RAPPORT DE PROJET SCIENTIFIQUE ET COLLECTIF

INF02: Détection de fake news

Tuteur : Gabriel OLYMPIE - Coordinateur : Giles SCHAEFFER

Nathan BOUT, Hamza MAJIOUD, Thomas LOUX, Adam EL BOUHDIDI

ÉCOLE POLYTECHNIQUE

INSTITUT POLYTECHNIQUE DE PARIS

Ekimetrics.

REMERCIEMENTS

Nous tenons à exprimer notre profonde gratitude à notre tuteur, Monsieur Gabriel OLYMPIE, pour son encadrement exceptionnel, sa disponibilité et son expertise tout au long de ce projet de recherche. Ses conseils précieux et son soutien inestimable ont grandement contribué à la réussite de notre travail. Nous remercions également Ekimetrics pour sa collaboration et la confiance qu'elle nous a accordée, en nous permettant d'accéder à ses ressources et en partageant ses connaissances pour enrichir notre recherche. Un grand merci au département d'informatique et à notre coordinateur, Monsieur Gilles SCHAEFFER, pour leur appui constant, l'organisation des ressources nécessaires et la création d'un environnement propice au développement de nos compétences et à l'aboutissement de notre projet.

TABLE DES MATIÈRES

INTRODUCTION

Les dernières années ont vu se développer un phénomène nouveau ayant pour but l'influence de la foule : la désinformation. L'outil informatique a permis de faciliter l'accès aux informations ainsi que la création d'informations. Auparavant la chaîne de l'information était maîtrisée par un nombre faible d'acteurs du fait du coût de la mise en place de la production et de la logistique avec les journaux. Ces derniers, pour subsister, devaient compter sur les ventes de leurs journaux, lesquels ne pouvant se permettre de déformer la vérité au risque de perdre leur audience. Néanmoins, l'émergence du modèle économique de la publicité a permis l'existence durable de la presse people. Ce modèle économique publicitaire surreprésenté sur internet repose sur le nombre de vues. Ainsi tous les coups sont autorisés pour attirer le plus de personnes.

Le problème des fausses informations et des fake news est devenu, par ailleurs, un enjeu majeur dans le contexte géopolitique actuel. Les informations fausses ou trompeuses peuvent être utilisées comme une arme politique pour manipuler l'opinion publique, influencer les élections, ou encore semer la confusion et la méfiance dans les relations internationales.

Un exemple récent de l'utilisation de la désinformation à des fins politiques est l'élection présidentielle américaine de 2016. Des rapports ont révélé que des groupes liés à la Russie ont mené une campagne de désinformation à grande échelle sur les réseaux sociaux pour influencer l'opinion publique en faveur du candidat Donald Trump. Les fausses informations ont été utilisées pour semer la confusion et la méfiance dans les esprits des électeurs, renforcer les divisions au sein de la société américaine et influencer l'issue de l'élection.

Un autre exemple est la pandémie de Covid-19, qui a vu la propagation rapide de fausses informations sur les réseaux sociaux, notamment sur l'origine du virus, les traitements et les vaccins. Certaines de ces informations ont été propagées délibérément dans le but de semer la confusion et de nuire à la réponse internationale à la pandémie.

La désinformation a également été utilisée dans les conflits armés, où elle peut être utilisée pour manipuler l'opinion publique internationale et pour justifier des actions agressives. Par exemple, le gouvernement russe a été accusé d'avoir propagé des fausses informations sur les événements en Ukraine en 2014 pour justifier son intervention militaire dans la région.

Ainsi les fake news et la désinformation augmentent considérablement en nombre et peuvent prendre des formes de plus en plus divers (Tweets, faux articles scientifiques, faux articles de journaux, etc...).

Une façon de contrer ces fausses informations est de trouver des outils permettant de vérifier les informations. Plusieurs initiatives visant à vérifier (fact-check) les faits ont ainsi vu le jour. Néanmoins, au vue de la quantité de posts et articles fallacieux, il faut songer à automatiser le processus. Une manière de parvenir à cette fin est l'utilisation de méthode de machine learning. Les outils issus du Traitement Automatique du Langages Naturel (TALN) ou Natural Language Processing (NLP) permettent notamment le traitement de l'information lorsque celui-ci est présenté sous forme de texte. D'autres méthodes de classification ou d'analyse sémantique des phrases permettent également de mettre en avant ces fake news. Cependant la mise en place

de ces méthodes nécessite un apprentissage sur des données. Or la recherche de données et de datasets déjà établis est complexe de par la présence de nature différentes dans les types de fake news. Ainsi, des méthodes de scrapping de données doivent etre mise en place mais présentent elles aussi de nombreux défauts. Cette implémentation des techniques théoriques existantes est donc plus complexes qu'une simple application d'algorithme. Le premier objectif de ce travail de groupe est d'arriver de façon satisfaisante à mettre en place différentes méthodes pour détecter les fausses informations à l'aide de programme de machine learning. Ce projet nous permettra notamment de nous familiariser avec ces puissants outils.

1
CADRE DE TRAVAIL

1.1 Qu'est-ce que la Fake News ?

La notion de désinformation est en réalité assez floue[1]. Selon le dictionnaire le Larousse[2], la désinformation est " Utilisation des techniques de l'information de masse pour induire en erreur, cacher ou travestir les faits". Concernant ce concept, la langue française est en fait plutôt restrictive. La désinformation indique en effet une action volontaire de la personne. De plus, le sens de ce mot se rapporte généralement à une volonté de tromper. Pour définir correctement le concept général que nous cherchons à prendre en compte, il faut s'intéresse au terme anglais misinformation.

Selon le dictionnaire anglais Cambridge[3], misinformation se rapport à de l'information fausse (wrong information). Il peut aussi se rapport à la volonté de tromper (information intended to deceive). Note : le concept de désinformation existe aussi en anglais sous la forme desinformation est possède alors le même sens[4]. On peut en fait décomposer la fausse information en plusieur catégories :

- La satire : Le texte est intentionnellement faux avec des informations loufoques donc le but est de faire rire son lecteur. C'est par exemple le cas du journal français le Gorafi ou du site anglais The Onion. Ces textes souvent ne se cachent pas d'être à viser humoristique, par contre ceci n'est pas forcément écrit au sein des articles et peuvent par effet secondaire se répandre sur les réseaux sociaux ;
- L'erreur : l'erreur humaine fait aussi partie des fausses informations que l'on peut retrouver. On peut néanmoins voir différents genres : l'erreur de type typographique et l'erreur de fonds. Dans le premier une coquille se glisse dans l'article alors que dans le second l'auteur peut se tromper complètement sur la théorie ou l'information qu'il rapporte. On pourrait considérer certaines théories complotistes comme une mécompréhension de faits réels. ;
- la désinformation : La désinformation est caractérisée par la volonté de tromper le lecteur pour servir l'objectif de l'auteur en modifiant l'opinion publique.

Il est intéressant de noter dans ce cadre qu'une théorie complotiste peut être considérée différemment selon l'intention de son auteur. Ça serait par exemple le cas pour des personnes opposées à la vaccination entre celles qui pensent réellement que les vaccins sont nocifs contrairement à ceux qui le revendiquent pour des objectifs de nuisances politiques. Dans le cadre de politique publique de santé, il a été important de faire des états des lieux des fausses nouvelles pouvant se retrouver sur internet[5].

Le terme générique Fake News englobe donc l'intégralité de ces notions. Ces notions ont des caractéristiques différentes ce qui les rend difficilement discernable dans leur ensemble par un seul système d'analyse. Le choix de datasets spécifiques ciblés sur un type de fake news permet

de voir la cohérence d'une méthode vis à vis de ce type. Par exemple, le nombre de smileys dans un tweet peut donner une indication sur sa véracité, mais n'est pas décisif sur un article de journal qui n'en contient pas.

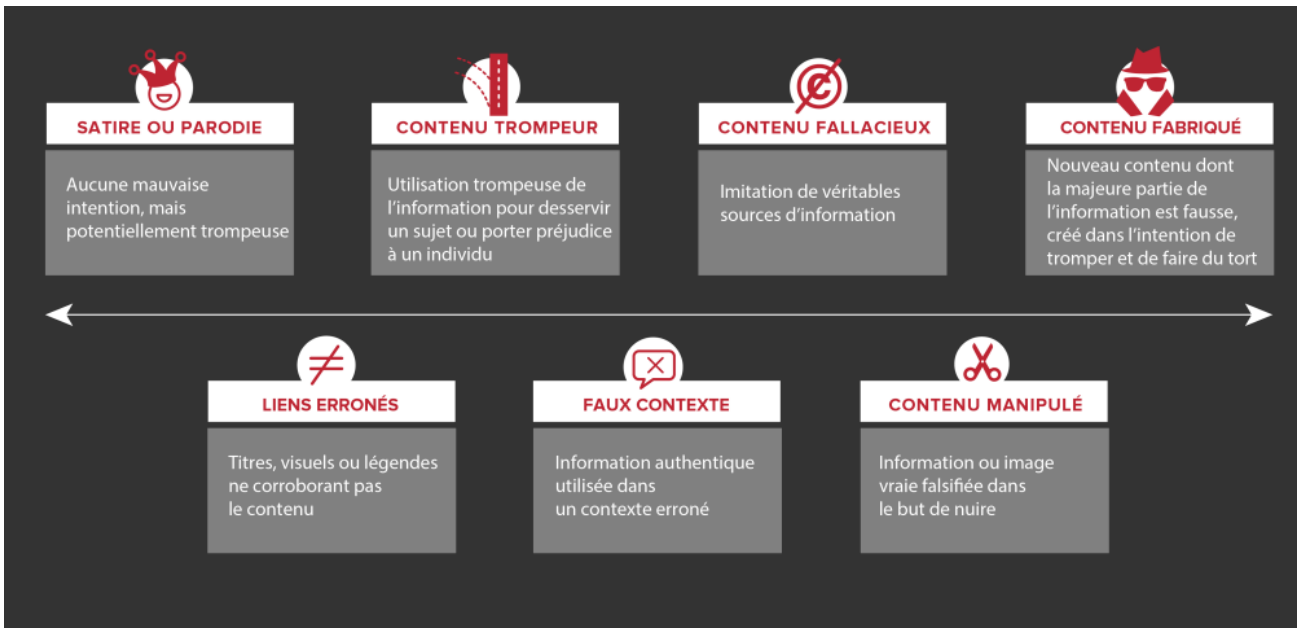


Figure 1 – 7 types de Fake News

1.2 Difficultés d'obtenir un dataset et scrapping de données

Lors des recherches de l'état de l'art, nous avons trouvé de nombreux datasets sur le thème de la mésinformation. Néanmoins, ces banques de données couvrent des thèmes assez large et ne sont donc pas spécifiques au domaine médical. On retrouve assez souvent des sujets sur le thème politique et technologique. En cherchant plus spécifiquement sur le domaine, les banques de données sont majoritairement aux sujets du Covid-19 sur la période de l'année 2021. En effet, cette période a vu de nombreux postes conspirationnistes, de désinformation intentionnelle ou non. Néanmoins, les articles ne mettaient pas à disposition leur base de données afin de pouvoir les utiliser.

Afin d'entrainer les différents modèles sur un grand nombre d'articles, nous nous sommes tournés vers des datasets déjà mais suffisamment récents pour que le style de fake news soient en accord avec la problématique actuelle. Nous nous sommes orientés vers le dataset 'LIAR'[6]. C'est un ensemble de données publiquement disponible, créé pour évaluer les systèmes de détection de fausses informations en utilisant des données réelles.

Il est composé de plus de 12 000 déclarations politiques avec des annotations qui indiquent si elles sont vraies ou fausses. Les déclarations ont été collectées à partir de vérificateurs de

faits, de politiciens et d'autres sources publiques. Les déclarations ont été évaluées selon six niveaux de véracité : vrai, mostly true, half true, barely true, false et pants on fire.

En plus de l'étiquetage de vérité, il contient également des métadonnées, telles que les sujets abordés dans la déclaration, le contexte dans lequel elle a été faite, la source d'information, etc. Cela permet aux chercheurs de développer des modèles de détection de fausses informations plus sophistiqués, qui prennent en compte le contexte et la source de la déclaration.

Il est devenu une référence dans la recherche sur la détection de fausses informations et est largement utilisé pour évaluer les performances des algorithmes de détection de fausses informations. Le dataset a également inspiré la création d'autres ensembles de données similaires, qui permettent d'élargir la recherche sur la détection de fausses informations et de développer des modèles plus robustes et fiables.

