IFT 714 - Traitement automatique des langues naturelles
Groupe 1 : Détection des fausses nouvelles par vérification des faits
Rapport final

Anne-Sophia Lim, Yanis Perrin, Caroline Wang

1 Résumé

L'émergence d'Internet et des réseaux sociaux permet de mettre en lumière le problème majeur que pose la propagation rapide et difficilement contrôlable des fausses nouvelles. Notre projet porte sur la détection de ces informations, grâce à la combinaison de techniques de traitement automatique des langues naturelles avec des méthodes d'apprentissage machine avancées, en implémentant des modèles de classification. Après augmentation de notre jeu de données par Easy Data Augmentation et paraphase à l'aide du modèle de langage Bert ainsi que l'extraction de nouvelles caractéristiques, nous voulons déterminer le degré de véracité des informations, tout en mesurant l'impact des données augmentées. Les résultats obtenus de nos différents modèles sont prometteurs sur l'efficacité de notre nouvelle approche par rapport aux approches habituelles adoptées dans la littérature actuelle.

2 Introduction

L'avènement d'Internet et des médias sociaux a bouleversé notre manière de consommer l'information, facilitant la propagation mondiale d'informations non vérifiées. La désinformation met en évidence l'importance de vérifier les informations diffusées. Pour cela, les plateformes mettent en place des méthodes de détection de fausses informations telles que l'analyse textuelle et les algorithmes de traitement automatique du langage. Ce projet vise à comparer plusieurs modèles de classification multiclasse permettant de déterminer le degré de véracité des informations véhiculées et de mesurer l'impact d'une augmentation de données sur les résultats obtenus, ce procédé est représenté par la figure 1.

Les actuels classificateurs de fausses nouvelles sont très complexes et demandants en ressources. Le présent objectif est d'utiliser des modèles simples et peu gourmands, et d'optimiser leurs résultats à l'aide d'opérations en amont telles que l'augmentation de données, le travail sur l'extraction de caractéristiques du texte à vérifier et sur la représentation du texte.

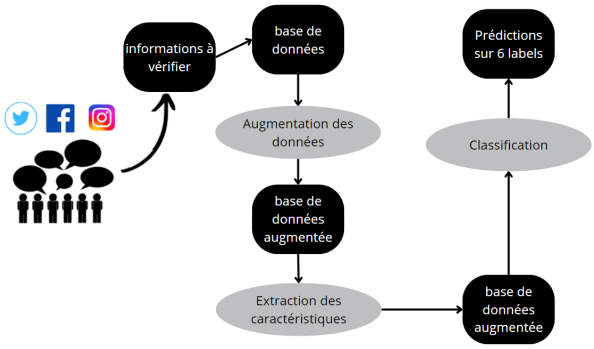

Figure 1: Processus de vérification de faits

3 Etat de l'art

La détection des fausses nouvelles représente souvent un défi de classification, habituellement binaire (vrai/faux) mais parfois multi-classes (graduation entre le vrai et le faux). Certaines approches, comme l'analyse de la neutralité du vocabulaire et la comparaison avec des textes similaires, traitent la détection comme un problème de régression, où l'authenticité des textes est évaluée à l'aide de scores de probabilité (Nakashole and Mitchell, 2014). Cette approche est limitée par le risque que les textes qualifiés de fausses nouvelles peuvent être sur-représentés par un certain type de vocabulaire. Des méthodes comme la représentation de texte à travers des méthodes comme TF-IDF ou des modèles N-gram ont été développées pour traiter les textes (Ahmed et al., 2017). Des modèles performants tels que les réseaux de neurones à convolution ont été utilisés pour effectuer une classification multiclasse sur le jeu de données choisi, mais les résultats étaient disparates (0.274 et 0.962 de

