FAKE NEWS - DEEP FAKES

LE VRAI DU FAUX

OPTIMISATION ET COMPARAISON DE MODÈLES D'APPRENTISSAGE AUTOMATIQUE POUR LA DÉTECTION DE FAKE NEWS POLITIQUES

PROJET PRÉSENTÉ PAR

Shabboo ALEAGHA et Ophélie ENGASSER

mai 2024 - M2-Artificial Intelligence & Management-Data Science

TABLE DES MATIÈRES

I. Résumé page 3

II. Introduction page 4

III. Problématique page 5
1. La détection des fake news dans la littérature scientifique page 5
2. La détection des fake news dans l'industrie et l'ingénierie page 7
3. Démarche opérationnelle page 8

IV. Données page 9
1. Les données d'apprentissage - dataset TI-CNN page 9
2. Les données inédites - webscraping page 10
3. Data cleaning page 10
4. Exploratory Data Analysis (EDA) page 11
5. Natural Language Processing (NLP) page 14
6. Méthodologie d'optimisation page 16

V. Résultats page 17
1. Benchmark des classifieurs - Machine Learning page 17
2. Benchmark des réseaux de neurones - Deep Learning page 19

VI. Discussion page 22
1. Interprétation des résultats page 22
2. Notre proposition de valeur : création de l'application FastGuard page 24

VII. Mise en place du projet page 25
1. Finalité et durée du projet page 25
2. Outils de gestion de projet page 25
3. Découpage page 25
4. Ordonnancement page 26
5. Planification page 27

VIII. Conclusion page 28

Références page 29

Annexes page 31
A. Explication des classifieurs de ML et du choix des hyperparamètres page 31
B. Logique de construction des réseaux et réglage des hyperparamètres page 35

RÉSUMÉ

A l'heure du partage massif et libre des informations sur internet et les réseaux sociaux, la diffusion de fausses informations ou fake news est de plus en plus problématique, en particulier dans le domaine politique. Ce phénomène constitue un fléau sur de nombreux plans, car il influence les opinions, incite à la peur ou à la haine, et finalement cherche à orienter les prises de décisions. Dans ce travail, nous proposons de recourir à des approches s'appuyant sur l'Intelligence Artificielle - Natural Language Processing et Machine Learning - pour développer un modèle capable de réaliser de manière efficace une tâche de détection de fake news, par la classification d'articles en vrais ou faux. Nous avons privilégié une méthodologie de benchmarking pour comparer différents modèles selon leurs performances à la tâche : 7 classifieurs de Machine Learning dits "traditionnels" optimisés à l'aide d'une recherche par grille et d'une validation croisée (Naive Bayes, Random Forest, SVM, Logistic Regression, Passive-Aggressive, AdaBoost et Gradient Boosting), et 7 modèles de Deep Learning construits selon une logique d'architecture augmentant en complexité, et optimisés par ajustement des hyperparamètres (Simple embedding network, simple RNN, LSTM, bidirectional LSTM, bidirectional LSTM + CNN, Word2Vec - bidirectional LSTM + CNN, et BERT). Les résultats mettent en évidence une supériorité des performances du modèle combinant Word2Vec avec un LSTM bidirectionnel et un CNN (> 86% d'accuracy sur des données inédites), démontrant l'efficacité d'une stratégie alliant une architecture qui capture la séquentialité du langage et une architecture saisissant les informations locales, alimentées par des embeddings qualifiés dans la représentation sémantique. Nous discutons les résultats et les poursuites possibles de ce travail dans le futur, et proposons un outil applicatif issu de ce thème, à savoir une application nommée FactGuard, basée sur l'intervention humaine et artificielle pour lutter contre la désinformation.

Mots-clés : intelligence artificielle, machine learning, fake news, classification, politique.

In the age of widespread and free sharing of information on the internet and social media, the dissemination of false information or fake news has become increasingly problematic, especially in the political domain. This phenomenon constitutes a scourge on many fronts, as it influences opinions, incites fear or hatred, and ultimately seeks to steer decision-making. In this study, we propose to use approaches based on Artificial Intelligence - Natural Language Processing and Machine Learning - to develop a model capable of effectively performing the task of fake news detection by classifying articles as true or false. We favored a benchmarking methodology to compare different models based on their performance in the task: 7 so-called "traditional" Machine Learning classifiers optimized using grid search and cross-validation (Naive Bayes, Random Forest, SVM, Logistic Regression, Passive-Aggressive, AdaBoost, and Gradient Boosting), and 7 Deep Learning models built on a logic of increasing complexity, optimized by tuning hyperparameters (Simple embedding network, simple RNN, LSTM, bidirectional LSTM, bidirectional LSTM + CNN, Word2Vec - bidirectional LSTM + CNN, and BERT). The results highlight the superior performance of the model combining Word2Vec with a bidirectional LSTM and a CNN (> 86% accuracy on unseen data), demonstrating the effectiveness of a strategy combining an architecture that captures the sequentiality of language and an architecture capturing local information, fueled by embeddings qualified in semantic representation. We discuss the results and possible future pursuits of this work, and propose an applicative tool stemming from this theme, namely an application called FactGuard, based on human and artificial intervention to combat misinformation.

Keywords *: artificial intelligence, machine learning, fake news, classification, politics.*

INTRODUCTION

Notre société accorde une place prépondérante aux médias et à l'information, reflétant l'appétence de l'être humain à s'informer et à comprendre le monde qui l'entoure. Depuis l'avènement des réseaux sociaux et la généralisation de leur utilisation, l'information est accessible gratuitement, facilement - quels que soient l'endroit et l'heure - et surtout elle est partageable massivement. Au moment où les registres d'immédiateté et d'instantanéité dominent le lien social et régulent nos manques, les médias rivalisent de technologies pour diffuser l'information en temps réel afin de satisfaire ce besoin. L'immédiateté possède un autre sens : elle caractérise également la possibilité accrue de pouvoir partager des informations sans véritable instance de vérification, et cette absence de source fiable permet l'intrusion de fausses informations.

Une **fake news** est définie comme un élément médiatique factuellement inexact mais présenté par son émetteur comme ayant une valeur de véracité avec une volonté explicite de tromper le récepteur (Pennycook & Rand, 2021, cité par Nelson, 2022). Les objectifs primordiaux sont - entre autres - d'attirer l'attention des lecteurs, d'inculquer des sentiments de peur, ou encore de diviser. Le **deep fake** serait le pendant photographique, vidéographique ou sonore de la fake news. Leur impact est fort car l'information constitue une porte d'accès sur le monde et un critère dans les prises de décisions (Mridha et al., 2021).

Les fausses informations et les rumeurs ont toujours existé, mais selon Nelson (2022), le phénomène est particulièrement prégnant depuis l'arrivée des réseaux sociaux, et de surcroît dans un contexte politique marqué par deux événements importants : l'élection du Président Trump en 2017, et le Brexit en 2020. On le voit, la politique est le lieu privilégié pour relayer ce genre d'information car la volonté est grande d'influencer, voire de manipuler les opinions. La période du Covid a également été propice à la diffusion des fake news, leurs diffuseurs misant sur le manque de compréhension et l'inquiétude que les individus tentaient d'apaiser en cherchant des réponses sur internet. D'un point de vue plus optimiste, si ces périodes marquées ont été propices à la recrudescence des fake news, elles ont aussi permis une sensibilisation du public à leur existence et leur impact. C'est alors que des méthodes d'explication, de prévention et de repérage ont vu le jour.

Certaines approches cherchent à expliquer d'un point de vue psychologique les facteurs influençant la crédibilité des fake news (Nelson, 2022 ; Pennycook & Rand, 2021), d'autres approches technologiques notamment basées sur l'Intelligence Artificielle (IA), cherchent à enrayer le phénomène en proposant de la détection de fake news grâce à l'apprentissage automatique. C'est cette approche que nous souhaitons développer dans notre travail. Nous avons choisi pour cela de nous pencher sur la **thématique politique**, que nous jugeons à fort potentiel d'impact actuellement (e.g. les guerres, le contexte électoral en Europe, aux Etats-Unis et en France à plus long terme). **L'objectif étant de mettre en œuvre un modèle d'IA détectant les fake news et pouvant être déployé dans une application**.

PROBLÉMATIQUE

En quoi l'IA peut-elle répondre à l'objectif que nous nous fixons, à savoir détecter les fake news dans l'univers médiatique en ligne ? L'IA est une branche de l'informatique qui s'intéresse à la manière dont une machine peut se substituer à l'intelligence humaine, pour résoudre des problèmes complexes. L'ambition est de profiter de la puissance calculatoire de la machine, de sa rapidité et de son absence de fatigabilité, pour imiter des tâches cognitives telles que la perception, le traitement de l'information, l'apprentissage, la détection ou la compréhension du langage (dans le cas qui nous occupe). Cependant, l'ordinateur ne traite pas l'information comme le fait le cerveau humain, aussi, l'enjeu est de lui fournir des données adéquates.

Dans le contexte de la détection des fake news, deux approches principales de l'IA sont utilisées : (i) le traitement du langage naturel ou **Natural Language Processing (NLP)** qui consiste à traiter les données textuelles dans le but de les traduire en un langage que la machine pourra prendre en charge, et (ii) l'apprentissage automatique ou **Machine Learning** (englobant les méthodes d'apprentissage profond ou Deep Learning) qui consiste à programmer un ordinateur pour qu'il réalise des tâches déterminées (détection, classification, régression, clustering, etc.) Cette technique est cruciale dans la détection des fake news en raison de sa capacité à analyser de grandes quantités de données et à identifier des schémas et des tendances cachées à l'intérieur des textes (Berrondo-Otermin & Sarasa-Cabezuelo, 2023). L'apprentissage peut être supervisé ou non supervisé selon que l'on donne au modèle une cible ou des étiquettes sur lesquelles se baser pour son apprentissage. **La tâche sur laquelle nous avons entraîné nos modèles dans ce présent travail est une tâche de classification binaire (Real = 1 ou Fake = 0), et ce à partir de plusieurs classifieurs et architectures de réseaux neuronaux, dont nous avons réalisé un benchmark**. De ce comparatif, et après optimisation, nous avons dégagé les modèles les plus robustes, c'est-à-dire les plus à même de généraliser sur des données jamais rencontrées, ce qui est l'objectif ultime pour une mise en production du modèle finalement retenu.

Quel est l'état de l'art concernant la détection des fake news par l'IA ? Effectuons une revue de la littérature scientifique et des cas d'usage industriels actuels.

1. La détection des fake news dans la littérature scientifique

Les articles s'intéressant à cette thématique se focalisent sur diverses approches selon la **modalité des données** utilisées (images, textes, association des deux), la **typologie des données** (données issues d'articles de presse ou issues des réseaux sociaux, associées ou non à des métadonnées), la **thématique des données** (politique, technologie, société, faits divers, santé, etc.), la **structuration des données** (relationnelle, graphe, json, etc.), le **traitement NLP**, la **méthode d'extraction des caractéristiques**, ou encore le **type d'apprentissage machine** (traditionnel, profond, par transfert, multimodal, etc.) Les études sont souvent comparatives, et tentent par le biais de benchmarking, de dégager le modèle le plus performant à la tâche, et ce à partir des métriques de test. L'optimisation est systématiquement réalisée.

De nombreuses études effectuent également des revues de la littérature dans le but de recenser l'état de l'existant dans le domaine. Parmi les plus récentes, on trouve celle de Hu, Mao & Zhang (2024), de Berrondo-Otermin & Sarasa-Cabezuelo (2023), ou encore de Yuan et al. (2023). Ahmed et al. (2021) s'intéressent en particulier aux recherches sur la détection de fake news avec des méthodes classiques de Machine Learning, et Al-Tai, Nema & Al-Sherbaz (2023), Hu et al. (2022), ou encore Mridha et al. (2021), déploient les techniques les plus récentes et avancées dans le domaine du Deep Learning et des réseaux de neurones profonds. Hu et al. (2022) exposent des taxonomies très détaillées de modèles supervisés, semi- et non supervisés. Enfin, un autre type d'étude se concentre sur des benchmarks de datasets (Sharma & Garg, 2023 ; Wang, 2017), ou introduit un dataset qui sera ensuite largement utilisé par d'autres (c'est le cas du dataset FakeNewsNet de Shu et al., 2019, ou du dataset Liar de Wang, 2017).

Les études basées sur des modèles de Machine Learning traditionnels. Dans ces études, un certain nombre de modèles courants de Machine Learning sont testés de manière comparative. Sharma & Garg (2023) expliquent que le Naive Bayes performe bien sur de grands ensembles de données. Cependant, ils démontrent, en utilisant le dataset IFND pour une tâche de détection de fake news, que le Naive Bayes présente la moins bonne précision (87.5%), comparativement au KNN, Decision Tree, Logistic Regression et Random Forest, ce dernier présentant la meilleure accuracy (94%). En revanche, Khan et al. (2019) montrent, sur le même cas d'usage, que la meilleure performance provient d'un Naive Bayes, combiné à une extraction de caractéristiques TF-IDF bigramme, ce qui rejoint la remarque des auteurs précédents concernant le potentiel de ce modèle sur de grands datasets. Agarwal et al. (2019) quant à eux, découvrent pour les fake news, de bons résultats avec un SVM et une régression logistique, avec un avantage pour le SVM en termes d'accuracy. Ils nuancent cependant avec un manque de généralisation des modèles pour des textes s'écartant du domaine politique et économique. Choudhury & Acharjee (2023) remarquent également une supériorité pour le SVM parmi SVM, Naive Bayes, Régression Logistique et Random Forest. Enfin, Nagashri & Sangeetha (2021) mettent en évidence le bien-fondé du modèle Passif-Agressif, d'autant plus pour le traitement de données en temps réel. Nous pouvons donc déduire de ces quelques exemples qu'il n'existe pas de consensus, y compris sur des datasets assez comparables, concernant les performances des modèles traditionnels de Machine Learning, hautement dépendants des données d'entraînement.

Les études basées sur des modèles de Deep Learning. De nombreuses architectures de réseaux de neurones peuvent s'avérer efficaces pour la détection des fake news. Par exemple, Ma et al. (2016) modélisent la relation séquentielle entre des publications grâce à des réseaux de neurones récurrents (RNN). Yu et al. (2017) utilisent des réseaux convolutionnels (CNN) pour représenter des relations sémantiques de haut niveau. Une architecture CNN permet également à un groupe d'auteurs de traiter des fake news à partir d'une entrée multimodale texte-image (Yang et al., 2023). Bian et al. (2020) exploitent les réseaux neuronaux basés sur des graphes (GCN) selon un process bottom-up et top-down pour le même type de tâche. Khattar et al. (2019) proposent l'auto-encodeur variationnel multi-modal (MVAE) pour extraire les représentations multi-modales cachées dans des nouvelles. De nombreux auteurs tirent profit de la

bidirectionnalité du modèle pré-entraîné BERT pour engager de bonnes performances dans ce domaine. Ainsi, Kaliyar, Goswami & Narang (2021) obtiennent une accuracy de presque 99% à l'aide d'un BERT combiné à des couches convolutives. Kula, Choras & Kozik (2021), quant à eux, obtiennent un bon modèle de détection avec un BERT combiné à une architecture RNN. Ces deux dernières études suggèrent, comme de nombreuses autres, que le modèle BERT serait particulièrement réceptif à ce genre de cas d'usage.

