Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis
37200 Tours, France
Tél. +33 (0)2 47 36 14 14
polytech.univ-tours.fr

Projet Recherche & Développement

2018-2019

Fact checking

Tuteur académique
Sabine Barrat

Étudiant
Pauline Lauron (DI5)

3 avril 2019

Liste des intervenants

Nom	Email	Qualité
Pauline Lauron	pauline.lauron@etu.univ-tours.fr	Étudiant DI5
Sabine Barrat	sabine.barrat@univ-tours.fr	Tuteur académique, Département Informatique

Avertissement

Ce document a été rédigé par Pauline Lauron susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Sabine Barrat susnommé le tuteur académique.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnait assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnait qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable du tuteur académique et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.

Pour citer ce document

Pauline Lauron, *Fact checking*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2018-2019.

```
@mastersthesis{
    author={Lauron, Pauline},
    title={Fact checking},
    type={Projet Recherche \& Développement},
    school={Ecole Polytechnique de l'Université François Rabelais de Tours},
    address={Tours, France},
    year={2018-2019}
}
```

Document mis en forme sous LaTeX

grâce à la classe de document `polytech.cls` V2.2.5 (2017/05/02)

Table des matières

Table des figures

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au bon déroulement de ce projet de recherche et développement.

Tout d'abord, je tiens à remercier mon encadrante, Sabine BARRAT, qui m'a accompagnée et conseillée durant l'intégralité de mon projet de recherche et développement.

Je remercie les journalistes de l'école publique de journalisme de Tours, plus particulièrement Laurent BIGOT et Jeremie NICEY pour m'avoir accordée du temps et fait avancer dans la réflexion des enjeux et des caractéristiques des *fake news*.

Enfin, je souhaite remercie toutes les personnes qui m'ont aidé pour la rédaction et la relecture de ce rapport.

1 Introduction

Figure 1 – *fakes news*

1 Qu'est ce que le *fact-checking* ?

Le ***fact-checking*** (vérification des faits en français) est une technique consistant, d'une part, à vérifier en instantané la véracité de faits et l'exactitude des chiffres présentés dans les médias par des personnalités et des experts, et d'autre part, à évaluer le niveau d'objectivité des médias eux-mêmes dans leur traitement de l'information. Cette notion est apparue aux États-Unis dans les années 1990 sous l'appellation de *fact-checking*.
C'est un enjeu majeur pour les plate-formes de réseaux sociaux, les moteurs de recherche, et les médias. Les journalistes produisent des vérifications informationnelles et en font des rubriques spécifiques. Toute la question repose sur "comment proposer un dispositif de contrôle et de vérification des informations ?"

2 Pourquoi les *fakes news* se développent ?

Les *fakes news* (informations fallacieuses, infox ou fausses nouvelles) sont des informations délibérément fausses, délivrées dans le but de tromper un auditoire. Elles ont toujours existé, mais leur résonance est amplifiée par la vitesse de circulation de l'information et la non-vérification dans les réseaux sociaux. Les chercheurs ont remarqué que les *fakes news* se propagent plus rapidement et atteignent plus de personnes que les informations dite fiables.

3 Pourquoi est-ce important de les détecter ?

Les *fakes news* peuvent avoir des conséquences néfastes dans certains environnements comme la politique. Un exemple qui illustre très bien les conséquences peut être les *fakes news* diffusées durant l'élection présidentielle des Etats-Unis et la dernière élection de Donald Trump. L'une des plus populaires *fakes news* relayées par des miliers de personne est le soutien du pape François à Donald Trump durant sa campagne électorale. Lors de l'élection de *Trump*, beaucoup de *fakes news* ont été émises. Des chercheurs se sont alors intéressés à savoir quel impact avait eu ces *fakes news* sur la victoire de *Donald Trump* : ils en ont conclu qu'elles ont eu peu de poids à l'échelle du pays. Selon leur calcul, il aurait donc fallu que ces fausses informations soient en moyenne 36 fois plus efficaces que les campagnes télévisées pour être responsables de sa victoire. Cependant, ces chercheurs nous alertent sur le fait que ce phénomène n'en est qu'à ces débuts et donc qu'ils ne peuvent faire de conclusion hâtive.

Un autre exemple serait durant la dernière élection présidentielle française où des publications avec les titres suivants étaient parus :

— Jean-Luc Mélenchon porterait une Rolex à son poignet
— Emmanuel Macron serait financé par l'Arabie Saoudite

Ces informations ont été démenties par les concernés.

Il est alors dans notre intérêt de lutter contre ces *fakes news*. Il y a en a de plus en plus qui circulent de plus en plus rapidement et qui peuvent se contredire, il est difficile pour les journalistes de démêler le vrai du faux et surtout très coûteux en temps. Pour limiter cela et apporter une aide considérable, il faut créer une méthode permettant que les détecter automatiquement. Les approches utilisant l'intelligence artificielle semblent être les plus pertinents à mettre en place. Cependant, le problème qui paraît simple cache plusieurs points de discussions.

4 Quels sont les challenges de ce problème ?

Un des premiers freins de ce problème est la collecte de données contenant des données dites "vraies" et des données dites "fausses" ainsi que le fait de les labelliser manuellement. Un nombre de données trop faible ne permet pas un apprentissage suffisant car toute la base de l'intelligence artificielle repose sur la comparaison du label prédit et du label originel, on parle de vérité terrain. Un jeu de données français devrait contenir au moins le titre et le corps de l'article ainsi que son label. Une autre possibilité serait de trouver un jeu de données avec la même structure mais, cette fois-ci, en anglais. De plus, il ne suffit pas d'apprendre à une machine à parler une langue, mais surtout à comprendre le contexte. Un mot peut avoir une signification différente étant donné le contexte.
Deuxièmement, ceux qui écrivent des *fakes news* veulent qu'elles soient le plus possible diffusées et donc, il faut paraître crédible aux vues des lecteurs, ce qui rend la détection d'autant plus

difficile.
Il faut aussi être vigilant sur l'utilisation de l'intelligence artificielle car elle peut, dans certain cas, exploser et possiblement amplifier des propos discriminants ou continuer d'entretenir des stéréotypes.
Ce qui est paradoxal ici, c'est que nous allons utiliser des méthodes de *Machine Learning* pour détecter des *fakes news* alors qu'il est tout à fait possible pour une intelligence artificielle de créer des *fakes news*. Un exemple de la puissance de ces intelligences est une application basée sur une IA qui permet sur une vidéo de faire dire ce que l'on veut à n'importe qui. Pas en réalisant un grossier doublage, mais en convertissant chaque parole prononcée en mouvement de la bouche. Dans le cadre d'une collaboration avec BuzzFeed, une vidéo a été réalisé dans laquelle Barack Obama parle de Black Panther et insulte Donald Trump au passage [19]. Cette vidéo nous rappelle à quel point ces infox ont de l'avenir et qu'il devient possible de modifier une vidéo de manière réalistique, on peut alors imaginer que le texte est encore plus facile à contourner.

5 Actualités françaises sur les *fakes news*

Lors de ses vœux à la presse, en janvier 2018, le Président Emmanuel Macron annonce un projet de loi visant à lutter contre les *fakes news* en donnant notamment la possibilité de saisir un juge et de déréférencer ou bloquer certains sites concernés.
Le 10 octobre 2018, les députés ont voté la proposition de loi ordinaire contre « la manipulation de l'information » en période électorale. Elle vise à permettre à un candidat ou parti de saisir le juge des référés pour faire cesser la diffusion de « fausses informations » durant les trois mois précédant un scrutin national. Les récentes prises de parole ou votes mettent en lumière à quel point c'est un sujet d'actualité et qui suscite beaucoup l'attention, que ce soit des journalistes, des politiques, des chercheurs ou même du grand public.

6 Objectifs

L'objectif ici est de mettre en place un outil de détection des *fakes news* automatique. Pour cela, nous allons utiliser un jeu de données pour permettre un apprentissage profond grâce à un réseau de neurones.

Partie recherche

2 Description générale

1 Environnement du projet

Ce projet ne dépend d'aucun environnement matériel.

Le programme sera développé dans le langage Python avec les bibliothèques Keras [12] et Tensorflow [17] pour le *Deep Learning*. L'IDE utilisé sera spyder qui fait partie de la distribution open-source Anaconda [1].

2 Bases méthodologiques

La première partie peut être assimilée à une partie recherche, qui permet :

— de prendre connaissance des techniques existantes,
— de définir vers quelle méthode se diriger.

Ensuite, nous passons dans une phase itérative (voire cyclique) dont le but est de trouver le modèle ainsi que les paramètres optimaux pour notre problème de détection de *fakes news*.
On parle d'une phase cyclique car :

1. création du modèle
2. entraînement de ce modèle selon certains paramètres
3. évaluation le modèle
4. retourner en 1. jusqu'à être satisfait des résultats

3 Caractéristiques des utilisateurs

La MOA sera l'utilisatrice de ces méthodes. Celle-ci possède de bonnes connaissances en informatique et en *Machine Learning*.

4 Fonctionnalités du système

Il y a 2 grandes étapes pour ce projet de recherche et développement :

— la phase d'apprentissage
— la phase de prédiction

Un diagramme de cas d'utilisation peut être retrouvé à la figure 1.

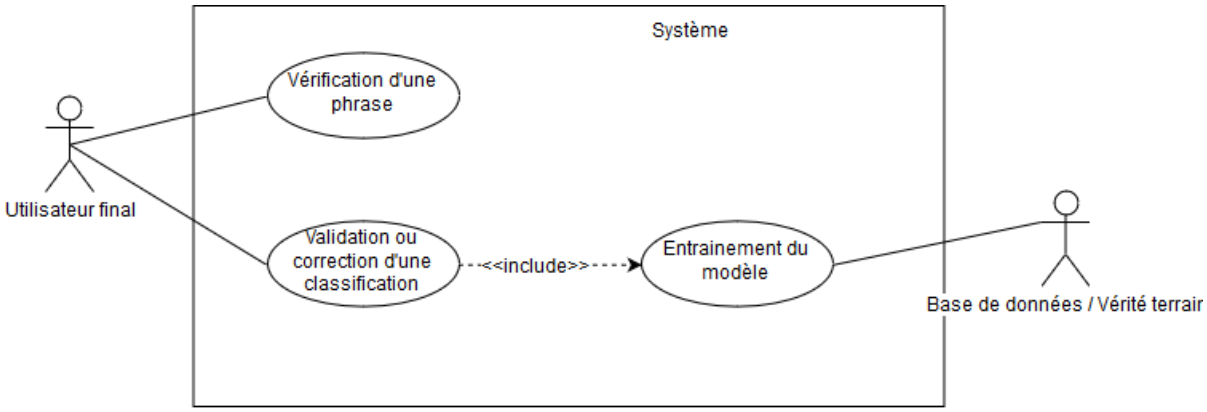


Figure 1 – *Diagramme de cas d'utilisation.*

L'utilisateur final, qui sera un journaliste, aura 2 possibilités :

— soit demander une aide à la décision, c'est-à-dire qu'on vérifie une phrase donné par le journaliste
— soit de valider ou de corriger une classification faite par le modèle déjà entraîné. Ce modèle aura interagit avec une base de données qui se compose d'articles dit "vrais" et d'autres dit "faux".

5 Structure générale du système

Le système sera défini selon l'organisation suivante :

— les jeux de données seront regroupés dans le dossier données.
— les modèles seront regroupés dans le dossier modèles.
— les résultats seront regroupés dans le dossier résultats.

Il y aura aussi un fichier de configuration sous forme de dictionnaire de données regroupant les paramètres des différentes parties. Cela permettra une modification plus rapide de ses paramètres ainsi qu'une vision générale des paramètres rentrant en compte pour les différentes parties du processus.

6 Spécifications fonctionnelles

Ce projet va être composé de plusieurs modules :

— pré-traitement des données
— conversion de mots vers vecteurs
— apprentissage
— prédiction

Le détail de ces modules peut être trouvé à l'annexe A.

3 État de l'art

Pour une meilleure compréhension du sujet, une première partie de cette section sera dédiée à l'explication du *Deep Learning* et l'utilisation des réseaux de neurones, plus spécifiquement des réseaux de neurones à convolution. Ensuite, un état de l'art concernant les CNN (Convolutional Neural Network) appliqués à des NLP (*Natural Langage Processing*) sera exposé.

1 Traitement automatique du langage naturel

Le traitement automatique du langage naturel (*Natural Langage Processing*) a pour but de créer des outils de traitement de la langue naturelle qui analyse les interactions entre les ordinateurs et le langage humain et en particulier pour traiter et analyser des grandes quantités de données. Un des challenges les plus difficile dans le traitement du langage est la prise en compte du contexte.
Prenons l'exemple du mot froid : dans la phrase, *"il fait froid"*, le froid réfère à une sensation glaciale alors que dans la phrase *"cet homme est froid"*, froid veut dire que l'homme ne manifeste aucun sentiment. On remarque alors que pour un même mot le contexte change significativement son sens.
Pour essayer de résoudre cette problématique, les chercheurs ont commencé à développer des applications utilisant des approches de *Machine Learning*. Ces algorithmes vont alors détecter des motifs dans les données d'entrée et en tirer des informations pour essayer de trouver une façon de généraliser la représentation des *fakes news* pour pouvoir plus facilement les détecter.

Champs d'application

Plusieurs champs d'application du NLP sont possibles. En voici une liste (elle n'est pas exhaustive) :

- Au niveau syntaxique, il existe la lemmatisation qui est le fait de regrouper des mots d'une même famille dans un texte afin de réduire ces mots à leur forme canonique.
- Au niveau sémantique, cela est utilisé lors de la traduction automatique ou la correction orthographique.
- Au niveau du traitement du signal, il y a la reconnaissance automatique de la parole.
- Au niveau de l'extraction d'informations, cela peut être utilisé lors de fouille de textes, c'est-à-dire la recherche d'informations spécifiques, la classification et catégorisation de documents, l'analyse de sentiment...

2 Le *Deep Learning*

Intelligence artificielle, *Machine Learning* et *Deep Learning* sont des mots qui deviennent de plus en plus courants.
Pour mieux comprendre leur lien, le schéma 1 est donnée. Très intuitivement, on remarque que l'intelligence artificielle est le grand domaine qui inclut le *Machine Learning* qui lui-même est composé du *Deep Learning*.

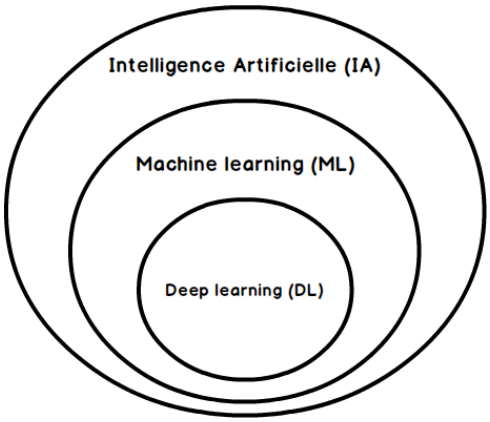


Figure 1 – *Lien entre IA, ML et DL*

Pour donner une définition assez généraliste, l'intelligence artificielle est l'ensemble des techniques et théories mise en oeuvre en vue de créer des machines qui permettent de simuler l'intelligence.
Parmi ces techniques, il existe le *Machine Learning* qui se base sur des approches statistiques dans le but de "comprendre" et "apprendre" à partir de jeu de données, tout cela dans le but d'améliorer leurs performances à résoudre des tâches sans être explicitement programmées pour chacune d'entre elles.
Enfin, le *Deep Learning* (apprentissage profond en français) est un ensemble de méthode d'apprentissage automatique tentant de modéliser avec un haut niveau d'abstraction des données grâce à des architectures articulées de différentes transformations non linéaires, on parle de réseau de neurones.
Étant donné que les réseaux de neurones artificiels imitent le fonctionnement du cerveau humain, les possibilités offertes par cette technologie sont vastes et peuvent toucher plein de domaines différents. Cette approche peut être retrouvée dans la conduite automatisée, la reconnaissance audio et vocale ou encore dans la recherche médicale.
Le *Deep Learning* permet de répondre à des questions du type "que peut-on déduire de ces données ?". L'intérêt du réseau de neurones est de modéliser l'impact des différents facteurs et leur relation entre eux. De plus, le *Deep Learning* offre l'avantage d'identifier les caractéristiques essentielles du traitement (qui été identifié par un traitement humain dans un algorithme préalable) directement. Cela peut permettre d'identifier des caractéristiques cachées ou des relations entre des données souvent impossibles à identifier pour l'homme [5].

On voit une évolution significative de l'utilisation du *Deep Learning* à partir de 2012, grâce à la combinaison de plusieurs facteurs :

- les réseaux neuronaux multi-couches (qui datent des années 50),
- les algorithmes d'analyse discriminante et apprenants (qui datent des années 80),
- et surtout, la prise en charge de traiter de données massives par des machines plus puissantes.

