chercher une solution qui est capable de surligner les éléments linguistiques appris par le CNN. Cette solution peut se référer aux travaux de (Vanni et al., 2018), dont un algorithme de déconvolution.

Pour résumer, dans le cas de notre étude, nous avons proposé 6 modèles pour la tâche d’attribution d’auteur, dont 2 modèles CNNs et 4 modèles LSTMs. Parmi ces modèles, nous avons obtenu les meilleurs résultats avec le modèle CNN-2 avec un taux d’exactitude de 83%. Ce modèle a bien identifié des tweets de Jean-Luc Mélenchon, François Asselineau, Jacques Cheminade et Nicolas Dupont-Aignan avec des f-mesures, respectivement, de 90%, 90%, 89% et 87%. La classification des tweets de Jean Lassalle et Marine Le Pen est aussi satisfaisante avec des f-mesures de 84%. Enfin, nous avons obtenu les résultats moins satisfaisants pour les candidats : Philippe Poutou (78%), François Fillon (78%), Nathalie Arthaud (77%), Benoît Hamon (75%) et Emmanuel Macron (73%). Parmi ces résultats, comme nous avons expliqué, les résultats plutôt faibles de Nathalie Arthaud et Philippe Poutou peuvent s’expliquer par le déséquilibre des données. Quant aux cas de Benoît Hamon, Emmanuel Macron et François Fillon, nous constatons que le modèle a des difficultés de distinguer des tweets de ces trois auteurs, c’est-à-dire qu’il y a une confusion entre ces trois auteurs.

5. Conclusion et perspective

Face à la croissance exponentielle des textes anonymes sur Internet, plus spécifiquement sur des réseaux sociaux, le problème d'authentification d'auteurs devient plus urgent que jamais. Cependant, ce problème est confronté à plusieurs difficultés et défis, dont la longueur des textes. A travers ce travail, nous avons pu montrer que l'application des méthodes d'apprentissage profond à l'attribution d'auteur des textes courts (des tweets dans notre cas) est une direction prometteuse.

Afin de construire un système neuronal, nous avons passé beaucoup de temps à comprendre les notions des réseaux de neurones et à maîtriser l'utilisation de différentes bibliothèques logicielles d'apprentissage profond. En plus, il faut lire des articles, des publications et effectuer beaucoup d'expérimentations en faisant varier non seulement les architectures des réseaux mais aussi des hyperparamètres pour trouver un système qui est robuste et adapté à notre recherche.

Le travail que nous avons présenté ici n'est qu'un premier pas vers le domaine d'attribution d'auteur associé à l'apprentissage profond. Comme perspectives de ce travail, plusieurs pistes peuvent être envisagés :

- Au lieu d'utiliser un seul trait linguistique, nous allons combiner plusieurs traits linguistiques en même temps (par exemple : mots + lemmes + parties du discours) avec l'espoir que cela va rendre le système d'attribution d'auteur plus robuste.
- Segmenter les tweets non pas au niveau de mots mais au niveau de lettres pour garder les majuscules, les ponctuations parce que ces éléments sont aussi utiles pour quantifier le style d'un auteur.
- Focaliser sur l'étape de prétraitement des données, dont la normalisation. Comme notre corpus constitue des tweets politiques, ils ne contiennent pas beaucoup des éléments « bruyants ». Pourtant, les tweets en général se caractérisent par les abréviations, les émoticônes, les mots allongés, etc,.. Normaliser ces éléments peut contribuer considérablement à la performance du système.
- Construire des modèles plus « profonds » par exemple : combiner les réseaux de neurones convolutionnels (CNN) avec les réseaux récurrents à mémoire court et long terme (LSTM) ; empiler les couches LSTM et ajouter une couche d'Attention.[24] En plus, nous voudrions tester le modèle BERT qui définit l'état de l'art pour plusieurs tâches de TAL.

[24] Les mécanismes d'attention permettent à un réseau de neurones d'apprendre sur quoi il doit se concentrer pour la décision.

- Augmenter le volume de corpus ainsi que le nombre d'auteurs. Une fois que nous obtenons un corpus assez grand, nous pouvons aussi entraîner notre propre *Word Embeddings* au lieu d'utiliser les *Word Embeddings* pré-entraînés.

Bibliographie

P. Juola. Authorship attribution. *Foundations and Trends in Information Retrieval*, 2006.

E. Stamatatos. Authorship attribution based on feature set subspacing ensembles. *International Journal on Artificial Intelligence Tools*, 15(5), 823-838, 2006.