Nous constatons à travers ces quelques références que de nombreuses données de la littérature (e.g. les travaux de Al-Tai, Nema & Al-Sherbaz (2023), ou de Mridha et al. (2021)), mettent en évidence de **moindres performances sur les modèles de Machine Learning traditionnels, comparativement à des modèles implémentés à partir de réseaux neuronaux**. Il se trouve que les modèles traditionnels, malgré leur avantage sur certaines tâches à haut niveau calculatoire, performent parfois moins bien sur des tâches plus proches de la cognition humaine (comme reconnaître une voix, classer des photos, etc.) Par ailleurs, l'apprentissage profond extrait ses propres caractéristiques, capturant davantage d'informations que les algorithmes classiques. Cependant, certaines études permettent de nuancer ce propos, en montrant par exemple pour un Naive Bayes prenant une matrice TF-IDF bigramme, des performances équivalentes à celles d'un réseau de neurones (Khan et al., 2019).

2. La détection des fake news dans l'industrie et l'ingénierie

Parallèlement aux travaux académiques, l'industrie avance au niveau de la technologie apportée à la détection des fake news. Après avoir lancé un projet de lutte contre la désinformation associé à des subventions, la Commission Européenne en a établi en 2019 un premier bilan, exhortant les GAFAM et autres acteurs du web à poursuivre leurs efforts dans ce sens (cf. article de l'Usine Nouvelle cité en références). En effet, de nombreux systèmes de vérification des faits en ligne les plus récents, tels que FactCheck.org et PolitiFact.com, sont encore basés sur des approches de détection manuelle par des humains, impliquant un temps de réaction problématique (Zhang & Ghorbani, 2020). Par ailleurs, la plupart des ressources existantes de vérification des faits en ligne se concentrent principalement sur des informations politiques, cernant donc seulement une partie de l'éventail des contenus. Selon les mêmes auteurs, Facebook permet aux utilisateurs de faire des signalements d'informations suspectes, et Google a lancé Google News Initiative, une plateforme collaborative destinée à lutter contre la désinformation et à aider les utilisateurs à se repérer. Mais ces initiatives, à l'instar de beaucoup d'autres, restent encore dépendantes de l'intervention humaine.

En France, une chercheuse en informatique, Ioana Manolescu, a été l'une des pionnières dans l'usage de l'IA et du Big Data pour la détection des faits, plus précisément à partir de technologies de NLP, et de stockage dans des bases de données orientées graphes. En collaboration avec des laboratoires de recherche, elle a lancé en 2016 le projet ContentCheck, qui s'adresse particulièrement aux journalistes.

Des start-up se sont lancées dans le développement de technologies basées sur l'IA, le NLP et le Big Data pour contribuer à lutter contre la désinformation (voir notamment l'article de StartUs insights cité en

références). Buster.AI est une start-up française proposant une solution basée sur l'apprentissage profond avec la possibilité d'interroger des bases de données distribuées pour attribuer un score de fiabilité à l'information. Dans le même ordre d'idée, NewsCheck, start-up américaine, exploite l'alliance homme-machine pour offrir une révision transparente et éthique. Ces start-up segmentent le milieu du journalisme, de l'entreprise, et même de la finance, potentiels victimes de la désinformation. Les marques risquent aussi d'être touchées par la calomnie, aussi, c'est à elles que s'adresse plus spécifiquement la start-up israélienne Cyabra qui propose une protection contre les abus en ligne. D'autres entreprises utilisent en plus la technologie de la blockchain, c'est le cas de la start-up singapourienne Primas, ou encore de la danoise Defudger qui cible les deepfakes.

Zhang & Ghorbani (2020) cernent les problématiques qu'occasionnent les fake news en ligne, et en ce sens permettent d'en dégager des besoins spécifiques pour proposer des solutions innovantes. Selon ces auteurs, la plupart des fausses informations circulent sur les réseaux sociaux, ce qui signifie qu'un volume massif de données sont produites sous des formats disparates, et ce, en temps réel. Nous retrouvons ici les 3 V bien connus du Big Data, à savoir le Volume, la Variété et la Vélocité des données fake. Un système de détection innovant et efficace pour l'identification des fake news doit donc prendre ces éléments en considération et développer des outils puisant dans des technologies telles que le **stockage massif et non structuré des données, l'ingestion et le traitement de ces données en temps réel, etc**.

3. Démarche opérationnelle

Pour déployer un modèle performant dans la détection des fake news, nous avons entraîné plusieurs modèles d'apprentissage automatique, à une **tâche de classification binaire (Real = 1 ou Fake = 0)** à partir des données d'entraînement, puis avons réalisé une étude comparative des meilleurs modèles dans un benchmark. Le plan opérationnel est le suivant (chaque partie possède un notebook dédié) :

- **ntbk#1. Modèles de Machine Learning dits "traditionnels"** : Naive Bayes, Random Forest, SVM, Logistic Regression, Passive-Aggressive, Boosted Methods (AdaBoost, Gradient Boosting). Librairie Python principale utilisée : **Scikit Learn**.
- **ntbk#2. Modèles de Deep Learning** : simple embedding network, RNN, uni & bidirectionnal LSTM, LSTM+CNN, Word2Vec / bidirectionnal LSTM + CNN. Librairie : **Tensorflow / API Keras**.
- **ntbk#3. Apprentissage par transfert avec BERT**. Librairies : **Tensorflow, PyTorch**.

Chaque partie a fait l'objet d'un pré-traitement des données, une modélisation, une évaluation, une optimisation et un test sur des données inédites.

DONNÉES

1. Les données d'apprentissage - dataset TI-CNN

La labellisation des données est une étape longue et coûteuse dans un projet d'IA. Dans le domaine de la détection de fake news, c'est en plus une démarche subjective si les informations sont récupérées directement sur internet. Aussi, nous avons privilégié la démarche de recourir à un dataset étiqueté fourni en open source, à savoir le dataset **Ti-CNN** proposé par Yang et al. (2023). Ce dataset a été souvent cité et utilisé dans d'autres études, notamment par Hu et al. (2022) dans leur article très exhaustif et documenté.

Il s'agit d'un dataset multimodal d'environ 100 MB, conçu spécifiquement pour la détection de fake news. Le dataset brut contient **54 colonnes**, **20015 articles sur la politique**, dont **11941 faux** (récupérés sur Kaggle) et **8074 vrais** (récupérés sur les sites du New York Times et du Washington Post). Les auteurs du dataset ont réalisé un travail de création de features qui nous a semblé très riche afin de réaliser un travail d'Exploratory Data Analysis (EDA) et de comprendre les features sous-jacentes à chaque classe (Real/Fake). Nous résumons dans le tableau ci-dessous les features les plus importantes pour le projet. La méthode **info()** de la librairie **Pandas** nous a permis d'obtenir les informations suivantes :

Nom colonne	Contenu	Count	Type de données Python	Nb de NaN
author	Nom de l'auteur	20015	object	0
comments	Nombre de commentaires à propos de l'article	16844	float64	3171
country	Pays source de l'article	20015	object	0
crawled	Date de récupération	16844	object	3171
language	Langue d'écriture	16844	object	3171
likes	Nombre de likes	16844	float64	3171
main_img_url	URL de l'image	20015	object	0
shares	Nombre de partages	16844	float64	3171
site_url	URL du site de l'article	16844	object	3171
text	Corps de l'article	20015	object	0
title	Titre de l'article	20015	object	0
type	**Classes Real/Fake = target**	20015	object	0
caps_title (caps_text)	Nombre de lettre capitales (titre et texte)	20015	int64	0

Nom colonne	Contenu	Count	Type de données Python	Nb de NaN
title_len (text_len)	Longueur du titre ou du texte en nombre de caractères	20015	int64	0
excl_title (excl_text)	Nombre de ponctuations exclamatives (titre et texte)	20015	int64	0
first_title (first_text)	Nombre de formulations à la 1re personne (titre et texte)	20015	int64	0
second_title (second_text)	Nombre de formulations à la 2e personne (titre et texte)	20015	int64	0
third_title (third_text)	Nombre de formulations à la 3e personne (titre et texte)	20015	int64	0
emotion columns (x10)	10 colonnes : Nombre d'occurrences d'émotions (colère, anticipation, dégoût, peur, joie, tristesse, surprise, confiance, négativité, positivité	20015	int64	0

2. Les données inédites - webscraping

Nous avons récolté des données servant de données de test "en production", à savoir des données que les modèles n'ont jamais rencontrées. Pour cela, nous avons scrappé les articles de deux journaux américains, à l'aide de la librairie **BeautifulSoup**. **1710 articles étiquetés Real** ont été scrappés sur le site du **Time Magazine,** et **1981 articles étiquetés Fake** ont été scrappés sur le site du journal satirique américain **The Onion**. Chaque classe a été récupérée dans la section "Politique" de chaque site, pour être en cohérence avec les données d'entraînement et la thématique de ce projet. Via la librairie **Pandas**, les articles ont été stockés dans des dataframes avec leurs index respectifs. Les deux dataframes ont été labellisés et concaténés pour constituer le dataframe de test final. Le code dédié au scraping figure dans un notebook spécifique contenu dans le répertoire du projet (ntbk#x).

3. Data cleaning

Concernant les données d'entraînement, assez bruitées dans leur version brute, plusieurs manipulations ont été nécessaires afin d'obtenir le dataset d'entraînement final. Une fonction a été appliquée sur la colonne contenant les articles, pour supprimer les balises HTML, les fins d'articles inutiles, les références à l'auteur, les mots vides, et les phrases contenant des adresses email. Ces manipulations ont été réalisées via la librairie **re** destinée au traitement des chaînes de caractères par des expressions régulières (regex). Puis, nous avons conservé uniquement les articles rédigés en Anglais. Grâce à la librairie **langdetect**, nous avons pu garder les articles anglais dont le langage était de type NaN dans la colonne dédiée. Ce traitement a permis de réduire légèrement le déséquilibre des classes du dataset, finalement passé à 58% d'articles fake pour 42% d'articles real.

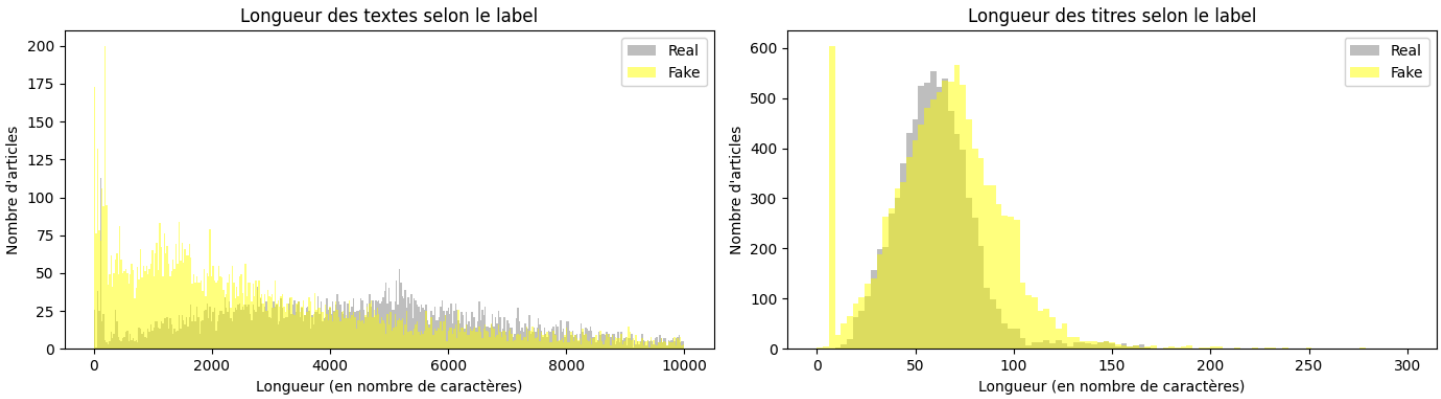
Nous avons pris le parti de nettoyer ce dataset de manière à pouvoir rendre les articles real et fake (issus de sources différentes) entièrement comparables entre eux **sémantiquement**, "toute chose étant égale par ailleurs", c'est-à-dire sans laisser la possibilité aux futurs modèles de pouvoir se baser sur une structure linéaire, syntaxique, formelle ou s'appuyer sur des noms d'auteurs ou emails présents dans une classe et pas dans l'autre. Par ailleurs, ce traitement nous a paru pertinent également dans une perspective de réduction de dimensionnalité. La même décision a été prise lors du traitement linguistique par le NLP.

Les données inédites quant à elles, étaient peu bruitées, car récupérées directement dans la balise HTML encapsulant le texte de chaque article de chaque page du site lors du webscraping. Par conséquent, il ne nous a pas été nécessaire de réaliser le même nettoyage.

4. Exploratory Data Analysis (EDA)

Comme nous l'avons mentionné, le dataset d'entraînement comporte de nombreuses colonnes permettant un travail d'EDA, offrant la possibilité de comprendre certaines caractéristiques explicites dans la structure des textes des deux classes. Nous avons exploré la longueur des textes et des titres, la fréquence des mots dans les titres, les éléments non verbaux des phrases (lettres capitales, ponctuation), ainsi que les sentiments exprimés.

Nous présentons ci-dessous un graphique représentant la **longueur des textes et des titres**.



Nous pouvons observer que les textes real ont une distribution plus aplatie, étant représentés par des longueurs plus variables, alors que les fake sont plus nombreux à être plus courts. Les titres des deux classes sont répartis davantage sous forme gaussienne, avec toutefois un léger décalage de la courbe des fake vers la droite, suggérant plus de longueurs plus importantes.

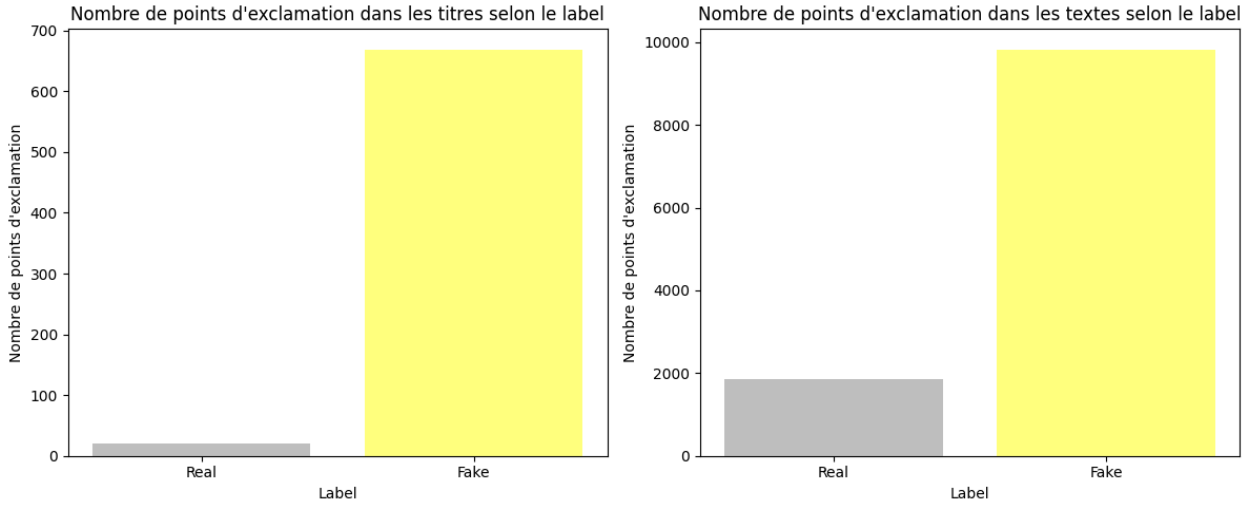
Visualisons à présent **les mots les plus représentés dans les titres** des deux corpus, sur un nuage de mots obtenu grâce à la librairie **wordcloud**, où la grandeur des mots est proportionnelle à leur fréquence dans le corpus.