Cet apprentissage profond permet, par exemple, de reconnaître le contenu d'une image ou de comprendre le langage parlé. Sa structure est basée sur des "réseaux de neurones" que l'on va détailler dans la section suivante.

2.1 Les réseaux de neurones

Le *Deep Learning* utilise un "réseau de neurones" pour apprendre des données d'entrées. Ce réseau de neurones s'inspire du cerveau humain : les neurones et leurs connexions. Il est donc composé de "neurones" répartis en une ou plusieurs couches.
Ces systèmes sont très souvent utilisés pour des problèmes de classification.
Un neurone au sens informatique est une somme pondérée de signaux auquel on affecte une fonction dite d'activation.

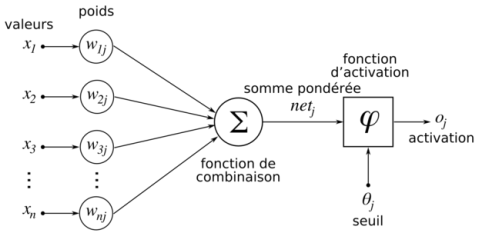


Figure 2 – *Structure d'un neurone.*

Le neurone calcule la somme de ses entrées puis cette valeur passe à travers la fonction d'activation pour produire sa sortie. La fonction de combinaison dans le cas de réseaux de type MLP (multi-layer perceptron) calculent une combinaison linéaire des entrées, c'est-à-dire que la fonction de combinaison renvoie le produit scalaire entre le vecteur des entrées et le vecteur des poids synaptiques. La fonction d'activation sert à introduire une non-linéarité dans le fonctionnement du neurone. Une des fonctions les plus utilisées est la fonction sigmoïde.

On associe à la couche finale une fonction de perte pour calculer l'erreur de classification. Pour minimiser cette erreur, les valeurs des poids des couches apprennent progressivement les paramètres qui minimisent la fonction de perte régularisée.

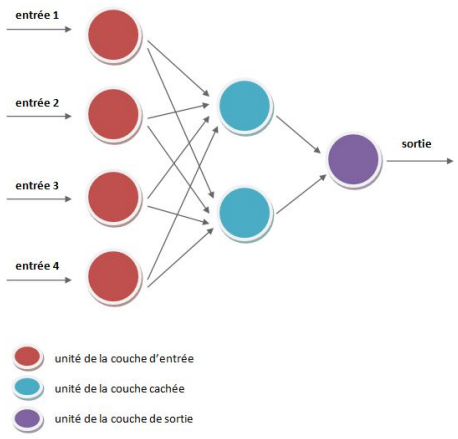


Figure 3 – *Exemple simple d'un réseau de neurones*

Comme on peut le voir dans la figure 3, nous avons une couche d'entrée composée de 4 neurones, 1 couche cachée composée 2 neurones et enfin la couche de sortie composée d'un seul neurone

qui produit les probabilités finales en utilisant pour fonction d'activation la fonction logistique (qui est le cas ici) ou la fonction softmax pour les multi-classes.
En général, plus le nombre de neurones dans les couches cachées est important, plus la capacité l'est. Le modèle permet alors de classifier des cas plus particuliers ou plus isolés lorsque l'on augmente le nombre de neurones. En contre partie, il sera plus gourmand en temps de calcul.

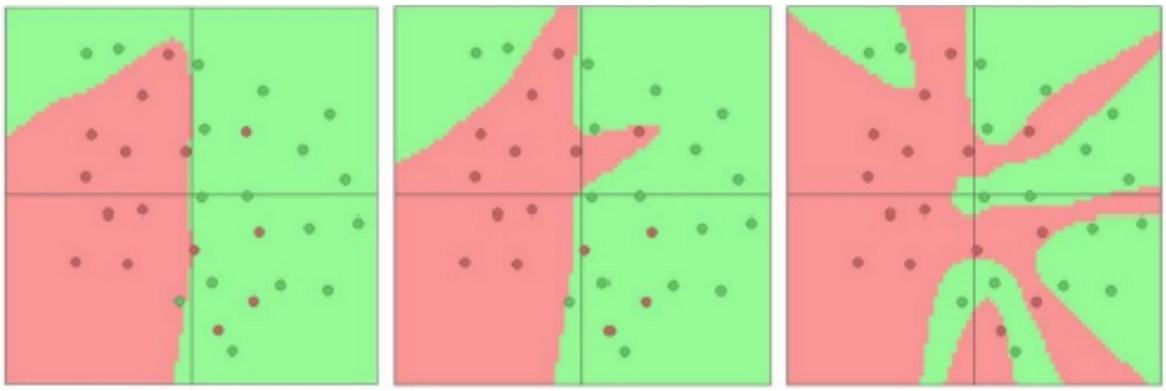


Figure 4 – *Exemple de l'influence du nombre de neurones dans les couches cachées.*

La figure 4 nous montre 3 images : celle de gauche est obtenu avec l'utilisation de 3 neurones cachées, celle du milieu avec 6 neurones cachées et celle de droite avec 20 neurones cachées.
Le but est de trouver une modélisation pour séparer les 2 classes : les points rouges et les points verts.
On remarque qu'avec 3 neurones cachées, 6 points rouges sont mal catégorisés et 1 point vert se situe dans la zone rouge. Si l'on double le nombre de neurones cachés (6), tous les points verts apparaissent dans la zone verte et seulement 4 points rouges sont mal classés. Finalement, si 20 neurones cachées sont utilisés, tous les points sont correctement classés.

Il semble que plus le nombre de neurones cachés augmentent, meilleures sont les performances. Cependant, il peut y avoir des risques de sur-apprentissage, c'est-à-dire que le modèle va apprendre "par coeur" les données fournies pour l'entraîner et donnera des résultats moindres si on lui donne un jeu de données qu'il n'a jamais vu. C'est une raison pour laquelle le jeu de données sera divisé en 2 parties (cf. 5.1 (Chapitre 4)).
Tout l'enjeu est de trouver un juste milieu entre un apprentissage assez profond pour trouver un motif général tout en évitant un sur-apprentissage qui rendrait ce motif trop spécifique.

2.2 Les réseaux de neurones à convolution

Nous allons parler d'un type de réseau de neurones en particulier : le CNN (Convolutional Neural Network) car nous allons utiliser ce type d'architecture et nous allons voir pourquoi dans la partie 3.2.

Une convolution est le fait d'appliquer un filtre sur une matrice comme on peut le voir dans la figure 5 suivante.
Ici, le filtre est une matrice $3 * 3$ représenté avec un fond jaune et des chiffres rouges. Ce filtre se déplace le long de la matrice pour donner une valeur de sortie avec un pas de 1. Cela veut dire que le filtre se déplace d'une case par une case.
La matrice d'entrée a une dimension de $5 * 5$ et le filtre de $3 * 3$, on obtient une matrice de sortie de dimension $3 * 3$.

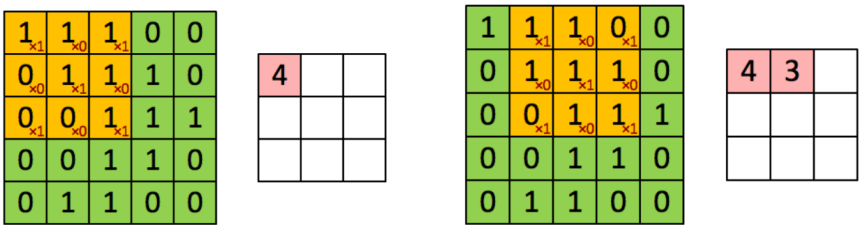


Figure 5 – *Les deux premières opérations de la convolution*

Un CNN est un ensemble de plusieurs couches de convolution couplé à des fonctions d'activation linéaires (ReLU ou tanh par exemple). Dans un réseau de neurones classique, chaque neurone source est connecté à une couche, on parle de couche entièrement connectée. Avec un CNN, ce n'est pas le cas. On utilise la convolution pour calculer la sortie à partir de l'entrée. Un CNN apprend automatiquement les valeurs de ses filtres selon les tâches qu'il a à effectuer (classification, segmentation...).

3 Le *Deep Learning* appliqué aux NLP

Nous avons donc présenté précédemment ce qu'était le NLP ainsi que le ML et le DL. Une représentation de leur intéraction peut être trouvée à la figure 6.

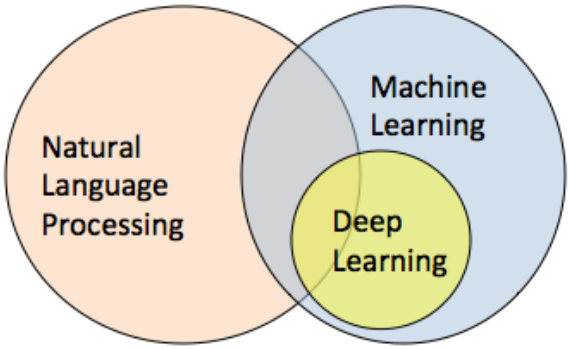


Figure 6 – *Lien entre NLP, ML et DL*

3.1 La représentation des mots

Comme nous le savons, les ordinateurs utilisent un langage binaire. Il faut alors traduire notre langue en une représentation numérique. Cette représentation ne doit pas seulement représenter le mot mais aussi son sens.

Pour cela, il existe plusieurs façon de créer ces représentation numérique, par des vecteurs ou des matrices [18]. Nous allons nous attarder sur les vecteurs.
Un vecteur de mot est un simple vecteur de poids. Il existe plusieurs façon de créer ces vecteurs, on parle de vectorisation. Une présentation des principaux "vectoriseurs" va suivre.

3.1.1 Bag of words

Bag of word (BoW) est un algorithme qui compte le nombre de fois qu'un mot apparaît dans un document. Cela permet de comparer des documents sur les similarités pour les classer par

exemple.

Pour mieux comprendre cet algorithme, prenons un exemple avec une phrase simple : *"Il fait beau aujourd'hui. Il fait presque chaud."*.
Le vocabulaire créé est alors constitué des mots uniques des document : ici, pour nos 2 phrases, le vocabulaire est : "il", "fait", "beau", "aujourd'hui", "presque", "chaud".

Nous pouvons alors créer les vecteurs :

— pour la première phrase : [1,1,1,1,0,0]
— pour la deuxième phrase : [1,1,0,0,1,1]

Le premier élément des vecteurs représente "il", le deuxième représente "fait" et ainsi de suite. Une phrase est alors représentée par un vecteur et chaque élément du vecteur représente un mot.

Dans cette approche, chaque mot est appelé un gram. Il est possible des créer des n-gram. Par exemple, si n=2, alors les mots uniques seront : "il fait", "fait beau", "beau aujourd'hui" pour la première phrase.

C'est un algorithme simple à comprendre et à implémenter. Malheureusement, il existe quelques limites :

— la taille du vocabulaire peut vite prendre de trop grandes proportions
— le contexte n'est pas pris en compte ici alors qu'il peut apporter de précieux renseignements

3.1.2 Word2vec

C'est un modèle très performant qui à l'inverse des algorithmes précédemment présentés tend à prendre en compte le contexte par l'intégration des mots voisins pour produire des vecteurs représentant les mots. Il consiste à représenter chaque mot par un vecteur dans un espace de grande dimension. Ces vecteurs sont calculés de sorte que les vecteurs proches correspondent à des mots qui apparaissent souvent dans le même contexte au sein d'un corpus. Il existe 2 type de méthodes d'entraînement :

— Skip-gram
— Continious Bag of Words (CBOW)

Skip-gram

Dans ce modèle, l'entrée est le mot cible et la sortie correspond aux les mots qui entourent le mot cible.
Par exemple, *Je suis un beau canard*, l'entrée sera *un* et la sortie sera *Je*, *suis*, *beau*, *canard* si la fenêtre prise en compte est égale à 5. Le modèle qui permet d'obtenir ces résultats contient une couche cachée et pour la couche de sortie, la fonction softmax est appliquée pour que chaque élément de la sortie décrive la probabilité qu'un mot apparaisse dans le contexte.

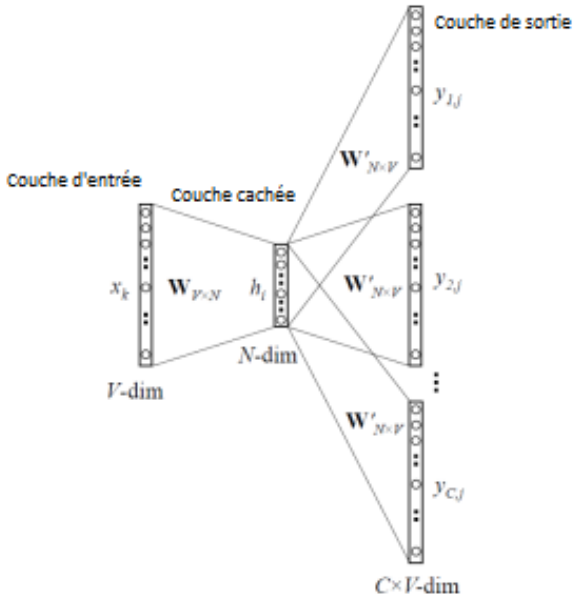


Figure 7 – *Architecture du skip gram*

Cette approche permet d'obtenir une information supplémentaire très précieuse : les relations entre les mots, les vecteurs sont alors plus "parlants". Ils permettent par exemple d'avoir des informations sur le genre ou les temps verbaux.

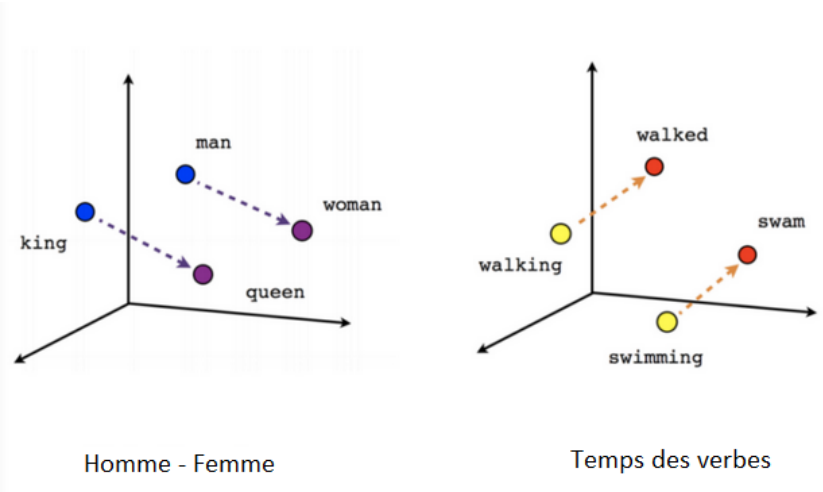


Figure 8 – *Exemples d'informations que l'on peut obtenir grâce à skip gram*

Continious Bag of Words

Le CBOW est très similaire à skip gram mais dans l'autre sens, c'est-à-dire que, cette fois-ci, l'entrée correspond aux mots entourant le mot cible et la sortie est le mot cible.
Si l'on reprend l'exemple précédent avec la phrase *Je suis un beau canard*, l'entrée sera *Je*, *suis*, *beau*, *canard* et la sortie sera *un*.
L'architecture du modèle est la suivante :

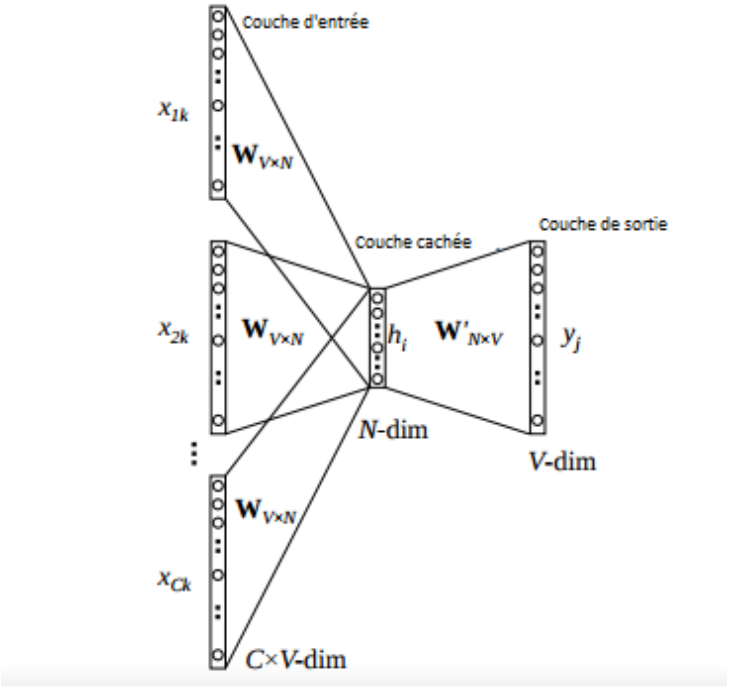


Figure 9 – *Architecture du CBOW*

Un exemple de l'algorithme peut être trouvé à la figure 10. Chaque mot est représenté par une suite de probabilités où chaque probabilité réfère à un critère.

