DEFI-AI: Rapport

Team name: France-INSA/ENSEEIHT/VALDOM-UPENDO

Membres:

- **GHOMSI KONGA Serge**
- **KEITA Alfousseyni**
- **RIDA Moumni**
- **SANOU Désiré**
- **WAFFA PAGOU Brondon**

Introduction

Avec les progrès technologiques et l'apparition d'internet, il devient difficile pour les étudiants ou les demandeurs d'emploi de trouver le travail adapté à leurs connaissances et compétences accumulées à l'école ou en cours de travail. En outre, l'entreprise de recrutement doit filtrer manuellement les profils des candidats pour choisir les personnes qui conviennent au poste qu'elle recrute, ce qui prend beaucoup de temps alors que le nombre de candidatures pourrait atteindre des centaines ou des milliers. C'est pourquoi l'objectif du Défi-IA cette année était d'attribuer la bonne catégorie d'emploi (parmi 28 catégories) à une description de poste.
La prédiction d'emploi est une tâche de classification qui utilise plusieurs techniques de machine Learning et de traitement du langage naturel pour essayer de prédire un emploi sur la base des descriptions de poste, y compris les exigences du poste, les connaissances, les compétences, les intérêts, etc. Pour ce projet, nous nous intéressons à la prédiction de la catégorie des emplois à partir des descriptions d'emploi et du sexe.

Dans ce rapport, nous vous présentons dans un premier temps l'implémentation de plusieurs modèles de classification supervisées dont BERT, SVC et LSTM-GRU-CNN avec des *embeding (Glove, FastText)* pré-entrainés. Nous présentons également les raisons de notre choix et les résultats que nous avons obtenus.
Afin d'améliorer l'efficacité, la robustesse et la stabilité de notre prédiction, nous avons opté pour une méthode d'ensemble combinant les modèles implémentés. Nous utilisons la méthode de vote majoritaire. Cette méthode est assez simple et a récemment prouvé son efficacité dans les problèmes de classification des textes [9].

Nos résultats ont montré que le modèle d'ensemble que nous avons proposé a obtenu le meilleur résultat avec un F1-socre de 78%. De plus, nous analysons les résultats expérimentaux pour avoir un aperçu de ce problème afin de trouver de meilleures solutions à l'avenir.

1. Exploration des données

Dans cette étude, nous utilisons des ensembles de données du Défi-AI 2021. L'ensemble de données d'entrainement est composé de 217 197 différentes descriptions de poste associées à 28 différentes catégories de poste. La Figure 1 présente la distribution des données par catégorie.