Si l'on compare les deux nuages de points, on observe que les fake n'ont souvent aucun titre, ce que l'on voit par la grandeur des termes "notitle" (les auteurs du dataset ont remplacé les NaN par "notitle"). Dans le corpus fake, certains mots en majuscule comme VIDEO, pourraient avoir pour but d'attirer l'attention du lecteur. On observe également que le corpus real semble contenir davantage de verbes, notamment des verbes d'actions (say, vote, watch, rise…) Un certain nombre d'acronymes non signifiants (e.g. WSJ) suppose par ailleurs la nécessité d'un traitement NLP pour réduire encore la dimensionnalité.

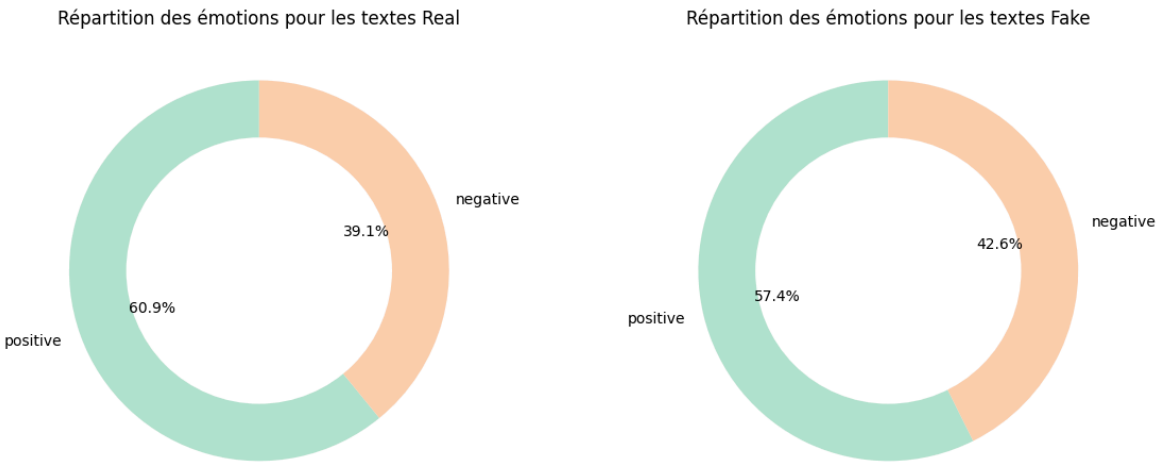
Le nombre d'éléments non verbaux sont représentés ci-dessous dans des graphiques à barres.

Nous remarquons que le corpus fake présente davantage de lettres capitales que le corpus real, que ce soit au niveau des titres ou des textes. Comme mentionné plus haut, cela pourrait se traduire par une volonté de la part des auteurs de fake news, d'attirer implicitement l'attention chez le lecteur (Yang et al., 2023). Cela pourrait également refléter un manque de standardisation dans les fake news qui ne sont pas soumis à un calibrage lié à des recommandations éditoriales.

La ponctuation exclamative (points d'exclamation) est également caractéristique des deux corpus.



A l'instar des lettres capitales, on note une différence significative entre le nombre de points d'exclamation dans les fake comparativement aux real. Cela va dans le sens de l'observation précédente, soulignant le caractère emphatique des articles fake, et ajoutant un objectif rhétorique destiné à accentuer l'effet du fond de l'écriture. Voyons maintenant si la ponctuation exclamative, qui véhicule une forme de sentiment non verbal, s'accompagne d'une tendance équivalente au niveau de **l'expression des sentiments**. Pour la répartition des sentiments positifs et négatifs, il existe légèrement plus de termes négatifs que de termes positifs parmi les fake, et inversement pour les termes positifs, confirmant la tendance évoquée.



En revanche, il n'existe pas de pattern véritablement différent entre les deux corpus concernant le nombre d'occurrences des autres sentiments (colère, joie, peur, etc.)

Cette partie EDA nous a montré de manière intéressante que les articles real et fake possèdent des structures syntaxiques et linguistiques différentes, les premiers étant rédigés en conformité avec les standards éditoriaux, tandis que les seconds, plus libres, utilisent des formes rhétoriques ayant pour but d'accentuer les sentiments, surtout négatifs. Inconsciemment, ce style d'écriture oriente le point de vue du

lecteur. Nous allons présenter la partie de pré-traitement NLP, ayant pour but de préparer les textes pour être pris en charge par les modèles. Si l'esprit humain se base, consciemment ou inconsciemment, sur les éléments repérés en EDA, il est important que les modèles puissent se baser sur leurs propres caractéristiques implicites, sans se référer à des patterns emblématiques des deux types de textes.

5. Natural Language Processing (NLP)

Le NLP donne la possibilité à un ordinateur de "comprendre", analyser, manipuler et potentiellement générer du langage humain (Mridha et al., 2021). Pour y aboutir, un pré-traitement et un encodage des données textuelles sous forme numérique (vectorielle) sont nécessaires.

Pré-traitement. Nous avons procédé aux pré-traitements suivants : tokenisation (division du texte en tokens, ici des mots), normalisation (mise en minuscules), suppression de la ponctuation, suppression des mots vides ou de liaison (stopwords) et des mots inférieurs à 2 caractères, stemming (réduction des mots à leur radical (stème), à l'aide de l'algorithme de Snowball, et enfin, ajout d'un filtre pour supprimer les abréviations communes (mr, mrs, dr, etc), les initiales suivies d'un point, et les chiffres. L'ensemble de ces traitements a été regroupé dans une fonction nommée **preprocess_text()** et a été réalisé via la librairie **nltk**. Dans un but de réduction de la taille du vocabulaire et pour éviter la sparsité de la matrice d'encodage, nous avons utilisé uniquement le stemming, non combiné avec la lemmatization, cette dernière ne nous ayant pas offert plus d'avantages compensant la complexité. Ce pré-traitement assez complet est justifié par deux choix de notre part : (i) celui de neutraliser le bruit dans nos données et ainsi de réduire leur dimensionnalité, afin d'optimiser les analyses en temps et en espace, et (ii) celui de construire des modèles robustes se basant sur des caractéristiques intrinsèques, cachées et non linéaires dans les données, plutôt que sur des caractéristiques formelles et structurelles. L'appui trop important d'un modèle sur ce type de caractéristique présente le risque, selon nous, d'un surajustement aux données d'entraînement et donc d'une généralisation problématique aux données non vues.

Préparation des données. Les données d'entraînement ainsi traitées ont été stockées dans une matrice X, et les labels dans un vecteur y. Ces derniers étant encodés sous forme de chaînes de caractères (real - fake) dans le dataset, nous avons procédé à un encodage en classes numériques (0 - 1) grâce à un objet **LabelEncoder() de sklearn**, pour correspondre au type de données attendues par sklearn. Nous avons ensuite décomposé ces variables en tuples via la fonction **train_test_split() de sklearn**, avec un ratio de division de 80:20, selon le principe de Pareto (Mridha et al., 2021). Ce sont les entrées qui ont été fournies aux vectorizers lors de l'étape d'extraction des caractéristiques. Notons que l'ensemble de validation a été défini a posteriori au moment de la définition de la GridSearchCV pour les classifieurs Machine Learning, et au moment de l'entraînement pour les réseaux de neurones, par une sous-division sur l'ensemble de test, ce dernier ayant été réservé comme ensemble "hold-out", utilisé pour estimer de manière impartiale les performances des modèles sur les données non vues et ainsi mieux conclure quant à la détection d'un potentiel surajustement.

Vectorisation des mots et extraction de caractéristiques. Cette étape consistant à mapper un mot avec un vecteur et à en dégager des caractéristiques intrinsèques aux textes a été réalisée grâce à la technique TF-IDF pour les classifieurs de Machine Learning, et grâce à une couche d'embedding ou un modèle spécialisé (ici Word2Vec) intégrés à nos architectures de réseaux de neurones, ou utilisé seul en apprentissage par transfert (BERT).

A l'instar de Sharma & Garg (2023), Khan et al. (2019), Agarwal et al. (2019), ou encore Pal, Kumar et Pal (2019) qui ont démontré l'efficacité de cette technique, nous avons utilisé le **vectorizer TF-IDF** pour réaliser l'extraction de caractéristiques préalablement à l'entraînement des classifieurs de Machine Learning. TF-IDF est une mesure statistique qui évalue l'importance d'un terme dans un document, par rapport à un corpus entier. Cette technique est intéressante car elle prend en compte non seulement la fréquence d'un mot, mais sa fréquence relative par rapport à son importance, ou sa rareté, dans le vocabulaire d'une langue donnée (par ex., le mot "donc" en français est très répandu, pourtant il ne va pas apporter beaucoup d'information sémantique dans une phrase, par contre si l'on rencontre souvent le mot "terroriste", cela peut apporter de l'information, car il s'agit d'un mot moins fréquent "relativement" à d'autres).

Mathématiquement, cet algorithme est calculé de la manière suivante :

$$TF - IDF(i,j) = \frac{tf(i,j)}{k+tf(i,j)} \times log\left(\frac{N}{df(i)}\right)$$

où $tf(i,j)$ représente le nombre d'occurrences du terme i dans le document j ; k est un paramètre pour lisser la fréquence de terme, souvent défini à 1 ; N est le nombre total de documents, et $df(i)$ est le nombre de documents contenant le terme i (pour une explication mathématique, voir Sharma & Garg, 2023). Le résultat est une matrice qui s'interprète de la manière suivante : une ligne représente un texte, une colonne représente un mot unique dans le corpus global, les coefficients a_{ij} représentent les poids de chaque terme dans le document (leur valeur est comprise dans l'intervalle $[0, 1]$). Cela signifie que si un mot apparaît dans le texte, il aura un poids TF-IDF non nul dans la matrice correspondante. Il aura une valeur de zéro sinon. La matrice TF-IDF a ensuite été fournie comme données d'entrée à chaque modèle testé.

Pour les réseaux de neurones, nous avons effectué une conversion des textes en séquences d'entiers. Après avoir divisé les textes en unités de mots (tokenizing), nous avons encodé les textes en entiers grâce à la méthode **text_to_sequences() de l'objet Tokenizer de Keras**, et ce en fonction du vocabulaire de notre corpus (étape de **sequencing**). Chaque vecteur a été fixé de manière à correspondre à un nombre constant d'entiers correspondant à la longueur moyenne du corpus total. Les vecteurs de séquences trop longues ont ainsi été tronqués, et ceux de séquences trop courtes ont été remplis de zéros (étape de **padding**). Ces vecteurs ont servi d'entrée à la première couche d'embedding des différents réseaux de neurones. Plus loin, nous avons également utilisé le vocabulaire d'un modèle **Word2Vec** entraîné sur notre corpus, comme entrée du réseau de neurones ayant obtenu les meilleures performances. Word2Vec est une méthode de représentation de mots à l'aide de vecteurs de réels denses. Des mots proches en termes de sens

possèdent une représentation vectorielle proche. Le modèle a été développé par Google et se base sur un réseau de neurones peu profond avec une couche intermédiaire dont les poids forment la représentation vectorielle du mot. Il offre un point de départ utile pour de nombreuses tâches liées au NLP, et nous verrons dans la partie résultats que c'est cette configuration (Word2Vec + réseau de neurones) qui nous a permis d'obtenir le modèle le plus robuste, devançant même le **modèle pré-entraîné BERT**, basé sur l'architecture des transformers (stack d'encoders), qui a pour principe de capturer des informations entre les mots d'une séquence de manière simultanée.

6. Méthodologie d'optimisation

Pour les **classifieurs de Machine Learning**, chaque modèle testé a fait l'objet d'une optimisation par une recherche du meilleur estimateur, à l'aide d'une GridSearchCV fournie par la librairie **sklearn**. Cette approche a répondu à deux attentes : (i) gagner du temps sur le réglage des hyperparamètres, et (ii) développer une méthodologie valide et non biaisée par une **validation croisée "k-fold"** (ici nous avons choisi 5 plis), en réservant un ensemble dit "hold-out" servant d'ensemble test final, parfaitement indépendant, contenant des données que les modèles ont réellement non vues préalablement. Pour les **réseaux de neurones**, la recherche par GridSearch étant trop coûteuse, chaque modèle a été optimisé par ajustement des hyperparamètres de manière indépendante, c'est-à-dire "toutes choses égales par ailleurs" pour les autres. Cette approche nous a permis de contrôler les changements obtenus. La démarche d'optimisation a été exposée dans le notebook dédié au Deep Learning pour le premier modèle, puis, par souci d'économie, seuls les modèles optimaux ont été présentés. La logique de construction des modèles est présentée en **annexes**, les architectures testées allant de la plus simple à la plus complexe, le point commun à toutes étant la première couche d'embedding (avec ou sans entraînement) et la dernière couche dense à une unité, associée à une fonction d'activation sigmoïde pour l'obtention de la classe de sortie.

Enfin, les modèles ayant présenté la meilleure accuracy ont été exportés dans le répertoire du projet à l'aide de la librairie **pickle**. Ils ont ensuite été testés "en production" sur les données inédites. Ces évaluations ont été synthétisées dans un benchmark que nous présenterons.

Synthèse - Fiche technique du projet

✓ **Besoin ciblé** : détection de fake news politiques
✓ **Approche IA** : Natural Language Processing - Apprentissage automatique
✓ **Tâche** : classification binaire (real = 1 ou fake = 0)
✓ **Classifieurs ML** : Naive Bayes, Random Forest, SVM, Logistic Regression, Passive-Aggressive, AdaBoost, Gradient Boosting
✓ **Architectures DL** : simple embedding, RNN, LSTM (uni and bidirectional), w2v/LSTM+CNN, BERT
✓ **Input** : matrice TF-IDF (ML) - vecteurs d'entiers ou de réels (DL)
✓ **Output** : prédictions argmax
✓ **Métriques** : accuracy, precision, recall, f1-score

RÉSULTATS

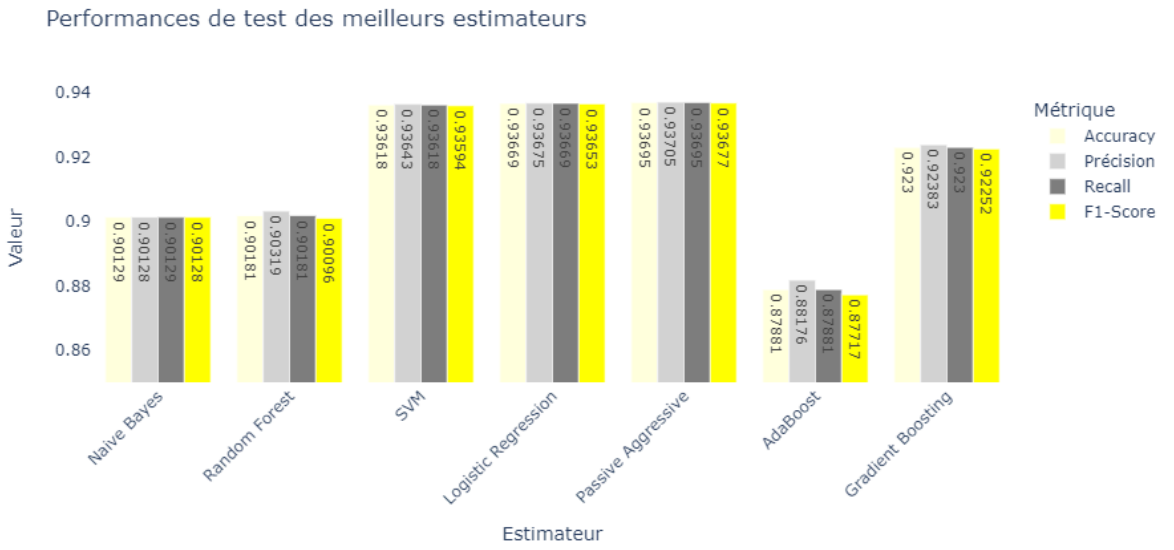
Le choix des modèles s'est fait en regard des performances mises en évidence dans les différentes études de la littérature. Pour une explication de leur pertinence et du choix des hyperparamètres, voir les **annexes**. Les résultats et leurs graphiques les plus illustratifs sont présentés ici, la totalité figure dans les notebooks.