Figure 10 – *Un exemple de représentation de l'algorithme Word2Vec*

3.2 Les CNN pour le langage

Les réseaux de neurones à convolution ont d'abord été expliqués pour des images.
En imagerie, les pixels proches ont de grandes chances d'être sémantiquement lié, ce qui n'est pas vraiment le cas pour les mots. La question est alors de savoir comment cela fonctionne exactement et si le niveau d'abstraction est assez élevé pour que cela soit significatif. Si nous nous basons que sur cette approche, il semble que CNN ne semble pas être la meilleure solution. Un autre type d'architecture RNN (Recurrent Neural Network) semble plus intuitif. Cependant, cela ne veut pas dire que CNN ne fonctionne pas. Au contraire, il s'avère que les CNN appliqués à des problèmes NLP donnent de bons résultats comme nous le montre cette article [20] qui montre que les CNN, même sans aucune base, ont une bonne habilité à comprendre le langage.

RNN : *Recurrent Neural Network*

Un RNN est représenté comme sur la figure 11. Sa spécificité est le fait de posséder des neurones récurrents, représentés en bleu.

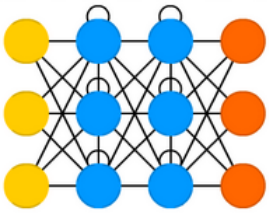


Figure 11 – *Architecture d'un RNN*

Un des avantages de ce genre de réseau de neurones, c'est qu'il prend en entrée des données de taille variable. Ils sont beaucoup utilisés dans le traitement du langage. Cela est dû au fait que, comparé aux autres réseaux de neurones, ses entrées sont liées les unes aux autres, ce qui lui permet de se souvenir des liens entre les mots et ainsi prendre en compte le contexte. Cela permet d'avoir une sortie de meilleure qualité. Cette prise en compte du contexte est parfaitement adapté à notre problème, sauf que les RNN souffrent d'un gros désavantages : ils créent des liens qui au fur et à mesure du nombre d'entrée, perdent en mémoire ces liens. Cela amène souvent à une performance faible.

CNN : Convolutional Neural Network

Un CNN a une architecture définie comme sur la figure 5 (Chapitre 5).

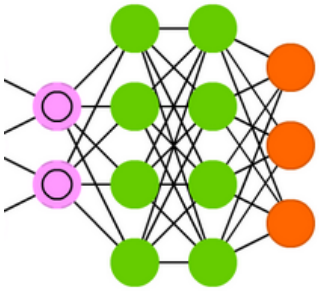


Figure 12 – *Architecture d'un CNN*

Le modèle Bag-of-word a été, pendant des années, l'approche standard, étant donné qu'il amenait de bons résultats. L'argument principal de l'utilisation des CNN est sa rapidité. Comparé aux n-grams, les CNN sont aussi performants en terme de représentation. On peut se rendre compte que les n-grams utilise beaucoup de ressource car même Google n'a pas souhaité mettre des jeux de données après 5-grams, alors qu'il est tout à fait raisonnable d'utiliser des filtres plus grand que 5 pour les CNN.

Malgré le fait que l'on perde cette récursivité et la prise en compte du contexte, cela pourra être compensé par l'utilisation de Word2Vec.

4 Où en sommes-nous aujourd'hui ?

Les approches de *Machine Learning* pour la détection de *fakes news* sont à leur début. Cependant il existe quelques travaux de premières approches avec des résultats atteignant une précision aprochant les 90% de fiabilité pour des jeux de données particuliers.

Actuellement, il existe des détecteurs pour la fiabilité des articles que les utilisateurs lui soumettent comme **ClaimBuster** [4] ou encore il existe des bases de données concernant la

fiabilité cette fois-ci de sites web comme **Decodex** [15]. Ces derniers proposent aussi un plug-in navigateur qui permet de vérifier les sites visitées.

Cependant, ces outils ne sont qu'à leurs prémices et leur exactitude est encore loin de la perfection.

Plusieurs travaux proposent des méthodes pour détecter automatiquement les *fakes news*. Nous allons nous attarder sur certains de leur approche.

fakes news Challenge

Une des publications décrit une approche "simple but tough-to-beat" (simple mais difficile à battre) [11]. Effectivement, le modèle utilisé est composé de caractéristiques lexicales et de similarités alimenté par un perceptron multi-couche (MLP) avec une couche cachée (hidden layer). Cet article entre dans le cadre d'une compétition appelé *fakes news* Challenge [9]où ils se sont hissés à la troisième place.
Le fait de privilégier une architecture plus simpliste et un unique modèle, à l'inverse des CNN ou des ensembles, permet une plus grande facilité de compréhension de ces algorithmes et ainsi déterminer les paramètres qui ont une plus grande influence sur les résultats. Un jeu de données de qualité, c'est-à-dire avec un nombre de données suffisant par type de label, semble être un facteur de réussite important. Cet article permet de mettre en avant des modèles moins profonds pouvant donner d'excellents résultats.

Toujours dans cette même compétition, la deuxième équipe a opté pour une approche avec MLP pour le modèle d'apprentissage et ils ont mis l'accent sur le prétraitement des données pour extraire un maximum d'informations (features). [3].
Pour finir, l'équipe qui est arrivée première a proposé un ensemble composé d'un gradient-boosted decision trees et d'un CNN profond [6].

TI-CNN : un modèle combinant textes et images

Une autre publication nous décrit TI-CNN (Text and Image information based on Convolutional Neural Network) [13] qui, comme son nom l'indique, utilise 2 sources de données : le texte et les images provenant d'articles.
La précision lorsque l'on combine ces deux sources de données externes atteint 0.92. Leur modèle est décomposé en 4 parties : une partie pour le texte explicite, une autre pour le texte latent, une suivante pour les images explicites et, enfin, une partie pour les images latentes. Ce qui est décrit d'explicite correspond à l'analyse du texte comme la taille des articles, le nombre de phrases... Le côté latent correspond aux informations utilisées en entrée pour un CNN. De plus, pour la partie texte, l'algorithme word2vec est utilisé.
Un des challenges avec ce genre de modèle est qu'il faut obtenir un jeu de données qui combine du texte avec des articles dits fiables et d'autres dits non fiables ainsi que des images d'une qualité suffisante et avec les deux labels : images non retouchées et images retouchées.

La comparaison entre CNN et RNN

Un autre article, qui me paraissait important de citer est "Early Detection of *fakes news* on Social Media Through Propagation Path Classification with Recurrent and Convolutional Networks" [14] qui compare une dizaine d'approches différentes comme un RNN et un CNN ainsi que sa combinaison. On remarque que le CNN surpasse légèrement le RNN et que la combinaison des deux donne le meilleur résultat.

Les différents vectoriseurs appliqués pour un problème de fact checking

Pour finir, [2] est un article qui compare les différentes vectoriseurs tels que tf-idf ou bag-of-word ou encore word2vec. Il semblerait qu'avec leur modèle, qui est un DNN (Dense Neural Network).
Il utilise aussi le jeu de données fourni par *fakes news* Competition. Le modèle utilisé (DNN) reste le même pour pouvoir se concentrer sur les vectoriseurs. Leur résultat annonce que la méthode tf-idf est la meilleure avec une précision d'environ 94%, suivie du bag-of-word avec 89% et, finalement, word2vec avec environ 75% de précision. Cette fois-ci, si le modèle utilisé est un MLP, la méthode bag-of-word atteint 92%.

Tous ces articles tendent à utiliser un pré-traitement du texte comme la suppression des stop-words et l'utilisation d'un vectoriseur, bag-of-word ou word2vec puis la création d'un modèle qui semble être en majorité un DNN. Il serait intéressant de voir les performances d'un CNN couplé à ces pré-traitements étant donné que ce dernier peut atteindre des performances très intéressantes même pour ce genre de problématique.

4 Analyse et conception

1 Jeu de données

Le jeu de données utilisé provient du site Kaggle.
Il est composé de 20800 articles. Pour les données, il est fourni pour chaque article : le titre, le corps de l'article et les labels 0 pour fiable et 1 pour non fiable sont fournis. Les données sont divisées en 2 fichiers :

— train.csv qui contient le titre, le corps de l'article, son auteur et son label.
— test.csv qui contient le titre, le corps de l'article et son auteur.

La distribution est la suivante :

	Fiable	Non fiable
Nombre d'articles	10387	10413

2 Pré-traitement du langage

Suppression de la ponctuation et mise en minuscule

Le fait de supprimer la ponctuation permet d'enlever des informations non significatives. De même que le fait de mettre en minuscule permet de regrouper des mots qui peuvent être les mêmes en les harmonisant.

Le résultat obtenu est le suivant :

Séparer les mots

Pour pouvoir appliquer un algorithme comme word2vec, il faut qu'il puisse extraire chaque mot de chaque phrase pour chaque article. Nous allons donc séparer tous les mots, tous en faisant attention de bien garder l'ordre des articles pour que les labels correspondent aux bons articles.

```
'House Dem Aide: We Didn't Even See Comey's Letter Until Jason Chaffetz Tweeted It
By Darrell Lucus on October 30, 2016 Subscribe Jason Chaffetz on the stump in
American Fork, Utah ( image courtesy Michael Jolley, available under a Creative
Commons-BY license) \nWith apologies to Keith Olbermann, there is no doubt who the
Worst Person in The World is this week-FBI Director James Comey.'

'House Dem Aide  We Didn t Even See Comey s Letter Until Jason Chaffetz Tweeted It
By Darrell Lucus on October          Subscribe Jason Chaffetz on the stump in
American Fork  Utah   image courtesy Michael Jolley  available under a Creative
Commons BY license   With apologies to Keith Olbermann  there is no doubt who the
Worst Person in The World is this week FBI Director James Comey '
```

Figure 1 – *Avant et après suppression de la ponctuation.*

Supprimer les stop words

Les stop words sont des mots très courants, ici en anglais, et ils ralentissent la progression de l'apprentissage.
Pour cela, nous allons utiliser la bibliothèque NLTK [16] qui possède une liste par défaut des stopwords. Le résultat obtenu est le suivant :

```
['house', 'dem', 'aide', 'we', 'didn', 't', 'even', 'see', 'comey', 's', 'letter',
'until', 'jason', 'chaffetz', 'tweeted', 'it', 'by', 'darrell', 'lucus', 'on',
'october', 'subscribe', 'jason', 'chaffetz', 'on', 'the', 'stump', 'in',
'american', 'fork', 'utah', 'image', 'courtesy', 'michael', 'jolley', 'available',
'under', 'a', 'creative', 'commons', 'by', 'license', 'with', 'apologies', 'to',
'keith', 'olbermann', 'there', 'is', 'no', 'doubt', 'who', 'the', 'worst',
'person', 'in', 'the', 'world', 'is', 'this', 'week', 'fbi', 'director', 'james',
'comey']

['house', 'dem', 'aide', 'even', 'see', 'comey', 'letter', 'jason', 'chaffetz',
'tweeted', 'darrell', 'lucus', 'october', 'subscribe', 'jason', 'chaffetz',
'stump', 'american', 'fork', 'utah', 'image', 'courtesy', 'michael', 'jolley',
'available', 'creative', 'commons', 'license', 'apologies', 'keith', 'olbermann',
'doubt', 'worst', 'person', 'world', 'week', 'fbi', 'director', 'james', 'comey']
```

Figure 2 – *Avant et après suppression des stop words.*

3 Conversion mot vers vecteur : Word2Vec

Comme vectoriseur, l'utilisation de word2vec semble être cohérente. Elle offre une compréhension du contexte bien plus importante que des méthodes comme bag-of-word ou encore tf-idf qui ne sont que des statistiques et ne prennent pas en compte le contexte. Comme nous l'avons évoqué, une des limites de ce modèle n'est pas l'algorithme en soi mais plutôt la qualité de la données.

Le modèle déjà entraîné : pretrained word2vec

Google met à disposition un modèle qui comprend un vocabulaire de 3 millions de mots. Ils ont été entraînés sur plus de 100 millions de mots du jeu de données Google News.
Ce modèle est constitué de vecteurs d'une longueurs de 300 critères. On s'aperçoit aussi que, dans ce modèle, certains *stop words* sont exclus (*a*, *and*...) mais d'autres sont inclus (*the*, *also*...).

4 Modèle

4.1 Modèle de référence

L'architecture du modèle de référence est la suivante :

- une couche Embedding
- une couche Dropout
- une couche Convolution en 1D
- une couche MaxPooling
- une couche Dense avec une fonction d'activation relu
- une couche BatchNormalizarion
- une couche Dropout
- une couche Dense avec la fonction d'activation relu
- une couche BatchNormalization
- une couche Dense avec la fonction d'activation sigmoid

Ce modèle s'inspire fortement de l'article cité précédemment dans 4 (Chapitre 3) : le TI-CNN [13]. Il peut être amener à évoluer dans la deuxième partie de ce PRD.

4.2 Explications des différentes couches

La couche Embedding

C'est une simple matrice qui transforme des mots en leur représentation vectorielle. Dans la bibliothèque Keras, plusieurs paramètres sont à renseigner comme la taille du vocabulaire, la dimension de sortie, les matrices

La couche Convolution

Nous avons déjà expliqué ce qu'était une convolution dans 2.2 (Chapitre 3).

La couche Pooling

Le pooling revient à sélectionner un échantillon de la matrice d'entrée. Il existe plusieurs types de pooling :

- average pooling : fait la moyenne dans la fenêtre définie
- max pooling : prends la valeur maximum dans la fenêtre définie

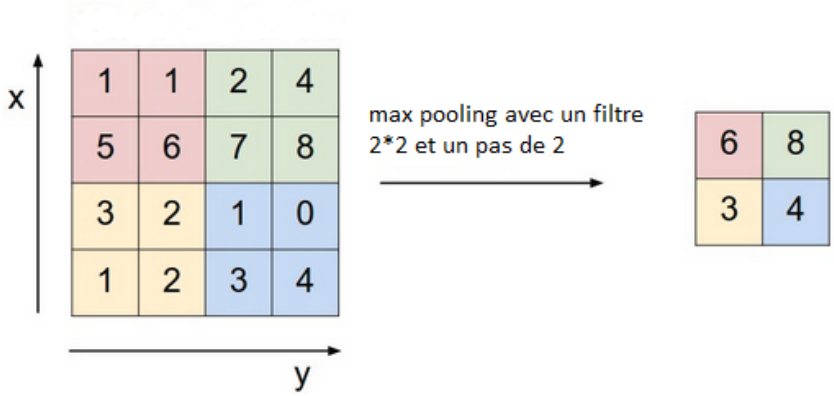


Figure 3 – *Exemple de l'effet d'un maxpooling sur une matrice.*

Pourquoi choisir le max-pooling ?
Il permet de réduire la taille de la sortie alors que l'on garde les activations les plus élevés. La mise en commun introduit une sorte d'invariance de traduction dans le réseau.

La couche BatchNormalization

Cette couche normalise les données à la sortie de cette couche. Cela permet de ne pas avoir à prendre en charge des échelles trop importantes.

La couche Dropout

Le Dropout a pour but de limiter le sur-apprentissage. Le fonctionnement est le suivant : à chaque couche Dropout, il sélectionne aléatoirement des neurones et met leurs poids à 0.

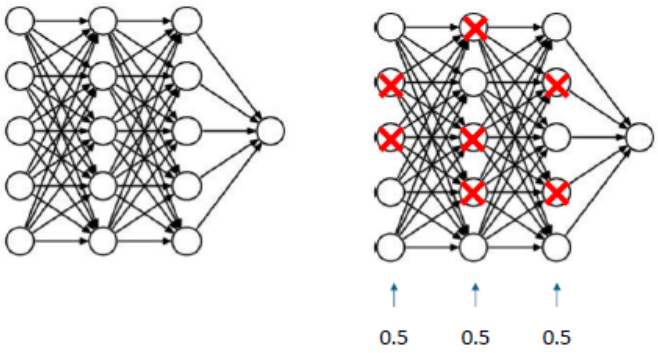


Figure 4 – *Représentation d'un dropout avec 50% de perte.*

La couche Dense

C'est une couche normale de neurones. Chaque neurone reçoit l'entrée de la couche précédente.