justesse respectivement). (Wang, 2017) (Das Bhattacharjee et al., 2017). Enfin, la méthode LSTM avec un mécanisme d'attention peut présenter une légère hausse du score de justesse suivant les différentes études (0.415 (Long et al., 2017) et 0.457 (Oshikawa et al., 2020)). Cette différence pourrait venir d'un biais sur l'apprentissage, causé par le volume faible des données. En effet, le traitement et l'extraction des caractéristiques des textes a permis à ces modèles d'apprendre assez efficacement, mais le manque de données sur les fausses nouvelles ne permet pas un apprentissage assez robuste pour certains modèles. Pour pallier ce problème, des méthodes d'augmentation de données qui améliorent la capacité de généralisation des modèles ont été développées. Parmi elles, l'EDA (easy data augmentation) repose sur quatre transformations pour chaque phrase (Wei and Zou, 2019) : la **substitution de synonymes** aléatoire de n mots de la phrase par l'un de ses synonymes, l'**insertion aléatoire** d'un synonyme de n mots, la **permutation aléatoire** de deux mots dans la phrase, effectuée n fois et la suppression aléatoire de chaque mot de la phrase avec une probabilité p. Ce procédé est simple à mettre en place, toutefois les variations sont limitées et la sémantique de la phrase n'est pas nécessairement conservée. Il est également possible d'utiliser des modèles transformers préentrainés (Kumar et al., 2020) tels que BERT ou BART, dont l'efficacité a déjà été montrée sur diverses bases. Cependant, ces modèles requierent néanmoins un volume relativement important de données pour bien généraliser. Ces deux méthodes seront appliquées à la base de données choisie pour en évaluer les impacts.

4 Méthodologie

Notre méthodologie se décompose en six grandes étapes : le prétraitement de nos données, l'augmentation de nos données, l'extraction des caractéristiques, la représentation de nos données textuelles et enfin l'entrainement et le test de nos modèles. Notre approche se caractérise par une différence notable par rapport à l'état de l'art actuel: **nous n'utiliserons pas de modèles avancés telles que des transformers où des CNNs**. Notre but ici est d'étudier les effets de l'augmentation de nos données ainsi que des différentes représentations du texte afin de comprendre son impact sur des modèles simples tout en les comparant à des modèles modernes sans augmentation.

Prétraitement des données

Avant d'entamer la classification, il est essentiel d'effectuer une étape de prétraitement afin de réduire la taille de nos données pour pouvoir en tirer les informations les plus pertinentes pour notre tâche de classification. Cette étape de prétraitement contient notamment la tokenisation ainsi que la lemmatisation du texte mais aussi la gestion des mots non significatifs.

La **tokenisation** du texte consiste à découper le texte en unités plus petites, appelées tokens, qui peuvent ensuite être traitées par nos modèles. Pour ce faire, nous utilisons la bibliothèque python bien établie Natural Language Toolkit (nltk).

Une fois les tokens récupérés, il est important de supprimer les mots non significatifs, qui n'apportent pas ou très peu d'informations comme *a, about, by, for, from etc*. Cette suppression est dûe au fait que les mots non significatifs pourraient apporter du bruit non désiré lors de la classification du texte.

La **lemmatisation** cherche à transformer les tokens jusqu'à leurs formes fondamentales, cette forme résultante est appelée « lemma ». Par exemple, lors de la lemmatisation, les tokens "runs", "running" et "ran" sont réduits au lemma "run". Cela nous permet de décroitre le nombre de types de mots dans nos données et d'obtenir une classification plus rapide et efficace.

Augmentation de nos données

L'augmentation des données est au coeur de notre projet. Elle permet d'augmenter la taille et la diversité du jeu de données, d'améliorer la généralisation du modèle et de réduire le possible surapprentissage. Dans ce projet, nous augmentons nos données de deux manières différentes. Dans un premier temps, par la technique **EDA** que l'on a vu précédemment puis dans un second temps nous utiliserons un modèle préentrainé **BERT** (Devlin et al., 2019) afin de générer des exemples qui prennent en compte le contexte de la phrase.

Extraction des caractéristiques

Nous avons extrait plusieurs caractéristiques lexicales comme le nombre de caractères liés à la ponctuation mais aussi le nombre de mots et de digits dans les citations. Il est aussi intéressant d'extraire le nombre d'étiquetage des mots dans un texte avec leurs parties du discours correspondantes (verbes, adjectif, pronom. . . etc). Cela pourrait aider les algorithmes à comprendre la structure grammaticale et la signification de nos citations. Enfin, nous avons extrait une analyse sentimentale en évalu-

ant les sentiments dégagés par la citation (positif, neutre et négatif).