Néanmoins, ce dataset possède de nombreux défauts comme la classification en 6 catégories qui meme ramené à 2 catégories vraies et fausses ne donnent pas de résultats probants. Son nombre de texte de 12000 est conséquent mais leurs faibles longueurs de quelques dizaines de mots entraine notamment des problèmes d'overfitting. Il nous a cependant permis de comprendre les défauts de certains modèles et illustre la difficulté qui découle de la définition de fake news. Meme une analyse humaine et spécifique des articles est complexe pour une classification binaire et les articles sont classifiés en 6 catégories.

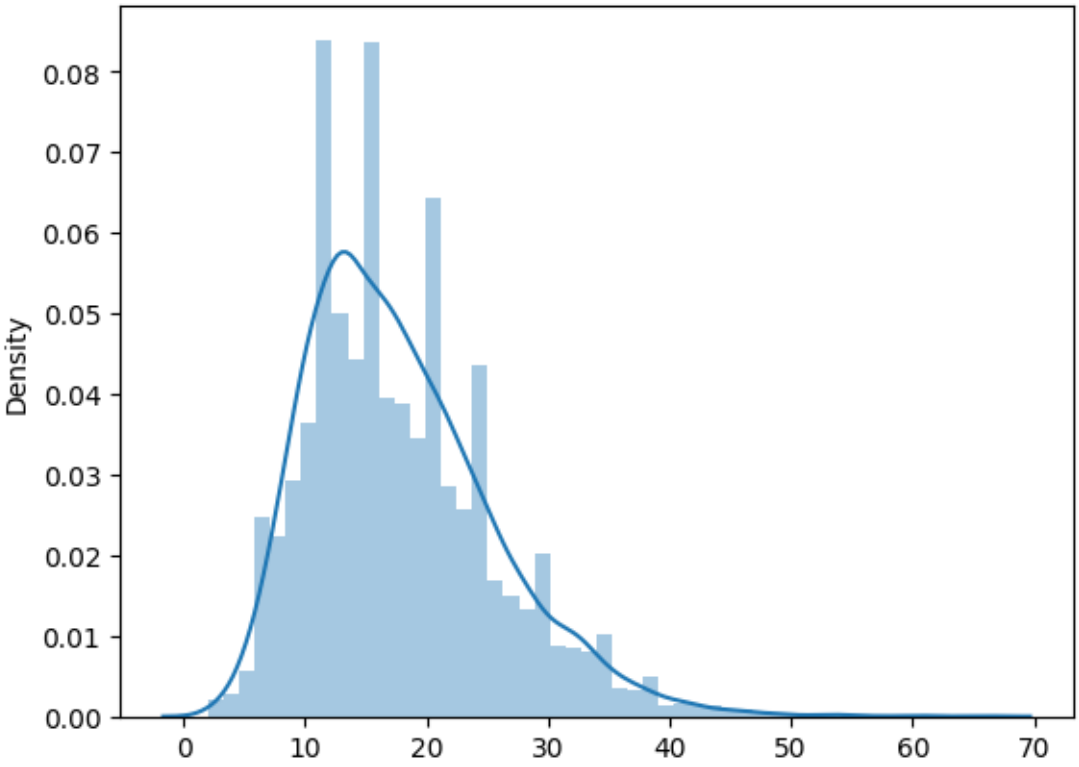
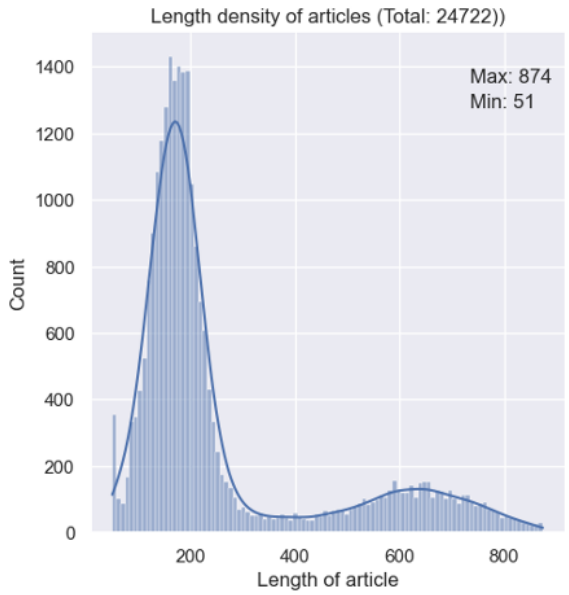


FIGURE 2 – Nombres de mots dans les articles de LIAR

Afin de simplifier la tâche, nous avons un temps considéré de nous restreindre à la satire. Pour cela, nous avons constitué notre propre jeu de données. La difficulté première de la création

d'un dataset est la labellisation lorsque le jeu de données est utilisé pour réaliser un entrainement supervisé. Afin d'obtenir la labellisation des données nous avons utilisé les articles de The Onion et du Guardian. Nous avons fait l'hypothèse que l'ensemble des articles de The Onion étaient à caractère satirique et que ceux du Guardian étaient véridiques.

Afin de récupérer la donnée, les articles de The Onion ont été téléchargés de façon asynchrone à l'aide du module asyncio et puis nous avons récupéré les données pertinentes des pages html à l'aide du module Beautiful Soup. Une analyse de la structure des pages html nous a permis de rapidement identifier les headers permettant d'obtenir le titre, la date, le thème et bien sûr le texte de l'article. Les url des pages ont été facilement obtenus en parcourant le sitemap du site. Le sitemap est un document présent généralement sur les sites pour décrire l'arborescence des pages se trouvant sur le site. C'est un fichier qui est notamment utilisé par les robots réalisant l'indexation des pages sur les moteurs de recherches. Les données avant préprocessing s'étendent de 2002 à 2023 avec un total de 33 189 articles. Après avoir enlevé les articles trop longs et trop courts et les pages ne comportant qu'une vidéo, le dataset comporte 24 722 articles. Une analyse permet de retrouver la distribution des longueurs des textes et la répartition des thèmes. La distribution est trouvée à l'aide du module seaborn qui réalise une estimation par noyau (kernel distribution estimation).



Number of different themes: 40	
News In Brief	7358
Sports	3908
Local	3576
Politics	2949
News	1645
Entertainment	1617
Infographic	1097
Commentary	904
Slideshow	446
American Voices	318
OGN	308
Breaking News	187

FIGURE 3 – Répartition des articles de The Onion

Les articles du Guardian sont accessibles via leur API Open Platform[7]. Du fait des limitations journalières de l'API, les articles ont été récupérés directement sur Kaggle[8].

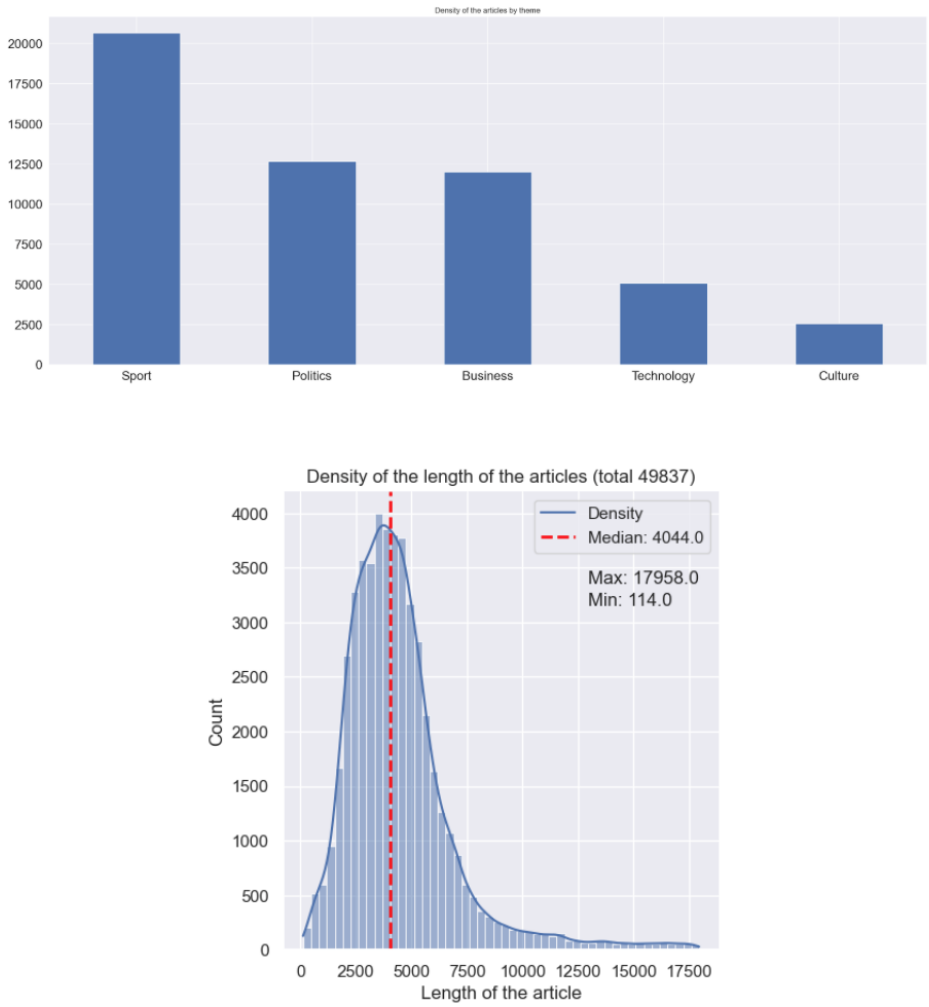


FIGURE 4 – Répartition des articles du Guardian

Malheureusement, ce dataset issue d'un scrapping de sites Internet contient des articles qui ne sont pas assez similaires. En effet, les articles de The Onion ont une longueur d'une moyenne de 250 mots alors que les articles de The Guardian sont proches de 5000 mots. Cette différence de tokens en entrée rend peu utilisable le dataset en raison de ce biais trop forte entre les deux catégories d'articles. L'idée de se retreindre à la satire sur ce dataset a donc été abandonné.

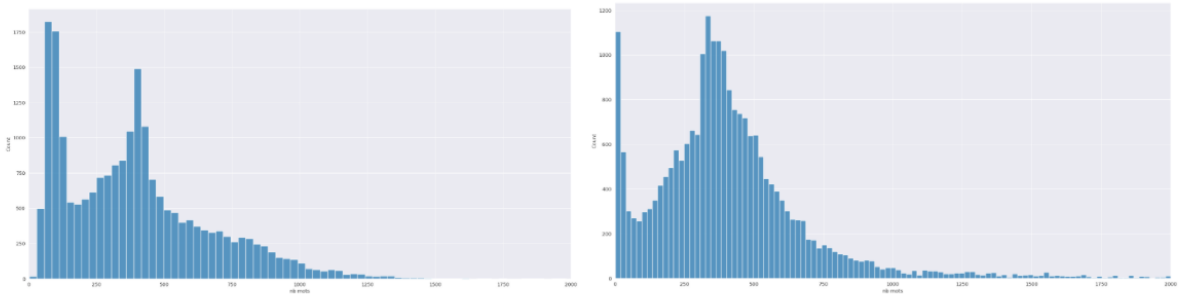
1.3 DATASETS RETENUS

En apprentissage approfondi, le choix des jeux de données d'entraînement est tout aussi important que le choix du modèle, en effet, des jeux de données biaisés peuvent entrainer une spécialisation du modèle à détecter le biais plutôt que les caractéristiques sous jacente, ce qui donne de bonne performance sur les données d'entrainement mais très mauvais pour une utilisation avec d'autres données.

De surcroît, le manque de datasets spécialisés disponibles librement sur internet nous a poussés à adopter deux jeux de données axés politique [9] [10] [11], avec quelques autres structurés et équilibrés par des experts.

1.3.1 • Dataset d'entrainement

Le premier dataset est une collection de 44 900 articles de faits divers, qui ont été rédigés dans un style professionnel. Les faux articles ont été créés de manière à être assez similaires aux articles véridiques, ce qui permet de réduire les biais liés à la taille des textes. Il est important de noter que certains faux textes sont plus courts, ce qui les rend similaires aux fakes news partagées sur les réseaux sociaux.



nb mots textes vrais | nb mots fakes

Nombres de mots	Moyenne	Variance
Textes Vrais	394.40	281.06
Textes faux	435.23	420.99

Figure 5 – Données sur le dataset d'entrainement

Une analyse des mots les plus fréquents dans chaque type de textes montre que les textes partagent globalement les mêmes mots clés, c'est-à-dire que des modèles statistiques ne seraient probablement pas suffisants, et qu'il faudrait pousser vers des modèles avec une meilleure compréhension sémantique.