La fonction de perte

Pour ce genre de problème, nous allons utilisé la fonction de perte nommée cross entropy.

$$\mathrm{CE} = \sum_{i}^{\mathrm{C}} t_i log(s_i) \tag{1}$$

où t_i est la vérité terrain et s_i est le score provenant du CNN pour chaque classe. Souvent un fonction d'activation est appliquée avec, comme la fonction sigmoid ou softmax.

Pour ce genre de problème, nous allons utiliser la fonction de perte categorical cross-entropy, car nous utilisons un CNN pour classifier les textes en 4 classes et un texte ne peut appartenir à plusieurs classes. Cette fonction est déjà implémenté dans la bibliothèque Keras.

5 Apprentissage

5.1 Découpage du jeu de données

Le jeu de données sera divisé en 2 parties :

— le jeu d'entraînement (80%)
— le jeu de test (20%)

Comme son nom l'indique, le jeu de test ne sera utilisé que pour tester la méthode, c'est-à-dire lorsque l'on considère que la méthode est prête à être testée avec des données qu'elle n'a jamais vues. La méthode ne verra jamais le jeu de test durant son apprentissage.

Validation croisée

Il faut cependant pouvoir évaluer notre modèle durant la phrase d'apprentissage. Pour cela, nous allons mettre en place une validation croisée.
Une validation croisée est le fait de diviser le jeu d'entraînement en k parties de taille égale. Chaque partie jouera le rôle du jeu de test (que l'on appellera jeu de validation) et les autres parties seront utilisées pour l'entraînement.

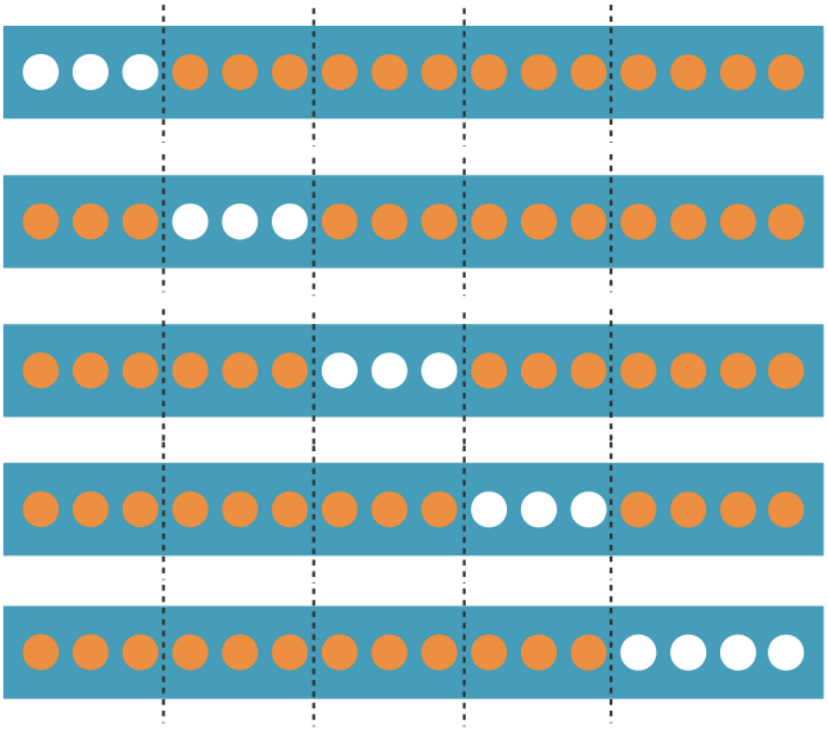


Figure 5 – *Représentation d'une validation croisée avec k=5*

On peut alors faire la moyenne des prédictions de chaque partie pour avoir une approximation de la performance de notre modèle.

Il existe plusieurs type de validation croisée, la plus simple étant celle décrite précédemment. On peut procéder à une validation croisée *leave-one-out* qui consiste à s'assurer que chaque pli est un échantillon unique. On peut aussi faire une validation croisée *shuffle-split*. On échantillonne de manière aléatoire un nombre prédéfini de données pour le jeu d'apprentissage et un autre pour le jeu de test.

Stratification

En plus de la validation croisée, on va mettre en place la stratification. Cela consiste à garder une homogénéité des proportions des classes (ici, fiable et non fiable). Cela peut être très utile lorsqu'il y a un gros déséquilibre entre les classes et permet ainsi au modèle de voir et apprendre sur les classes équitablement.

5.2 Hyper-paramètres

Lorsque le modèle sera défini, il restera à déterminer les paramètres optimaux pour ce problème. Pour cela, nous allons mettre en place une *grid search*. Le principe est le suivant : il faut définir une série de valeur pour chaque paramètre que l'on veut tester et cette recherche comparera les performance de chacun pour en déduire le meilleur. Lorsque les valeurs sont définies, la grid seach essaie chaque combinaison possible.
Par exemple, si nous avons 2 paramètres à optimiser avec 4 valeurs possibles pour chacun, cela fait 16 combinaisons à tester. On se rend vite compte que cette méthode peut vite exploser au niveau du temps de calcul.

Une autre solution pourra être la *random search* qui utilise le même principe, sauf qu'au lieu de tester toutes les combinaisons possibles, elle va sélectionner aléatoirement les valeurs à tester. Cela peut être intéressant lorsque le nombre de paramètres à tester et de combinaisons totales est très grands.

6 Test

6.1 Matrice de confusion

Une matrice de confusion sert à mesurer la qualité d'un système de classification.
Voici à quoi ressemble celle-ci :

		Classe réelle	
		-	+
Classe prédite	-	True Negatives *(vrais négatifs)*	False Negatives *(faux négatifs)*
	+	False Positives *(faux positifs)*	True Positives *(vrais positifs)*

Figure 6 – *Matrice de confusion entre 2 classes.*

VP signifie vrai positif, c'est-à-dire que l'élément a été correctement prédit dans la bonne classe +.
VN signifie vrai négatif, c'est à dire que c'est un élément qui a été correctement prédit dans la bonne classe -.
FP signifie faux positif, c'est-à-dire que l'élément a été mal prédit, ici il a été prédit de la classe + alors qu'il appartient à la classe -.
FN signifie faux négatif, c'est-à-dire que l'élément a été mal prédit, ici il a été prédit de la classe - alors qu'il appartient à la classe +.

Lors de l'apprentissage ou même de la phase de test, la mise en place et l'affichage de la matrice de confusion peut nous aider à comprendre dans quel domaine l'algorithme a des difficultés et ainsi cibler ces cas plus difficile à détecter.

6.2 Jeu de données test

Comme nous l'avons évoqué précédemment, nous utilisons un jeu de test. Ce jeu de test n'aura, à aucun moment, été utilisé lors de l'apprentissage de notre modèle. Ce sont des données nouvelles que le modèle voit pour la première fois. Cela nous permet d'avoir un résultat "objectif" quant à la précision de notre modèle et sa généralisation.
La mesure de test aura été définie auparavant : c'est une mesure d'accuracy. Cette exactitude de classification est le nombre de prédictions correctes par rapport à toutes les prédictions faites.

6.3 Comparaisons

Il peut être judicieux de comparer les performances de notre modèle avec des approches naïves.

Par exemple, on peut dire que chaque entrée donnera le label *fausse information*.
Avec le même principe, on peut affecter aléatoirement une classe pour chaque entrée.

Cette comparaison permet de tester à quel point notre modèle a appris (ou non) et si les résultats restent cohérents.

Partie développement

5 Développement

1 Données

Les données utilisées sont détaillées dans la partie 5.1 (Chapitre 4).

Pour obtenir une première idée du contenu de notre jeu de données et de ce qui pourrait caractériser une *fake news*, il est possible d'afficher nos données sous forme de nuage de mots.

Figure 1 – *Mots les plus fréquent dans les articles fiables*

Pour les articles fiables, on remarque que les mots les plus présents sont : *Donald*, *Trump*, *New*, *Fox*...

Figure 2 – *Mots les plus fréquent dans les articles non fiables*

Pour les articles non fiables, on remarque que les mots les plus fiables sont : *Trumps*, *Hillary*, *Home*, *Donald*, *US*...

On pourrait se dire qu'il suffit de calculer le nombre de fois qu'un mot apparaît et cela nous aiguille sur l'appartenance à un article non fiable. Cependant, on remarque que les mots *Donald* ou encore *Trump* sont des mots que l'on retrouve fréquemment dans les 2 catégories. Il semble alors pertinent de prendre en compte le contexte et de ne pas se limiter à une approche purement statistique.

Ces mots de nuages nous confirment que ce jeu de données a une dominante politique et qu'il se situe dans la période des élections présidentielles américaines de 2016.

2 Traitement des données

Pour obtenir les meilleurs performances possibles, il faut que les données soient traitées et épurées.
Dans notre cas, on s'oriente vers l'extraction des mots les plus courant dans les articles.

Après ce traitement, chaque phrase va alors être séparer en mots. Tous les mots ne seront alors pas gardés comme les mots vides (*and*, *the*, *or*...) ou encore les balises HTLM ainsi que la ponctuation.
Cependant, concernant la ponctuation, il nous a semblé judicieux d'en garder une trace. Pour cela, nous calculons l'occurrence de chaque symbole. De plus, nous avons aussi relevé le nombre de lettres en majuscule. On peut faire l'hypothèse que les articles moins fiables ont sont plus ponctués et avec plus de majuscules.

3 Vectorisation

La vectorisation est le fait de convertir nos mots en nombre. Pour cela, nous utilisons l'algorithme *Word2Vec*. Nous allons aussi ajouter quelques données statistiques tels que le nombre de points d'exclamation dans les articles.

3.1 Word2Vec

Il existe 2 approches :

— la création d'un modèle word2vec que l'on entraîne sur nos propres données
— l'utilisation d'un modèle déjà entraîné que Google a mis en libre service.

Création d'un modèle Word2Vec

Pour le premier cas, la création de notre propre modèle, nous avons décidé d'utiliser les paramètres suivants :

Paramètres	Valeur
workers	4
size	100
windows	10
min_count	40
sample	1e-3

Nous avons alors notre modèle Word2Vec entraîné. Il va falloir en extraire les informations qui vont être utile pour orienter notre réseau de neurones en initialisant ses poids à des valeurs judicieusement choisies. Mais avant cela, il faut que notre texte soit transformé en numérique.

Utilisation d'un modèle Word2Vec déjà entraîné

Le principe est dans l'ensemble le même. Sauf que cette fois-ci, le modèle n'a pas besoin d'être créer, ni entraîner, il suffit de le charger comme modèle Word2Vec et répéter le même procédé.

Représentation d'un modèle Word2Vec

Comme dit précédemment, chaque mot est représenté par un vecteur. Un exemple serait de regarder la représentation multi-dimensionnelle du mot *Trump* et l'on obtient le résultat suivant :

```
word2vec['trump'] =

    [ 5.40193031e-03, -6.62558898e-02,
    -5.20286933e-02, -1.00536630e-01,
       ..., ...,
     -1.35854378e-01, -6.58609197e-02,
     1.06397025e-01,  2.08864193e-02]
```

3.2 Tokenizer

Pour transformer nos articles en une séquence de numériques, nous allons utiliser la fonction *Tokenizer* fournie par Keras qui permet de convertir du texte en vecteur. Pour cela, on précise que l'on veut garder les 3000 mots les plus courants de nos articles. Puis, on va lui soumettre les mêmes articles pour qu'il puisse encoder chaque phrase selon le dictionnaire de 3000 mots que nous avons créé précédemment.

On peut prendre l'exemple suivant :

"Le chat est gris."
"C'est un chat gris."
"Un chat."

On compte alors le nombre de fois qu'un mot apparaît dans le corpus. On va alors les classer par ordre décroissant puis leur affecter un rang.
Et on obtient les résultats suivants :

Mots	Occurrence	Index
"le"	1	6
"chat"	3	1
"est"	2	4
"gris"	2	2
"c"	1	5
"un"	2	3

On peut alors traduire nos phrases selon les rangs définis précédemment et ensuite obtenir :

$$\text{"Le chat est gris."} \longrightarrow [6, 1, 4, 2]$$
$$\text{"C'est un chat gris."} \longrightarrow [5, 4, 3, 1, 2]$$
$$\text{"Un chat."} \longrightarrow [3, 1]$$

Malheureusement, les séquences retournées par la fonction précédente peuvent avoir des tailles différentes. Or, notre réseau de neurones à convolution doit avoir une entrée où les vecteurs ont une taille fixe. Pour cela, nous allons faire appel à du *padding*. Pour se faire, nous prenons le vecteur de plus grande valeur, puis, pour tous les autres vecteurs on ajoute autant de 0 qu'il faut pour que le vecteur est la même taille que le vecteur de plus grand.

$$[6, 1, 4, 2] \longrightarrow [0, 6, 1, 4, 2]$$
$$[5, 4, 3, 1, 2] \longrightarrow [5, 4, 3, 1, 2]$$
$$[3, 1] \longrightarrow [0, 0, 0, 3, 1]$$

Nous avons maintenant des données prêtes à être présentées à notre réseau de neurones à convolution.

Pour l'instant, nous n'avons pas encore utilisé notre modèle Word2Vec. Les rangs précédemment déterminés vont correspondre à l'index de notre matrice *embedding* créée à partir de notre modèle word2vec. Tous les mots, c'est-à-dire 3000, vont être énumérés un à un pour récupérer la correspondance dans notre modèle Word2Vec et ainsi créer la matrice *embedding* que nous fournirons à la première couche de notre réseau de neurones. Cette matrice sera alors composée des 3000 mots les plus présents de notre jeu d'apprentissage et chaque mot sera représenté en 100 dimensions.

Cela va servir lors du mappage dans la première couche du réseau de neurones : la couche *embedding* va pouvoir faire le lien entre les mots présents dans le texte en prenant son rang avec sa position dans la matrice *embedding* qui est la représentation multi-dimensionnelle de ces mots.

3.3 Prise en compte de la syntaxe

Une des hypothèses que l'on peut émettre sur les fake news est que la syntaxe peut ne pas respecter certains codes. Un exemple peut être la mise en majuscule d'un titre ou la succession de point d'exclamation pour rentre le titre plus accrocheur, visible et ainsi attirer le lecteur à le lire.

La première version qui a été appliquée à notre modèle est un simple compteur de point d'exclamation, point d'interrogation ainsi que le nombre de majuscule. Les résultats peuvent être trouvé dans la section 8.1. Cela a bien sûr été appliqué avant la suppression de la ponctuation ainsi que la mise en minuscule de tous les mots.

Dans un second temps, on a décidé de prendre en compte la présence des hyperliens. On peut supposer qu'un article fiable va tendre à citer ses sources pour justifier ses dires et donc ajouter le lien de ces sources.

4 Modèle

Pour l'implémentation du modèle, on utilise la bibliothèque Keras écrite en python. Elle est open source. Elle est interfaçable avec Tensorflow, une autre bibliothèque open source écrite en

python.
L'avantage de Keras est qu'elle permet de prendre rapidement en main les réseaux de neurones car elle met à disposition une API simple et cohérente. Elle permet aussi de pouvoir personnaliser facilement les implémentations déjà existantes. De plus, elle fonctionne sur CPU et GPU.

Comme vu précédemment, les données ont été converties en matrice. Cette matrice a une taille de 18720 par 12072. Elle va être présentée au réseau de neurones sous forme de lot, ici ce sont des lots de 40 articles. Le réseau verra alors 468 lots pour voir l'entièreté du jeu d'apprentissage.