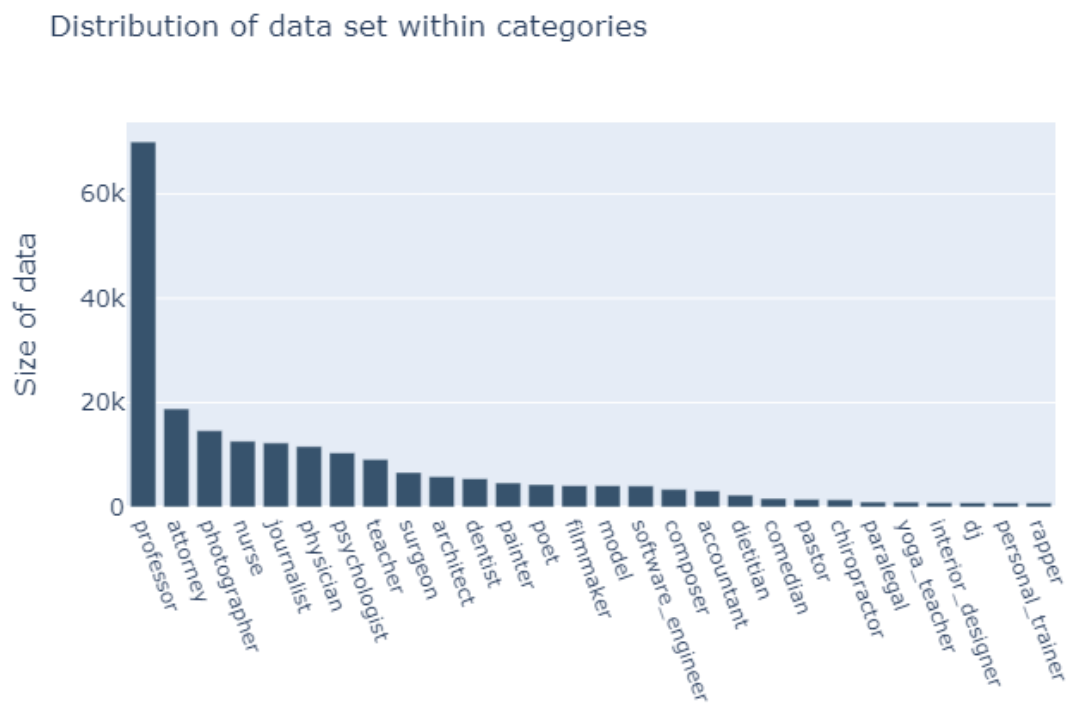

Figure 1: Répartition des données par catégorie

Nous faisons le constat selon lequel l'ensemble des données n'est pas équilibré.

Par exemple, la catégorie ***professor*** a beaucoup plus d'occurrence que les autres.

En conséquence, le vocabulaire des données est très déséquilibré.

2. Modèles

2.1. SVM : Support Vector Machine

Pour le deuxième choix, nous nous sommes tournés vers un modèle de support vector machine a noyau linaire. En effet, un noyau linéaire donne généralement de bons résultats quand on a un grand nombre de features, ce qui est notre cas : un nombre élevé de documents avec un grand nombre de mots.
Le choix du noyau effectué, on devait choisir quelle méthode d'extraction de features appliquée, on a comparé entre le TF_IDF et le Count_Vectorizer en prenant en compte les unigrams et les bigrams « ngram_**range = (1,2) »**. Le résultat observé est que le TF_IDF donnait un meilleur résultat et prenait peu de temps dans la phase d'entrainement.

L'implémentation du modèle a été faite en local, sur jupyter notebook en utilisant la librairie Scikit-learn.

2.1.a- Résultats :