1. Benchmark des classifieurs – Machine Learning

Réalisons un comparatif de **4 métriques de test** pour chaque meilleur estimateur (grid **best_estimator_**).

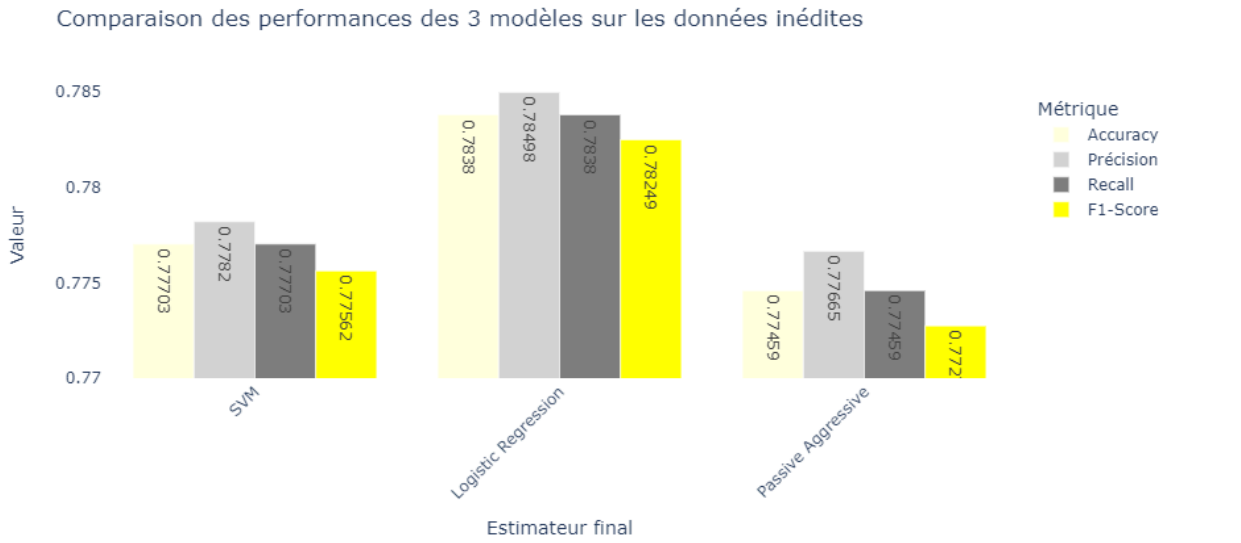
Métrique	Description, usage	Formule
Accuracy ou exactitude	✓ **Mesure le taux d'articles bien classés.** ✓ Évalue la précision globale d'un modèle en termes de prédictions correctes par rapport à toutes les prédictions effectuées. ✓ Limites : influencée par les classes fortement représentées.	$\frac{VP+VN}{VP+VN+FP+FN}$
Precision ou sur-détection	✓ **Mesure le nombre de prédictions positives correctes par rapport au total des prédictions positives**. ✓ Évalue la capacité d'un modèle à ne pas classer à tort un article comme Real. Si elle est élevée alors il y a moins d'erreurs de faux positifs.	$\frac{VP}{VP+FP}$
Recall (rappel) ou sous-détection	✓ **Mesure le nombre de prédictions positives correctes par rapport au total des positifs réels.** ✓ Évalue la capacité d'un modèle à identifier correctement les occurrences positives. S'il est élevé alors il y a moins d'erreurs de faux négatifs.	$\frac{VP}{VP+FN}$
F1-Score	✓ **Moyenne harmonique pondérée de la précision et du rappel.** ✓ Avantage : plus représentatif lorsque les classes sont déséquilibrées.	$\frac{2*Precision*Recall}{Precision+Recall}$

Pour la précision, le rappel et le F1-Score nous avons considéré la **moyenne pondérée** des deux classes.

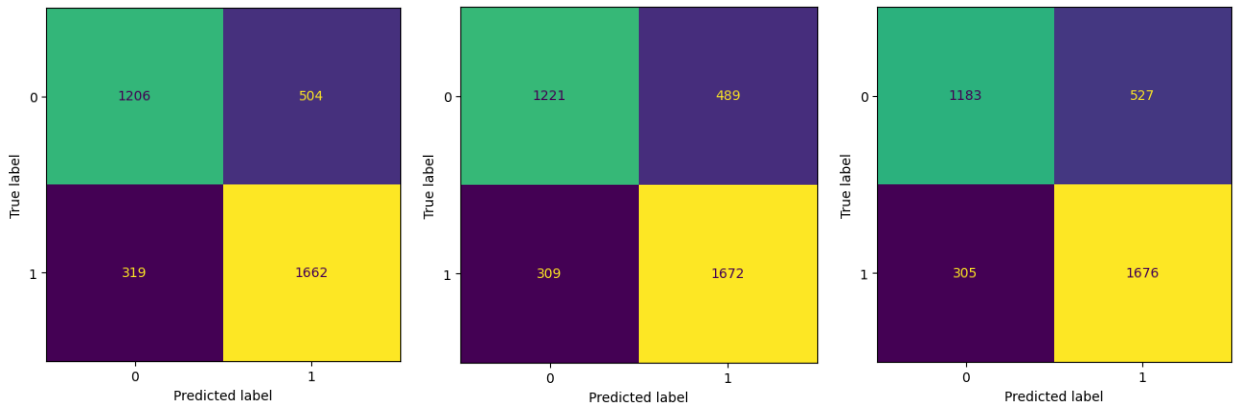


Nous voyons la supériorité de **SVM**, **Logistic Regression** et **Passive-Aggressive** en termes de performances de test. Pour ces 3 estimateurs, l'accuracy observée sur l'ensemble d'entraînement et sur

l'ensemble de test "hold-out" (cf. notebook) est quasiment identique (à .001 près), suggérant peu de pertes et de surajustement des modèles. Ce résultat souligne la pertinence de la recherche par grille, et la validité des modèles proposés, pour effectuer une comparaison de ceux-ci "en production" sur des **données inédites (scrappées)**, telle que présentée dans le graphique ci-dessous.



Nous remarquons une légère supériorité (à .01 près) du modèle de régression logistique. Comparons les **matrices de confusion** obtenues pour les données inédites (de gauche à droite : SVM, LR, PA).

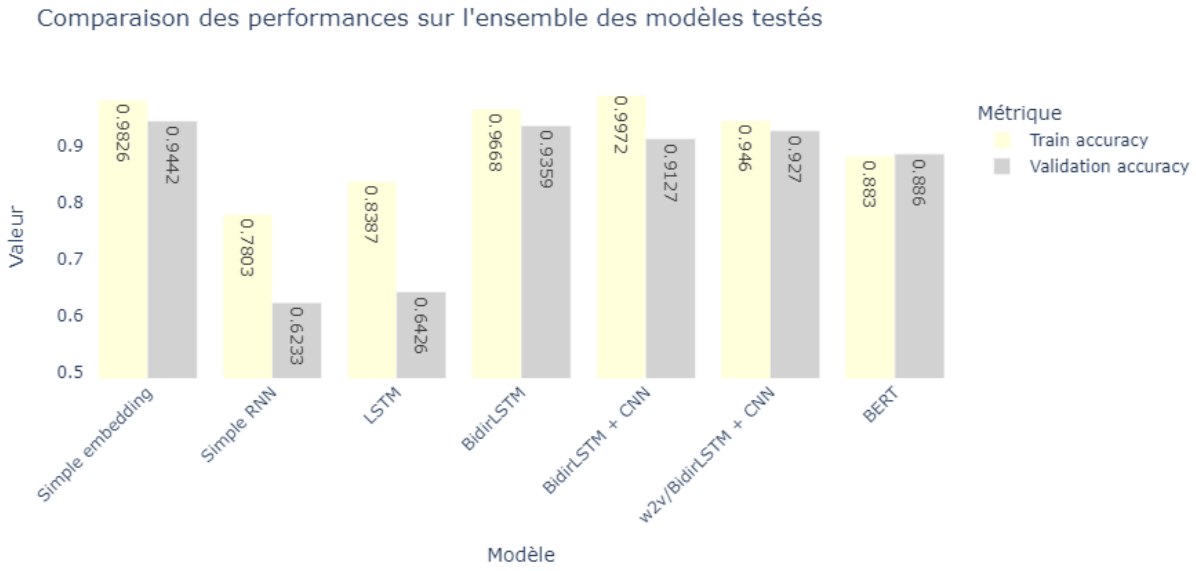


Nous remarquons que pour les trois estimateurs, on relève toujours **plus de faux positifs** (c'est-à-dire de cas où les modèles prédisent la classe Real=1 alors que l'étiquette réelle est Fake=0), que de faux négatifs (l'inverse). En d'autres mots, les modèles peinent davantage à reconnaître les articles fake et à les classer comme tels lorsqu'ils le sont. Ce résultat ne peut pas s'expliquer par le déséquilibre de nos classes qui est en faveur d'une plus grande proportion de fake (si cela était imputable au déséquilibre, le modèle produirait plus de faux négatifs). Une autre remarque est que l'accuracy pour les trois estimateurs est moindre que pour l'évaluation (avec une perte moyenne de 15 points d'accuracy). Cela suggère un surajustement des

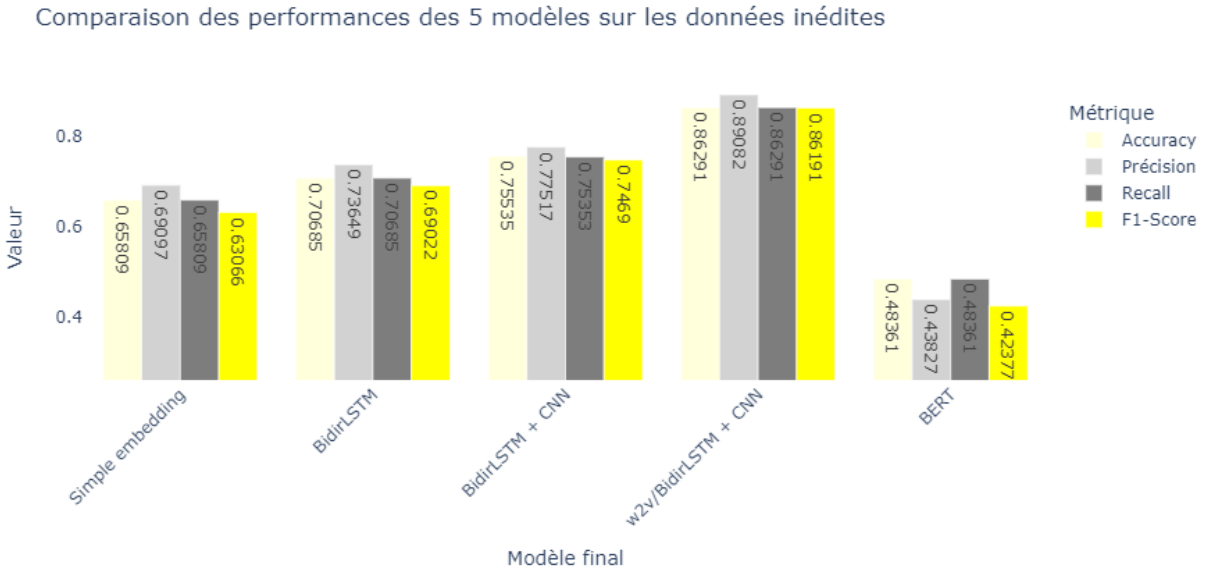
modèles aux données d'entraînement, qui peut s'expliquer par des modèles devenus trop complexes pour généraliser aux nouvelles données scrappées.

2. Benchmark des réseaux de neurones – Deep Learning

Visualisons le comparatif entre l'**accuracy train versus validation** pour l'ensemble des modèles testés et optimisés de cette partie.



Les modèles qui performent le mieux (seuil d'accuracy > .90), et présentent un bon compromis entre le train et la validation, ont été retenus pour être testés sur les données inédites. Le benchmark sur ces modèles nous donne les résultats suivants.



Nous voyons une **meilleure capacité du modèle combinant Word2Vec+BidirLSTM+CNN à généraliser sur les données inédites, avec un bon compromis entre la précision et le rappel**. Il est intéressant de constater à partir des **métriques par classe sur les données inédites** (cf. graphique ci-dessous) les tendances qui se dessinent entre les modèles concernant les capacités à prédire certaines classes.

Concernant la **précision** pour les 3 premiers modèles (**simple embedding**, **BidirLSTM** et **BidirLSTM+CNN**), on constate de meilleures performances pour la prédiction de fake, parmi tous les articles que les modèles ont prédit. Pour le **rappel**, parmi tous les articles réellement étiquetés de chaque classe, les modèles prédisent mieux les real, la différence entre les deux classes étant plus accentuée pour le simple embedding et le BidirLSTM que pour le BidirLSTM+CNN. Les **F1-Scores** des classes 0 sont plus faibles pour les 3 premiers modèles, l'écart entre les classes diminuant avec BidirLSTM et BidirLSTM+CNN, suggérant une amélioration des performances à mesure de la complexité de l'architecture des réseaux.

Nous observons un pattern inverse pour le modèle **w2v/BidirLSTM+CNN.** La précision est meilleure pour la classe 1 que pour la classe 0, et inversement pour le rappel. Ces scores engendrent un F1-Score très

acceptable pour les deux classes, révélant la qualité globale de ce modèle et sa capacité de prédiction de chacune des classes.

Le modèle pré-entraîné BERT rejoint le pattern des trois premiers pour le rappel et le F1-Score, mais présente une précision plus forte pour la classe 1, suggérant qu'il prédit et rappelle mieux les real, et n'est pas en capacité d'être compétent sur les fake.

Pour résumer, l'ensemble des résultats est en faveur d'une **supériorité du modèle combinant une architecture LSTM bidirectionnelle avec une couche de convolution CNN, utilisant en entrée des embeddings de mots Word2Vec**. Il affiche de très bonnes performances sur toutes les métriques, avec une bonne précision et un bon rappel, ce qui indique qu'il identifie correctement les nouveaux articles, quelles que soient leurs classes. Ce comportement offre un bon compromis entre faux positifs et faux négatifs, qui pourrait rendre ce modèle apte à être utilisé en production. Ces éléments nous apprennent que l'augmentation en complexité dans l'architecture des modèles a été en mesure de répondre à la tâche de classification de fake news, permettant de capturer plus de complexité et de caractéristiques intrinsèques dans les données textuelles même inédites. Cela est d'autant plus visible lorsque les modèles tirent profit de la combinaison de deux architectures efficaces et complémentaires telles que RNN - pour la prise en compte de l'aspect séquentiel du langage et une prise de recul sur la globalité, avec une extension LSTM conservant en mémoire les informations importantes même sur des séquences longues, et CNN - pour une extraction des caractéristiques locales des textes, et que les embeddings d'entrée sont issus d'un modèle pré-implémenté comme Word2Vec. Il apparaît également dans les résultats que certains classifieurs de Machine Learning dits "traditionnels" performent mieux que les réseaux de neurones les plus simples que nous avons testés, ce qui confirme certaines observations déjà évoquées (Khan et al., 2019).