Le modèle final comprend les couches suivantes :

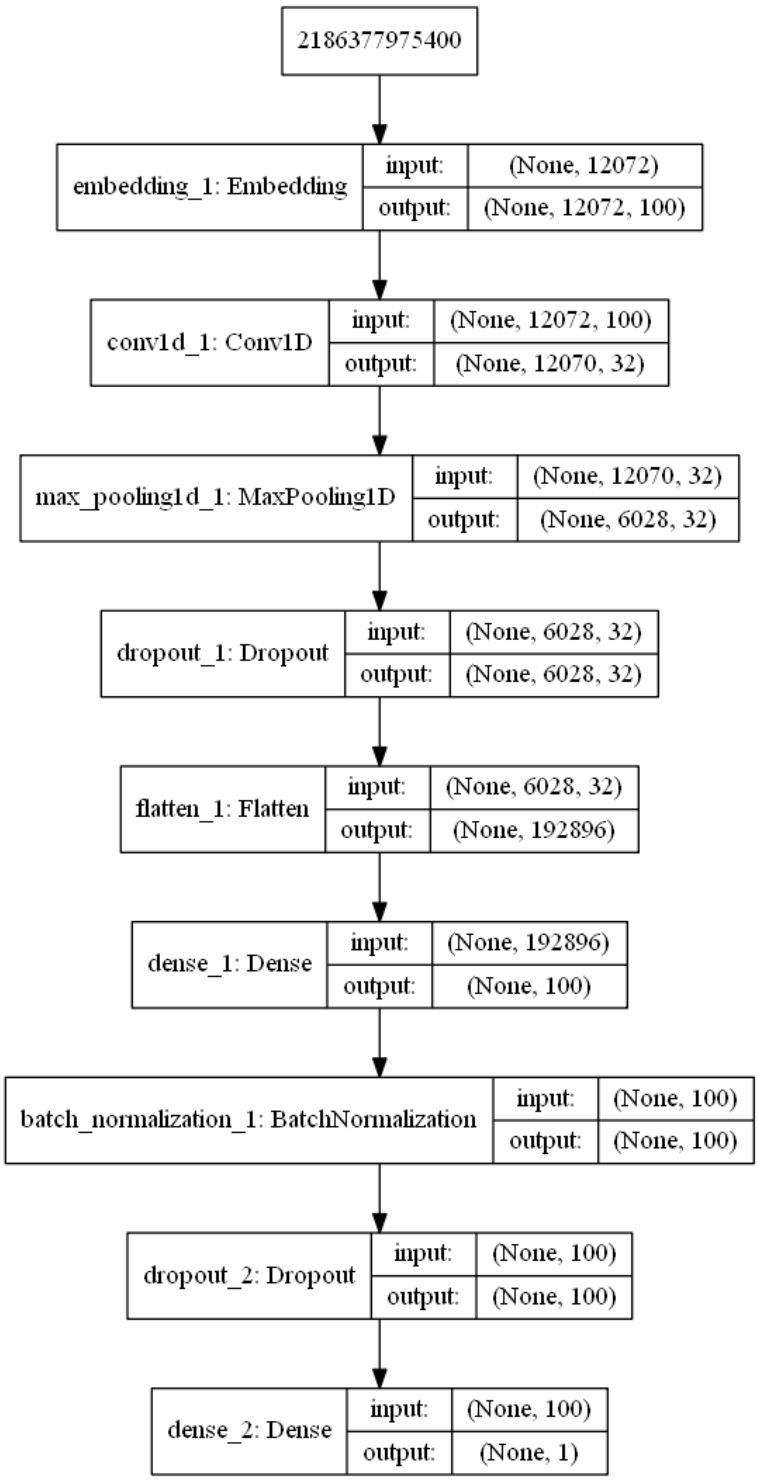


Figure 3 – *Architecture et nombre de paramètres du modèle utilisé*

La couche *Embedding* sert à mapper nos articles transformés en séquences d'entiers avec la matrice *embedding* qui a été construite grâce à notre modèle Word2Vec. Chaque article avait été construit avec 12072 caractéristiques (qui correspondent aux mots) qui vont elles-mêmes être les 100 caractéristiques qui correspondent à leur représentation spatiale extraites du modèle Word2Vec.

La couche de *Convolution* va permettre de réduire cette dernière dimension, c'est-à-dire que l'on va passé d'une représentation spatiale de 100 caractéristiques à 32 caractéristiques.

La couche *MaxPooling* va permettre encore de réduire drastiquement la dimension de la matrice étant donné que l'on passe de 12070 caractéristiques à 6028 caractéristiques.

Les 2 couches *Dropout* sont là pour éviter le surapprentissage avec un paramètre de 0.2 : c'est-à-dire que l'on perd 20% de nos neurones totaux.

La première couche *Dense* sert à appliquer une transformation non linéaire en appliquant la fonction *reLu*. C'est une couche qui est entièrement connectée.

La couche *BatchNormalisation* permet de normaliser les données.

La deuxième couche *Dense* permet de nous donner l'appartenance à la classe *fake news* avec en sortie un nombre compris entre 0 et 1.

Les callbacks

La bibliothèque Keras met à disposition des *callbacks* : ce sont un ensemble de fonctions qui sont appliquées tout au long de l'apprentissage du réseau.

Les *callbacks* utilisés sont :

- *ModelCheckpoint* : il permet de sauvegarder le modèle après chaque époques, on peut même préciser que l'on ne veut que sauvegarder le modèle si il est meilleur que le précédent selon la fonction de perte de notre jeu de validation.
- *EarlyStopping* : cela arrête l'apprentissage quand l'*accuracy* a cessé de s'améliorer pendant un certain nombre d'itération.
- *ReduceLROnPlateau* : cela permet de préciser la *learning rate* de départ, puis elle sera diminuée à chaque fois que les performances de notre réseau de neurones stagneront jusqu'à une certaine valeur limite que l'on aura précisé.
- *TensorBoard* : c'est un outil de visualisation fournit par Tensorflow.

5 Compilation du modèle

La fonction de perte utilisée est la binary crossentropy qui est la fonction de perte la plus utilisée lorsque le problème est binaire. Ici, soit l'article est une *fake news*, soit il ne l'est pas.

Pour l'optimiseur, nous utilisons Adam avec les paramètres suivants : la *learning rate* commence avec une valeur de 0.01 puis va diminuer de 0.05 pour chaque plateau (c'est-à-dire que la valeur de la fonction de perte de notre jeu de validation ne s'améliore pas) jusqu'à atteindre la valeur de 0.001.

6 Hyper-paramètres

L'utilisation d'une recherche par grille nous a permis de trouver les paramètres nous permettant d'obtenir les fiabilités maximales. Les paramètres utilisés sont les suivants :

epochs	10
batch size	40
loss	binary crossentropy
optimizer	adam

Les *epochs* correspondent au nombre de fois que le jeu d'apprentissage sera vu en entier par le modèle.
Le *batch size* correspond au nombre d'articles montré à notre modèle avant que les poids soient mis à jour. Plus le modèle verra d'articles en même temps, meilleur il sera. Cependant, plus ce nombre est élevé, plus le besoin de ressources augmente étant donné qu'il faut pouvoir gardé en mémoire les articles lus.

7 Apprentissage

Nous avons entraîné le modèle durant 20 époques obtenir les graphiques suivants.

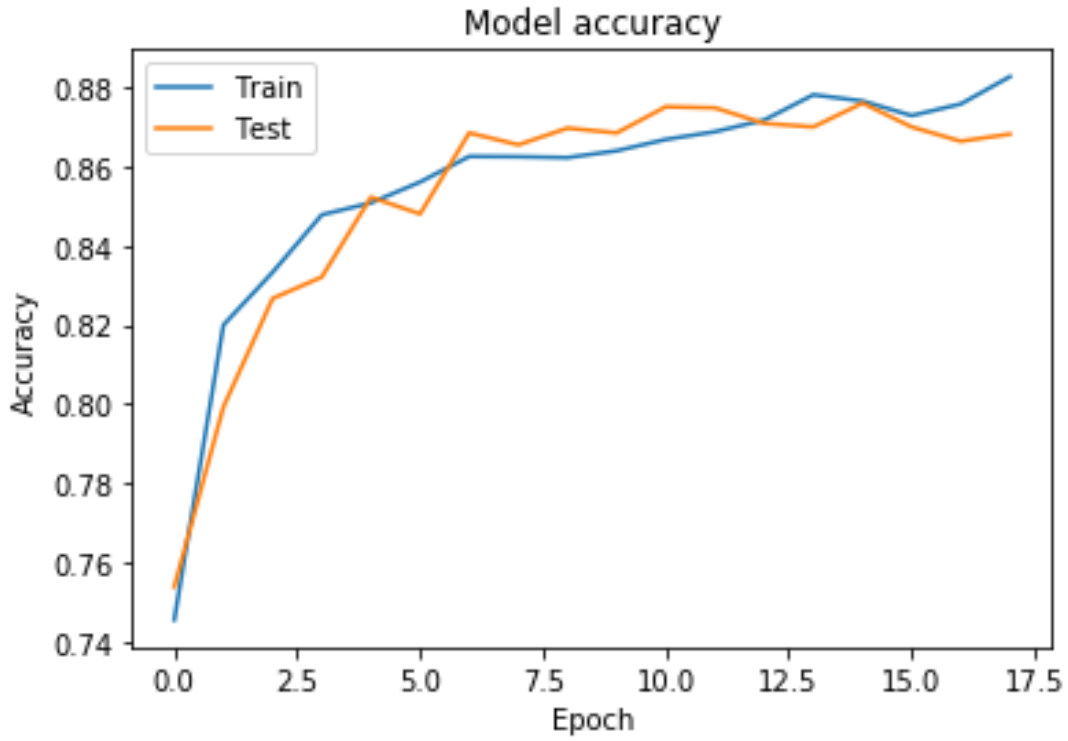


Figure 4 – *Graphique d'apprentissage selon la fiabilité*

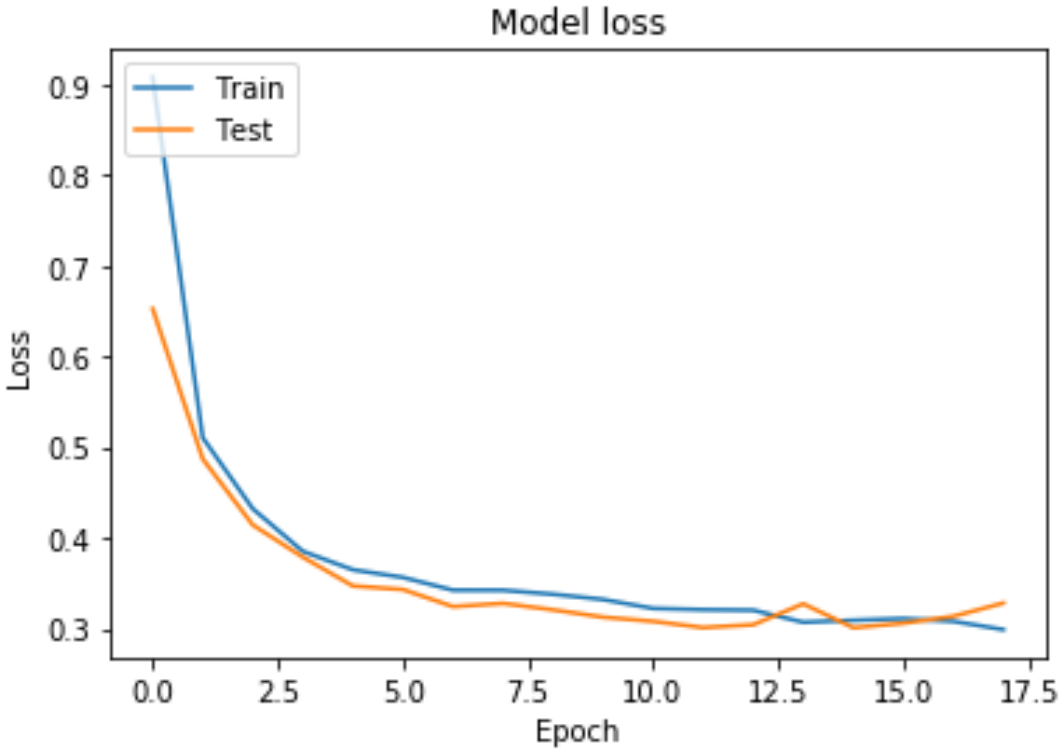


Figure 5 – *Graphique d'apprentissage selon la fonction de perte*

Ce graphique nous permet de nous rendre compte que le modèle apprend au fur et à mesure et tend à minimiser la fonction de perte. Plus cette fonction diminue, plus le taux d'erreur devient faible. On peut aussi relever que le modèle commence à être en sur-apprentissage aux environs de l'époque 13. On peut le voir par une chute de la courbe orange pour le graphique d'apprentissage selon l'exactitude et inversement, on remarque que pour la courbe d'apprentissage selon la fonction de perte, la courbe orange tend à remonter.

L'apprentissage final s'effectuera avec un découpage en 10-90, c'est-à-dire que le jeu de validation représentera 10% du jeu total et le reste composera le jeu d'apprentissage.
Dû à des problèmes de sur-apprentissage, on a finalement entraîné le modèle durant 10 époques.
Le paramétrage du réseau de neurones à convolution peut être trouvé dans la section 1 (Annexe C).

8 Résultats et interprétation

8.1 Résultats

Dans un premier temps, le modèle 4.1 (Chapitre 4) a été entraîné dans le but de trouver les meilleurs paramètres par l'utilisation d'une *grid search*. Puis, dans un second temps, on a entraîné notre modèle avec les paramètres optimaux.
Les résultats sont les suivants :

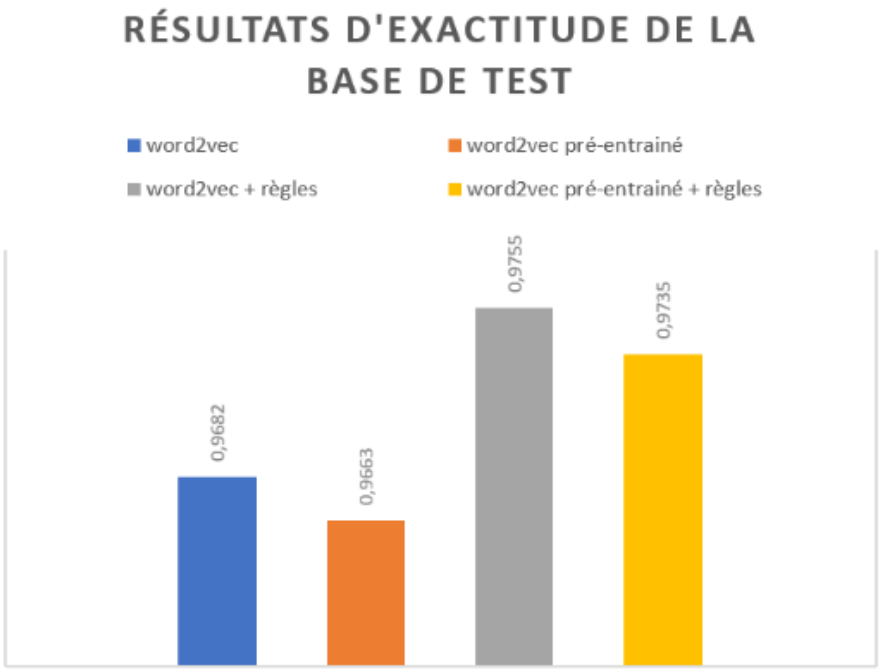


Figure 6 – *Résultats des différents modèles*

Le meilleur taux est de 0.9755 qui a été obtenu en utilisant le modèle Word2vec que l'on a entraîné à partir de notre jeu de données et où l'on a ajouté les règles de syntaxe.
Ces résultats ont été obtenu avec le paramétrages qui peut être trouvé en annexe.

8.2 Interprétation

On se rend compte que notre modèle nous montre des résultats très encourageant.
Si l'on regarde les résultats de plus près, on se rend compte que le modèle Word2Vec que nous avons nous-même entraîné à partir du jeu de données, a de meilleure performance que le modèle pré-entraîné de Google. Cela peut être expliqué par le fait que notre jeu de données est assez spécifique, il se centre sur le domaine de la politique alors que le modèle Word2Vec de Google a été entraîné sur des milliards de mots et n'est peut être pas assez spécifique.
Par ailleurs, on remarque que l'ajout de règle simple permet à notre réseau de neurones d'obtenir de meilleurs performances passant ainsi d'une exactitude d'environ 96% à 97%.

6 Axes d'amélioration

Ce projet est un projet fonctionnel mais il est toujours possible de l'améliorer. Voici une liste non exhaustive des points d'améliorations que l'on peut mettre en place dans le futur :

— **l'utilisation d'un jeu de données plus complet**
Avec le *deep learning*, plus le nombre de données est important, plus le résultat peut être fiable. De même que plus il y a d'informations diverses sur l'article, plus il est possible de trouver des liens et déterminer un motif permettant de déterminer si un article est fiable ou non.

— **la mise en place d'un retour sur pertinence**
Ce type d'approche rend l'utilisateur (ici, les journalistes) acteur de la décision de notre modèle. C'est-à-dire que lorsque le modèle obtient un pourcentage ne permettant pas de prédire de façon précise si un article est fiable ou non, entre 0.45 et 0.55 par exemple, on peut demander à l'utilisateur de nous dire si c'est une article fiable ou non. De là, on peut faire réapprendre à notre modèle en utilisant la nouvelle information que l'utilisateur vient de renseigner. Ainsi, notre modèle devient de plus en plus performant.

— **l'amélioration du design de l'interface**
Il est aussi possible d'améliorer l'apparence de l'interface actuelle qui est assez minimaliste pour le moment. Il sera toujours possible d'intégrer d'autres fonctionnalités par la suite.

7 Bilan et conclusion

La première partie de mon projet de recherche et développement m'a permis de définir les objectifs et le périmètre de ce sujet. Puis, de définir la méthode utilisée pour résoudre ce problème. La deuxième partie avait pour but la mise en place de cette solution : l'utilisation du modèle Word2Vec pour vctoriser les données et d'un réseau de neurones pour la classification.