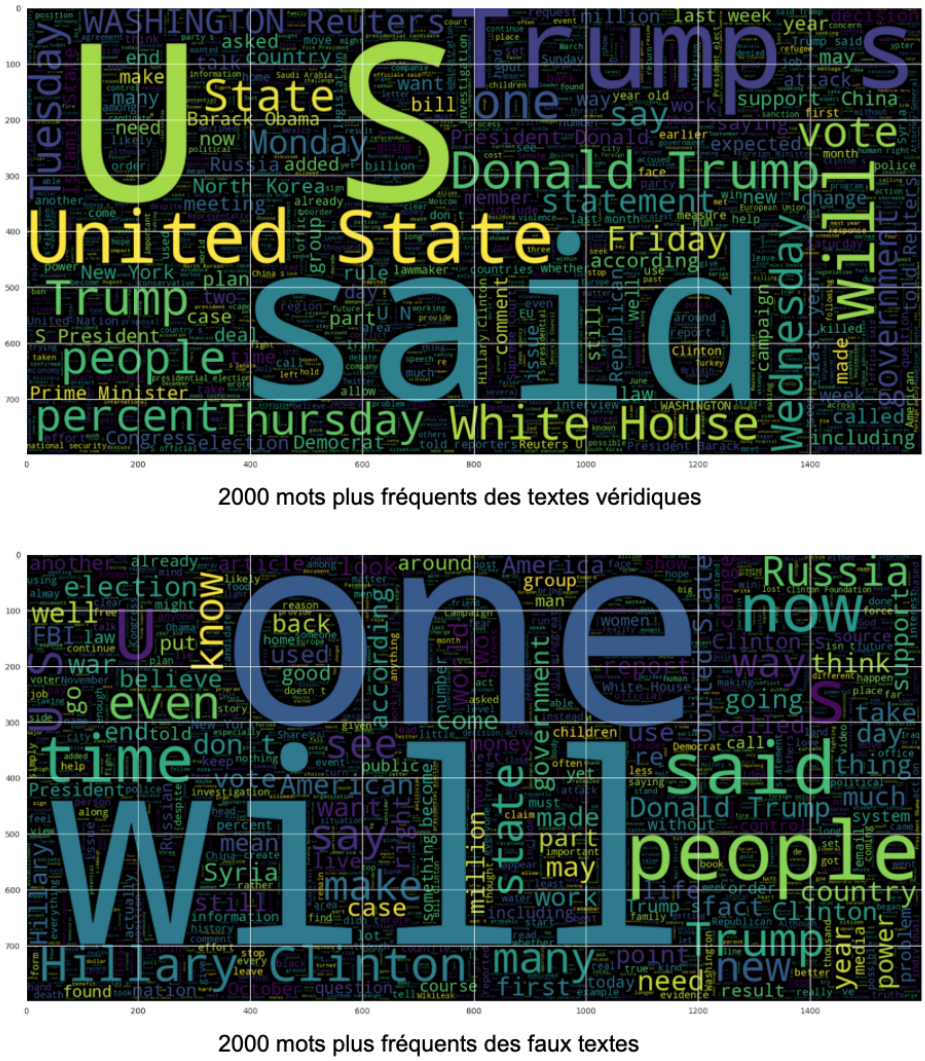


FIGURE 6 – Analyse de la fréquence d'apparition des mots

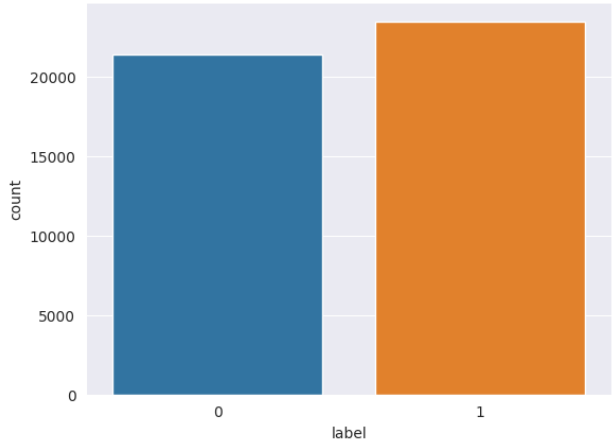
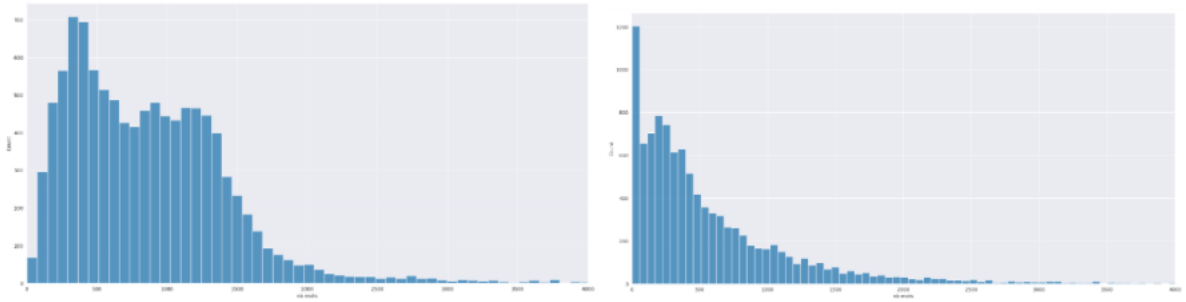


FIGURE 7 – Répartition des données

Ce jeu de données possède un nombre de fake news et de news véridiques identique, ceci pourrait poser problème à la généralisation de l'apprentissage vers d'autres types de textes.

1.3.2 • Dataset de validation

Le deuxième dataset est composé de 22 000 textes de longueurs variables, et il est beaucoup moins formel que le premier. Les textes sont hétérogènes dans leur ensemble, les textes véridiques étant principalement des articles, tandis que les faux textes peuvent être des articles, des tweets viraux, des faux débats ou des discours satiriques. Cette variété de représentations des fake news pose différents problèmes aux modèles lors de la généralisation de la classification d'un type de données à un autre.



Nombre de mots	Moyenne	Variance
Textes Vrais	907.62	765.39
Textes faux	639.11	965.26

Figure 8 – Données sur le dataset de validation

Notre dataset contient des articles très variés en longueur, contenue et thèmes traités. Ainsi, parmis tous les dataset trouvés, on utilisera ce dernier afin de tester nos modèles.

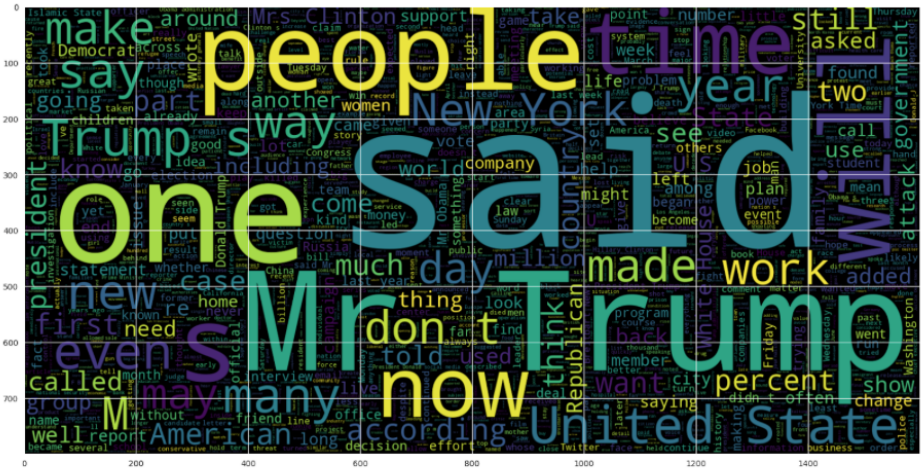


FIGURE 9 – Analyse de la fréquence d'apparition des mots

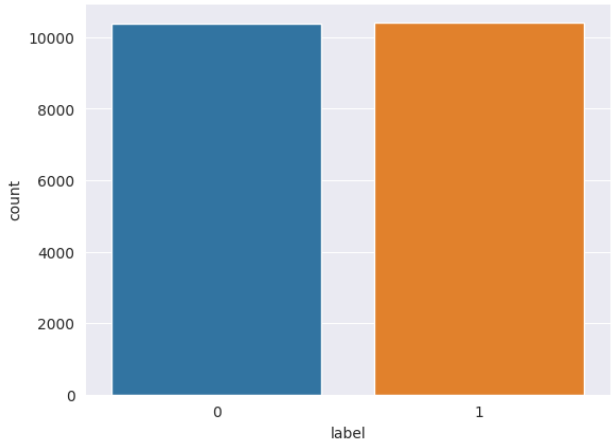


FIGURE 10 – Répartition des données

Bien que le dataset soit varié concernant la syntaxe et champs lexical, la longueur des articles pourrait induire un biais. En effet, les articles classés comme “True” sont de taille plus grande que les “Fake”, ainsi nos modèles pourraient considérer la longueur de l’article comme indicateur.

2
APPROCHE PAR MACHINE LEARNING CLASSIQUE

Dans cette partie, on s'inspire d'un papier de recherche réalisé par Dharmaraj R. Patil de Institute of Technology, Shirpur, Maharashtra, India.[12].

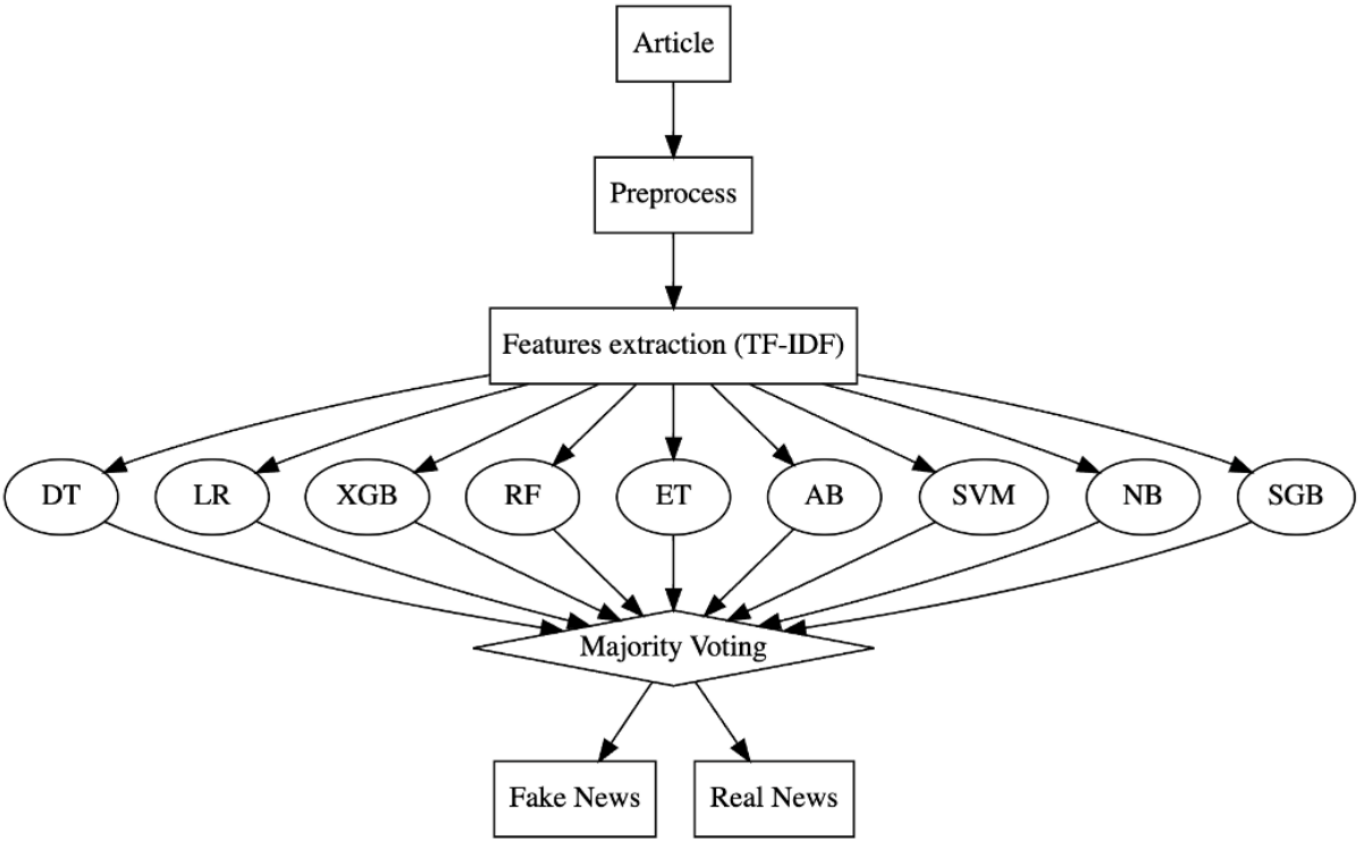


FIGURE 11 – Méthode employée

2.1 Présentation du Pre-processing et Features Extraction

De manière générale, après avoir traité nos données, notamment en retirant la ponctuation, tokenisation, suppression des mots vides, racinisation des mots (word stemming), suppression des URL et des noms, un premier réflexe serait d'utiliser un modèle basique de classification afin

de voir si les méthodes simples fonctionnent. Ces prétraitements utilisés permettent d'enlever de l'information inutile ce qui permet une meilleure performance des modèles de classification.