Réprésentation du texte

Les algorithmes modernes de traitement automatique des langues utilisent les "embeddings" comme représentation du sens des mots. Dans notre projet, nous utilisons des embeddings de phrases qui sont des représentations vectorielles de haute dimensionnalité qui capturent la sémantique et le contexte d'une phrase ou d'un paragraphe dans un espace vectoriel continu. Contrairement aux embeddings de mots qui représentent chaque mot individuellement, les embeddings de phrases représentent l'ensemble de la phrase dans un seul vecteur. Dans le projet, nous utilisons 3 différentes représentation du texte: **TF-IDF** (Robertson, 2004), **Doc2Vec** (Le and Mikolov, 2014) et les embedding générés par un modèle **Bert** préentrainé.

Entrainement, test et modèles de référence

Afin d'accomplir notre tâche, nous récupérons trois fichiers csv déjà existants (train, valid, test) qui représentent respectivement l'ensemble d'entraînement, de validation et de test. Les fichiers train et valid seront utilisés pour l'apprentissage de nos modèles tandis que le fichier de test servira uniquement à tester le modèle sur des données jamais vues. Pour ce projet, nous comparerons nos résultats à plusieurs modèle de références qui ont obtenus différents résultats au fil des années depuis l'apparition de notre jeu de données (Ali et al., 2022).

5 Expériences

Base de données

La base de données choisie est la base LIAR (Liar Dataset) (Wang, 2017). Elle contient plus de 12 000 citations politiques factuelles provenant du site internet POLITICFACT.COM[1] avec des métadonnées telles que le sujet, le contexte, l'auteur mais aussi l'historique de la véracité des faits qu'il a énoncés précédemment. Cet historique est défini par le nombre de déclarations inexactes ou exactes qu'a pu faire le politicien par le passé. Chaque déclaration est étiquetée par un éditeur du site avec l'une des six valeurs possibles de véracité. La base de données contient 14 attributs et nous en avons gardés 7. Ces attributs sont répertoriés dans le tableau 1.

Les données sont relativement bien balancées, à part pour l'étiquette "pants on fire" qui est moins

[1] https://www.politifact.com/

Nom de la colonne	Signification
label	étiquette de la citation ("pants on fire", "false", "barely true", "half true", "mostly true" ou "true")
statements	citation du politique
c_pants_on_fire	Nombre d'anciennes citations classées comme des mensonges éhontés ("pants on fire")
c_false	Nombre d'anciennes citations classées comme fausses ("false")
c_barely_true	Nombre d'anciennes citations classées comme étant à peine vraies ("barely true")
c_half_true	Nombre d'anciennes citations classées comme étant à moitié vraies ("half true")
c_mostly_true	Nombre d'anciennes citations classées comme étant principalement vraies ("mostly true")

Table 1: Tableau récapitulatif des attributs gardés de la base de données LIAR

représentée, comme le montre la figure 2. La classification s'effectue alors comme montré par la figure 3 : le modèle prend en entrée une citation et l'historique des crédits lié à son auteur pour prédire l'une des six classes.

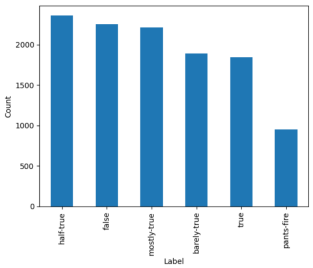

Figure 2: Histogramme des étiquettes

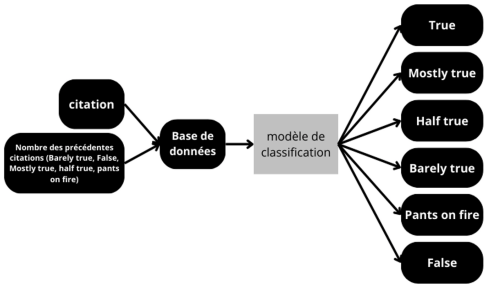

Figure 3: Illustration de la classification effectuée

Méthode d'évalutation

En ce qui concerne les métriques d'évaluation, afin de ne pas se limiter à la simple exactitude du modèle nous utilisons aussi le rappel, la précision et le score F1, ce sont des métriques claires utilisées

de nombreuses fois dans les travaux antérieurs.

Détails expérimentaux

Dans ce projet, nous avions choisis initialement quatres modèles classiques de machine learning : LinearSVM (Hearst et al., 1998), arbres de décisions (Quinlan, 1986), forêts d'arbres de décisions (Breiman, 2001) et les k plus proches voisins (Guo et al., 2003). Après plusieurs test, ils nous paraissaient intéressant d'inclure un tout nouveau modèle qui prendrait en compte séparémment les données textuelles des données numériques (cf : Figure 4). La couche LSTM (Hochreiter and Schmidhuber, 1997) et la couche complétement connecté transforment les entrées en variables latentes qui sont ensuite concaténées et passées à travers une couche fully-connected puis dans un softmax afin de données la prédiction de la classe.