Ce projet m'a permis d'acquérir des notions et compétences dans les domaines du *Machine Learning*. Il m'a aussi permis de mettre en place une gestion de projet et de devoir découper et extimer le temps des différentes tâches.

L'un des atouts de ce sujet est le fait qu'il se soit pas juste cantonné à un domaine : celui de l'informatique. Mais aussi d'autre domaine tel que le journalisme. Il m'a fait découvrir, partager et débattre sur le sujet des *fake news* ce qui fût très enrichissant pour moi.

On peut aussi noter que l'autonomie et la liberté accordées pour ce sujet m'ont permis d'axer ce sujet vers un domaine qui m'intéresse : le *Machine Learning*.

8 Perspectives d'avenir

Ce projet n'est qu'un prototype qui est amené à évoluer rapidement. Étant donné le délai court de ce projet, il reste quelques axes d'améliorations possibles qui sont :

— **une étude de faisabilité pour appliquer ce problème au français**
L'une des premières remarques que l'on peut faire est qu'il n'existe pour le moment pas de jeu de données en français contenant assez de données et classées de manière fiable.

— **l'intégration à Factoscope [7]**
Si ce problème est appliqué à du français et que les performances sont bonnes, nous pourrions l'intégrer au site Factoscope pour une aide à la décision pour les journalistes ou encore le grand public.
FactoScope 2017-2022 vérifie les propos des personnalités politiques et donne accès au travail de vérification de la majeure partie des vérificateurs français. Ce projet innovant est piloté par les journalistes-étudiants de l'École publique de journalisme de Tours (EPJT). Il permet de retrouver à une seule et même adresse la majeure partie du *fact-checking* français et de le mettre à disposition du citoyen, qui peut rechercher ces contenus par thème ou par personnalité politique.

— **une approche pédagogique pour le grand public**
On pourrait imaginer une forme de prévention sur la facilité dont les *fake news* peuvent nous induire en erreur sur des sujets d'actualités parfois sensibles.
Un site américain a mis en place un jeu qui invite les participants à créer des fausses nouvelles et à les diffuser le plus possible. Ils peuvent rédiger leurs propres articles ou en copier des déjà existants, puis les partager sur des faux réseaux sociaux. Et pour augmenter l'audience de leurs articles, ils peuvent acheter de faux profils sur les réseaux sociaux qui partageront l'information. Ce jeu est "Fake it to Make it" [8]. C'est une manière de faire prendre conscience aux joueurs de la facilité à fabriquer des rumeurs et à les diffuser massivement sur le Web.

Annexes

A Spécifications fonctionnelles

1 Module de pré-traitement

1. Présentation
   - Nom : Le pré-traitement
   - Priorité de réalisation : primordiale
2. Description
   Cette fonction a pour but de nettoyer les données pour éviter de polluer les informations. Pour cela, différentes techniques vont être appliquées : les stop words, stemming...
   - Entrée : le nom du fichier qui correspond au jeu d'apprentissage. Ce fichier sera au format csv avec certaines colonnes spécifiques : article, label.
   - Sortie : un tableau de la taille du nombre d'articles qui lui même contient un tableau avec les noms sélectionnés de chaque article.

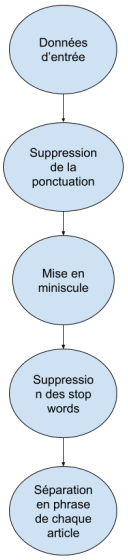


Figure 1 – *Différentes étapes du pré-traitement.*

2 Module de vectorisation

1. Présentation
   - Nom : Vectorisation
   - Priorité de réalisation : primordiale
2. Description
   Cette fonction a pour but de transformer des mots en vecteurs. Pour cela, nous allons nous appuyer sur l'algorithme développé par Google word2vec. Il existe la bibliothèque gensim [10] qui prend en charge ce type de modèle.
   - Entrée : un tableau a double dimension contenants pour chaque sous-ensembles les mots correspondant à un article.
   - Sortie : un modèle word2vec.

3 Module d'apprentissage

1. Présentation
   - Nom : Apprentissage
   - Priorité de réalisation : primordiale
2. Description
   Cette fonction a pour but de définir et d'entraîner le modèle développé. Keras permet de créer facilement des réseaux de neurones à convolutions et fournit des fonctions pour l'apprentissage. Déterminer les paramètres optimaux sera une partie importante de ce module.
   - Entrée : une matrice représentant les mots
   - Sortie : un modèle entraîné

4 Module d'évaluation

1. Présentation
   - Nom : Évaluation
   - Priorité de réalisation : primordiale
2. Description
   Cette fonction a pour but de tester les performances du modèle précédemment entraîné.
   - Entrée : un modèle à tester, le jeu de données à tester au format csvc
   - Sortie : un fichier csv avec les résultats des prédictions du modèle

5 Module d'IHM

1. Présentation
   - Nom : Interface
   - Priorité de réalisation : primordiale
2. Description
   C'est le point d'entrée de l'algorithme. Dans un premier temps, le terminal fera office d'interface et par la suite, pourra évoluer vers une interface plus élaborée.
   - Entrée : une phrase à analyser
   - Sortie : une catégorie

B Gestion de projet

1 Premier semestre

1.1 Aperçu de la gestion de projet

Le diagramme de Gantt de la planification du premier semestre se trouve à la figure 1.

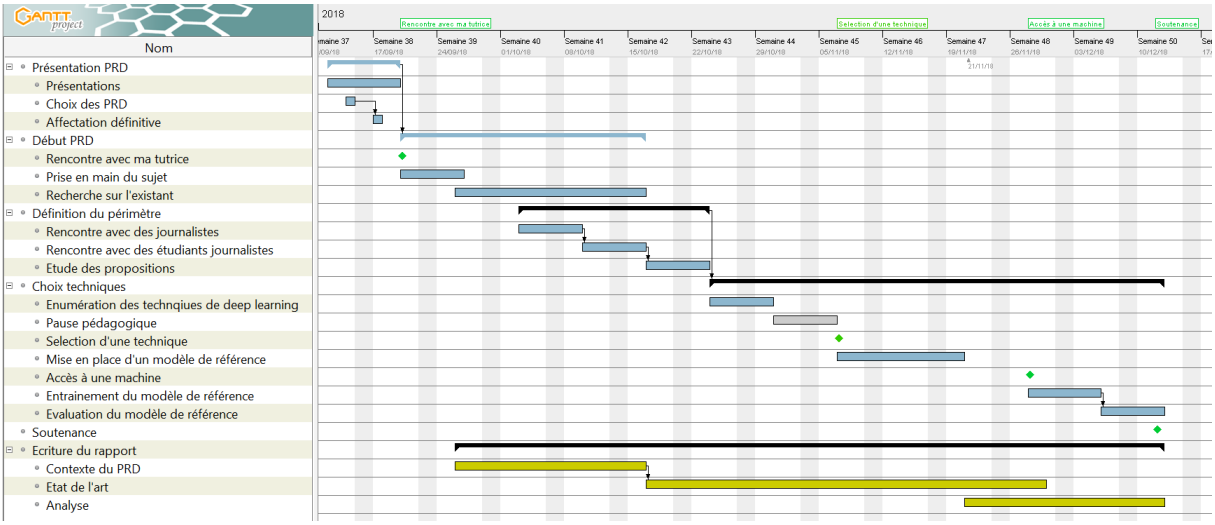


Figure 1 – *Diagramme de Gantt sur les tâches réalisées au cours du premier semestre.*

1.2 Découpage des tâches

Nom	Date de début	Date de fin
Présentation PRD	12/09/18	19/09/18
Présentations	12/09/18	19/09/18
Choix des PRD	14/09/18	14/09/18
Affectation définitive	17/09/18	17/09/18
Début PRD	20/09/18	16/10/18
Rencontre avec ma tutrice	20/09/18	20/09/18
Prise en main du sujet	20/09/18	26/09/18
Recherche sur l'existant	26/09/18	16/10/18
Ecriture du rapport	26/09/18	12/12/18
Contexte du PRD	26/09/18	16/10/18
Etat de l'art	17/10/18	29/11/18
Analyse	21/11/18	12/12/18
Définition du périmètre	03/10/18	23/10/18
Rencontre avec des journalistes	03/10/18	09/10/18
Rencontre avec des étudiants journalistes	10/10/18	16/10/18
Etude des propositions	17/10/18	23/10/18
Choix techniques	24/10/18	06/12/18
Enumération des technqiues de deep learning	24/10/18	30/10/18
Pause pédagogique	29/10/18	02/11/18
Selection d'une technique	01/11/18	01/11/18
Mise en place d'un modèle de référence	07/11/18	13/11/18
Accès à une machine	28/11/18	28/11/18
Entrainement du modèle de référence	28/11/18	05/12/18
Evaluation du modèle de référence	06/12/18	06/12/18
Soutenance	12/12/18	12/12/18

Figure 2 – *Tâches effectuées au premier semestre*

1. Présentation PRD
   — Le but était d'avoir une vision globale des différents sujets proposés, de pouvoir faire des voeux puis avoir l'affectation définitive des sujets.
2. Début PRD
   — Dans un premier temps, j'ai rencontré mon encadrante, puis nous avons discuté du sujet et j'ai ainsi commencé mes recherches sur le thème du fact checking.
3. Écriture du rapport
   — L'écriture du rapport s'est faite au fur et à mesure. Au début, seul le contexte apparaissait, puis quand le périmètre a été défini, j'ai pu commencé à rédiger la partie état de l'art et enfin l'analyse.
4. Définition du périmètre
   — Le but était de définir vers quelle direction ce sujet allait se tourner. Nous avons choisi d'y appliquer une approche utilisant le *Machine Learning*.
5. Choix techniques
   — Le *Machine Learning* et le *Deep Learning* proposent un panel de technique d'apprentissage. Il a fallu trancher sur une méthode en particulier qui est l'utilisation de word2vec comme vectoriseur et d'un CNN comme classificateur.
6. Soutenances
   — Cela fait référence à la préparation de la soutenance de mi-projet.

2 Deuxième semestre

2.1 Aperçu prévisionnel de la gestion de projet

Le diagramme de Gantt prévisionnel se trouve à la figure 3

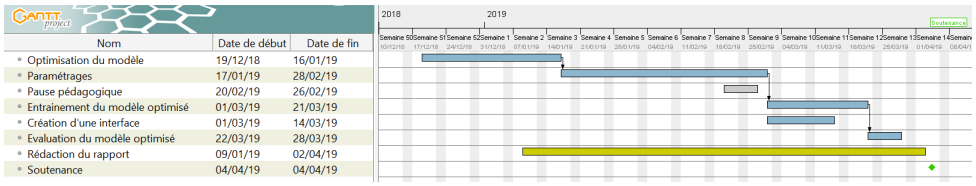


Figure 3 – *Tâches à effectuer au deuxième semestre*

2.2 Découpage des tâches

Nom	Date de début	Date de fin
Optimisation du modèle	19/12/18	16/01/19
Paramétrages	17/01/19	28/02/19
Pause pédagogique	20/02/19	26/02/19
Entrainement du modèle optimisé	01/03/19	21/03/19
Création d'une interface	01/03/19	14/03/19
Evaluation du modèle optimisé	22/03/19	28/03/19
Rédaction du rapport	09/01/19	02/04/19
Soutenance	04/04/19	04/04/19

Figure 4 – *Tâches à effectuer au deuxième semestre*

1. Optimisation du modèle
   - Le but est de trouver un modèle qui permet d'obtenir les meilleurs résultats possibles. Cette recherche se fera de façon itérative, tant que le modèle ne sera pas assez bon, la recherche continue.
2. Paramétrage
   - Le but est de trouver les paramètres qui optimisé au mieux le modèle pour obtenir les meilleurs résultats possibles. Cette recherche se fera de façon itérative, tant que les paramètres ne sont pas optimaux, la recherche continue.
3. Entraînement du modèle optimisé
   - Le temps d'entraînement d'un modèle peut varier. Pendant cette phase, on entrainera le modèle selon les paramètres optimaux trouvés.
4. Création d'une interface
   - Création d'une interface pour lancer l'algorithme lorsqu'une phrase est donnée.
5. Évaluation du modèle optimisé
   - L'évaluation consiste à juger à quel point l'algorithme est performant lorsqu'il voit des données qu'il n'a jamais vu.
6. Rédaction du rapport
   - L'écriture du rapport se fera au fur et à mesure des résultats et des recherches effectuées.
7. Soutenances
   - Cela fait référence à la préparation de la soutenance finale de ce projet de recherche et développement.

C Paramétrages

1 Paramétrage du réseau de neurones

Couches	Paramètres				
Embedding	**input_dim**=3000	**output_dim**=100	**weights**=embedding_matrix	**input_length**=12072	**trainable**=True
Convolution	**filters**=32	**kernel_size**=3	**activation**='relu'		
MaxPooling	**pool_size**=16	**strides**=2			
Dropout	**rate**=0.2				
Dense	**units**=100	**activation**='relu'			
Dropout	**rate**=0.2				
Dense	**units**=100	**activation**='sigmoid'			

Figure 1 – *Paramètres du réseau de neurones à convolution*

D Guide d'installation et d'utilisation

Guide d'installation et d'utilisation

Pauline LAURON

Mars 2019

Contents

1 Introduction

Ce manuel a pour but d'expliquer les différentes fonctionnalités développées lors du projet de recherche et développement consistant à la détection automatique de *fakes news*. Pour cela, nous avons utilisé le *deep learning* qui est une technique d'apprentissage.
Les *fakes news* (informations fallacieuses, infox ou fausses nouvelles) sont des informations délibérément fausses, délivrées dans le but de tromper un auditoire. Cet algorithme a été développé dans le but d'être un outils d'aide à la décision.

2 Installation

2.1 Instructions

2.1.1 Récupérer le projet

```
$ git clone https://github.com/passelacet/fake_news.git
$ cd fake_news
```

Le projet est installé.

2.1.2 Lancer le programme

La commande pour lancer le programme d'entraînement du modèle est la suivante :

```
$ python3 main.py --train-file <path_train_file>
--test-file <path_test_file>
```

train_file et *test_file* sont des paramètres exigés. Etant donné la multitude de paramètres modifiables, ils ont été regroupés dans le fichier config.py.
La commande pour lancer l'interface est la suivante :

```
$ python3 interface/controller_gui.py
```

2.1.3 Dépendances

Ce programme a été développé sous Python 3.6. Les *packages* utilisés sont :

- pandas
- keras
- sklearn
- matplotlib
- PyQt5
- Tensorflow

– numpy
– gensim
– scipy

A partir du moment où le projet est cloné, il est possible d'installer ces modules rapidement grâce à la commande suivante :

```
$ pip install -r requirements.txt
```

2.1.4 Modifier les paramètres

Tous les paramètres ont été répertorié dans le fichier *config.py*. Il regroupe les paramètres de pré_traitement des données, de la création du modèle Word2Vec, la préparation des données pour le CNN, le modèle du CNN, l'entrainement de ce dernier et la prédiction. Une explication de ces différents paramètres y est faîte.

2.2 Architecture

Le projet principal est composé des fichiers suivants :

– config.py : fichier de configuration

– cross_validation.py : fichier permettant d'entraîner le modèle selon une validation croisée

– extract_features.py : fichier permettant d'extraire des caractéristiques particulières

– grid_search.py : fichier permettant d'exécuter une grid search dans le but de trouver les paramètres optimaux

– prepare_data_CNN.py : fichier permettant de transformer les mots en valeurs numériques

– preprocessing.py : fichier permettant le traitement des mots

– pretrained_word2vec.py : fichier permettant d'extraire une partie du fichier que Google met à disposition

– word2vec_model.py : fichier qui permet de générer ou de charger un modèle Word2Vec

Plusieurs dossiers appartiennent à ce projet et sont :

– data : dossier qui contient la base de données

– interface : dossier qui contient l'interface

– result : dossier contenant les différentes sorties (modèles...) qui est lui même composé de :

  - cv : dossier de sortie de la validation croisée
  - gs : dossier de sortie de la grid search
- tools : dossier qui regroupe des outils tels que la visualisation des données...

Une documentation détaillée des fonctions des classes peut être retrouvée dans le dossier doc.