Ainsi, on peut extraire les caractéristiques de notre dataset, c'est un processus de conversion des données textuelles brutes en un ensemble structuré de caractéristiques ou d'attributs qui peuvent être utilisés pour entraîner des modèles d'apprentissage automatique ou pour réaliser des analyses. Ces caractéristiques peuvent inclure des informations telles que la fréquence des mots, la présence de certaines phrases, la structure grammaticale, la similarité sémantique et d'autres éléments qui aident à comprendre et à représenter le texte de manière plus significative pour les algorithmes. Dans notre cas, concernant ces modèles de classification, on a utilisé TFIDF Le premier terme (TF) représente l'importance relative des mots dans un ensemble de documents. Plus un mot apparait fréquemment, plus sa valeur est élevée.

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{1}$$

ft,d représente le décompte brut d'un terme dans un document, c'est-à-dire le nombre de fois que le terme t apparaît dans le document d. Le second terme (IDF) mesure l'importance d'un mot dans l'ensemble des articles. Elle est calculée en prenant le logarithme du rapport entre le nombre total de documents et le nombre de documents contenant le mot en question. Un mot qui apparaît rarement dans l'ensemble des documents aura une valeur IDF plus élevée, ce qui signifie qu'il est considéré comme plus important et informatif.

$$IDF(t,D) = log\frac{N}{|\{d \in D : t \in d\}|} \tag{2}$$

Où N représente le nombre total d'articles du dataset. Et | d Î D : t Î d | nombre total d'articles ou le mot t apparaît. Finalement, on utilise le terme TFIDF calculé par :

$$TFIDF(t,d,D) = TF(t,d) * IDF(t,D) \tag{3}$$

2.2 Présentation de 9 modèles de classification

Nous avons évalué les performances de 9 classificateurs d'apprentissage automatique pour la détection des Fake News, y compris Decision Tree (DT), la Régression Logistique (LR), XGBoost (XGB), Random Forest (RF), Extra Trees (ET), AdaBoost (AB), Support Vector Machine (SVM), Descente de Gradient Stochastique (SGD) et le Naive Bayes (NB). Parmi ces 9 classificateurs d'apprentissage automatique, nous avons construit un classificateur multi-modèles en utilisant le vote majoritaire pour la décision finale. Nous avons utilisé l'implémentation de chaque classificateur dans la bibliothèque Python scikit-learn [13]. Une brève discussion sur chaque classificateur est présentée ci-dessous[14][15].

1. Decision Tree

Un arbre de décision est un type d'algorithme de classification utilisé dans l'analyse de données, y compris le traitement du langage naturel (NLP). Il fonctionne en divisant répétitivement les données en sous-groupes plus petits en se basant sur les caractéristiques des données, créant ainsi une structure d'arbre. L'objectif est d'organiser les données de manière à ce que les groupes formés soient le plus homogène possible en termes de variable cible (la classe à prédire). Imaginez un arbre avec des branches et des feuilles. Chaque branche représente une décision basée sur une caractéristique, et chaque feuille représente une prédiction de la variable cible. Lorsqu'une nouvelle observation doit être classée, on la fait traverser l'arbre en suivant les branches correspondant à ses caractéristiques, jusqu'à ce qu'elle atteigne une feuille qui fournit la prédiction. Les arbres de décision sont appréciés pour leur interprétabilité, car ils peuvent être visualisés et compris facilement.[16][17]

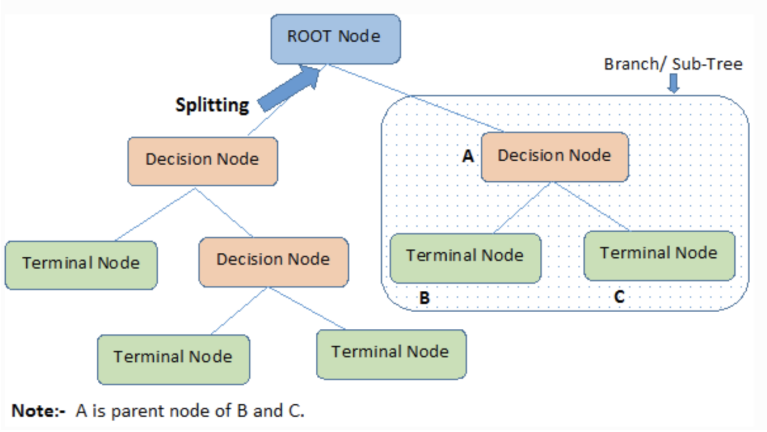


FIGURE 12 – Tree of decisions
[7]

2. Régression Logistique

La régression logistique est un algorithme populaire de machine learning utilisé dans l'approche d'apprentissage supervisé. Elle est utilisée pour prédire la variable dépendante catégorique à partir d'un ensemble de variables indépendantes. La régression logistique prédit le résultat d'une variable dépendante catégorique. Par conséquent, le résultat doit être une valeur catégorique ou discrète. Il s'agit dans notre cas de "0" ou "1 ". Au lieu d'afficher des valeurs précises comme 0 et 1, la régression logistique offre des valeurs de probabilité qui se situent entre 0 et 1.

$$\hat{p}(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + ... + \beta_n x_n)}} \tag{4}$$

où : $\hat{p}(x)$: Probabilité prédite pour la variable cible.
$\beta_0, \beta_1, ..., \beta_n$: Coefficients de régression.
$x_1, ..., x_n$: Caractéristiques des données.

3. XGBoost

   L'approche XGBoost (eXtreme Gradient Boosting) est bien connue et réussie. Le boosting de gradient est une approche d'apprentissage supervisé qui combine des estimations à partir d'une série de modèles plus simples et plus faibles pour prédire correctement une variable cible. En raison de sa forte gestion d'une large gamme de types de données, de relations et de distributions, ainsi que de l'énorme gamme d'hyperparamètres qui peuvent être ajustés, la technique XGBoost fonctionne bien dans les problèmes d'apprentissage automatique. XGBoost est capable de traiter des problèmes de régression, de classification (binaire et multiclasse) et de classement.[18]

$$Ł(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}i) + \sum k = 1^K \Omega(f_k) \tag{5}$$

   où $L(\phi)$: Fonction objective à minimiser.
   $(y_i, \hat{y}_i)$: Fonction de perte pour la i-ème observation.
   $\Omega(f_k)$: Terme de régularisation pour le k-ème arbre.
   n : Nombre d'observations.

4. Random Forest

   Une forêt aléatoire, comme son nom l'indique, est composée d'un grand nombre d'arbres de décision individuels qui collaborent en tant qu'ensemble. La forêt aléatoire génère une prédiction de classe pour chaque arbre, et la classe ayant le plus de votes devient la prévision de notre modèle. Le principe de base de la forêt aléatoire est la connaissance communautaire, qui est à la fois simple et puissante. Le modèle de forêt aléatoire est particulièrement réussi car il est composé d'un grand nombre de modèles (arbres) largement non corrélés qui collaborent pour surpasser chacun des modèles constituants individuels.

5. Extra Tree

   Le classificateur Extra Trees est un type de technique d'apprentissage ensembliste qui agrège les résultats de classification de plusieurs arbres de décision non corrélés rassemblés dans une "Forêt" pour obtenir ses résultats de classification. Il est conceptuellement très similaire au classificateur Random Forest et diffère principalement dans la manière dont les arbres de décision de la forêt sont formés. Les arbres de décision de la forêt Extra Trees sont construits à partir d'échantillons d'entraînement. Ensuite, à chaque nœud de test, chaque arbre reçoit des échantillons aléatoires de k caractéristiques à partir des ensembles de caractéristiques. À partir desquelles chaque arbre de décision doit sélectionner les meilleures caractéristiques pour diviser les données en utilisant un critère mathématique important. Cette sélection aléatoire de caractéristiques aboutit à la construction de plusieurs arbres de décision non corrélés.

6. AdaBoost

AdaBoost est un algorithme d'apprentissage automatique qui combine un grand nombre de classificateurs faibles pour former un classificateur fort et précis. Il assigne à chaque classificateur un poids en fonction de sa performance, de sorte que les classificateurs les plus performants ont un poids plus élevé dans la combinaison finale. Cela permet à AdaBoost de donner plus de poids aux exemples mal classés, ce qui améliore la précision du modèle.

$$w_i^{(t+1)} = w_i^{(t)} \cdot e^{-\alpha_t y_i h_t(x_i)} \tag{6}$$

où $w_i^{(t+1)}$: Poids de la i-ème observation à l'itération (t+1).
$w_i^{(t)}$: Poids de la i-ème observation à l'itération t.
α_t : Poids du modèle à l'itération t.
y_i : Vraie valeur de la i-ème observation.
$h_t(x_i)$: Prédiction du modèle à l'itération t pour la i-ème observation.

7. Support Vector Machine

Support Vector Machine (SVM) est un algorithme d'apprentissage automatique utilisé pour la classification et la régression. Il cherche à créer un hyperplan qui sépare les exemples de différentes classes de manière optimale dans un espace multidimensionnel. L'objectif de SVM est de maximiser la marge, c'est-à-dire la distance entre l'hyperplan et les exemples les plus proches de chaque classe. Les exemples qui se trouvent à la marge sont appelés vecteurs de support, d'où le nom de l'algorithme. SVM est souvent utilisé pour des tâches de classification binaire, mais peut être étendu à des problèmes de classification multiclasse.

$$\min_{w,b} \frac{1}{2}||w||^2 \quad sous\, contrainte \quad y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, n \tag{7}$$

où : w : Vecteur des poids.
b : Biais.
y_i : Vraie valeur de la i-ème observation.
x_i : Caractéristiques de la i-ème observation.
n : Nombre d'observations.

8. Stochastic Gradient Descent

La descente de gradient stochastique (SGD) est une approche simple mais très efficace pour ajuster des classificateurs et des régresseurs linéaires avec des fonctions de perte convexe telles que les Machines à Vecteurs de Support (SVM) linéaires et la régression logistique. Bien que la SGD soit présente depuis longtemps dans la communauté de l'apprentissage automatique, elle a reçu une attention considérable récemment dans le

contexte de l'apprentissage à grande échelle. La SGD a été appliquée avec succès à des problèmes d'apprentissage automatique à grande échelle et clairsemés, souvent rencontrés dans la classification de texte et le traitement du langage naturel. Le classificateur SGD implémente essentiellement une routine d'apprentissage SGD simple prenant en charge diverses fonctions de perte et pénalités pour la classification. Scikit-learn fournit le module SGDClassifier pour implémenter la classification SGD.[19]

$$w_{t+1} = w_t - \eta_t \nabla L(w_t) \tag{8}$$

où : w_{t+1} : Vecteur des poids à l'itération (t+1).
w_t : Vecteur des poids à l'itération t.
η_t : Taux d'apprentissage à l'itération t.
$\nabla L(w_t)$: Gradient de la fonction de perte par rapport aux poids à l'itération t.