DISCUSSION

1. Interprétation des résultats

Dans la partie dédiée aux **classifieurs de Machine Learning traditionnels**, les meilleurs estimateurs retenus sont ceux qui ont présenté les meilleures métriques après optimisation par GridSearchCV, à savoir les modèles SVM, Logistic Regression et Passive-Aggressive. Le test sur les nouvelles données "en production" a révélé que le modèle de régression logistique était le plus robuste, car le plus en capacité de généraliser sur les nouvelles données, à .001 près toutefois par rapport aux deux autres. Les 3 modèles présentent une perte de 15 points en accuracy par rapport à l'évaluation sur le dataset de test. Les 3 modèles présentent également une tendance à produire davantage de faux positifs que de faux négatifs. Pour interpréter ces résultats, nous avons rejeté plus haut une imputation au déséquilibre dans nos données, mais nous allons principalement discuter trois éléments.

(i) **Remarques à propos du data cleaning**. Notre travail a présenté deux phases de cleaning, avec une seconde étape de cleaning plus approfondi : suppression des adresses mails, de certains acronymes privés de sens, de certaines fins de phrases non informatives, toute la ponctuation et les chiffres (pour une démonstration du bien-fondé d'un cleaning approfondi, voir Umer et al., 2020). Ce nouveau traitement nous a fait constater une légère baisse dans la performance des modèles (de l'ordre de .01). Cependant, cela nous a semblé important afin que le modèle n'apprenne pas à reconnaître les articles sur la base de features non pertinentes. La réduction du bruit peut être considérée comme une forme de régularisation, qui vise à limiter la capacité du modèle à s'ajuster excessivement aux données d'entraînement. Malgré ces traitements et l'optimisation apportée par la recherche par grille et la cross-validation, la perte de 15 points sur les prédictions à partir des données inédites est questionnante. C'est pourquoi nous avons poursuivi nos investigations vers des architectures plus complexes.

(ii) **Remarques à propos de l'utilisation de la méthode TF-IDF**. Cette méthode très utilisée en NLP présente de nombreux avantages, notamment celui de prendre en compte l'intégralité du corpus de données, comparativement à d'autres méthodes, ainsi que sa simplicité et son interprétabilité. Toutefois, TF-IDF peut entraîner une forte sparsité des vecteurs surtout si une part de bruit n'a pas été neutralisée lors du pré-process (ce qui est difficile à vérifier sur des datasets conséquents). Nous pouvons également ajouter que TF-IDF traitant chaque mot de manière indépendante, ne capture pas la sémantique des mots et le contexte des phrases, ce qui l'empêche de prendre du recul dans un contexte plus large. En d'autres termes, TF-IDF traiterait le corpus de manière lexicale plutôt que sémantique. Ces inconvénients peuvent conduire la méthode à perdre certaines informations capitales (la relation entre les mots) mais à en traiter d'autres moins signifiantes, ce qui pourrait expliquer la moindre capacité des modèles basés sur cette méthode, à généraliser à de nouvelles données jamais rencontrées.

(iii) **Remarques à propos de l'utilisation des modèles de Machine Learning classiques**. La combinaison entre une matrice TF-IDF et un classifieur traditionnel peut être limitée en termes d'extraction de features,

comparativement à d'autres méthodes d'apprentissage profond qui auront la capacité d'extraire leurs propres caractéristiques à partir de relations complexes et non linéaires dans les données. Cela nous a naturellement conduites vers des architectures de réseaux neuronaux, en appui également sur les données de la littérature allant dans ce sens.

Dans la partie dédiée aux **modèles de Deep Learning**, nous avons testé différentes architectures, en partant de la plus simple pour aller vers la plus complexe, et en optimisant les modèles par ajustement des hyperparamètres. Parmi les modèles retenus (c'est-à-dire affichant la meilleure accuracy lors de l'entraînement), nous avons mis en évidence que plus l'architecture des réseaux était complexe, plus ces derniers étaient en mesure d'augmenter leurs métriques lors des tests sur des données jamais rencontrées. D'après les graphiques montrant les performances lors du test et sur données inédites, on voit que les résultats à ces deux étapes ne sont pas forcément corrélés. Les meilleures promesses de généralisation proviennent, comme on l'a vu, des architectures plus complexes. Nos résultats valident l'efficacité déjà démontrée des RNN et CNN, ceux-ci étant parmi les plus utilisés dans la détection de fake news (Mridha et al., 2021). Les résultats vont également dans le sens de Fernandez-Reyes & Shinde (2018) pour qui ces deux types de réseaux sont les plus intéressants dans le domaine des fake news politiques. Les auteurs évoquent même une popularité grandissante des CNN pour des tâches de NLP, offrant leurs atouts dans la recherche des caractéristiques locales. Dans notre travail, nous les avons combinés, afin d'obtenir la meilleure expérience des deux, en utilisant aussi l'extension LSTM des RNN, en plus d'une alimentation du réseau avec les embeddings de mots obtenus par le Word2Vec. Nous constatons indéniablement que ce réseau a permis plus que les autres de capturer le langage dans son aspect contextuel et sémantique, et ce de manière plus fine que ne l'a fait BERT, réputé généralement pour être à la pointe dans ce type de cas d'usage. Certains auteurs font le même constat : une supériorité de l'intégration Word2Vec - LSTM pour la détection de fake news, comparativement à des modèles d'apprentissage traditionnels ou un BERT (Mallik & Kumar, 2023). Adrian, Prasetyowati & Sibaroni (2023) montrent également de meilleures performances d'une intégration Word2Vec - LSTM pour la même tâche, par rapport à une intégration GloVe - LSTM, avec une précision moyenne de 95% et 90% respectivement. Selon les auteurs, Word2Vec pourrait mieux capturer les relations sémantiques entre les mots, tandis que GloVe serait plus à même de relever les relations de distribution et la cooccurrence des mots. Notre travail nous amène à nous dire que dans le cas d'usage de la détection des fake news, un accent mis sur la sémantique dans la construction d'un modèle, serait une stratégie payante. C'est cette stratégie que nous avons accentuée dans ce projet, misant comme nous l'avons dit, sur un nettoyage de tout bruit permettant éventuellement aux modèles de se baser sur autre chose que la sémantique. L'utilisation de données inédites issues d'un journal satirique écrit dans le même style journalistique qu'un article vrai, renforce la confiance que nous pouvons avoir dans les capacités de généralisation du modèle "gagnant".

Proposons à présent quelques **axes d'amélioration pour des travaux futurs**. Nous avons choisi ici à dessein de nous concentrer sur les fake news politiques, du fait du contexte politique actuel tendu. Par conséquent, nous avons entraîné nos modèles à partir d'un dataset entièrement constitué d'articles du

domaine politique, et les avons testés “en production” sur des articles scrappés dans des rubriques du même thème. Les modèles les plus performants étant construits sur la base d’architectures captant la sémantique (Word2Vec), nous pouvons postuler qu’il sera nécessaire d’élargir nos données d’apprentissage à d’autres thématiques si nous voulons un modèle qui transfère ses connaissances à toutes les fake news. A l’avenir, il serait intéressant également d’entraîner notre modèle sur davantage de features, éventuellement des images selon un protocole d’architecture multimodale, à l’instar de Yang et al. (2023), ou en intégrant les méta-données disponibles. De plus, la poursuite de notre optimisation de BERT est primordiale, celui-ci s’étant montré plutôt décevant dans ce projet. Les Large Language Models (LLM) tels que ChatGPT 3.5 ou Llama-2, s’avèrent aussi très prometteurs dans l’avancée de la technologie de détection de fake. Kareem & Abbas (2023) introduisent une stratégie pionnière qui se sert des LLM pour détecter les fake news grâce à la génération d’un raisonnement logique, offrant, en plus de la détection, des justifications pour chaque classification accessibles à la compréhension humaine. Cela permet d’accéder à la “boîte noire” de l’IA, alors que le process interne de modèles tels que BERT reste difficile à appréhender.

2. Notre proposition de valeur : création de l’application FactGuard

Le cours de Design Thinking proposé pendant cette année de M2 a fait germer dans notre groupe l’idée de la création d’une application qui allie la technologie de l’IA avec une équipe d’ambassadeurs, citoyens volontaires, pour détecter des fake news. Nous l’avons nommée **FactGuard**. L’application permettrait à des utilisateurs, internautes surfant sur les sites internet d’informations et les réseaux sociaux, de sélectionner un article ou un texte suspect, et de le proposer à l’application - en collant l’URL ou le texte lui-même - pour le faire vérifier. Sur l’interface, un réseau d’ambassadeurs formés prendrait en charge l’article, celui-ci étant soumis au premier ambassadeur disponible et ce dernier, après un premier filtrage répondant à un protocole strict, proposerait l'article douteux au modèle de classification le plus performant. Suite au processus, l’utilisateur aurait un retour sur sa partie réservée de l’application, lui donnant le résultat de l’analyse, l’article vérifié étant alors rangé dans une liste et coché comme “article vérifié”. Ce résultat serait ensuite inscrit sur une base de données immuable, sécurisée mais aussi transparente, grâce à la technologie de la blockchain. Cela permettrait de traquer l’article, mais aussi les métadonnées qui lui sont liées. Cette solution se veut innovante dans le sens où elle conjugue les éléments ayant été démontrés comme les plus efficients dans le domaine de la détection de fake news, à savoir **l’intervention et l’interprétation humaines** suivant un protocole défini à l’avance, **l’IA** et sa rapidité d’exécution, et enfin **la blockchain** et son potentiel de garantie de l’authenticité. L’appel à la communauté et à la collaboration d’individus sensibilités à la question - mais non nécessairement professionnels de l’IA - met l’accent davantage sur la collaboration et le “faire-ensemble”, que sur la réserve d’une technologie de pointe à une élite inaccessible. Combattre le mal des réseaux sociaux par le réseau social, rendre les utilisateurs acteurs de leurs lectures, constitue selon nous la meilleure arme pour lutter contre la désinformation.

MISE EN PLACE DU PROJET

1. Finalité et durée du projet

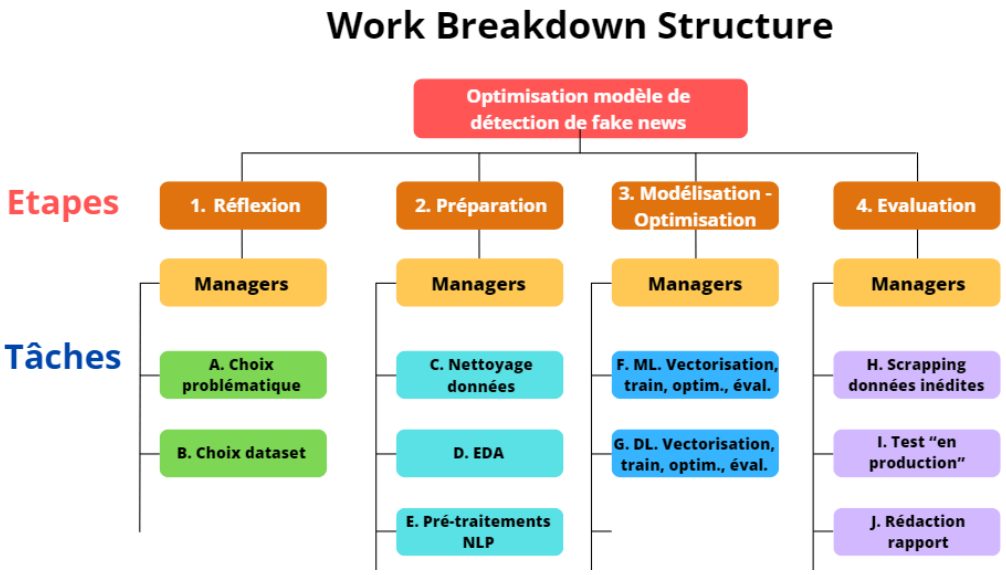
L'objectif final est de créer un modèle optimisé et satisfaisant un seuil d'acceptabilité fixé à **85%+ d'accuracy sur des données non vues par le modèle**. La durée du projet est envisagée du 15 octobre 2023 au 15 mai 2024, soit **7 mois**. Le coût du projet n'est pas présenté, celui-ci n'ayant pas engagé de charges particulières.

3. Outils de gestion de projet

✓ **Découpage :** Work Breakdown Structure – WBS (tâches) / Ressource Breakdown Structure – RBS (ressources)
✓ **Ordonnancement :** Méthode des Potentiels Métra – MPM
✓ **Planification des tâches et affectation des ressources :** diagramme de Gantt (via le logiciel Gantt Project)
✓ **Dashboard :** Trello
✓ **Revues de projet :** Google Meet / What's App
✓ **Versioning, stockage et partage du code :** Git, GitHub

2. Découpage

Nous avons réalisé au préalable un **découpage WBS** structuré en termes de **tâches**, la liste hiérarchique et chronologique des étapes et des tâches étant la suivante :



Concernant le **découpage WBS** structuré en termes de **ressources**, 2 ingénieures "Managers" sont allouées au projet ainsi que leur matériel respectif, décrit dans le tableau ci-dessous.

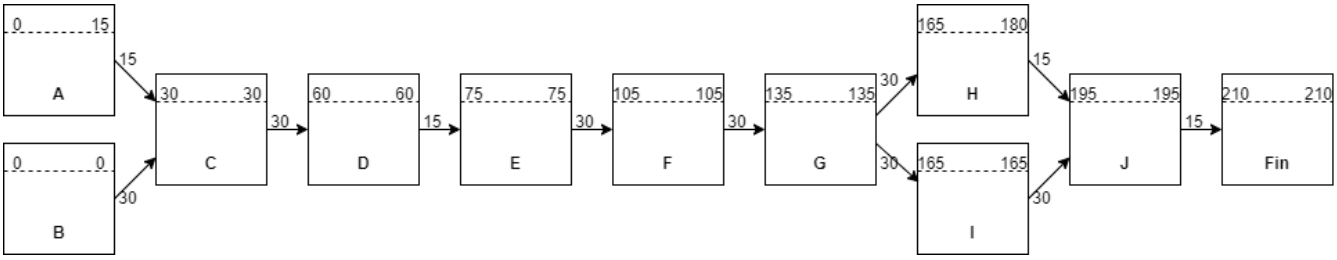
Environnement technique

✓ **Ordinateurs** : M3 ARM chip Apple, 8 core CPU, 8 core GPU, 16 GB RAM - Intel(R) Core(TM) i5-1135G7 - 2.40GHz - 1382 MHz 4 core
✓ **Langage** : Python 3.11.5
✓ **Librairies principales** : Scikit-Learn 1.4.1.post1 - Tensorflow 2.16.1 - Tf-Keras 2.16.0 - Transformers 4.40.1 - Torch 2.2.2

3. Ordonnancement

Tâche	Durée (jours)	Prédécesseur	Niveau	Ressources
A	15	-	0	2 ING
B	30	-	0	2 ING
C	30	A, B	1	2 ING
D	15	C	2	2 ING
E	30	D	3	2 ING
F	30	E	4	2 ING
G	30	F	5	2 ING
H	15	G	6	2 ING
I	30	G	6	2 ING
J	15	H, I	7	2 ING

Graphique MPM. Il permet de visualiser les tâches, leur succession, les marges libres et totales, la durée du projet (210 jours, soit 7 mois), ainsi que le chemin critique, c'est-à-dire les dates à ne pas dépasser sans risquer de perturber la deadline du projet.