- Temps entrainements : 1.12 min
- Accuracy : 81%
- F1_score : 75%

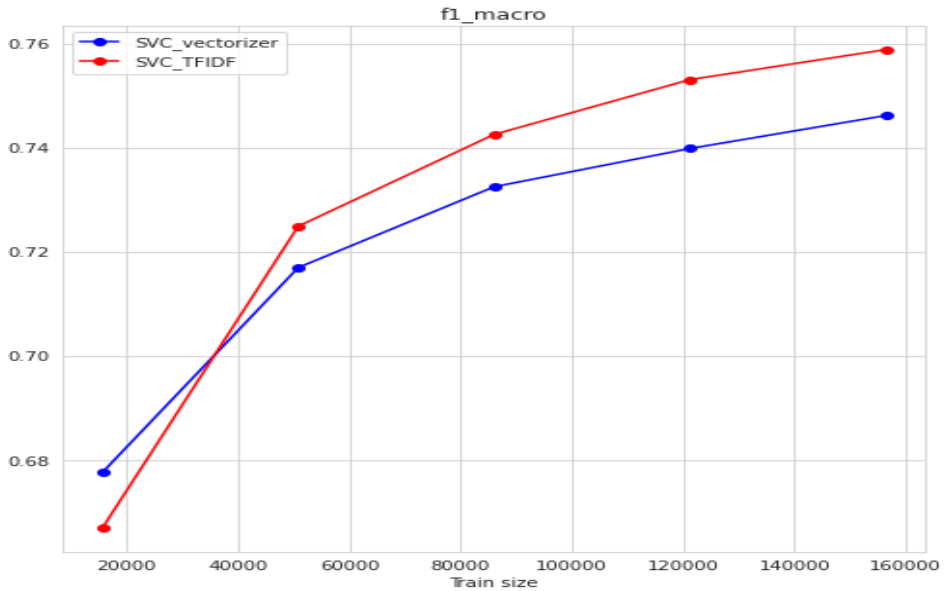

Figure 2 : f1_score svc: count_vectorizer vs tf_idf

0	0.662	0.502	0.571	293
1	0.815	0.744	0.778	818
2	0.681	0.681	0.681	182
3	0.620	0.549	0.582	1881
4	0.756	0.699	0.726	146
5	0.809	0.826	0.817	921
6	0.683	0.766	0.722	2516
7	0.809	0.585	0.679	188
8	0.826	0.719	0.768	1330
9	0.833	0.701	0.761	632
10	0.826	0.621	0.709	161
11	0.784	0.713	0.747	2339
12	0.885	0.699	0.781	342
13	0.737	0.635	0.682	797
14	0.898	0.817	0.855	2532
15	0.745	0.728	0.736	845
16	0.933	0.921	0.927	1096
17	0.856	0.606	0.709	284
18	0.777	0.800	0.788	775
19	0.837	0.910	0.872	13898
20	0.826	0.890	0.857	2878
21	0.815	0.697	0.752	152
22	0.794	0.710	0.750	2151
23	0.944	0.483	0.639	209
24	0.764	0.639	0.696	1162
25	0.811	0.837	0.824	670
26	0.847	0.897	0.871	3779
27	0.874	0.827	0.850	463
accuracy			0.811	43440
macro avg	0.802	0.721	0.755	43440
weighted avg	0.810	0.811	0.808	43440

Figure 3: Rapport de classification SVC

2.2. BERT: Bidirectional Encoder Representations from Transformers

Le premier modèle intégré dans notre stratégie est Bert, ce choix est motivé par les résultats encourageant qu'il retourne et du fait que son champ d'application coïncide parfaitement avec notre sujet.
Il s'agit d'un modèle de représentation de langage qui s'appuie sur des Transformers : Modèle construit sur la base d'encodeur et de décodeurs se basant sur des méthodes d'attention.
Pour l'implémentation, nous avons utilisé la librairie Pytorch et effectuer l'entrainement sur Colab sous GPU.

2.2.a – Résultats :