L'architecture de ce projet peut être représentée par le diagramme d'activité suivant :

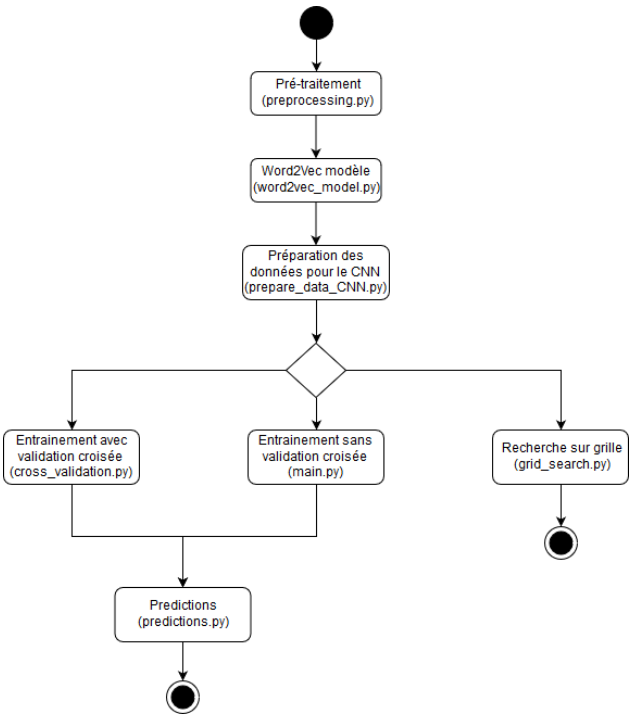


Figure 1: Diagramme d'activité de la partie apprentissage

3 Partie développeur

3.1 Entraîner le modèle

Il existe plusieurs façon d'entraîner le modèle. Celles-ci sont détaillées dans les parties suivantes.

3.1.1 Entraînement par défaut

L'entraînement par défaut proposé est un entraînement "classique". Il utilise 3 jeux de données : un jeu d'apprentissage, un jeu de validation et un jeu de test.

3.1.2 Entraînement par validation croisée

Il est possible d'entraîner le modèle par validation croisée. Pour cela, dans le fichier config.py, il faut passer le paramètres *cross_validation* à *True*.

3.2 Sauvegarde du modèle

4 Partie utilisateur

5 L'interface

L'interface est à destination des personnes à la recherche d'une aide à la décision tels que les journalistes par exemple.
Elle est découpée en 2 parties :

- une partie "article" qui contient une zone de texte ainsi qu'un bouton lançant la prédiction
- une partie "prédiction" qui affiche si le texte donnée en entrée est un article fiable ou non avec le pourcentage d'appartenance à une infox.

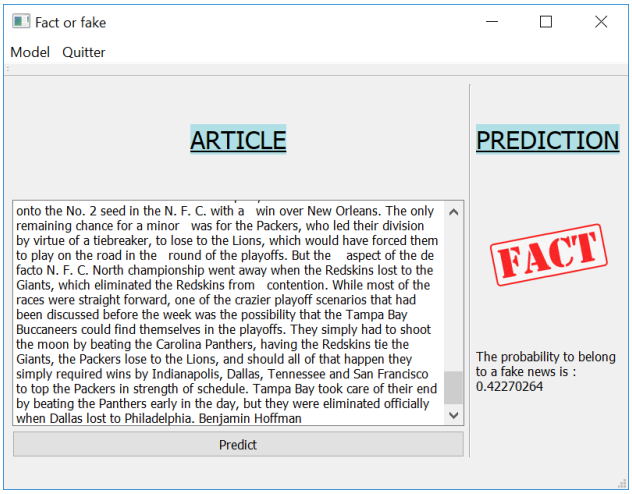


Figure 2: Interface utilisateur

Pour pouvoir utiliser cette interface, il faut que le modèle soit déjà entraîné. Pour cela, un modèle par défaut a été renseigné. Cependant, il est toujours possible d'utiliser son propre modèle en allant dans *Model ¿ Open* pour sélectionner le modèle que l'on veut utiliser.

Partie article Comme on peut le voir sur l'image XX, on retrouve une zone de texte qui permet à l'utilisateur de saisir l'article qu'il souhaite traiter.
Pour obtenir des résultats optimaux, il est recommandé de fournir l'article de la façon suivante : le titre de l'article suivi du corps de l'article et pour finir l'auteur de l'article. Lorsque la saisi est terminée, il reste à l'utilisateur à cliquer sur le bouton *predict*.

Partie prédiction A partir du moment au l'utilisateur clique sur le bouton *predict*, l'algorithme se lance. Lorsqu'il se termine, une prédiction est donnée : soit c'est un article fiable représenté par l'image XX, soit c'est un article non fiable. De plus, un indicateur de probabilité apparaît et permet de connaître la probabilité que l'article soit une *fack news*.

Exemples Voici 2 exemples d'article : le premier est un article non fiable et le deuxième un article non fiable. Les textes utilisées peuvent être retrouvés dans en annexe 7.1.

On remarque que le modèle le classifie comme *fake news*, ce qui est correct avec un pourcentage d'environ 97% d'appartenance à la classe fake news.

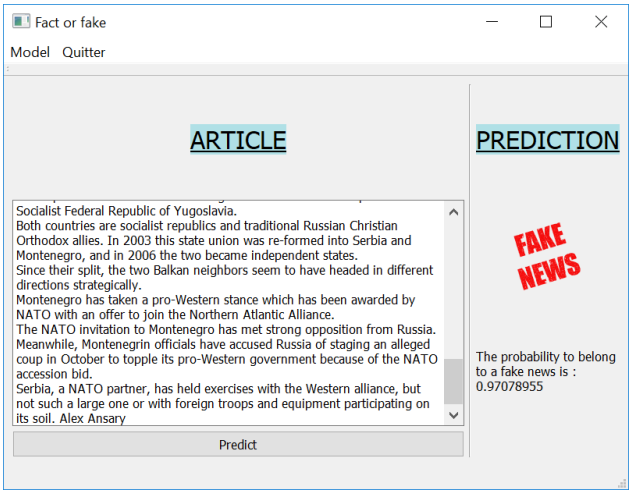


Figure 3: Interface utilisateur

On remarque que le modèle le classifie comme *fact*, ce qui est correct avec un pourcentage d'environ 97% d'appartenance à la classe fake news.

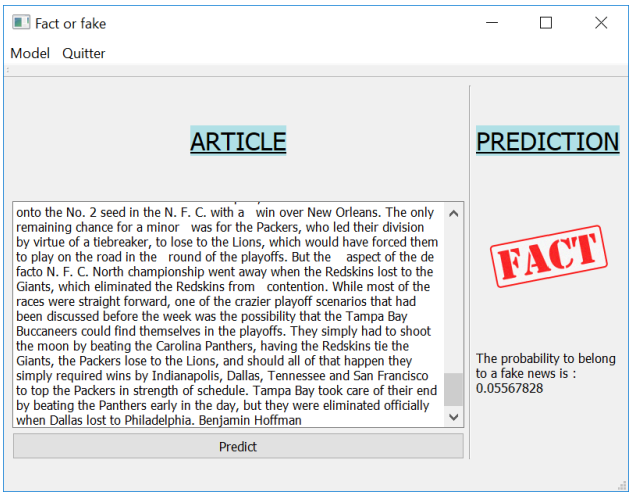


Figure 4: Interface utilisateur

Pour rappel, les prédictions sont données avec un taux d'erreur de 11%.

6 Conclusion

7 Annexe

7.1 Textes pour prédiction

7.1.1 Article non fiable

NATO, Russia To Hold Parallel Exercises In Balkans NATO, Russia To Hold Parallel Exercises In Balkans 11/02/2016 PRESS TV Russia's military and NATO forces are holding parallel military exercises in two neighboring Balkan countries. Russian troops will participate in war games in Serbia while NATO is conducting military drills in Montenegro, media reported on Monday. Russian forces' 13-day military exercise in Serbia is named "The Slavic Brotherhood 2016" and begins on Wednesday. It will include 150 Russian paratroopers, 50 air force staffers, three transport planes and an unspecified number of troops from Serbia and Belarus, Russia's Defense Ministry said. The five-day NATO drill in Montenegro started on Monday and involves responding to floods and chemical attacks. It will involve 680 unarmed personnel from seven NATO countries and 10 partner states. In the past both Serbia and Montenegro were constitutional republics of the Socialist Federal Republic of Yugoslavia. Both countries are socialist republics and traditional Russian Christian Orthodox allies. In 2003 this state union was re-formed into Serbia and Montenegro, and in 2006 the two became independent states. Since their split, the two Balkan neighbors seem to have headed in different directions strategically. Montenegro has taken a pro-Western stance which has been awarded by NATO with an offer to join the Northern Atlantic Alliance. The NATO invitation to Montenegro has met strong opposition from Russia. Meanwhile, Montenegrin officials have accused Russia of staging an alleged coup in October to topple its pro-Western government because of the NATO accession bid. Serbia, a NATO partner, has held exercises with the Western alliance, but not such a large one or with foreign troops and equipment participating on its soil. Alex NATO, Russia To Hold Parallel Exercises In Balkans NATO, Russia To Hold Parallel Exercises In Balkans 11/02/2016 PRESS TV Russia's military and NATO forces are holding parallel military exercises in two neighboring Balkan countries. Russian troops will participate in war games in Serbia while NATO is conducting military drills in Montenegro, media reported on Monday. Russian forces' 13-day military exercise in Serbia is named "The Slavic Brotherhood 2016" and begins on Wednesday. It will include 150 Russian paratroopers, 50 air force staffers, three transport planes and an unspecified number of troops from Serbia and Belarus, Russia's Defense Ministry said. The five-day NATO drill in Montenegro started on Monday and involves responding to floods and chemical attacks. It will involve 680 unarmed personnel from seven NATO countries and 10 partner states. In the past both Serbia and Montenegro were constitutional republics of the Socialist Federal Republic of Yugoslavia. Both countries are socialist republics and traditional Russian Christian Orthodox allies. In 2003 this state union was re-formed into Serbia and Montenegro, and in 2006 the two became independent states. Since their split, the two Balkan neighbors seem to have

headed in different directions strategically. Montenegro has taken a pro-Western stance which has been awarded by NATO with an offer to join the Northern Atlantic Alliance. The NATO invitation to Montenegro has met strong opposition from Russia. Meanwhile, Montenegrin officials have accused Russia of staging an alleged coup in October to topple its pro-Western government because of the NATO accession bid. Serbia, a NATO partner, has held exercises with the Western alliance, but not such a large one or with foreign troops and equipment participating on its soil. Alex Ansary

7.1.2 Article fiable

N.F.L. Playoffs: Schedule, Matchups and Odds - The New York Times When the Green Bay Packers lost to the Washington Redskins in Week 11, dropping to Aaron Rodgers vowed to "run the table" in a march to the playoffs. With a victory over the Detroit Lions on Sunday night, the team fulfilled Rodgers' promise. Much of the drama of the matchup between division rivals was eliminated earlier in the day when the Redskins lost to the Giants, thus guaranteeing both the Packers and Lions would be playoff teams, but the N. F. C. North bragging rights, and a home game in the first round of the playoffs, were sufficient motivation for Green Bay to push hard enough to secure the team's sixth consecutive win and the third consecutive loss for Detroit. Pundits had spent the week deciphering all of the wild scenarios that could play out for positioning among the remaining teams. But when all was said and done, ten of the teams that were in line for a playoff spot remained in the same seeding order. No. 6 Detroit Lions at No. 3 Seattle Seahawks Time: 8:15 p. m. Eastern SATURDAY on NBC The Seahawks' title hopes took a crushing blow when Earl Thomas was lost for the season with a broken leg. After the injury, the Seahawks went with the wins coming with major asterisks as they came against the Rams and 49ers. That collapse paled in comparison to the Lions, who lost their final three games, blowing what had been a large division lead against Green Bay. Line: Seahawks (: 43) No. 5 Giants at No. 4 Green Bay Packers Time: 4:40 p. m. Eastern SUNDAY on Fox Thanks to playing in the N. F. C. East, home of the Dallas Cowboys, the Giants managed to tie Atlanta for the record in the N. F. C. but got stuck with the No. 5 seed in the playoffs and a road game against a Packers squad that won its final six games. The good news for the Giants is that superstitious fans will note that the last two times they played the Packers on the road in the playoffs, they not only won the games but went on to win the Super Bowl both times. Line: Packers (: 44. 5) Bye weeks: Dallas, Atlanta No. 5 Oakland Raiders at No. 4 Houston Texans Time: 4:35 p. m. Eastern SATURDAY on ESPN Line: Texans (: 37) The Texans were the least inspiring of the N. F. L. 's division champions and that was complicated further when Tom Savage, whom the team had elevated to starting quarterback after the benching of Brock Osweiler, was forced to leave Week 17's loss to Tennessee with a concussion. As bad as that sounds, it may still be enough against a reeling Oakland squad that lost Derek Carr to a broken leg in Week 16, Matt McGloin to a shoulder injury in Week 17, and fell all the

way from the No. 2 seed in the A. F. C. to No. 5. It is unclear at this point if McGloin or rookie Connor Cook will start at quarterback against Houston. No. 6 Miami Dolphins at No. 3 Pittsburgh Steelers Time: 1:05 p. m. Eastern SUNDAY on CBS The Dolphins were a contender when the team's quarterback, Ryan Tannehill, was lost in Week 14 with injured ligaments in his left knee. Thanks to backup quarterback Matt Moore, and Jay Ajayi, the team's running back, they won two of three games and secured a berth. But going up against a offense like Pittsburgh is a tough test for Miami's middling defense, even if Tannehill's knee heals enough to allow him to return. Line: Steelers .5 (: 47. 5) Bye weeks: New England, Kansas City With four teams vying for two N. F. C. playoff spots, all eyes were on the game on Sunday. A Redskins victory could have caused movement in the seedings, with the Lions and Packers playing a evening matchup. The Giants, who had already locked up the No. 5 seed and had nothing to gain, threw a wrench in the Redskins' plans, eliminating their division rivals with a decisive victory. With the drama essentially taken out of the N. F. C. all of the playoff movement Sunday occurred in the A. F. C. where there was a at the top of the standings in the A. F. C. West. Just a week after losing Derek Carr, the team's quarterback and a legitimate candidate for most valuable player, to a broken leg, the Oakland Raiders were crushed by the Denver Broncos. That, combined with the Kansas City Chiefs' victory over the San Diego Chargers vaulted the Chiefs from a spot all the way to the No. 2 seed in the A. F. C. which comes with a bye in the playoffs. The loss for Oakland added to the misery of the Raiders, who have gone from Super Bowl contenders last week to a team that will play on the road in Houston next week potentially with a quarterback under center as Carr's backup, Matt McGloin, injured his shoulder in the loss to the Broncos. Beyond the switch to the Chiefs as the No. in the A. F. C. it was business as usual for the teams that will get bye weeks in the playoffs. The New England Patriots secured the No. 1 spot in the A. F. C. with a win over Miami, the Dallas Cowboys were already guaranteed the No. 1 spot in the N. F. C. before their loss to Philadelphia, and the Atlanta Falcons held onto the No. 2 seed in the N. F. C. with a win over New Orleans. The only remaining chance for a minor was for the Packers, who led their division by virtue of a tiebreaker, to lose to the Lions, which would have forced them to play on the road in the round of the playoffs. But the aspect of the de facto N. F. C. North championship went away when the Redskins lost to the Giants, which eliminated the Redskins from contention. While most of the races were straight forward, one of the crazier playoff scenarios that had been discussed before the week was the possibility that the Tampa Bay Buccaneers could find themselves in the playoffs. They simply had to shoot the moon by beating the Carolina Panthers, having the Redskins tie the Giants, the Packers lose to the Lions, and should all of that happen they simply required wins by Indianapolis, Dallas, Tennessee and San Francisco to top the Packers in strength of schedule. Tampa Bay took care of their end by beating the Panthers early in the day, but they were eliminated officially when Dallas lost to Philadelphia. Benjamin Hoffman

E Documentation

doc_fake_news Documentation

Pauline LAURON

Mar 21, 2019

CONTENTS

CHAPTER

ONE

SCRIPT PRINCIPAL

main.**make_args_parser**()
Parse les arguments données en ligne de commande

main.**main**()

Fonction principale qui appelle des différentes étapes [] lecture des données, traitement des données, création du modèle word2vec, préparation des données pour le CNN, entrainement, prédiction

CHAPTER

TWO

MODULES ET PACKAGES

2.1 Chargement des données

load_data.**load_data**(*path_train*, *path_test*)
Fonction qui charge les données

Parameters

- **path_train** – chemin du jeu d'apprentissage
- **path_test** – chemin du jeu de test

Returns le jeu d'apprentissage, les étiquettes du jeu d'apprentissage, le jeu de test

2.2 Pré-traitement des données

class preprocessing.**Preprocessing**
Classe qui permet le pré-traitement des données

cleaning_text(*text*, *remove_stopwords=False*)
Fonction qui permet de nettoyer et de séparer une phrase en mot

Parameters

- **text** – la phrase à nettoyer
- **remov_stopwords** – suppression des stop words ou non