4. Planification

Diagramme de Gantt. Le diagramme nous permet de visualiser le **chemin critique**, représentant l'ensemble des tâches qui présentent une **marge totale égale à 0 jours**.

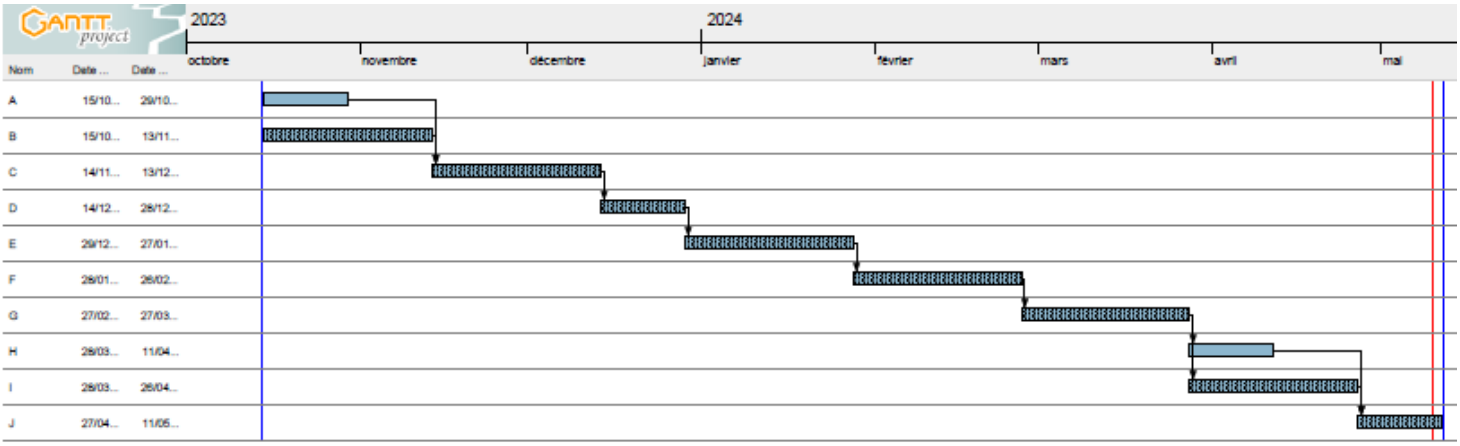
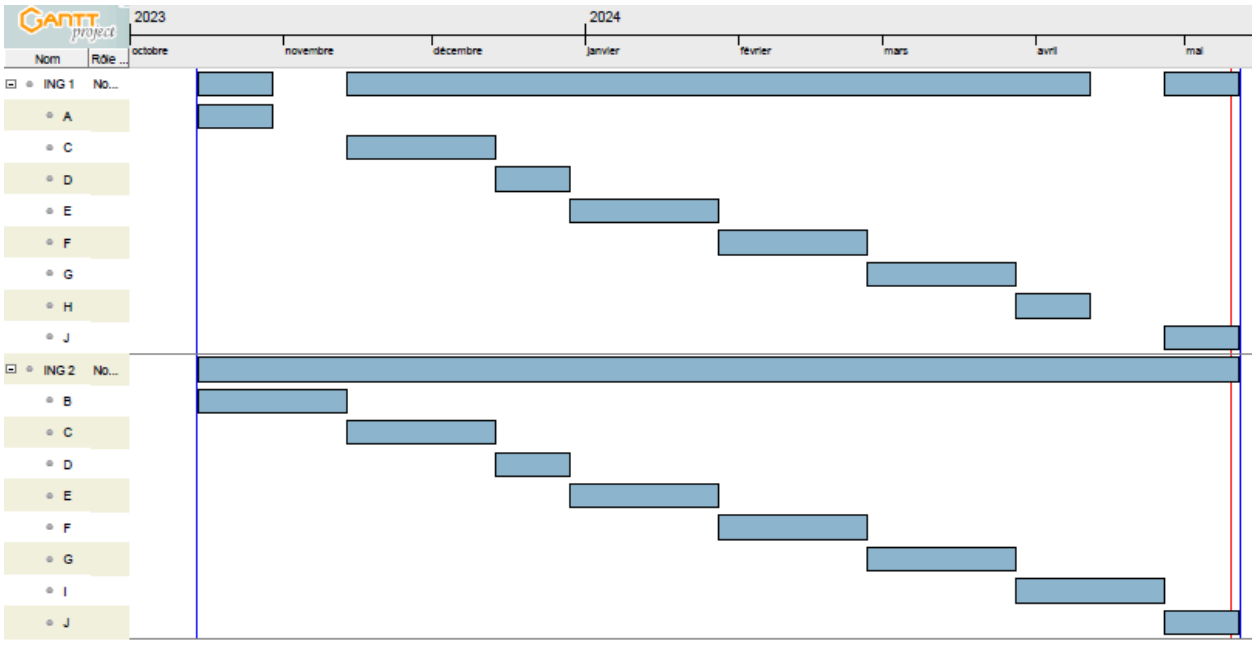


Diagramme des ressources. Il n'existe pas de **contrainte disjonctive** ou **cumulative**. En effet, chaque membre de l'équipe possède son ordinateur et son environnement de développement, et le temps alloué à chaque tâche pour chaque ressource est suffisant, certaines tâches étant même non critiques.



CONCLUSION

A travers ce projet axé sur la détection des fake news politiques, nous avons mené une approche de benchmark de différents modèles optimisés, à la fois de Machine Learning "traditionnels" et de Deep Learning, ainsi qu'une réflexion sur les applications possibles d'un modèle performant sur le terrain, en regard des besoins métiers.

Après une étape de nettoyage, de pré-traitement NLP et de vectorisation des données, nous avons testé : **7 classifieurs en Machine Learning optimisés à l'aide d'une recherche par grille** et d'une **validation croisée** à 5 plis (Naive Bayes, Random Forest, SVM, Logistic Regression, Passive-Aggressive, AdaBoost et Gradient Boosting), et **7 modèles en Deep Learning, avec des architectures de complexité croissante, construites et optimisées successivement** (Simple embedding network, simple RNN, LSTM, bidirectional LSTM, bidirectional LSTM + CNN, Word2Vec - bidirectional LSTM + CNN, et le modèle pré-entraîné BERT).

Le benchmark a révélé une supériorité du modèle le plus complexe sur le plan architectural, combinant une **intégration Word2Vec avec LSTM bidirectionnel et CNN**, réseaux dont nous avons pu relever les spécificités, à savoir la séquentialité du langage avec une forme de rétention en mémoire pour le LSTM, et l'extraction de caractéristiques locales pour le CNN, le tout alimenté par des embeddings de mots générés par le modèle Word2Vec, ayant prouvé sa capacité à saisir efficacement la sémantique des articles. Cette architecture nous a permis d'obtenir une bonne accuracy, supérieure à **86%**, pour la classification des articles jamais rencontrés par le modèle, devançant le modèle pré-entraîné BERT.

Ce résultat suggère qu'une approche combinant des architectures ayant chacune leurs avantages, en utilisant un modèle optimal de représentations de mots, est efficace pour ajuster un modèle à une tâche précise. Il serait intéressant de poursuivre ces investigations afin de comparer d'autres modèles d'embeddings de mots, comme GloVe ou encore FastText, non testés ici, voire une approche différente axée sur les LLM afin d'obtenir une conceptualisation des démarches qu'un modèle peut entreprendre pour arriver à classer des fake news.

Nous avons enfin proposé un outil issu de notre travail : une application nommée FastGuard à destination des utilisateurs, basée sur une logique de réseau citoyen et collaboratif de lutte contre la désinformation, à partir de l'alliance homme-IA et de la blockchain. Cette application résulte d'un travail de co-construction selon la méthodologie du Design Thinking en équipe, après des itérations basées sur l'idéation, le prototypage et le test sur de potentiels utilisateurs.

Ces résultats sont prometteurs et engagent à poursuivre nos recherches dans le futur.

RÉFÉRENCES

Adrian, M.G., Prasetyowati, S.S. & Sibaroni, Y. (2023). Effectiveness of word embedding GloVe and Word2Vec within news detection of indonesian using LSTM. *Jurnal Media Informatika Budidarma*, 7(3), 1180-1188.

Agarwal, V., Sultana, H.P., Malhotra, S. & Sarkar, A. (2019). Analysis of classifiers for fake news detection. *Procedia Computer Science*, 165, 377-383.

Ahmed, A.A.A., Aljabouh, A., Donepudi, P.K. & Choi, M.S. (2021). Detecting fake news using Machine Learning: a systematic review. *Computers and society*, arXiv.

Al-Tai, M.H., Nela, B.M. & Al-Sherbaz, A. (2023). Deep learning for fake news detection: literature review. *Al-Mustansiriyah Journal of Science*, 34(2), 70-81.

Berrondo-Otermin, M. & Sarasa-Cabezuelo, A. (2023). Application of Artificial Intelligence techniques to detect fake news: a review. *Electronics*, 12(24), 5041.

Bian, T., Xiao, X., Xu, T., Zhao, P., Huang, W., Rong, Y. & Huang, A. (2020). Rumor detection on social media with bi-directional graph convolutional networks. *Proc. AAAI Conf. Artif. Intell*., 34(1), 549-556.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45, 5-32.

Choudhury, D. & Acharjee, T. (2023). A novel approach to fake news detection in social networks using genetic algorithm applying machine learning classifiers. *Multimedia Tools and Applications*, 82(6), 9029-9045.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20, 273-297.

Fernandez-Reyes, F.C. & Shinde, S. (2018). Evaluating deep neural networks for automatic fake news detection in political domain. *Iberico-American Conference on AI*, 206-216.

Hu, L., Wei, S., Zhao, Z. & Wu, B. (2022). Deep learning for fake news detection: a comprehensive survey. *AI Open*, 3, 133-155.

Hu, B., Mao, Z. & Zhang, Y. (2024). An overview of fake news detection: from a new perspective. *Fundamental Research*, In press.

Kaliyar, R.K., Goswami, A. & Narang, P. (2021). FakeBERT: Fake news detection in social media with a BERT-based deep learning approach. *Multimedia Tools and Applications*, 80, 11765-11788.

Kareem, W. & Abbas, N. (2023). Fighting lies with intelligence: using large language models and chain of thoughts technique to combat fake news. *Lecture Notes in Computer Science*, 14381, 253-258.

Khan, J.Y., Khondaker, T.I., Iqbal, A. & Afroz, S. (2019). A benchmark study on machine learning methods for fake news detection. *ResearchGate preprint*.

Khattar, D., Goud, J.S., Gupta, M. & Varma, V. (2019). Mvae: Multimodal variational autoencoder for fake news detection. *Proceedings of the International World Wide Web Conferences*, 2915-2921.

Kula, S., Choras, M. & Kozik, R. (2021). Application of the BERT-based architecture in fake news detection. *Advances in Intelligent Systems and Computing*, 1267, 239-249..

Ma, J., Gao, W., Mitra, P., Kwon, S., Jansen, B.J., Wong, K.F. & Cha, M. (2016). Detecting rumors from microblogs with recurrent neural networks. *Proceedings of the International Joint Conference on Artificial Intelligence*, 3818-3824.

Mallik, A. & Kumar, S. (2024). Word2Vec and LSTM based deep learning technique for context-free fake news detection. *Multimedia Tools and Applications*, 83, 919-940.

Mridha, M. F., Keya, A. J., Hamid A., Monowar, M. M. & Rahman, S. (2021). A comprehensive review on fake news detection with deep learning. *IEEE Access*, 9, 156151- 156170.

Nagashri, K. & Sangeetha, J. (2021). Fake news detection using passive-aggressive classifier and other machine learning algorithms. In: S. M. Thampi, E. Gelenbe, M. Atiquzzaman, V. Chaudhary, K.-C. Li (Eds.) *Advances in Computing and Network Communications*. Lecture Notes in Electrical Engineering, vol. 736, p. 221-233. Springer, Singapore.

Nelson, J. (2022). Fake news et deep fakes: une approche cyberpsychologique. *Interfaces numériques*, 11(2).

Pal, S., Kumar, T.S. & Pal, S. (2019). Applying Machine Learning to detect fake news. *Indian Journal of Computer Science*, 4(1), 7-12.

Pennycook, G., Cannon, T.D., & Rand, D.G. (2018). Prior exposure increases perceived accuracy of fake news. *Journal of Experimental Psychology: General*, 147(12), 1865–1880.

Pennycook, G & Rand, D.G. (2021). The psychology of fake news. *Trends in Cognitive Sciences*, 25(5), 388-402.

Sharma, D.K. & Garg, S. (2023). IFND: a benchmark dataset for fake news detection. *Complex & Intelligent Systems*, 9, 2843-2863.

Shu, K., Mahudeswaran, D., Wang, S., Lee, D. & Lui, H. (2019). FakeNewsNet: a data repository with news content, social context and spatiotemporal information for studying fake news on social media. *Social and Information Networks,* arXiv.

Umer, M., Imtiaz, Z., Ullah, S., Mehmood, A., Choi, G.S. & On, B.W. (2020). Fake news stance detection using deep learning architecture (CNN-LSTM). *IEEE Access*, 8, 156695–156706.

Wang, W.Y. (2017). "Liar, Liar Pants of Fire": a new benchmark dataset for fake news detection. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2, 422-426.

Yang, Y., Zheng, L., Zhang, J., Cui, Q., Zhang, X., Li, Z. & Yu, P.S. (2023). TI-CNN: Convolutional neural network for fake news detection. *Computation and Language*, arXiv.

Yu, F., Liu, Q., Wu, S., Wang, L. & Tan, T. (2017). A convolutional approach for misinformation identification. In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 3901–3907.

Yuan, L., Jiang, H., Shen, H. & Shi, L. (2023). Sustainable development of information dissemination: a review of current fake news detection research and practice. *Systems*, 11(9), 458.

Zhang, X. & Ghorbani, A. (2020). An overview of online fake news: characterization, detection, and discussion. *Information Processing & Management*, 57(2), 102025.

https://www.usinenouvelle.com/article/fake-news-ces-technologies-qui-les-traquent.N1818347

https://www.startus-insights.com/innovators-guide/5-top-startups-tackling-fake-news-disinformation/

https://www.udemy.com/course/data-science-et-machine-learning-masterclass-python

https://www.udemy.com/course/cours-complet-de-deep-learning-avec-tensorflow-et-keras

ANNEXES

A. Explication des différents classifieurs de Machine Learning et du choix des hyperparamètres

Naive Bayes

Le modèle Naive Bayes est une méthode de classification probabiliste basée sur le **théorème de Bayes**

$$P(x_i) = \frac{P(C_k).P(C_k)}{P(x_i)}$$

qui estime la probabilité qu'un point de données appartienne à une classe particulière $C_{k'}$ en utilisant les probabilités conditionnelles des caractéristiques x_i. A la base, le modèle postule "naïvement" que toutes les caractéristiques sont indépendantes les unes des autres. Par ex. dans un problème de classification de texte, le modèle suppose que la présence d'un mot particulier dans un document est indépendante de la présence d'autres mots. De cette façon, en connaissant les probabilités d'une caractéristique sachant une classe, la probabilité d'une classe (ici 0.5) et d'une caractéristique, on peut "remonter" vers la connaissance de la probabilité d'une classe, sachant une caractéristique.