- Temps entrainement : 71 min (3 epochs)
- Accuracy : 82%
- F1_score : 76 %

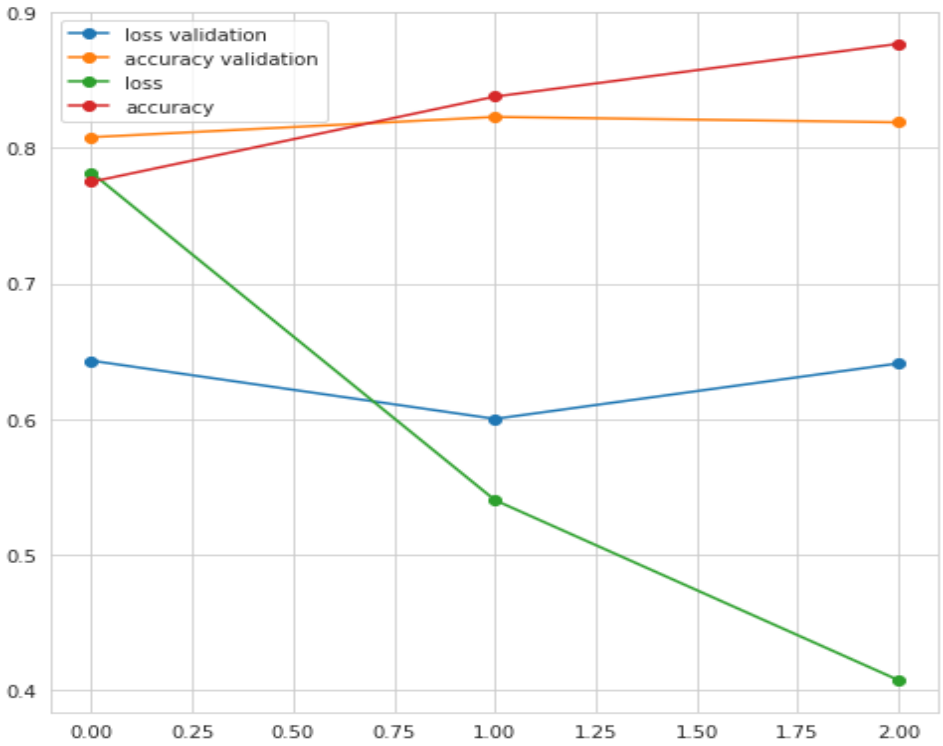

Figure 4: Loss et accuracy pour les phases entrainement et validation par epoch

Le modèle donne de bon résultats et l'accuracy ne diverge pas lors de la phase de validation, on constate aussi grâce au rapport de classification ci-dessous, qu'il n'y a pas de grand écart entre le score de chaque classe du jeu de données.

	precision	recall	f1-score	support
0	0.61	0.59	0.60	293
1	0.89	0.72	0.80	818
2	0.71	0.70	0.70	182
3	0.61	0.58	0.60	1881
4	0.71	0.73	0.72	146
5	0.74	0.86	0.80	921
6	0.77	0.74	0.75	2516
7	0.64	0.68	0.66	188
8	0.81	0.77	0.79	1330
9	0.81	0.71	0.75	632
10	0.65	0.70	0.67	161
11	0.76	0.75	0.75	2339
12	0.75	0.76	0.76	342
13	0.77	0.55	0.64	797
14	0.87	0.84	0.85	2532
15	0.71	0.82	0.76	845
16	0.93	0.94	0.93	1096
17	0.79	0.66	0.72	284
18	0.80	0.81	0.80	775
19	0.87	0.90	0.89	13898
20	0.88	0.85	0.86	2878
21	0.79	0.74	0.76	152
22	0.74	0.77	0.75	2151
23	0.87	0.61	0.72	209
24	0.66	0.69	0.67	1162
25	0.83	0.86	0.85	670
26	0.87	0.89	0.88	3779
27	0.82	0.85	0.84	463
accuracy			0.82	43440
macro avg	0.77	0.75	0.76	43440
weighted avg	0.82	0.82	0.82	43440

Figure 5: Rapport de classification Bert

2.3. Modèles Bi-GRU-LSTM-CNN & Bi-GRU-CNN

Ces dernières années, les modèles de CNN ont montré leur efficacité dans le traitement du langage naturel. Ils donnent de bons résultats pour la classification des textes [1] et pour la détection de discours de haine en ligne [2]. Dans ces modèles, les mots sont généralement représentés dans un espace vectoriel, pour bien exprimer la relation sémantique entre les mots en utilisant des modèle pré-entrainés de *words embdeding* [3] tels que *Glove et FastText,* pour améliorer le résultat prédictif du modèle.

Dans la classification des textes, la combinaison de différents modèles tels que le modèle Bi-RNN [4] et le modèle Bi-LSTM-CNN [5] donne des résultats intéressants. Ces méthodes de classification montrent une bonne optimisation des résultats prédits, de meilleurs résultats que les modèles simples.

Dans cette section du rapport, nous avons mis en œuvre deux modèles de combinaison (Bi-GRU-CNN et Bi-GRU-LSTM-CNN) pour la classification des descriptions de poste. En outre, nous y avons intégré deux modèle pré-entrainés (Glove et FastTex).