9. Naive Bayes

Le classificateur Naive Bayes est un modèle de machine learning probabiliste utilisé pour des tâches de classification. Sa base repose sur le théorème de Bayes. En utilisant ce théorème, nous pouvons trouver la probabilité que l'événement A se produise, sachant que l'événement B s'est produit. Ici, B est l'évidence et A est l'hypothèse. L'hypothèse naïve de ce modèle est que les prédicteurs ou les caractéristiques sont indépendants les uns des autres, c'est-à-dire que la présence d'une caractéristique particulière n'affecte pas les autres. Cette hypothèse simplifie grandement le calcul des probabilités et permet d'utiliser des données de grande dimension avec un temps de calcul raisonnable. En somme, le classificateur Naive Bayes est une méthode simple mais puissante pour la classification qui est largement utilisée dans des applications de classification de texte et de reconnaissance de caractères.

$$P(y_k|x_1, \ldots, x_n) \propto P(y_k) \prod_{i=1}^{n} P(x_i|y_k) \tag{9}$$

où : $P(y_k|x_1, ..., x_n)$: Probabilité conditionnelle de la classe $\mathrm{y}_k étantdonnéles caractéristiques \mathrm{x}_1, ..., $
$P(y_k)$: Probabilité a priori de la classe y_k.
$P(x_i|y_k)$: Probabilité conditionnelle de la caractéristique $\mathrm{x}_i étantdonnélaclasse \mathrm{y}_k$.
n : Nombre de caractéristiques.

Après avoir entrainé les 9 différents modèles sur notre dataset d'entraînement, on obtient des valeurs de d'accuracy, précision, rappel et f1-score assez satisfaisante. Le graphique suivant résume toutes les valeurs des métriques afin de comparer entre les différents modèles de machine learning utilisés.

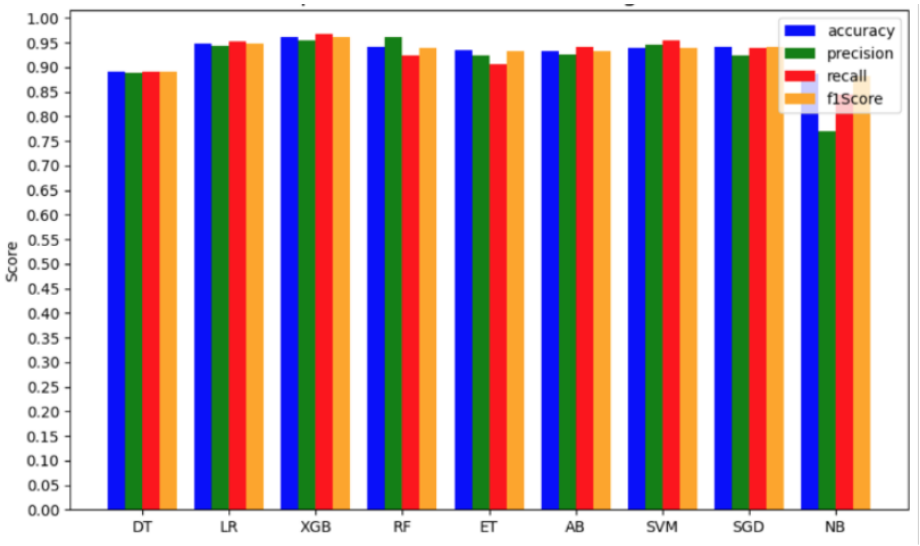


FIGURE 13 – Comparaison des 9 modèles

On voit que seuls, Decision Tree et Naive Bayes sortent du lot avec des résultats inférieurs aux autres, ainsi pour notre modèle et ce dataset, Decision Tree et Naïve Bayes sous performe comparer aux autres. On peut aussi visualiser ces métriques en faisant 5 itérations au lieu d'une seule afin de confirmer les valeurs d'accuracy obtenues. On voit que l'évolution est assez homogène et donc le résultat de nos métriques serait fiable. En voyant ces graphiques on peut émettre la même conclusion ci-dessus.

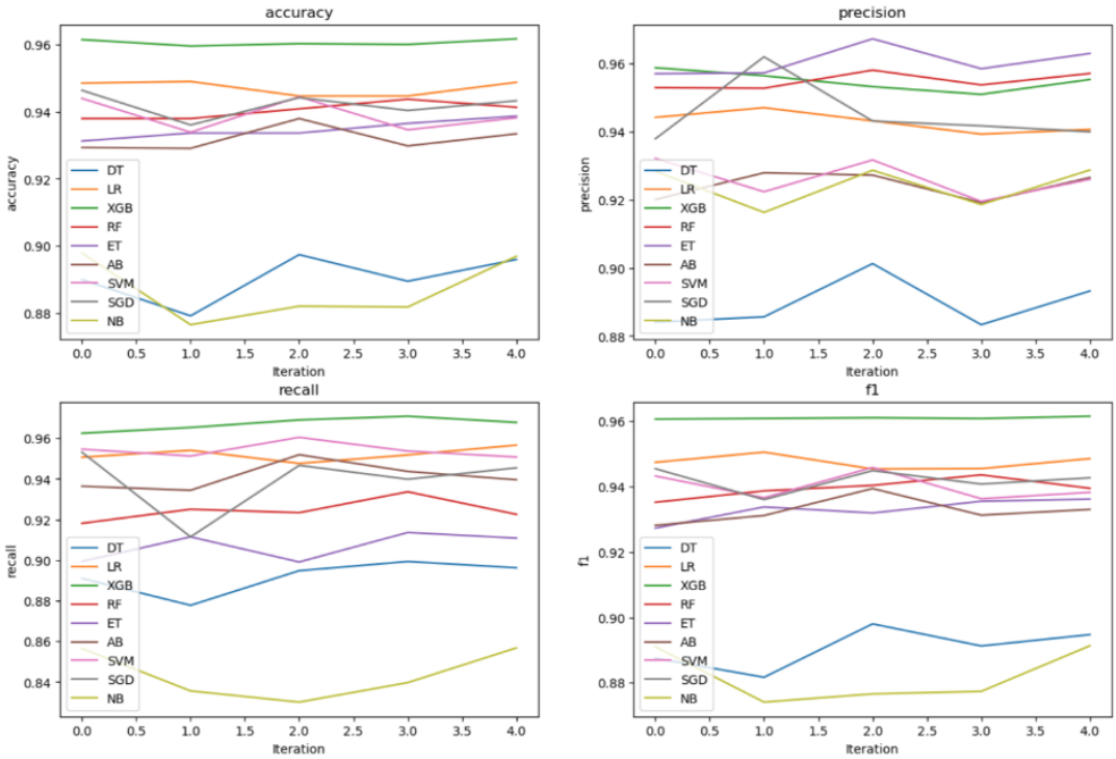


FIGURE 14 – Test des 9 différents modèles sur 5 itérations différentes

Cependant, en suivant le papier de recherche, afin d'avoir une vision globale ou définitive, on devrait utiliser le jugement majoritaire avec de sortir avec une seule réponse (fake ou pas) pour chaque article.

2.3 Majority Voting

Comme décrit précédemment, nous avions 9 modèles basiques de machine learning qui donnent des résultats différents entre chaque article, une harmonisation des résultats s'impose.

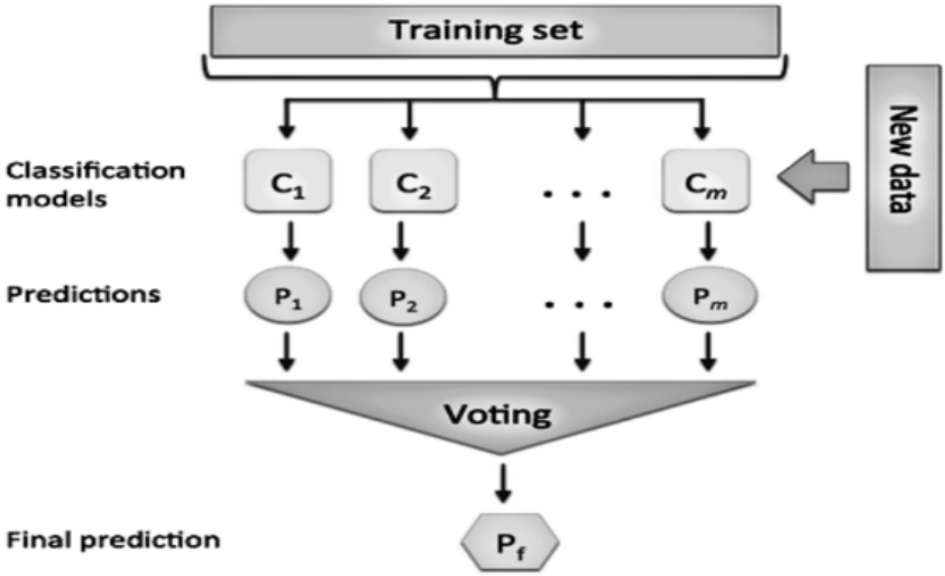


FIGURE 15 – Classifieur du vote majoritaire

Une méthode souvent utilisée est le majority voting (ou vote majoritaire), qui est une technique de combinaison de modèles en machine learning. L'idée principale derrière le majority voting est de prendre la décision finale en fonction de la majorité des prédictions des modèles individuels. On prend la classe qui a été prédite le plus grand nombre de fois, ensuite la classe majoritaire prédite par le vote majoritaire est ensuite considérée comme la décision finale. Le majority voting est généralement utilisé pour augmenter la précision des modèles en utilisant les prédictions des différents modèles. Cependant, il est important de noter que le vote majoritaire ne fonctionne que si les modèles individuels ont des erreurs non corrélées. Si tous les modèles ont des erreurs similaires, le vote majoritaire ne donnera pas une amélioration significative car les modèles prédiront tous la même classe. Nous allons vérifier cela en visualisant le coefficient de corrélation des prédictions des 9 différents modèles.

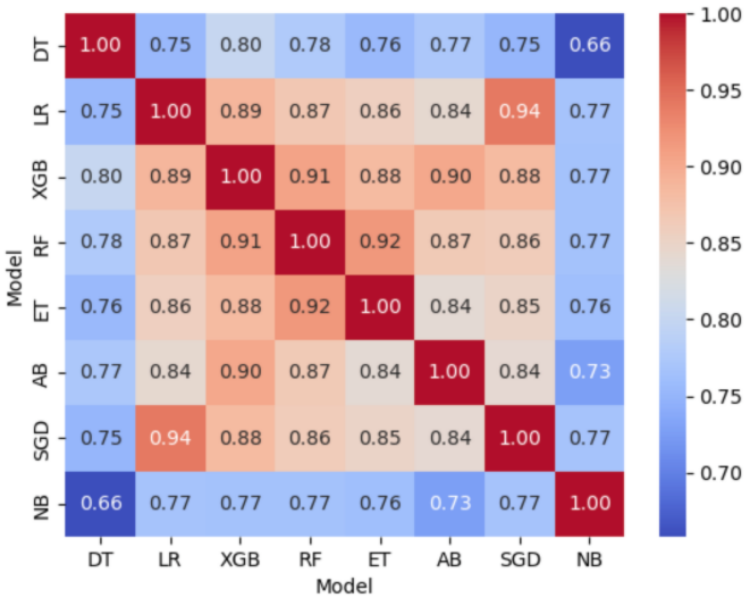


FIGURE 16 – Coefficient de corrélation des prédictions des 9 différents modèles

Sur cette matrice, on voit que bien que les différents classifieur aient des métriques (accuracy) satisfaites, ils sont tous assez décorrélés puisqu'ils oscillent entre 0.65 et 0.92.

Maintenant qu'il est légitime d'utiliser cette méthode de vote, on voit qu'elle nous donne des résultats encore plus satisfaisants que chacun des 9 modèles, avec cette matrice de confusion, et une accuracy de 0.9602 sur le test.

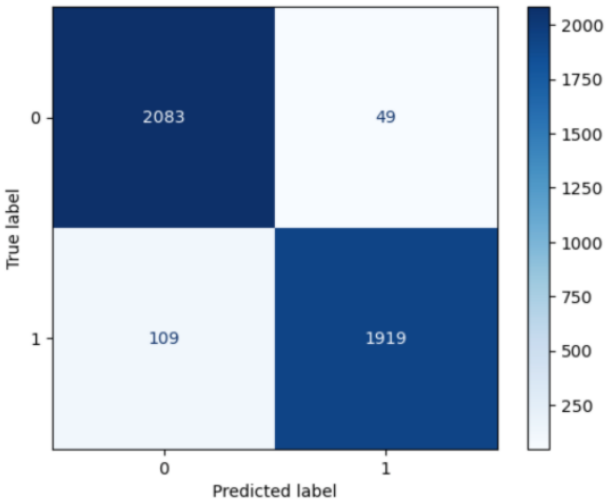


FIGURE 17 – Matrice de confusion

2.4 GÉNÉRALISATION

Une fois que le modèle a été entraîné, on décide de le tester sur un autre dataset, afin de confirmer ce modèle, qui de prime-abord a l'air de fonctionner parfaitement. La généralisation

se fait sur le dataset de test présenté précédemment.