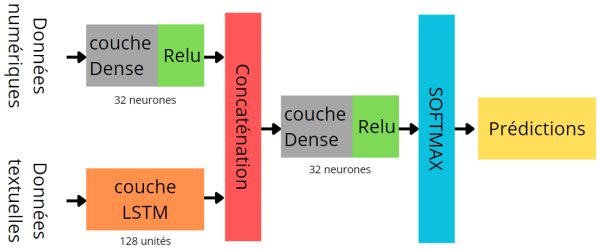

Figure 4: Nouveau modèle implémenté

Résultats

		No augment				EDA augment				BERT augment			
	Representation	A	R	P	F1	A	R	P	F1	A	R	P	F1
LSVM	TF-IDF	0.27	0.27	0.27	0.27	0.25	0.25	0.25	0.25	0.26	0.26	0.26	0.26
DecisionTree	TF-IDF	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.39	0.36	0.36	0.36	0.36
RandomForest	TF-IDF	0.39	**0.44**	**0.47**	**0.43**	0.43	**0.43**	**0.47**	**0.42**	0.44	**0.44**	0.47	**0.43**
KNN	TF-IDF	0.30	0.3	0.32	0.3	0.30	0.3	0.3	0.3	0.29	0.29	0.3	0.29
LSVM	BERT embed	0.28	0.28	0.29	0.28	0.28	0.28	0.29	0.28	0.29	0.29	0.3	0.29
Decision Tree	BERT embed	0.33	0.33	0.33	0.33	0.35	0.35	0.35	0.35	0.35	0.35	0.35	0.35
Random Forest	BERT embed	0.38	0.38	0.42	0.37	0.40	0.4	0.44	0.39	0.39	0.39	0.43	0.38
KNN	BERT embed	0.30	0.3	0.31	0.29	0.30	0.3	0.3	0.3	0.29	0.29	0.3	0.29
LSVM	Doc2Vec	0.3	0.3	0.42	0.26	0.28	0.28	0.3	0.25	0.29	0.29	0.31	0.26
Decision Tree	Doc2vec	0.33	0.33	0.33	0.32	0.35	0.35	0.35	0.35	0.39	0.39	0.39	0.38
Random Forest	Doc2vec	0.39	0.39	0.43	0.38	0.42	0.42	**0.47**	0.41	0.41	0.41	**0.48**	0.40
KNN	Doc2Vec	0.30	0.3	0.32	0.29	0.30	0.3	0.3	0.29	0.29	0.29	0.3	0.29
LSTM+DENSE	LSTM embed	**0.457**				**0.443**				**0.441**			

Table 2: Résultat après entrainement des modèles

Les résultats du tableau 2 montrent les résultats obtenus par nos modèles sur la tâche de classification multiclasse. On peut observer que parmi les modèles classiques le modèle le plus performant est le random forest. On remarque que l'augmentation de nos données peut permettre par moment de gagner 1 à 2% de précision. Notre modèle LSTM+dense est celui qui performe le mieux avec près de 45% de précision sans augmentation de données. Ce qui est étonnant reste le fait que le modèle perfome mieux sans augmentation qu'avec. Nos intuitions portent sur le fait que les données générées sont encore trop similaires aux données initiales. Une autre explication pourrait être que l'augmentation des données peut introduire du bruit ou des distorsions qui rendent les exemples d'entraînement moins pertinents ou plus difficiles à généraliser. Cependant, si l'on observe les résultats de notre meilleur modèle, nous battons pratiquement tous les modèles avant 2021 (CNN, bi-LSTM, hybrid CNN)(Ali et al., 2022).

Résultats supplémentaires (pour aller plus loin)

D'autres données ont été générées à l'aide de ChatGPT par OpenIA. Les prompts utilisé sont : "Could you give me some facts pronounced by politicians labeled as true, i.e. true information and verified and put them in a Python list ?", "Could you do the same thing but this time generating false facts ?". Elles ont permis de tester l'augmentation de données, l'extraction des caractéristiques et représentation du texte sur des données hors distribution à travers une classification binaire (étiquettes vrai/faux), sans prendre en compte l'historique de véracité des citations précédentes. Les résultats n'ont toutefois pas été concluants, car les données générées sont trop éparses. Nous obtenons des modèles avec une précision autour de 55-60%, ce qui se rapproche d'un classifieur aléatoire. Les données générées par chatgpt sont trop diverses et la distribution est bien trop différente de notre jeu de données actuel qui est bien trop unique pour permettre à nos modèles de généraliser sur des citations sans aucun contexte particulier.