2.3.1 Modèle Bi-GRU-CNN

Le modèle Bi-GRU-CNN a été utilisé dans le cadre du problème de prédiction des salaires [6], sur la base des descriptions de poste telles que le contenu du poste, les exigences du poste, le temps de travail, la fonction et le type d'emploi. La Figure 6 présente l'architecture de ce modèle. Les principales composantes sont

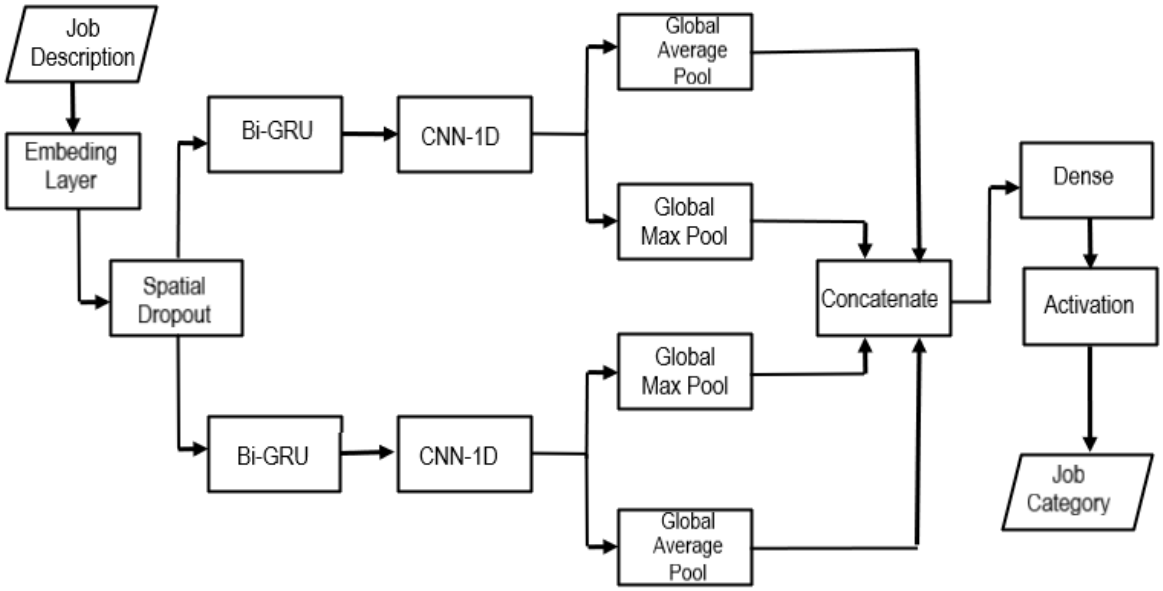

Figure 6: The Bi-GRU-CNN architecture for the job classification

● Couche de représentation des mots : Après avoir obtenu la fréquence des mots de la description de poste et essayé différentes valeurs du nombre maximum de mots à conserver, nous avons choisi l'entrée comme une matrice de 70.000x300 dimensions. En particulier, chaque description de poste comporte au maximum 70 000 mots et chaque mot est représenté par un *embeding* de 300 mots. Nous utilisons un « *pre-training word-level vector* de Glove et FastText » qui est une sorte de représentation des mots pour les modèles de réseaux de neurones profonds.
● Couche CNN-1D : Dans cette architecture, nous utilisons un *1D spatial drop* avec un taux d'abandon de 0,2. Cela permet d'éviter le surajustement et d'obtenir de meilleures généralisations pour ces modèles.
● *Bidirectional GRU* : Comme dans la figure 1, nous construisons deux blocs Bidirectional GRU ayant 128 unités pour chacun, en parallèle, et chaque Gated Recurrent Units (GRU) est sans porte de sortie.