Dans le premier dataset, le modèle a été entraîné à partir de données spécifiques, tandis que dans le deuxième dataset, les données ont été collectées auprès d'une source différente. Bien que le modèle puisse ne pas être capable de généraliser parfaitement à de nouveaux ensembles de données, il a tout de même réussi à produire une précision (accuracy) de 73% sur le deuxième dataset, ce qui est encourageant. A titre de comparaison, un modèle aléatoire aurait une précision de 50%, le modèle ayant une chance sur deux de prédire la bonne classe.

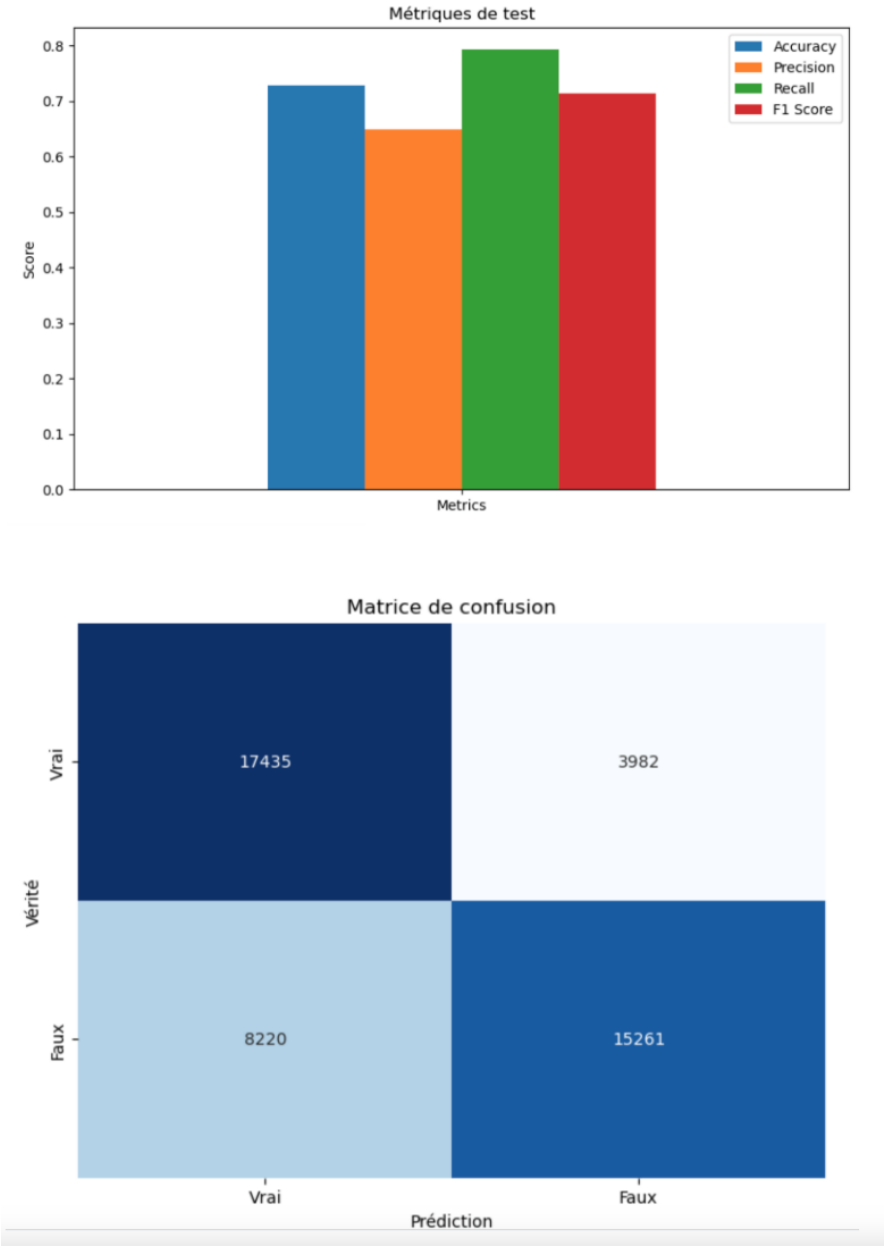


FIGURE 18 – Résultats sur le dataset de validation

Il convient de noter que la méthode du vote majoritaire pour la classification de fake news est déjà présentée dans la littérature scientifique. En effet, nous nous sommes inspirés d'un papier

de recherche qui a utilisé cette approche avec succès sur des ensembles de données similaires. Bien que notre étude ait produit des résultats encourageants, il est important de souligner que les performances de notre modèle peuvent différer de celles du modèle présenté dans le papier de recherche d'origine. De plus, il est essentiel de prendre en compte les différences dans les ensembles de données utilisés, ainsi que les caractéristiques et les hyperparamètres spécifiques du modèle. Dans l'ensemble, le vote majoritaire reste une approche prometteuse pour la classification de fake news, et elle peut être utilisée en conjonction avec d'autres approches pour améliorer la précision globale.

3
APPROCHE PAR LE DEEP LEARNING

3.1 Importance du plongement (Embedding)

L'un des points centraux de l'analyse du texte est de parvenir à un prolongement efficace : éviter les grandes dimensions et préserver la proximité lexicale. Deux approches sont possibles pour cela : les méthodes non ordonnées et les méthodes syntaxiques.
Les méthodes non ordonnées ne tiennent pas compte de l'ordre des mots dans la phrase ou le paragraphe afin de réaliser l'embedding contrairement aux méthodes syntaxiques où l'on tient compte de cette ordre. Bien évidemment l'ordre est important : "Il n'est pas méchant" pour marquer une double négation. "Brutus a organisé le meutre de Cesar" et "Cesar a organisé le meutre de Brutus" n'ont pas le même sens, d'ailleurs l'une est vraie et l'autre fausse. Pourtant, ces deux phrases auraient la même représentation. Néanmoins, l'approche non ordonnée peut avoir des résultats assez bons et en contrepartie est plus simple à mettre en place et peut être plus rapide.
On pourra utiliser la similarité cosinus selon la formule suivante :

$$similarity = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

On cherche par ce prolongement à avoir une valeur de similarité cosinus d'autant plus proche de 1 que le sens des mots ou documents sont proches. Au contraire, si les phrases non pas de rapport, on s'attend à ce que cette valeur soit proche de 0. Ces représentations vectoriels sont apprises et optimisées par la descente de gradient lors de l'entrainement.

Parmi les approches non ordonnées, on retrouve le Count Vectoriser et le Term Frequency-Inverse Document Frequency (TF-IDF). Ces approches ont été de nombreuses fois comparées [21][22] [23]. On part d'un corpus de M documents, on note N le nombre de mots différents. Pour le Count Vectoriser, on associe à chaque document un vecteur de taille N où les coordonnées de 1 à N correspondent aux décomptes de mots uniques du paragraphe.
En fait, vectoriser en réalisant le comptage ou en notant la fréquence des mots dans le document sont deux approches similaires. En effet, la propriété important pour les vecteurs est la similarité cosinus (cosine similarity) selon la formule A INSERE. Ainsi ce qui compte est le vecteur normalisé. L'approche fréquentiste est proposé dès 1957 par Hans Peter Lun[24] . Par contre, l'approche TF-IDF prend en compte la fréquence d'apparition du mot au sein de l'ensemble des documents. La simple approche fréquentiste donne des poids importants à des mots communs ayant peu de charge sémantique comme "le", "la", "un". Ces mots ne participent pas au fond du texte, on peut donc les retirer sans changer le sens du texte. En les enlevant des vecteurs, on retire une contribution qui augmente la valeur de la similarité cosinus. Avec le terme IDF, leur poids au sein du vecteur devient 0 car ces mots sont présents dans tous les textes. Ces mots sont appelés mots vides ou stop words.

Ces prolongements sont faciles et rapides à réaliser. Ces approches sont généralement utilisées lors de tâche de classification. Elles reposent sur l'hypothèse que les textes ont des distribution de mots différents selon leur catégorie. Cela peut relativement bien fonctionner, un exemple étant le filtrage de spam pour les mails. Dans un cadre professionnel, les spams traitent de sujets non professionnels ou sinon de thèmes souvent différents aux mails généralement reçus. Les mots utilisés sont alors différents et la distinction est facile à réaliser.

Néanmoins le problème principal de ces prolongements est la taille importantes des dimensions des vecteurs. En effet, sans traitement préalable, le nombre de mots unique peut rapidement dépasser 10 000. Les algorithmes que l'on peut utiliser par la suite risque de souffrir du fléau de la dimension ou curse of dimensionality en anglais. Plusieurs algorithmes ont une complexité exponentielle selon la dimension des vecteurs. De plus, les données deviennent de plus en plus dispersées dans l'espace des caractéristiques, ce qui signifie qu'il est plus difficile de trouver des modèles qui peuvent séparer les différentes classes de données.

Pour réduire la dimension, il est possible d'effectuer une analyse des composantes principales ou principal component analysis (PCA). La PCA transforme un ensemble de variables corrélées en un ensemble de variables non corrélées, appelées composantes principales, qui représentent la variance maximale des données. La PCA est basée sur la décomposition en valeurs propres de la matrice de covariance des données. La matrice de covariance mesure la relation linéaire entre les variables d'un ensemble de données, et la décomposition en valeurs propres permet de déterminer les directions de plus grandes variances dans les données. Les premières composantes principales correspondent aux vecteurs propres associés aux valeurs propres les plus élevées de la matrice de covariance. Une fois que les composantes principales ont été calculées, elles peuvent être utilisées pour projeter les données d'origine dans un nouvel espace de dimension réduite. Les données projetées conservent les informations les plus importantes sur les données d'origine, ce qui permet d'effectuer des analyses plus simples et plus rapides.

3.2 Utilisation de CNN et de LSTM

Un réseau de neurones récurrent (RNN) est un type de réseau de neurones artificiels utilisé pour traiter des données séquentielles, telles que des séquences de mots dans un texte ou des séquences de signaux dans une série temporelle.

L'architecture LSTM (Long Short-Term Memory) est un type de réseau récurrent qui a été développée pour surmonter le problème de la disparition du gradient dans les RNN standard. Dans les RNN standards, le gradient qui est utilisé pour ajuster les poids du réseau lors de l'apprentissage peut devenir très petit au fil du temps, ce qui rend difficile la propagation de l'information sur de longues séquences temporelles.

Les LSTM résolvent ce problème en introduisant des unités de mémoire spéciales appelées

"cellules LSTM". Ces cellules ont des portes (gate en anglais) qui contrôlent l'accès à la mémoire de la cellule et régulent le flux d'informations dans le réseau. Les portes permettent à la cellule de mémoriser l'information pertinente pour le traitement des séquences à long terme et de la récupérer en fonction des besoins.

Les portes sont gérées par des neurones spéciaux appelés "neurones de portes". Il y a trois types de neurones de portes dans une cellule LSTM : la porte d'entrée, la porte de sortie et la porte d'oubli. La porte d'entrée permet de décider quelles informations de la séquence d'entrée doivent être stockées dans la cellule de mémoire. La porte d'oubli permet de décider quelles informations de la cellule de mémoire doivent être oubliées ou conservées. La porte de sortie permet de décider quelle information stockée dans la cellule de mémoire doit être renvoyée à la sortie du réseau.