6 Conclusion

Le présent travail a ainsi permis d'observer l'impact de l'augmentation de données par EDA ou paraphrase à l'aide de Bert, sur la classification multiclasse de faits. L'utilisation d'un LSTM après une augmentation de données a été la plus performante, produisant en moyenne 44,27% de justesse. Toutefois, l'augmentation de données n'a eu un impact assez significatif, les résultats sont mitigés.

Ces comparaisons pourraient être affinées en utilisant la base de données LIAR Plus, qui ajoute un attribut "justification" qui explique la motivation de l'étiquette attribuée. Afin d'augmenter les performances tout en prenant en compte les contraintes matérielles, il serait possible de combiner les résultats de modèles simples.

References

Hadeer Ahmed, Issa Traore, and Sherif Saad. 2017. Detection of online fake news using n-gram analysis and machine learning techniques. pages 127–138.

Abdullah Ali, Fuad Ghaleb, Bander Al-rimy, Fawaz Alsolami, and Asif Khan. 2022. Deep ensemble fake news detection model using sequential deep learning technique. *Sensors*, 22:6970.

Leo Breiman. 2001. Random forests. 45.

Sreyasee Das Bhattacharjee, Ashit Talukder, and Bala Venkatram Balantrapu. 2017. Active learning based news veracity detection with feature weighting and deep-shallow fusion. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 556–565.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. 2003. Knn model-based approach in classification. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 986–996, Berlin, Heidelberg. Springer Berlin Heidelberg.

M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China. Association for Computational Linguistics.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents.

Yunfei Long, Qin Lu, Rong Xiang, Minglei Li, and Chu-Ren Huang. 2017. Fake news detection through multi-perspective speaker profiles. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 252–256, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Ndapandula Nakashole and Tom M. Mitchell. 2014. Language-aware truth assessment of fact candidates. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1019, Baltimore, Maryland. Association for Computational Linguistics.

Ray Oshikawa, Jing Qian, and William Yang Wang. 2020. A survey on natural language processing for fake news detection.

J. R. Quinlan. 1986. Induction of decision trees. 1.

Stephen Robertson. 2004. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation - J DOC*, 60:503–520.

William Yang Wang. 2017. "liar, liar pants on fire": A new benchmark dataset for fake news detection. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 422–426, Vancouver, Canada. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Annexes

Contribution des membres

Anne-Sophia : Recherche de propositions. Recherche sur les enjeux du sujet. Explication des problématiques posées par le sujet. Introduction et état de l'art. Description des données. Prétraitement de la base de données et représentation du texte (Doc2Vec et TF-IDF).

Yanis : Recherche de propositions. Recherche sur les bases de données disponibles. Explication du contexte et de l'importance du projet. Prétraitement de la base de données, représentation du texte (Bert représentation), extraction des caractéristiques et application des modèles.

Caroline : Recherche de propositions. Recherche sur les modèles applicables pour le projet. Explication de la méthode adoptée. Description des données et recherche sur l'augmentation de données. Augmentation de données (BERT + EDA).

Ressources Informatiques

Dans notre projet, python est le langage de programmation utilisé, celui-ci est le plus utilisé et conseillé dans la manipulation des données. Nous faisons appel à plusieurs librairies python dans ce projet (Numpy, Pandas, Tensorflow, Scikit-Learn, NLTK, Nlpaug, Sentence-Transformers, textaugment). Numpy et Pandas nous facilitent la manipulation des données grâce aux diverses méthodes

qu'elles présentent. NLTK est une libraire utilisée pour le traitement du langage naturel. Tensorflow et Scikit-Learn nous permettront de créer, d'entraîner et de tester nos modèles. Nlpaug, Sentence-Transformers et textaugment sont utilisées pour l'augmentation des données. Enfin, nous utilisons Jupyter Notebook qui est une plate-forme informatique interactive basée sur le Web qui nous permet de tester notre code via des cellules de codes. Github est utilisé pour le partage et la gestion des versions du projet.