Nous avons testé plusieurs valeurs de l'**hyperparamètre alpha** dans une GridSearchCV. Alpha est un paramètre de lissage, utilisé pour éviter les probabilités nulles, notamment pour les caractéristiques qui n'apparaissent pas dans toutes les classes. Cela ajoute des observations fictives de chaque caractéristique dans chaque classe. Un alpha égal à zéro signifie aucun lissage, avec un risque de surajustement, alors qu'un alpha trop fort lisse excessivement.

Random Forest

L'utilisation de ce modèle répond à l'amélioration des performances et à la levée des limitations liées aux **arbres de décisions classiques**. Le modèle random forest est un **apprenant dit d'ensemble** (constitué de plusieurs arbres de décisions) qui créé des sous-ensembles de features sélectionnées aléatoirement pour chaque division potentielle. Toute feature utile finira donc par être sélectionnée. Finalement, la prédiction retenue sera celle de la majorité des arbres (prédiction agrégée par un système de vote), ou alors une probabilité de classe sortante. La majorité des hyperparamètres sont communs avec ceux des arbres de décision.

Pour les hyperparamètres spécifiques au random forest, nous avons choisi, grâce à la grid search, de faire varier:

- **Le nombre d'estimateurs (n_estimators)** : combien d'arbres ? De manière très intéressante, l'étude princeps de Breiman (2001) nous montre que naturellement les possibilités d'apprendre augmentent avec le nombre d'estimateurs, mais qu'en plus, il n'existe pas de risque d'overfitting car à mesure que le nombre d'arbres augmente, ceux-ci finissent par utiliser les mêmes features et à se corréler. Il n'y a donc pas d'augmentation de la complexité, mais plutôt une duplication des prédictions. La littérature présente empiriquement une valeur raisonnable par défaut aux environs de 100 pour cet hyperparamètre.

- **Le nombre de features (max_features)** : combien de features dans chaque sous-ensemble aléatoire ? A nouveau, la publication d'origine (Breiman, 2001) suggère de fixer le nombre de features aléatoires à $log_2(N + 1)$ dans le sous-ensemble. D'autres constats empiriques proposent $\sqrt{N}$, d'autres encore $\frac{N}{3}$ (pour la régression).

- **Échantillons Bootstrap (bootstrap)** : échantillonnage aléatoire avec remplacement, fixé à True par défaut. Il s'agit d'entraîner chaque estimateur sur un sous-ensemble de lignes et de features, limitant ainsi la corrélation des arbres entre eux, et favorisant la généralisation à de nouvelles données inédites.

Remarque sur l'hyperparamètre spécifique Erreur OOB (oob_score) : découlant du paramètre précédent, si l'ensemble ne voit pas un certain nombre de données, il sera alors possible de s'en servir pour calculer une métrique d'erreur additionnelle et facultative (réglée à False par défaut). Toutefois, du fait que ce paramètre n'affecte aucunement l'entraînement ni les arbres, nous ne l'avons pas fait intervenir dans notre grid. Pour les hyperparamètres communs avec Decision Tree, nous avons fait varier la qualité de la division des noeuds (**criterion**) : nous avons ainsi testé :

- **L'indice (impureté) de Gini**, soit la probabilité qu'un élément choisi au hasard soit mal classé.

- **L'entropie**, soit la mesure de l'incertitude et du désordre.

- **La log loss**, généralement utilisée dans des tâches de classification (notons que son utilisation comme critère de division pour un arbre de décision est peu fréquente dans la littérature).

Nous n'avons pas fait varier **max_depth** et **max_leaf_nodes**, paramètres qui permettent de minimiser la complexité de l'ajustement du modèle, considérant que dans notre cas, nous laissons le modèle "libre" d'explorer toute la variance possible dans les données pour maximiser les performances.

SVM

Mathématiquement, ce modèle se base sur la création d'**hyperplans** (sous-espaces de dimension $N-1$) pour séparer des classes, l'enjeu étant de trouver le placement adéquat de ce séparateur pour minimiser l'erreur de prédiction, sachant que ce séparateur doit à la fois admettre des marges qui maximisent la distance entre les classes, mais aussi des marges qui ont suffisamment de souplesse (et donc introduisent un terme d'erreur ϵ) pour offrir un bon compromis biais-variance dans le modèle. Plus formellement, on peut écrire la formule décrivant les classificateurs à vecteurs de support comme suit:

$$y_i\left(\beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_p X_{ip}\right) \geq M\left(1 - \epsilon_i\right)$$

où il s'agit de choisir les meilleurs coefficients β pour maximiser la marge M, tout en jouant avec ϵ. La publication princeps (Cortes & Vapnik, 1995) introduit le concept de "kernel trick", utilisant le produit scalaire (mesure de la similarité entre l'espace de features d'origine et l'espace de features élargi) pour réduire le coût en calcul lié à des noyaux non linéaires qui engendreraient beaucoup de complexité dans le modèle.

Pour ce modèle complexe, comportant plusieurs hyperparamètres qui ne sont pas faciles à régler intuitivement, la GridSearch est particulièrement indiquée. Ainsi, nous avons fait varier:

- **Le paramètre de régulation (C)** : dans le modèle, plus C est grand, plus nous acceptons d'erreurs de classification ; dans sklearn, pour rester dans la cohérence par rapport aux autres modèles, C a été prévu pour être inversement proportionnel à la tolérance en erreur (donc plus C est petit, plus on acceptera de points à l'intérieur des marges).

- **Le noyau (kernel)** : le type de noyau qui projette les données dans une dimension supérieure (linéaire, ou non linéaire - e.g. à fonction de base radiale rbf). Sklearn fixe un noyau rbf par défaut, car c'est celui qui performe le mieux.

- **La flexibilité dans la frontière de décision (gamma)** : une valeur faible admettra une frontière plus lisse (et donc un risque d'underfitting), alors qu'une valeur élevée admettra une frontière plus complexe (avec un risque d'overfitting). Nous avons testé les gamma 'scale' (par défaut, qui utilise $\frac{1}{nfeatures \times X.var(\ )}$ comme valeur) et 'auto' (qui utilise $\frac{1}{nfeatures}$).

Logistic Regression

La régression logistique est un modèle largement utilisé pour la classification, en prédisant la probabilité qu'une observation appartienne à une certaine classe. Elle fonctionne par la transformation d'un modèle de régression linéaire

$$\hat{Y} = \sum_{i=0}^{n} \beta_i x_i$$

en un modèle de classification, grâce à la fonction logistique (ou sigmoïde)

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

et ceci afin de transformer toute entrée en une valeur comprise entre 0 et 1, qui pourra ainsi être traitée comme une probabilité. Pour quantifier l'erreur, ce modèle fonctionne avec le maximum de vraisemblance, ou la plausibilité du modèle vis-à-vis de vraies données. Il s'agira donc de trouver les valeurs des β qui maximisent la probabilité d'obtenir les données réellement observées. Cela revient à minimiser le négatif du log de la fonction de vraisemblance (log loss).

Pour l'optimisation, nous avons fait varier dans une GridSearch:

- **La pénalité de régularisation (penalty)** : nous avons testé la régularisation L1 (Lasso) et L2 (Ridge).

- **La valeur C de la régularisation** : il est recommandé d'utiliser des valeurs C espacées logarithmiquement. Ses valeurs sont, comme pour le SVM, inversement proportionnelles à la tolérance à l'erreur.

- **Le solveur (solver)** pour spécifier l'algorithme à utiliser pour résoudre le problème d'optimisation lors de l'estimation des coefficients : par défaut 'lbfgs', 'sag' et 'saga' recommandés pour des grands ensembles de données.

- **Le nombre maximal d'itérations autorisées lors de l'optimisation des coefficients (max_iter)** : nous avons fixé une valeur de 500 dès l'instanciation du classifier.

Passive-aggressive

Ce modèle est principalement utilisé pour des tâches de classification binaire. Son nom provient de son comportement lors de la mise à jour des paramètres du modèle. En effet, il s'adapte de manière "passive" lorsque les prédictions du modèle sont correctes, mais devient "agressif" en cas d'erreur de classification, en effectuant alors des mises à jour de ses paramètres, afin de minimiser l'erreur. Cette mise à jour se réalise de manière incrémentale en utilisant une projection, et s'effectue comme suit :

$$w = w + \eta y x$$

où η est le taux d'apprentissage, et y le label. Cet algorithme est particulièrement intéressant pour des données traitées en temps réel et de manière continue, cas d'usage où le modèle doit s'adapter en permanence à l'arrivée de nouvelles données.

Les hyperparamètres que nous avons retenu et fait varier sont :

- **Le paramètre de régularisation C** : testé à différentes échelles.

- **La fonction de coût (loss)** : 'hinge' pour le PA-I dans la publication de référence, et 'squared_hinge' pour le PA-II.

- **Le maximum d'itérations** : nous n'avons pas spécifié ce paramètre et avons ainsi laissé la valeur par défaut, 1000.

Méthodes de Boosting

Les méthodes de Boosting ne sont pas des algorithmes de Machine Learning à proprement parler, mais plutôt des **algorithmes d'ensemble ou méthodes de méta-apprentissage**, qui fonctionnent selon un principe d'agrégation de résultats. Ils peuvent être utilisés avec toutes sortes de modèles, mais fonctionnent généralement bien avec les arbres de décision, dont ils parviennent à améliorer les performances (à ce titre on remarque que les paramètres définis par sklearn sont adaptés aux decision trees). C'est avec les arbres que nous les avons utilisés ici. Ils consistent en une agrégation de modèles dits **"apprenants faibles"** (un modèle trop simpliste pour performer seul - les petits arbres "souches" en sont une

bonne illustration), dans un ensemble d' **"apprenants forts"**. Pour cela, une combinaison d'estimateurs avec un certain coefficient appliqué, pourrait agir comme un estimateur d'ensemble efficace. En d'autres termes il s'agirait d'une sorte de concaténation de modèles auxquels on a appliqué un coefficient α. Nous avons donc:

$$F_T(x) = \sum_{t=1}^{T} f_t(x)$$

avec l'ensemble d'apprenants faibles représenté par:

$$f_t(x) = \alpha_t h(x)$$

où T représente le nombre total d'arbres, t un arbre particulier, et h une hypothèse de sortie. De manière globale, comme pour tous les modèles, l'enjeu est de minimiser les erreurs de prédiction. Dans ce type de méthode, cela est réalisé en trouvant le meilleur nombre d'apprenants faibles, les apprenants les plus "doués", ainsi qu'un coefficient α optimal. Nous nous sommes penchées sur les performances de deux estimateurs : **AdaBoost** et **Gradient Boosting**. Les principales différences entre les 2 méthodes sont :

- **Un coefficient** α qui diffère selon l'apprenant pour AdaBoost (plus il est élevé, plus l'apprenant a d'influence), mais qui est commun à tous pour Gradient Boosting.

- **Une stratégie d'ajustement des poids différente** : si AdaBoost se base sur les exemples mal classés dont il augmente les poids pour faciliter la décision de l'apprenant suivant, Gradient Boosting se base sur la fonction de perte.

1. AdaBoost

Pour AdaBoost, nous avons fait varier les hyperparamètres suivants (nous savons que par défaut c'est decision tree qui est utilisé comme estimateur):

- **Le nombre d'estimateurs (n_estimators)** : fixé à 50 par défaut, nous avons pris 50 et 100.

- **Le taux d'apprentissage (learning_rate)** : taux par défaut, 2x plus lent, et 2x plus rapide.

- **La profondeur maximale des apprenants (max_depth)** : par défaut 1 (une souche).

2. Gradient Boosting

Pour Gradient Boosting, nous avons fait varier dans la grid search:

- **Le nombre d'estimateurs (n_estimators)**.

- **Le taux d'apprentissage (learning_rate)** : taux par défaut, 2x plus lent, et 2x plus rapide.

- **La profondeur maximale des apprenants (max_depth)**.

Notons que la grid search est ici particulièrement appropriée pour trouver un compromis entre le taux d'apprentissage et le nombre d'estimateurs : c'est crucial car plus le taux d'apprentissage est élevé, plus les arbres suivants auront un effet important, il suffira alors de quelques arbres. Mais s'il est trop élevé, on n'obtiendra pas les meilleures performances car on n'aura pas pris le temps d'apprendre. A contrario, si le taux est trop faible, les arbres suivants auront peu d'influence et il faudra créer plus d'arbres, ce qui augmentera d'autant le temps d'apprentissage.

B. Logique de construction des architectures de réseaux neuronaux et réglage des hyperparamètres

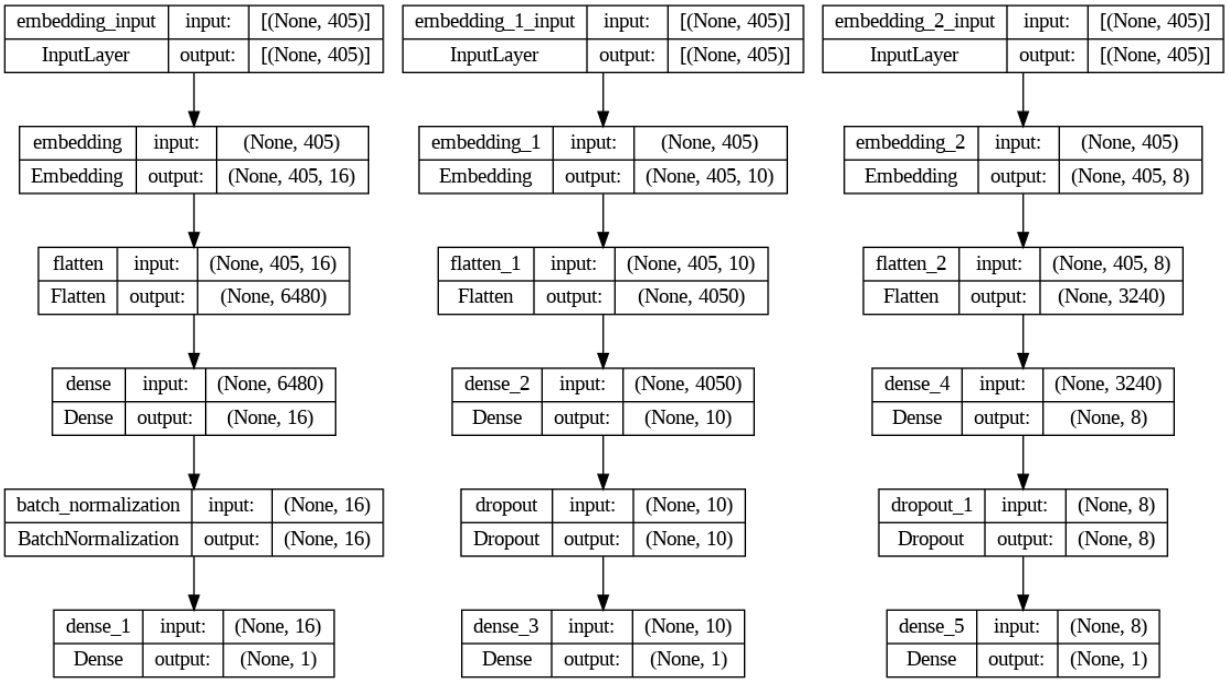
Chaque architecture propose une première **couche d'embedding** (avec ou sans entraînement) et une dernière **couche dense** à une unité, associée à une **fonction d'activation sigmoïde** pour la classe de sortie.