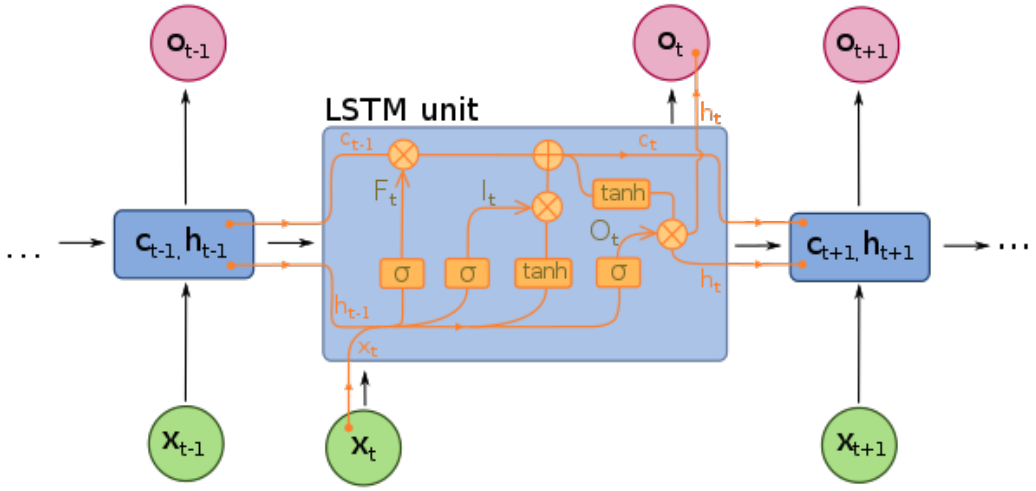


FIGURE 19 – Architecture d'un LSTM
[19]

```python
def define_model(max_vocab_size = 10_000, embedding_dim = 16):
    #Using an LSTM model
    model = tf.keras.Sequential([
        tf.keras.layers.Embedding(max_vocab_size, embedding_dim, mask_zero=True),
        tf.keras.layers.LSTM(64, return_sequences=True),
        tf.keras.layers.LSTM(32),
        tf.keras.layers.Dense(64, activation='relu'),
        tf.keras.layers.Dense(1, activation='sigmoid')
    ])
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

FIGURE 20 – Modèle séquentiel

La première couche de modèle est une couche d'Embedding. L'objectif de cette couche est d'effectuer le plongement en attribuant un vecteur aux mots. On se restreint aux mots les plus

utilisées, ce nombre est définir préalablement. Le vecteur associé est appris par le modèle lors de l'entrainement.
Le modèle est ensuite constitué de deux couches de LSTM afin d'extraire l'information syntaxique puis de deux couches denses.

FIGURE 21 – Perte et précision du modèle

Le modèle fournit de bons résultats avec une précision sur le segment de test de 98%. On remarque par la suite un overfitting avec une perte sur le segment de test qui augmente.

Une architecture que nous avons envisagé a été l'architecture CNN (Convolutional Neural Network). Les CNN sont capables d'extraire des caractéristiques importantes à partir d'images en utilisant des filtres de convolution, et d'utiliser ces caractéristiques pour effectuer des tâches telles que la classification d'images ou la détection d'objets. Les CNN sont particulièrement efficaces pour les tâches d'analyse d'images, car ils sont capables de traiter efficacement les données volumineuses telles que les images en conservant leur structure spatiale.

Ces carcatéristiques peuvent être utilisé en 1D. Les CNN peuvent ainsi être utilisés pour la compréhension de texte en appliquant des filtres de convolution sur les séquences de mots, considérées comme des vecteurs de dimensions 1. Les caractéristiques importantes extraites par les filtres de convolution incluent les n-grammes, les séquences de mots qui se répètent ou qui ont des relations sémantiques importantes.

Les caractéristiques extraites par les filtres de convolution peuvent être ensuite utilisées pour effectuer des tâches telles que la classification de texte, la génération de texte ou la recherche d'informations. Par exemple, un CNN utilisé pour la classification de texte peut extraire des caractéristiques importantes des séquences de mots d'un texte, puis utiliser ces caractéristiques pour prédire la catégorie à laquelle le texte appartient.

Les CNN en 1D sont donc une méthode efficace pour l'analyse de texte, car ils peuvent capturer les relations locales entre les mots d'une phrase tout en conservant la structure linéaire du texte.

La mise en place d'un tel réseau a été fait avec une première couche d'Embedding en utilisant des Embeddings déjà préentrainés comme le Word2Vec Google News 300 qui convertit les mots

en vecteur de taille 300 qui possèdent un emplacement vectoriel en fonction de leurs proximités dans les texte extrait de google news.

Suite à cet embedding, le vocabulaire des articles est enregistré dans une embedding matrice pour y avoir accès rapidement lors de l'apprentissage. Cette couche d'embedding découle sur une alternance de couche de convolution 1D et de MaxPooling 1D selon le schéma suivant.

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, None)]            0

embedding (Embedding)        (None, None, 300)         3678900

conv1d (Conv1D)              (None, None, 64)          96064

max_pooling1d (MaxPooling1D  (None, None, 64)          0
)

conv1d_1 (Conv1D)            (None, None, 64)          20544

max_pooling1d_1 (MaxPooling  (None, None, 64)          0
1D)

conv1d_2 (Conv1D)            (None, None, 64)          20544

global_max_pooling1d (Globa  (None, 64)                0
lMaxPooling1D)

dense (Dense)                (None, 64)                4160

dropout (Dropout)            (None, 64)                0

dense_1 (Dense)              (None, 1)                 65

=================================================================
Total params: 3,820,277
Trainable params: 141,377
Non-trainable params: 3,678,900
_________________________________________________________________
```

FIGURE 22 – Structure du réseau CNN

Cette structure donnera un overfitting immédiat sur le dataset d'entrainement avec un taux de réussite de 0.99 et une accuracy proche de la répartition des documents sur le dataset de validation. L'approche par les CNN est donc intéressante par la puissance de l'outil et sa capacité à reconnaitre des paternes récurrents. Malheureusement, cela nous apprends que la classification des fake news ne peut se faire uniquement sur cette relation locale entre les mots mais doit aussi porter sur un aspect plus global du texte.

3.3 UTILISATION DE BERT

BERT, ou Bidirectional Encoder Representations from Transformers, est un modèle de traitement automatique du langage naturel (NLP) développé par Google AI Language. Ce modèle repose sur l'architecture des Transformers, introduite par Vaswani et al. en 2017[20], qui a révolutionné le domaine de l'intelligence artificielle et du traitement du langage naturel.

BERT est pré-entraîné sur de vastes corpus de texte en deux étapes principales : pré-entraînement et ajustement fin. La première étape consiste en l'apprentissage non supervisé de

représentations linguistiques, tandis que la seconde étape consiste en l'entraînement supervisé sur une tâche spécifique.

L'innovation clé de BERT est l'utilisation d'une approche bidirectionnelle pour comprendre le contexte des mots dans une phrase. Contrairement aux modèles précédents qui lisaient le texte de gauche à droite ou de droite à gauche, BERT est capable de lire dans les deux directions. Cela permet au modèle de saisir un contexte plus riche et d'améliorer sa compréhension de la langue.

Le modèle BERT a été pré-entraîné sur des tâches de prédiction de mots masqués (Masked Language Model, MLM) et de prédiction de la phrase suivante (Next Sentence Prediction, NSP). Dans la tâche MLM, certains mots sont masqués et le modèle doit les prédire en utilisant le contexte bidirectionnel. Dans la tâche NSP, le modèle apprend à prédire si une phrase donnée est la phrase suivante d'une autre phrase.

Les architecture des Transformers ou celle de BERT étant très puissantes, il ne faudrait pas se contenter d'un entraînement standard (train, test) sur un même jeu de données, mais on entraîne le modèle sur le premier dataset, constitué d'articles, et on le teste sur les deux datasets en même temps (le deuxième dataset contient différents types de textes).

Dans notre modèle, nous utilisons comme entrée Embedding standard de BERT pour une compréhension de la sémantique du texte bien meilleure que celle des modèles précédents, cet embedding ne sera pas entraîné pendant l'apprentissage du modèle. L'embendding d'une phrase dans ce cas est un vecteur de dimension fixe, qui ne peut excéder 512. Le coeur du modèle est l'Encoder du modèle de base, qui sera ajusté finement (fine-tuned) sur l'un des deux jeux de données politiques, pour la détection de la désinformation.

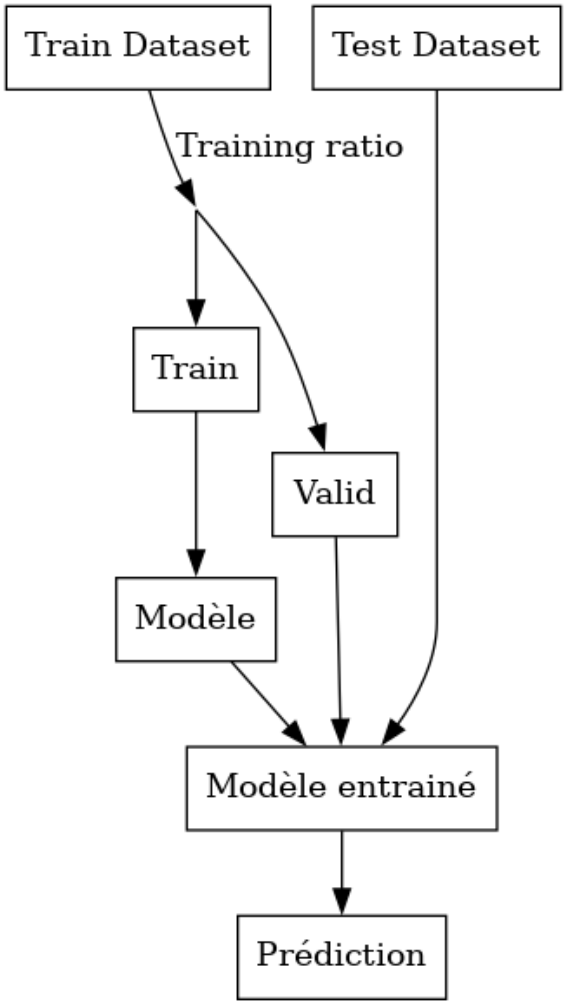
Selon le papier original, Bert est très sensible aux ajustements du fine-tuning, qui risque facilement de se surentraîner (over-fitting), et de ne plus classifier correctement des données de nature différentes. La difficulté de l'expérience est ainsi de trouver tous les hyperparamètres qui permettent de mieux généraliser l'apprentissage à partir d'un dataset vers un autre.

Le modèle reçoit en entrée le premier dataset, et ne s'entraîne que sur une partie de celui-ci, déterminée par le paramètre training ratio, l'autre fraction de ce dataset sera utilisée pour la validation. Le modèle s'entraîne pendant la durée epochs. L'entrée est divisée en batch de taille batch size, et chaque phrase en input sera transformée par le Tokenizer de BERT (embedding) en un vecteur de dimension max length.

Les paramètres du modèle seront entraînés par l'optimiser ADAM, doté d'un learning rate et d'un weight decay. La fonction de perte utilisée est la BinaryCrossEntropy, de formule :

$$-(y \cdot log(p) + (1 - y) \cdot log(1 - p)) \tag{10}$$

Enfin, le modèle sera testé sur le premier dataset, utilisé pour l'entraînement et la validation, mais aussi sur le deuxième dataset, vu par le modèle pour la première fois.



[h]

FIGURE 23 – Structure d'entraînement

Le modèle témoin est le modèle BERT de base pré entraîné, mais sans aucun réglage fin de ses paramètres. Le test de ce modèle renvoie une précision de l'ordre de 0.5 sur les deux jeux de données, prouvant que le modèle de base n'est pas à même de réaliser cette tâche sans entraînement. Une première contrainte en matière de tokenisation est que le modèle BERT ne peut accepter une entrée de vecteur de dimension supérieure à 512. Cependant, chaque mot encodé occupe généralement une dimension supplémentaire (ou plus), ce qui pose un problème. Plusieurs techniques sont mises en place pour y remédier, par exemple, il est possible d'utiliser une fenêtre glissante d'une certaine size et d'un certain stride. Expérimentalement, on trouve qu'il n'y a pas de différence entre les différentes combinaisons, et que globalement, la performance chute lors de l'utilisation de cette technique. En effet, le modèle est exposé à une grande quantité d'informations similaires et se surentraîne, réduisant sa capacité d'inférence.