A. Rocha, W.J. Scheirer, C. W. Forstall, T. Cavalcante, A. Theophilo, B. Shen, A. R. B Carvalho, E. Stamatatos. Authorship Attribution for Social Media Forensics. In *IEEE Transactions on Information Forensics and Security*, 12(1), 5-33, 2017.

J. Savoy. Attribution d'auteur par ensembles de séparateurs. *CORIA 2013 Conférence en Recherche d'Informations et Applications – 10th French Information Retrieval Conference*, 2013.

D.I. Holmes. Authorship Attribution. *Computers and the Humanities*, 28, 87-106, 1994.

D.I. Holmes. The evolution of stylometry in humanities scholarship. *Literary and Linguistic Computing*, 13(3), 111-117, 1998.

J. Rudman. The state of authorship attribution studies: Some problem and solutions. *Computers and Humanities*, 31, 351-365, 1998.

J.F. Burrows. 'Delta': A measure of stylistic difference and a guide to likely authorship. *Literary and Linguistic Computing*, 17(3), 267-287, 2002.

J. Diederich, J. Kindermenn, E. Leopold & G. Pass. Authorship Attribution with Support Vector Machines. *Applied Intelligence*, 109-123, 2003.

D.L. Hoover. Testing Burrow's Delta. *Literary and Linguistic Computing*, 19(4), 453-475.

Y. Zhao & J. Zobel. Effective and scalable authorship attribution using function words. In *Asian Information Retrieval Symposium*, 174-189. Springer, 2005.

E. Stamatatos. A survey of modern authorship attribution methods. *Journal American Society for Information Science and Technology*, 60(3), 538-556, 2009.

E. Stamatatos, N. Fakotakis & G. Kokkinakis. Computer-based authorship attribution without lexical measures. *Computers and the Humanities*, 35(2), 193-214, 2001.

M. Gamon. Linguistic correlates of style: Authorship classification with deep linguistic analyses features. In *Proceedings of the 20th International Conference on Computational Linguistics*, 611-617, 2004.

Y. Zhao, J. Zobel & P. Vines. Using relative entropy for authorship attribution. In *Information Retrieval Technology*, 92-105. Springer, 2006.

M. Koppel, J. Schler, & K. Zigdon. Determining an author's native language by mining a text for errors. In *ACM SIGKDD Intl. Conference on Knowledge Discovery in Data Mining*, 624–628. ACM, 2005.

M. Koppel, J. Schler & E. Bonchek-Dokow. Measuring differentiability: Unmasking pseudonymous authors. *Journal of Machine Learning Research*, 8:1261-1276, 2007.

M. Koppel, J. Schler, S. Argamon, & E. Messeri. Authorship attribution with thousands of candidate authors. In *Intl. ACM SIGIR Conference on Research and Development in Information Retrieval*, 659–660. ACM, 2006.

M. L. Pacheco, K. Fernandes & A. Porco. Random forest with increased generalization: A universal background approach for authorship verification. *CLEF 2015 Evaluation Labs*, 2015.

S. E. M. El Bouanani & I. Kassou. Authorship analysis studies: A survey. *International Journal of Computer Applications*, 86, 22–29, 2014.

K. Lagutina, N. Lagutina, E. Boychuk, I. Vorontsova, E. Shliakhtina, O. Belyaeva, I. Paramonov, P.G. Demidov. A Survey on Stylometric Text Features. *2019 25th Conference of Open Innovations Association (FRUCT)*, 184-195, 2019.

G. Ledger & T. Merriam. Shakespeare, fletcher, and the two noble kinsmen. *Literary and Linguistic Computing*, 235–248, 1994.

R. Forsyth & D. Holmes. Feature finding for text classification. *Literary and Linguistic Computing*,11(4), 163–174, 1996.

J. Grieve. Quantitative authorship attribution: An evaluation of techniques. *Literary and linguistic computing*, 22(3), 251–270, 2007.

C. E. Chaski. Who's at the keyboard? authorship attribution in digital evidence investigations. Intl. *Journal of Digital Evidence*, 4(1), 1–13, 2005.

R. Layton, P. Watters, & R. Dazeley. Authorship attribution for Twitter in 140 characters or less. In *Cybercrime and Trustworthy Computing Workshop*, 1–8, 2010.