Simple embedding neural network

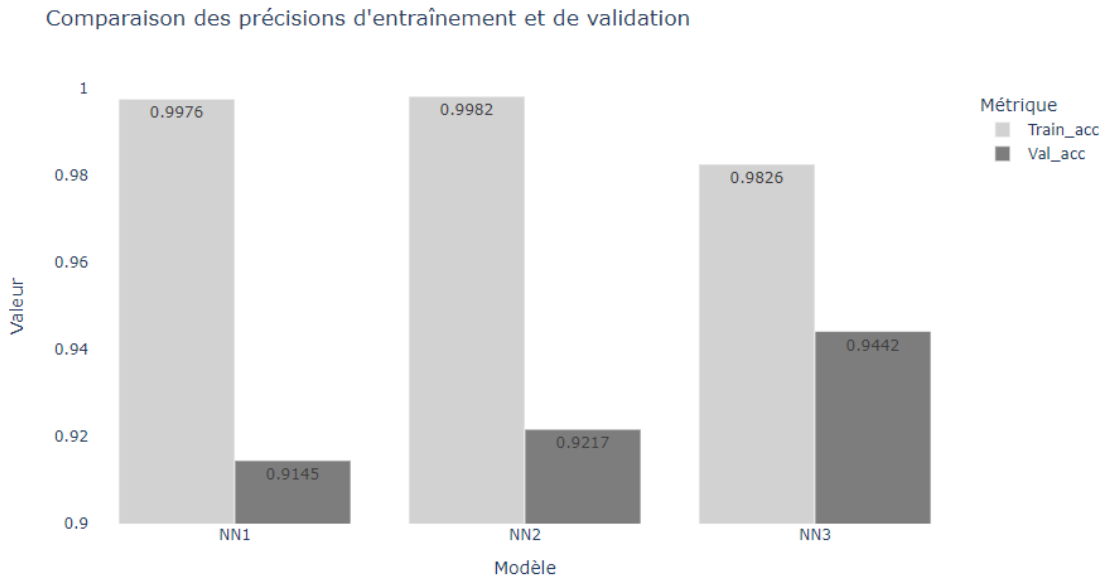
Nous avons dans un premier temps commencé notre travail avec une architecture simple, fully connected avec une **couche d'embedding**, puis l'avons progressivement optimisée pour réduire l'overfitting, en deux étapes :

- **Réduction de dimensionnalité du vecteur d'embedding** (16 à 10) et ajout d'une **couche dropout** pour désactiver aléatoirement et temporairement certains neurones.
- **Nouvelle réduction de dimensionnalité d'entrée avec ajout d'une régularisation L2**.

Ces étapes ont donné naissance à 3 réseaux selon les résumés suivants :



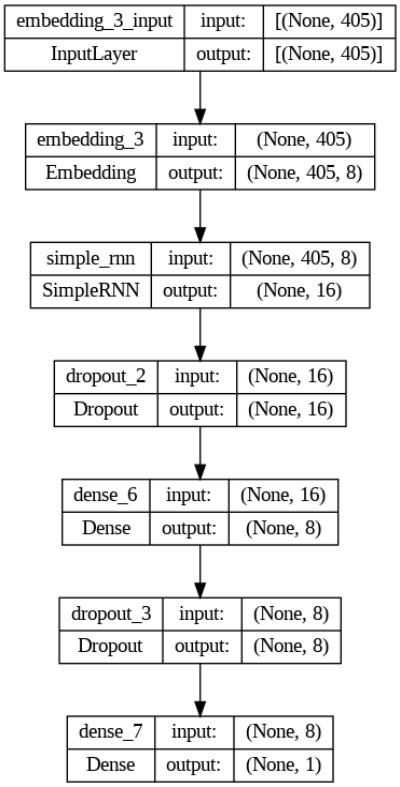
Ce réglage a resserré l'écart entre les scores de validation et de train, réduisant ainsi l'overfitting, comme nous le voyons sur le graphique suivant.



Les paramètres gardés constants car générant les meilleures performances ont été les suivants : **optimisateur Adam**, taille de **batchs de 200**, et ajout d'un callback **early stopping**.

Simple RNN

Jusqu'à présent, les modèles passés en revue ont pris en considération des caractéristiques d'importance des mots dans un corpus. TF-IDF a permis de caractériser des fréquences relatives, et le word embedding a élaboré un dictionnaire de mots fréquents, tout en conservant leur agencement dans une séquence. Le premier modèle a pu tenir compte de séquences fixes, mais sans que le modèle ne prenne en charge de manière ciblée cette séquentialité. **Nous avons donc introduit une dimension de séquentialité dans le traitement du texte, à partir des réseaux de neurones récurrents ou RNN**. Dans la plupart des langages, l'ordre des mots est la caractéristique la plus discriminante pour la compréhension humaine. Il est à noter que c'est principalement le cas pour l'Anglais, langue que nous traitons ici. Or, certaines langues dites casuelles, sont centrées davantage sur les liaisons grammaticales (e.g. les cas en Allemand). Dans la langue qui nous occupe, l'utilisation d'une architecture séquentielle nous semble très pertinente pour augmenter les performances d'un modèle à une tâche classification de textes. Ainsi, les modèles se basent sur une probabilité de voir un mot apparaître, en fonction du/des mot(s) précédent(s). Al-Tai et al. (2023) comparent cette fonctionnalité à une forme de mémoire. Ici nous considérons un modèle de type **'many-to-one'** (séquence à vecteur) dans le sens où à partir d'une séquence de mots est générée une seule prédiction finale. Ce RNN est agencé comme suit :



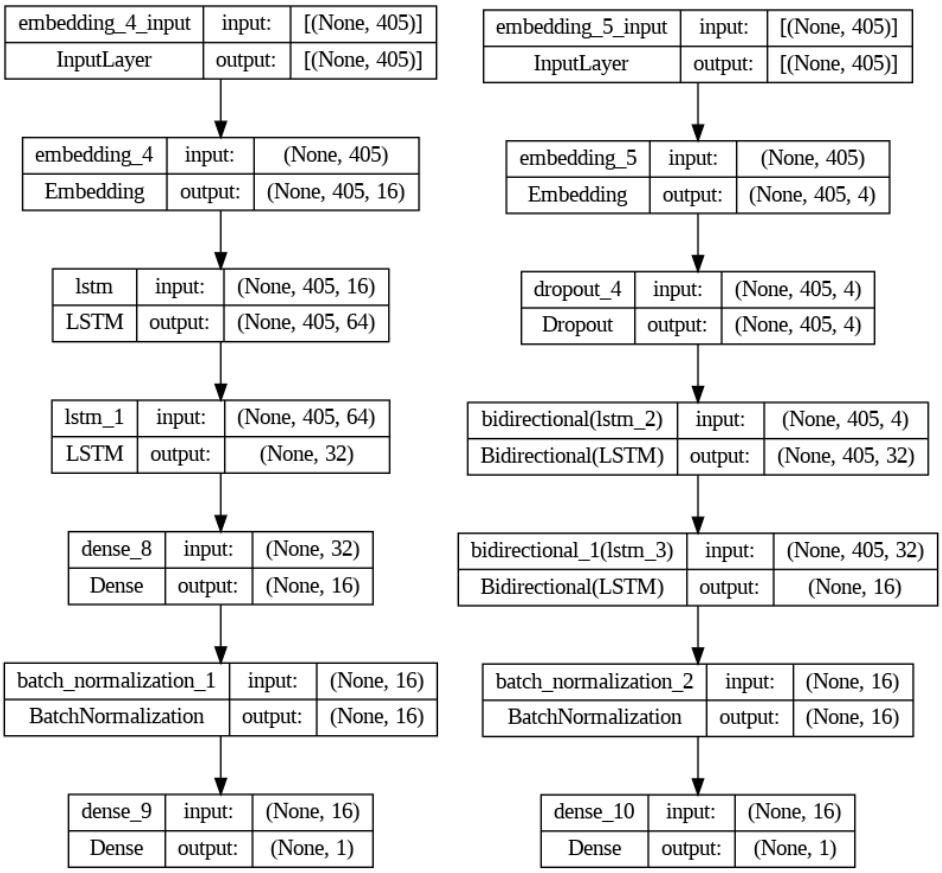
Pour l'optimisation, l'ajout d'une couche Dropout par rapport au réseau précédent a légèrement fait augmenter les performances, mais l'augmentation du nombre d'unités dans la couche RNN a rendu le modèle instable, elles ont donc finalement été fixées à 16. Les autres hyperparamètres sont restés constants.

LSTM

Un RNN simple présente la limitation de ne pas saisir de dépendances à long-terme. L'instabilité observée dans un RNN simple peut être attribuée à un problème d'**exploding gradient**, phénomène courant dans ce type d'architecture, lorsqu'un traitement séquentiel est réalisé sur des séquences très longues. Mathématiquement, lors de l'entraînement et plus précisément de la rétropropagation, les gradients des poids sont calculés par rapport à une fonction de perte à l'aide de la règle des chaînes. Si les poids sont grands, le gradient peut exploser à mesure du processus et empêcher la convergence de la fonction vers un minimum. Selon Al-Tai et al. (2023), l'utilisation d'une **extension LSTM** dans un RNN présente l'avantage de contourner le biais des vanishing/exploding gradients, tout en autorisant l'apprentissage sur de longues séquences. Cela s'explique par l'existence d'une **'mémoire à long terme'** et d'une **'mémoire à court terme'** inhérentes à la couche LSTM qui permettent au modèle de n'être pas autant affecté par la rétropropagation que les RNN simples. Des mécanismes de contrôle optimisent par l'intermédiaire de **'gate'** les quantités d'informations à traiter à court ou à long terme, et celles à oublier, un peu à la manière dont travaille la mémoire humaine.

Nous sommes parties d'un réseau avec deux couches **LSTM unidirectionnelles** (en augmentant la dimension de l'espace vectoriel de sortie à 16) associées à des fonctions d'activation tanh, suivies de deux couches denses entourant une couche BatchNormalization pour normaliser les activations et stabiliser le modèle. Puis nous avons remplacé les couches LSTM unidirectionnelles par des **LSTM bidirectionnelles**, toujours associées à une tanh, en diminuant également le nombre d'unités pour espérer réduire la

complexité, et en ajoutant une couche dropout préalable. Ces deux architectures donnent les résumés suivants, la 2e architecture ayant bien performé.

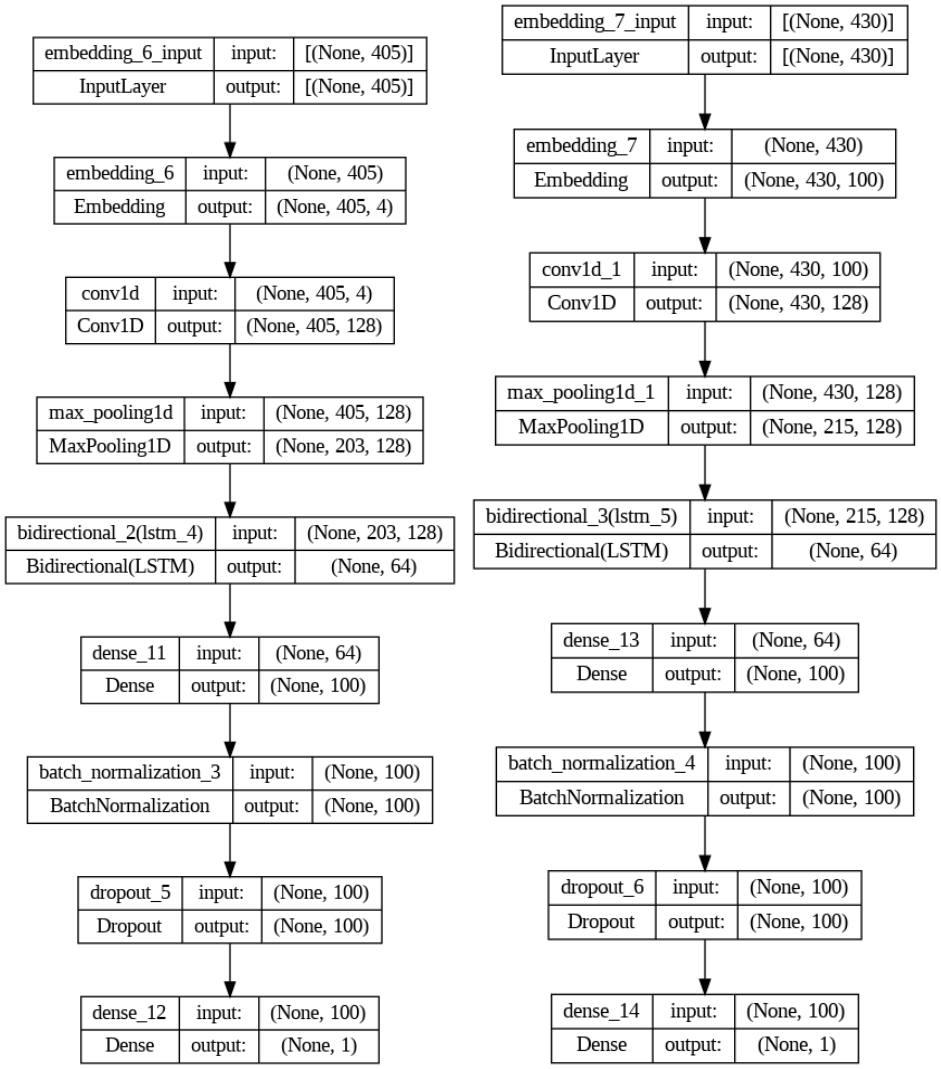


Word2Vec / LSTM + CNN

Nous avons ensuite testé l'approche combinant un réseau avec couches LSTM bidirectionnelles (les plus efficaces) et des couches de convolution propres au CNN, afin d'allier deux perspectives complémentaires dans l'extraction de caractéristiques propres aux documents, d'une part la capture de la séquentialité des mots avec LSTM (dans une perspective globale), associée à une extraction de caractéristiques locales par l'intermédiaire du CNN. Ce modèle a pu se montrer efficace en ajoutant un régularisateur L1 dans chaque couche dense, en diminuant la dimension de l'espace vectoriel de sortie à 4 et en augmentant l'input_shape au maximum.

Ensuite, nous avons donné en entrée d'un modèle LSTM+CNN un input issu des vecteurs de mots entraînés sur notre corpus avec Word2Vec, très efficace pour capter des représentations sémantiques et globales des corpus de texte. La couche d'embedding du réseau LSTM+CNN a donc pris ces vecteurs-résultats sans être entraînée (**trainable=False**). Le résultat a montré une performance très intéressante pour ce dernier modèle.

Le résumé des deux réseaux est le suivant :



BERT

Pour terminer, nous avons choisi de tester le modèle BERT selon une approche d'apprentissage par transfert, qui consiste à utiliser un modèle pré-entraîné et à l'adapter à une tâche cible en le ré-entraînant sur un corpus de données spécifique. Cette adaptation est possible grâce au fine-tuning du modèle. BERT est un modèle de pointe qui peut être utilisé dans de nombreuses applications de NLP (Mridha et al., 2021). Il intègre des représentations linguistiques pré-entraînées développées par Google et est construit sur une architecture codée par transformers. Il se distingue par sa capacité à identifier et capturer le sens contextuel dans un texte, et élimine les limitations habituellement liées à l'uni-directionnalité. Pour le fine-tuning, nous l'avons d'abord initialisé à partir de **'bert-base-uncased'** avec les paramètres pré-entraînés, puis l'avons ré-entraîné à partir des données labellisées et adapté les paramètres à notre tâche, notamment en réglant plusieurs fois le learning rate.