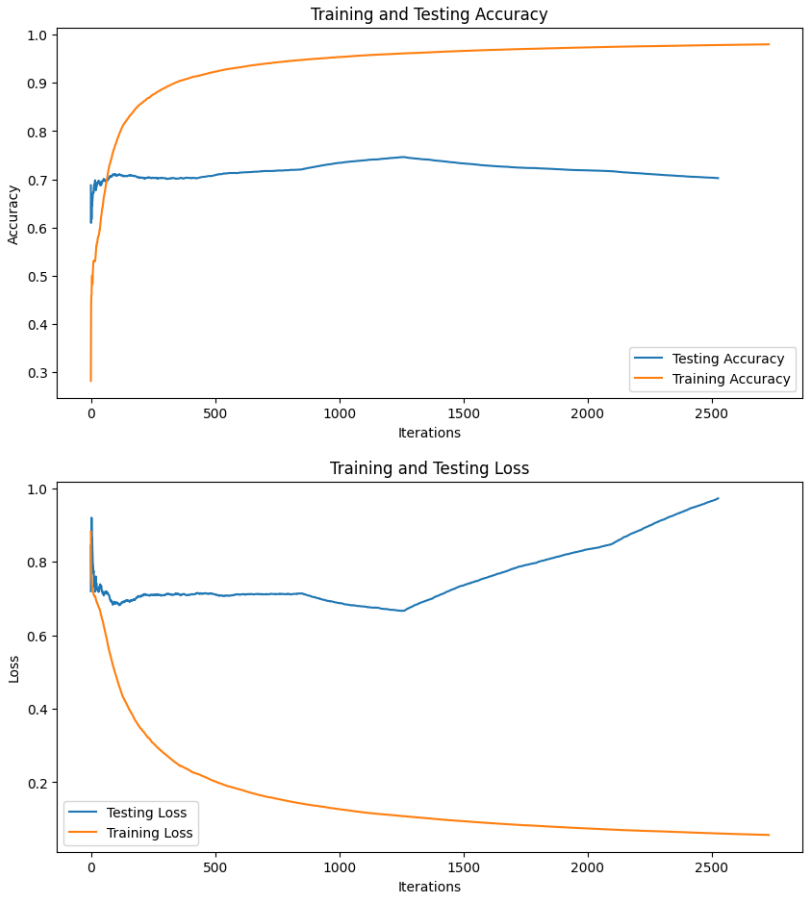


FIGURE 24 – Overfit du modèle à 4 epochs

Le papier de BERT suggère que la longueur maximale optimale pour le fine-tuning est de 128, pour un batch size de 64 ou 128. Nous adopterons une approche similaire pour notre embedding, où, de chaque texte en entrée, seules 3 fenêtres aléatoires de size = max length = 128 seront prises en entrée, la prédiction du modèle sera basée sur un jugement majoritaire de la prédiction de chacune des trois fenêtres.

De plus, par soucis de mémoire vidéo, nous avons limité le batch size à 32, et expérimentalement, l'accumulation de gradient n'a pas prouvé de grande différence contre son calcul direct pour chaque batch.
D'après l'article de BERT, les valeurs optimales pour le learning ratio sont de l'ordre de 10e-4, 5.10e-5, 2.10e-5 et 10e-5. Cependant, même en utilisant ces meilleurs hyperparamètres, le modèle parvient rapidement à apprendre les caractéristiques du jeu de données d'entraînement, mais ne peut pas bien performer sur un autre jeu de données.

Dans ces circonstances, la méthode d'entraînement la plus efficace consiste à limiter davantage l'exposition du modèle aux données d'entraînement en réduisant le training ratio encore plus bas. Paradoxalement, la réduction de l'entraînement conduit à une meilleure généralisation.

Les tableaux ci-dessous représentent les résultats de 48 expériences menées pour trouver la meilleure combinaison de learning ratio et de training ratio. L'entraînement a été effectué sur

une carte Nvidia P 100 16Go et a duré 35 heures au total.

Training ratio Learning rate	0.8	0.5	0.3	0.2	0.1	0.05
0.0001	0.9716	0.9813	0.9834	0.971	0.9702	0.9499
5e-05	0.987	0.9855	0.9863	0.9781	0.9704	0.9637
2e-05	0.9875	0.988	0.9816	0.9772	0.9308	0.9466
1e-05	0.9882	0.9866	0.9747	0.9763	0.9563	0.9421
5e-06	0.9803	0.9742	0.9712	0.9695	0.9517	0.9277
2e-06	0.9793	0.9698	0.9605	0.9616	0.9221	0.8654
1e-06	0.9695	0.9423	0.9344	0.9096	0.8564	0.7007
1e-07	0.775	0.7027	0.6977	0.5092	0.5514	0.5086

Précision du modèle sur le dataset d'entraînement

Training ratio Learning rate	0.8	0.5	0.3	0.2	0.1	0.05
0.0001	0.3772	0.3212	0.4024	0.4675	0.5894	0.5065
5e-05	0.4726	0.5852	0.5987	0.4717	0.7368	0.7076
2e-05	0.4995	0.559	0.6131	0.7724	0.7343	0.6466
1e-05	0.5877	0.8603	0.8528	0.846	0.823	0.6901
5e-06	0.5048	0.7894	0.7622	0.8273	0.7101	0.669
2e-06	0.8083	0.7662	0.704	0.6012	0.5593	0.4981
1e-06	0.7141	0.646	0.6721	0.6675	0.6725	0.5605
1e-07	0.7223	0.6228	0.6001	0.51	0.5245	0.513

Précision du modèle sur le dataset de test

FIGURE 25

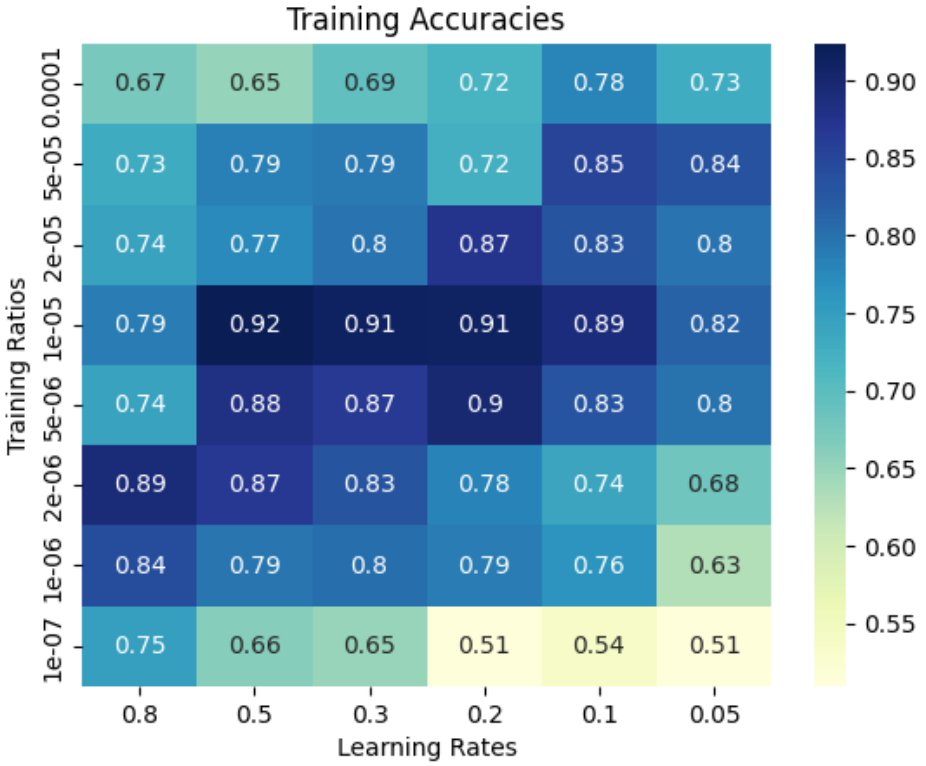


FIGURE 26 – Moyenne simple des précisions entre les deux datasets

En utilisant cette méthode, il est remarquable que la capacité de généralisation soit grandement améliorée dans certaines circonstances. En effet, il est souvent observé qu'entraîner un modèle sur une partie du jeu de données produit des résultats supérieurs à ceux obtenus en s'entraînant sur l'ensemble du jeu de données.

De surcroît, les learning rate optimaux concordent bien avec les recommandations du papier de BERT, bien qu'il est possible de descendre bien au-dessous de 10e-5 et d'avoir d'aussi bons résultats.

Le paramètre weight decay a été utilisé uniquement afin d'assurer la stabilité des modèles, et nous avons vérifié que sa variation n'a pas beaucoup influencé les résultats finaux.

Pour conclure, notre démarche expérimentale a su mettre en évidence la puissance de l'engin BERT à généraliser ses prédictions, à condition que les jeux de données partagent un fond sémantique similaire dans leur ensemble, bien que la manière de l'exprimer soit différente (satire, article, mensonge, etc)

CONCLUSION

Dans le cadre de notre projet de recherche sur la détection des fake news, nous avons mené une étude approfondie pour identifier et analyser des ensembles de données pertinents, en veillant à choisir des dataset de type varié. Ces données ont été utilisées pour entraîner et tester plusieurs modèles de détection. Nous avons adopté une approche de machine learning basée sur le majority voting combinant 9 modèles, ainsi qu'un modèle de réseau de neurones à longue mémoire à court terme (LSTM) et un modèle BERT. Les résultats obtenus démontrent une performance remarquable de notre approche, en particulier avec le modèle BERT, qui a atteint un taux de précision de 92% sur l'ensemble de test. Cette recherche confirme ainsi la faisabilité et l'efficacité de l'utilisation des techniques d'apprentissage profond et de traitement automatique du langage naturel pour identifier et contrer les articles qui s'avèrent être fake news.

Dans le futur, ce projet pourrait être étendu et amélioré de plusieurs façons pour renforcer davantage la détection des fake news. L'une des perspectives d'avenir pourrait consister à explorer l'intégration d'autres modèles de traitement du langage naturel plus avancés, tels que GPT-4 ou des architectures Transformer, pour améliorer la performance de détection. De plus, une analyse sémantique plus poussée et l'incorporation de la vérification des faits automatisée pourraient contribuer à rendre notre solution plus robuste face à diverses formes de désinformation. Il serait également intéressant de développer une plateforme en temps réel qui permettrait aux utilisateurs d'analyser la véracité des informations en ligne au fur et à mesure de leur navigation. En outre, une collaboration avec les réseaux sociaux et les moteurs de recherche pourrait aider à prévenir la propagation de fausses informations en identifiant et en signalant les contenus suspects de manière proactive. Enfin, il est crucial de continuer à mettre à jour et à enrichir les ensembles de données utilisés pour l'entraînement et l'évaluation des modèles, afin de garantir une détection efficace face à l'évolution rapide des techniques de désinformation et des sujets abordés par les fake news. Une approche interdisciplinaire incluant des domaines tels que la psychologie, la sociologie et la science politique pourrait également apporter des perspectives précieuses pour mieux comprendre et lutter contre la propagation des fake news dans notre société.

RÉFÉRENCES

[1] https ://www.science.org/doi/full/10.1126/science.aao2998

[2] https ://dictionnaire.lerobert.com/definition/desinformation

[3] https ://dictionary.cambridge.org/dictionary/english/misinformation

[4] https ://dictionary.cambridge.org/dictionary/english/disinformation

[5] https ://www.ncbi.nlm.nih.gov/pmc/articles/PMC7117034/

[6] LIAR : A Benchmark Dataset for Fake News Detection

[7] https ://open-platform.theguardian.com

[8] https ://www.kaggle.com/datasets/sameedhayat/guardian-news-dataset

[9] https ://www.kaggle.com/competitions/fake-news/data

[10] https ://www.kaggle.com/datasets/clmentbisaillon/fake-and-real-news-dataset

[11] https ://www.researchgate.net/publication/320300831_Detection_of_Online_Fake_News_Using_N-Gram_Analysis_and_Machine_Learning_Techniques

[12] https ://arxiv.org/abs/2203.09936 (visité la dernière fois le 27/04/2023)

[13] scikit-learn Machine Learning in Python, bibliothèque en ligne, https ://scikit-learn.org/stable/

[14] Quinlan, J. R. Induction of decision trees. Machine learning, 1986, 81-106.

[15] sklearn.tree.DecisionTreeClassifier : https ://scikit-learn.org/stable/modules/generated/sklearn.tree.De

[16] Stochastic Gradient Descent : https ://scikit-learn.org/stable/modules/sgd.html

[17] https ://www.kdnuggets.com/2018/08/unveiling-mathematics-behind-xgboost.html

[18] https ://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

[19] Par fdeloche — Travail personnel, CC BY-SA 4.0, https ://commons.wikimedia.org/w/index.php ?cu- rid=60149410

[20] https ://arxiv.org/abs/1810.04805

[21] https ://www.sciencedirect.com/science/article/pii/S1877050920323954,

[22] https ://www.researchgate.net/figure/Accuracy-comparison-between-TF-IDF-Vectorizer-and-Count-Vectorizer_fig2_343313467

[23] https ://www.semanticscholar.org/paper/Fake-News-Detection-on-Reddit-Utilising-and-Term-Patel-Meehan/72f1fe87f96567b3b788204a0992ed96db4cf309).

[24] https ://web.stanford.edu/class/linguist289/luhn57.pdf