Returns mots du texte nettoyés

lemmatization()
Fonction qui applique la lemmatisation

Parameters **sentences** – phrases à lemmatiser

Returns la phrase lemmatisée

run(*X*, *data*)
Fonction principale

Parameters

- **X** – le jeu de données
- **data** – 0 si jeu d'apprentissage 1 si jeu de test

Returns les données pré-traitées (séparation, suppression des mots vides...)

text_to_sentences(*text*, *tokenizer*, *remove_stopwords=False*)
Fonction qui sépare chaque phrase de chaque article

Parameters

- **text** – l'ensemble du document
- **tokenizer** – permet de séparer les phrases
- **remov_stopwords** – suppression des stop words ou non

Returns les articles sous forme de phrases

2.3 Modèle Word2Vec

class word2vec_model.**Word2Vec**
Classe qui permet de la création, l'entrainement ou le chargement d'un modèle Word2Vec

loading()
Fonction pour charger un modèle word2Vec

loading_custom()
Fonction pour charger un dictionnaire personnalisé préalablement créé

run(*sentences*)
Fonction principale

Parameters **sentences** – les données pré-traitées et séparées en mots par article

saving()
Fonction pour sauvegarder un modèle word2Vec

training(*sentences*)
Fonction qui créé et entraine le modèle word2vec avec le jeu de données sentences

Parameters **sentences** – jeu de données

2.4 Préparation des données pour le CNN

class prepare_data_CNN.**Prepare_data_CNN**
Classe qui converti les mots en numériques

add_features(*x_train_seq*, *x_test_seq*)
Fonction qui permet d'ajouter certaines caractéristiques

Parameters

- **x_train_seq** – matrice de représentation de la séquence du jeu d'apprentissage
- **x_test_seq** – matrice de représentation de la séquence du jeu de test

Returns la nouvelle matrice correspondant au jeu d'apprentissage

Returns la nouvelle matrice correspondant au jeu de test

loading()
Fonction de chargement

Returns matrice de représentation de la séquence du jeu d'apprentissage

Returns matrice de représentation de la séquence du jeu de test

Returns matrice des poids initiaux

matrix_emb(*word2vec_model*)
Fonction qui créer la matrice de représentation des mots présent dans le jeu d'apprentissage

Parameters **word2vec_model** – le modèle word2vec

Returns la matrice de représentation dans un espace vectoriels

padding(*sequences_train*, *sequences_test*)
Fonction de padding : chaque vecteur a la même taille

Parameters

- **sequences_train** – sequence du jeu d'apprentissage
- **sequences_test** – sequence du jeu de test

Returns matrice de représentation de la séquence du jeu d'apprentissage

Returns matrice de représentation de la séquence du jeu de test

predict_interface(*text*)
Fonction de prédiction pour l'interface

Returns le texte transformé en numérique

run(*sentences_train*, *sentences_test*, *word2vec_model*)
Fonction principale

Returns la matrice correspondant au jeu d'apprentissage

Returns la matrice correspondant au jeu de test

Returns la matrice des poids initiaux

saving(*x_train_seq*, *x_test_seq*, *embedding_matrix*)
Fonction de sauvegarde

Parameters

- **x_train_seq** – matrice de représentation de la séquence du jeu d'apprentissage
- **x_test_seq** – matrice de représentation de la séquence du jeu de test
- **embedding_matrix** – matrice des poids initiaux

tokenizing(*sentences_train*, *sentences_test*)
Fonction de convertion de mot en nombre

Parameters

- **sentences_train** – le dictionnaire de mot du jeu d'apprentissage
- **sentences_test** – le dictionnaire de mot du jeu de test

Returns numériques du jeu d'apprentissage

Returns numériques du jeu de test

2.5 Entrainement

class training.**Training**(*input_length*, *embedding_matrix*)
Classe qui permet d'entrainer le modèle et d'afficher les courbes de fiabilité et de perte

create_model (*embedding_matrix*, *input_length*)
Fonction de création du réseau de neurones (ici à convolution)

Parameters

- **embedding_matrix** – matrice des poids initiaux
- **input_length** – la taille d'entrée de la première couche

Returns le modèle

display (*history*)
Fonction qui affiche les courbes de fiabilité et de perte

Parameters **history** – l'historique de l'entrainement

run (*x_train*, *y_train*)
Fonction principale

Parameters

- **x_train** – le jeu d'apprentissage
- **y_train** – les étiquettes du jeu d'apprentissage

Returns le modèle entrainé

save ()
Fonction de sauvegarde du modèle

2.6 Validation croisée

class cross_validation.**Cross_validation** (*embedding_matrix*, *input_length*)
Classe permettant de mettre en place une valdiation croisée lors de l'apprentissage

model_vc ()
Fonction de création du réseau de neurones (ici à convolution)

Returns le modèle

run (*x_train*, *y_train*)
Fonction qui exécute la validation croisée

Parameters

- **x_train** – le jeu d'apprentissage
- **y_train** – les étiquettes du jeu d'apprentissage

Returns le modèle entrainé

2.7 Recherche par grille

class grid_search.**Grid_search** (*embedding_matrix*, *input_length*)
Classe qui permet de faire une recherche d'hyperparamètres

model_gs (*n_cnn=10*, *pool_size=100*, *strides=4*, *kernel_size=3*)
Fonction de création du réseau de neurones (ici à convolution)

Parameters

- **n_cnn** – nombre de neurones dans la couche de convolution

- **pool_size** – la taille de la fênetre de pooling
- **strides** – le nombre de fois que l'on divise la matrice
- **kernel_size** – la taille de la fênetre du noyau

Returns le modèle compilé

run(*x_train*, *y_train*)
Fonction qui exécute la recherche par grille

Parameters

- **x_train** – le jeu d'apprentissage
- **y_train** – les étiquettes du jeu d'apprentissage

Returns le modèle entrainé

saving()
Fonction de sauvegarde du modèle

2.8 Prédictions

class predictions.**Prediction**
Classe permettant de donner des prédictions à partir d'un modèle

prediction(*model*, *x_test*, *y_test*)
Fonction de prédiction

Parameters

- **model** – modèle entrainé
- **x_test** – jeu de test
- **y_test** – étiquette

submission(*x_test*, *y_test*)
Fonction pour soumettre

Parameters

- **model** – modèle entrainé
- **x_test** – jeu de test

2.9 Interface

2.9.1 View

class view_gui.**Ui_MainWindow**

retranslateUi(*MainWindow*)
Traduction

setLabelImage(*image_path*)
To display image :param self: the current object :param nameGraphic: the path of the image

setupUi(*MainWindow*)
Initialisation de l'interface

2.9.2 Main

class main_gui.**MyWindow**(*parent=None*)
Classe controleur de l'interface

display_fact()
Fonction qui affiche une image pour signaler un fait

display_fake()
Fonction qui affiche une image pour signaler une fake news

display_percentage(*percentage*)
Fonction qui affiche le pourcentage d'apparenance à une fake news

get_article()
Fonction qui obtient le texte saisi par l'utilisateur :return: le texte ou un message pour dire que rien n'a été saisi

open_model()
Fonction pour ouvrir un modèle

predict_button()
Fonction d'action du bouton predict

2.9.3 Predict

predict.**predict**(*text*, *model_name*)
Fonction pour prédire à quel classe appartient le texte :param text: le texte à classifier :return: la probabilité d'appartenance à la classe fake news

2.10 Boite à outils

2.10.1 Extractions de caractéristiques

tools.extract_features.**extract_features**()
Fonction qui permet d'extraire des caractéristiques syntaxiques d'un jeu de données

2.10.2 Personnalisation de modèle pré-entrainé Word2Vec

tools.pretrained_word2vec.**extract_custom_dict**(*sentences_train*, *sentences_test*, *path_output*)
Fonction qui permet d'extraire une partie du modèle pré-entrainé Word2Vec

Parameters

- **sentences_train** – le dictionnaire de mot du jeu d'apprentissage
- **sentences_test** – le dictionnaire de mot du jeu de test
- **path_output** – le chemin de sortie du dictionnaire personnalisé

2.10.3 Visualisation des données

tools.vizualize_data.**wordcloud_fn**()
Fonction qui créer un nuage de mots

2.10.4 Visualisation du modèle Word2Vec

tools.vizualize_word2vec.**tsne_plot**(*model*)
Creates an TSNE model and plots it

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

T

U

W

Bibliographie

[1] *Anaconda*. URL : https://www.anaconda.com/ (visité le 28/11/2018).

[2] Thota Aswini, Tilak Priyanka, Ahluwalia Simrat et Lohia Nibrat. *Fake News Detection : A Deep Learning Approach*. Application de différents vectoriseurs avec un DNN. 2018. URL : https://scholar.smu.edu/cgi/viewcontent.cgi?article=1036&context=datasciencereview.

[3] Andreas Hanselowski et Avinesh PVS et Benjamin Schiller et Felix Caspelherr. *Description of the System Developed by Team Athene in the FNC-1*. Second place for Fake News Competition. Juil. 2017. URL : https://github.com/hanselowski/athene_system/blob/master/system_description_athene.pdf.

[4] Claimbuster. *Claimbuster*. URL : https://idir-server2.uta.edu/claimbuster/.

[5] *Deep Learning : définition, concept et usages potentiels*. URL : https://www.eurocloud.fr/deep-learning-definition-concept-usages-potentiels/.

[6] Yuxi Pan et Doug Sibley et Sean Baird. *Talos Targets Disinformation with Fake News Challenge Victory*. First place for Fake News Competition. Juin 2017.

[7] *Factoscope*. URL : http://rattrapages-actu.fr/factoscope.

[8] *Fake it to make it*. URL : https://www.fakeittomakeitgame.com/.

[9] *Fake News Competititon*. URL : http://www.fakenewschallenge.org/.

[10] *Gensim*. URL : https://radimrehurek.com/gensim/index.html.

[11] Benjamin Riedel et Isabelle Augenstein et Georgios P Spithourakis et Sebastian Riedel. *A simple but tough-to-beat baseline for the Fake News Challenge stance detection task*. Third place for Fake News Competition. Juil. 2017. URL : https://arxiv.org/pdf/1707.03264v1.pdf.

[12] *Keras*. URL : https://keras.io/ (visité le 28/11/2018).

[13] Yang Yang et Lei Zheng et Jiawei Zhang et Qingcai et Xiaoming Zhang et Zhoujun Li et Philip S Yu. *TI-CNN : Convolutional Neural Networks for FakeNews Detection*. Utilisation de textes et d'images. Juin 2018. URL : https://blog.talosintelligence.com/2017/06/talos-fake-news-challenge.html.

[14] Yang Liu et Yi-fang Brook Wu. *Early Detection of Fake News on Social Media Through Propagation Path Classification with Recurrent and Convolutional Networks*. Comparaison du RNN et CNN dans la classification de fake news. Avr. 2018. URL : https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16826.

[15] Le Monde. *Décodex*. URL : https://www.lemonde.fr/verification/.

[16] *Natural Language Toolkit*. URL : http://www.nltk.org/ (visité le 28/11/2018).

[17] *Tensorflow*. URL : https://www.tensorflow.org/ (visité le 28/11/2018).

[18] Joseph Turian, Lev Ratinov et Yoshua Bengio. *Word representations : A simple and general method for semi-supervised learning*. 2010. URL : http://delivery.acm.org/10.1145/1860000/1858721/p384-turian.pdf?ip=46.193.0.13&id=1858721&acc=OPEN&key=4D4702B0C3E38B35%5C%2E4D4702B0C3E38B35%5C%2E4D4702B0C3E38B35%5C%2E6D218144511F3437&__acm__=1544177672_6e22a9dcc5957d6b78ea1f9962cb9dff.

[19] *You Won't Believe What Obama Says In This Video !* URL : https://www.youtube.com/watch?v=cQ54GDm1eL0.

[20] Xiang Zhang et Yann LeCun. *Text Understanding from Scratch*. Démonstration des performances des CNN. 2016. URL : https://arxiv.org/pdf/1502.01710.pdf.

Projet Recherche & Développement

Fact checking

Pauline Lauron

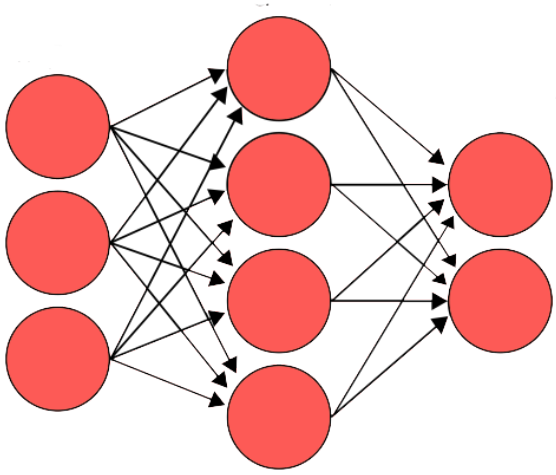
Encadrement : Sabine Barrat

Objectifs

Ce projet a pour but d'automatiser la détection de *fakes news*. Ces *fake news* peuvent avoir des conséquences néfastes dans certains domaines comme la politique. Il est donc intéressant de créer une méthode qui va automatiquement détecter ces *fake news* et ainsi devenir un outils d'aide à la décision pour les journalistes dans un premier temps.

Mise en œuvre

Utilisation du deep learning pour résoudre ce problème. Le modèle Word2Vec va être utilisé comme vectoriseur, c'est-à-dire que l'on va convertir un mot en une représentation numérique, et utiliser un réseau de neurones à convolution pour classifier et prédire si un article est dit fiable ou non fiable.



Résultats attendus

L'utilisation de la mesure d'exactitude a été utilisé et nous retourne un résultat au alentour de 97%. Ce résultat a été obtenu à partir du jeu de données de test.

Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis, 37200 Tours, France
polytech.univ-tours.fr

Informatique

Projet Recherche & Développement

Fact checking

Pauline Lauron

Encadrement : Sabine Barrat

Objectifs

Ce projet a pour but d'automatiser la détection de *fakes news*. Ces *fake news* peuvent avoir des conséquences néfastes dans certains domaines comme la politique. Il est donc intéressant de créer une méthode qui va automatiquement détecter ces *fake news* et ainsi devenir un outils d'aide à la décision pour les journalistes dans un premier temps.

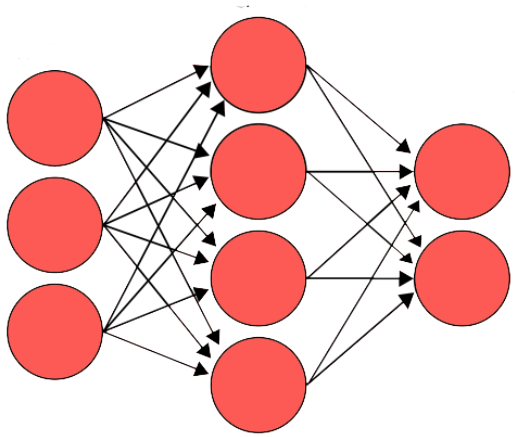
Mise en œuvre

Utilisation du deep learning pour résoudre ce problème. Le modèle Word2Vec va être utilisé comme vectoriseur, c'est-à-dire que l'on va convertir un mot en une représentation numérique, et utiliser un réseau de neurones à convolution pour classifier et prédire si un article est dit fiable ou non fiable.

Résultats attendus

L'utilisation de la mesure d'exactitude a été utilisé et nous retourne un résultat au alentour de 97%. Ce résultat a été obtenu à partir du jeu de données de test.

POLYTECH TOURS



Ecole Polytechnique de l'Université François Rabelais de Tours
Département Informatique
64 avenue Jean Portalis, 37200 Tours, France
polytech.univ-tours.fr

POLYTECH TOURS
Informatique

Fact checking

Résumé

Ce projet de recherche et développement a pour but de de créer une méthode de détection automatique des *fakes news*. De part la facilité de publications des informations et leur partage par les réseaux sociaux par exemple, la détection de ces *fake news* est un sujet qui prend de plus en plus d'importance car elle peut avoir un impact dans certains domaines comme la politique. Pour cela, nous avons utilisé des techniques de *Machine Learning* et plus précisément du *Deep Learning* pour entraîner un algorithme à les détecter.
Ce rapport présente une approche combinant word2vec avec un réseau de neurones à convolutions.

Mots-clés

fact checking, *Machine Learning*, *Deep Learning*, *réseau de neurones*

Abstract

This research and development project aims to create a method for automatic detection of fakes news. Due to the ease of publishing information and sharing it through social networks for example, the detection of these fake news is an increasingly important subject because it can have an impact in certain areas such as politics. To do this, we used techniques of Machine Learning and more precisely Deep Learning to train an algorithm to detect them. This report presents an approach combining word2vec with a convolutional neural network.

Keywords

fact checking, *Machine Learning*, *Deep Learning*, *neural network*

Tuteur académique
Sabine Barrat

Étudiant
Pauline Lauron (DI5)