L. Tanguy, A. Uriely, B. Calderone, N. Hathout & F. Sajous. A multitude of linguistically-rich features for authorship attribution. PAN Lab at CLEF, 2011.

L. Tanguy, F. Sajous, B. Calderone & N. Hathout. Authorship Attribution: Using Rich Linguistic Features when Training Data is Scarce. PAN Lab at CLEF, 2012.

J. Sun, Z. Yang, S. Liu & P. Wang. Applying stylometric analysis techniques to counter anonymity in cyberspace. *Journal of Networks, 7(2), 2012.*

R. Schwartz, O. Tsur, A. Rappoport & M. Koppel. Authorship attribution of micro-messages. In *Conference on Empirical Methods on Natural Language Processing*, 1880–1891. ACL, 2013.

E. Stamatatos. On the robustness of authorship attribution based on character n-gram features. *Journal of Law & Policy*, 21, 421–725, 2013.

A.M. Turing. Computing machinery and intelligence. *Mind*, 59, 433-460, 1950.

G. Fung. The disputed Federalist Papers: SVM feature selection via concave minimization. In *Conference on Diversity in Computing*, 42–46. ACM, 2003.

M. Koppel & J. Schler. Exploiting stylistic idiosyncrasies for authorship attribution. In *Workshop on Computational Approaches to Style Analysis and Synthesis*, 69, 72–80, 2003.

J. Savoy. The Federalist Papers revisited: A collaborative attribution scheme. *American Society for Information Science and Technology*, 50(1), 1–8, 2013.

A. Abbasi & H. Chen. Applying authorship analysis to extremistgroup web forum messages. *IEEE Intelligent Systems*, 20(5), 67–75, 2005.

R. Zheng, J. Li, H. Chen & Z. Huang. A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of the American Society for Information Science and Technology*, 57(3), 378–393, 2006.

S. Argamon, C. Whitelaw, P. Chase, S. R. Hota, N. Garg & S. Levitan. Stylistic text classification using functional lexical features. *Journal of the American Society for Information Science and Technology*, 58(6), 802–822, 2007.

S. Hedegaard & J. G. Simonsen. Lost in translation: Authorship attribution using frame semantics. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers*, 2, 65–70. Association for Computational Linguistics, 2011.

G. K. Mikros & Kostas P. Authorship attribution in Greek tweets using authors multilevel n-gram profiles. In *AAAI Spring Symposium Series*, 2013.

G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh & L. ChanonaHernandez. Syntactic n-grams as machine learning features for natural language processing. *Expert Syst. Appl.*, 41, 853–860, 2014.

M. Koppel & Y. Winter. Determining if two documents are written by the same author. *Journal of the Association for Information Science and Technology*, 65(1), 178–187, 2014.

R. Clement & D. Sharp. Ngram and bayesian classification of documents for topic and authorship. *Literary and Linguistic Computing*, 18(4), 423–447, 2003.

F. Peng, D. Schuurmans & S. Wang. Augmenting naïves Bayes classifiers with statistical language models. *Information Retrieval*, 7(3- 4)317–345, 2004.

S. R. Boutwell. Authorship attribution of short messages using multimodal features. Master's thesis, Naval Postgraduate School, Monterey, CA, USA, 2011.

M. Almishari, D. Kaafar, E. Oguz & G. Tsudik. Stylometric linkability of tweets. In *Workshop on Privacy in the Electronic Society*, 2014.

M. L. Jockers & D. M. Witten. A comparative study of machine learning methods for authorship attribution. *Literary and Linguistic Computing*, 25, 215–223, 2010.

O. Halvani, M. Steinebach & R. Zimmermann. Authorship Verification via kNearest Neighbor Estimation. Notebook for PAN at CLEF, 2013.

B. Kjell, W. A. Woods & O. Frieder. Information retrieval using letter tuples with neural network and nearest neighbor classifiers. In *IEEE Intl. Conference on Systems, Man and Cybernetics*, 2, 1222–1226. IEEE, 1995.

R. A. J. Matthews & T. V. N. Merriam. Neural computation in stylometry I: An application to the works of Shakespeare and Fletcher. *Literary and Linguistic Computing*, 8(4), 203–209, 1994.

D. Lowe & R. Matthews. Shakespeare vs. Fletcher: A stylometric analysis by radial basis functions. *Computers and the Humanities*, 29(6), 449–461, 1995.

F. Tweedie, S. Singh & D. I. Holmes. Neural network applications in stylometry: The Federalist Papers. *Computers and the Humanities*, 30(1), 1–10, 1996.

J. F. Hoorn, S. L. Frank, W. Kowalczyk & F. van Der Ham. Neural network identification of poets using letter sequences. *Literary and Linguistic Computing*, 14(3), 311–338, 1999.

D. Bagnall. Author identification using multi-headed recurrent neural networks. In *Working Notes Papers of the CLEF 2015 Evaluation Labs*, 2015.

D. Rhodes. Author Attribution with CNN's. Technical report, 2015 http://cs224d.stanford.edu/reports/RhodesDylan.pdf.

P. Shrestha, S. Sierra, F. A. Gonzalez, M. Montes-y-Gómez & T. Solorio. Convolutional Neural Networks for Authorship Attribution of Short Texts. In *Proceedings of the EACL*, 2017.

J. Hitschler, B. Esther & I. Rehbein. Authorship Attribution with Convolutional Neural Networks and POS-Eliding. In *Proceedings of the Workshop on Stylistic Variation*, 53-58, 2017.

N. Schaetti. Bidirectional Echo State Network-based Reservoir Computing for Cross-domain Authorship Attribution. *Notebook for PAN at CLEF*, 2018.

L. Gagala. Authorship attribution with neural networks and multiple features. *Notebook for PAN at CLEF*, 2018

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

T. Mikolov, W. Yih, G. Zweig. Linguistic Regularities in Continuous Space Word Representations. In *Association for Computational Linguistics*, 746-751, 2013.

Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of Empirical Methods on Natural Language Processing*, 2014.

N. Kalchbrenner, E. Grefenstette & P. Blunsom. A convolutional neural network for modelling sentences, 2014.

T. Zhang, C. Li, N. Cao & R. Ma. Text feature extraction and classification based on convolutional neural network (CNN). In *Data Science*, 472-485, 2017

A. Conneau, H. Schwenk, L. Barraul, & Y. Lecun. Very Deep Convolutional Networks for Text Classification. In *Association for Computer Linguistics*, 1, 107-1116, 2017.

Y. Sari & M. Stevenson. Exploring Word Embeddings and Character N-Grams for Author Clustering. *CLEF*, 2016.

A. Sittar, H. R. Iqbal & R. M. A. Nawab. Author Diarization Using Cluster-Distance Approach. *Working Notes Papers of the CLEF*, 2016

O.S. Rao, N. GanapathiRajuDr & Y. SrilalithaDr. Authorship Attribution using Unsupervised Clustering Algorithms on English C50 News Articles. *International Advanced Research Journal in Science, Engineering and Technology, 4*, 272-276, 2017.

H. Gomez- Adorno, Y. Alemán, D. Vilariño, M. Sanchez-Perez, D. Pinto & G. Sidorov. Author Clustering using Hierarchical Clustering Analysis, 2017.

J. L. Elman. Finding structure in time. In *Cognitive science*, 14(2), 179–211, 1990.

S. Hochreiter & J. Schmidhuber, "Long Short-Term Memory," In *Journal of Neural Computation*, 9(8), 1735-1780, 1997.

J. Savoy. Authorship Attribution: A comparative study of three text corpora and three languages. In *Journal of Quantitative Linguistics*, 19, 132-161, 2012.

A.L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers". In *IBM Journal of Research and Development*, 3(3), 210-229, 1959.

J. Pennington, R. Socher & C.D. Manning. **Glove : Global vectors for word representation**. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12, 2014.

P. Bojanowski, E. Grave, A. Joulin, T. Mikolov. **Enriching Word Vectors with Subword Information**. In *TACL*, 2017.

J. Delvin, M. W. Chang, K. Lee, K. Toutanova. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,** 2018**.**

N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever & R. Salakhutdinov. Dropout : a simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research*, 15, 1929–1958, 2014.

L. Vanni, D. Mayaffre & D. Longrée. ADT et deep learning, regards croisés. Phrases-clefs, motifs et nouveaux observables. *JADT*, 2018.

Z.S. Harris. Distributional structure. Word, 10(2-3), 146– 162, 1954.

Y. Bengio, R. Ducharme, P. Vincent & C. Janvin. A neural probabilistic language model. In *The journal of Machine Learning Research*, 3, 1137–1155, 2003.

R. Collobert & J. Weston. A unified architecture for natural language processing : Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, 160–167, 2008.

Liste des Figures

Liste des Tableaux