2.3.2 Modèle Bi-GRU-LSTM-CNN

Ce modèle a été proposé par Huynh et al [7] pour résoudre le problème de la détection de la propagande haineuse dans le cadre du *VLSP Share 2019*, et a obtenu la cinquième place au tableau d'affichage du test public. Huynh et al [8] ont obtenu un bon résultat pour prédire le titre du poste en utilisant la classification du modèle Bi-GRU-LSTM-CNN sur la description du poste. En principe, le modèle Bi-GRU-LSTM-CNN a la même structure que le modèle Bi-GRU-CNN, mais au lieu de deux Bi-GRU en parallèle, ce modèle utilise une combinaison de Bi-GRU et de Bi-LSTM (Long Short-Term Memory) en parallèle. Nous avons utilisé deux blocs LSTM bidirectionnels parallèles ayant 128 unités pour chacun. Nous avons utilisé des activations récurrentes *sigmoïdes*.

2.3.3 Expérimentations

A. Pré-processing

Pour le prétraitement des données, nous utilisons les techniques suivantes :
● Conversion des descriptions de poste en chaînes minuscules.
● Segmentation des descriptions de poste en un ensemble de mots
● Représentation des mots en vecteurs avec des ensembles d'insertion de mots pré-entrainés.
Au cours de notre mise en œuvre, nous avons remarqué que la suppression du mot-clé dans les données, diminuait la performance du modèle. Nous avons choisi de ne pas supprimer les mots-clés.

B. Résultats

Dans notre travail, nous avons divisé au hasard l'ensemble de données en trois ensembles différents, dont 20 % pour l'ensemble de test, 80 % pour l'ensemble d'entrainement. Dans l'ensemble d'entrainement, nous obtenons 20 % pour l'ensemble de validation.
Pour évaluer nos modèles, nous utilisons deux mesures telles que la précision et le F1-score.

Dans l'implémentation, après avoir *tuné* les paramètres, nous avons décidé de prendre *learning rate = 1e-3, batch size = 128, epochs = 5, optimizer = Adam.*
Tout d'abord, nous avons implémenté Bi-GRU-LSTM-CNN-Glove dans notre PC, qui présente les caractéristiques suivantes : Processeur : Inter ® Core(™) i7-6500U CPU @ 2.50GHz 2.59GHz, Ram : 16Go, Système : 64 bits.

Avec l'exécution en mode CPU, le modèle donne une *accuracy* de test de 82,11% et un F1-score de 75,95. Mais cela prend beaucoup de temps, environ 12 heures pour toutes les étapes de prétraitement et d'entrainement, et 2 heures pour l'entrainement d'une époque.
Ensuite, nous passons à Kaggle afin d'utiliser le GPU et de réduire le temps d'exécution. Nous implémentons avec les ressources disponibles GPU = 16 GB, CPU = 13GB.

Tableau 1: Résultats expérimentaux de différents modèles

Models	Accuracy	F1-score	Training comp. time (for one epoch)
Bi-GRU-LSTM-CNN-Glove (300)	82.00	**76.11**	144 seconds
Bi-GRU-LSTM-CNN-FastText(300)	**82.05**	75.91	142 seconds
Bi-GRU-CNN-FastText(300)	81.77	75.25	136 seconds
Bi-GRU-CNN-Glove(300)	81.91	75.74	138 seconds
Ensemble	**82.85**	**77.10**	

Nous avons procédé à une série de mises en œuvre et nos résultats expérimentaux sont présentés dans le Tableau 1 et sur la Figure 7. En particulier, nous avons remarqué que le modèle Bi-GRU-LSTM-CNN avec le Glove et FastText embedding est plus performant que le

modèle Bi-GRU-CNN. En particulier, le résultat du modèle Bi-GRU-LSTM-CNN-FastText atteint une accuracy de 82,05 %, et un F1-score de 75,91 %. Nous avons également remarqué que le modèle Bi-GRU-LSTM-CNN-Glove donne un F1-score de 76,11 % et une accuracy de 82,00 %. L'architecture Bi-GRU-LSTM combinée avec CNN a été plus efficace que l'architecture Bi-GRU lorsqu'elle est combinée avec le modèle CNN.

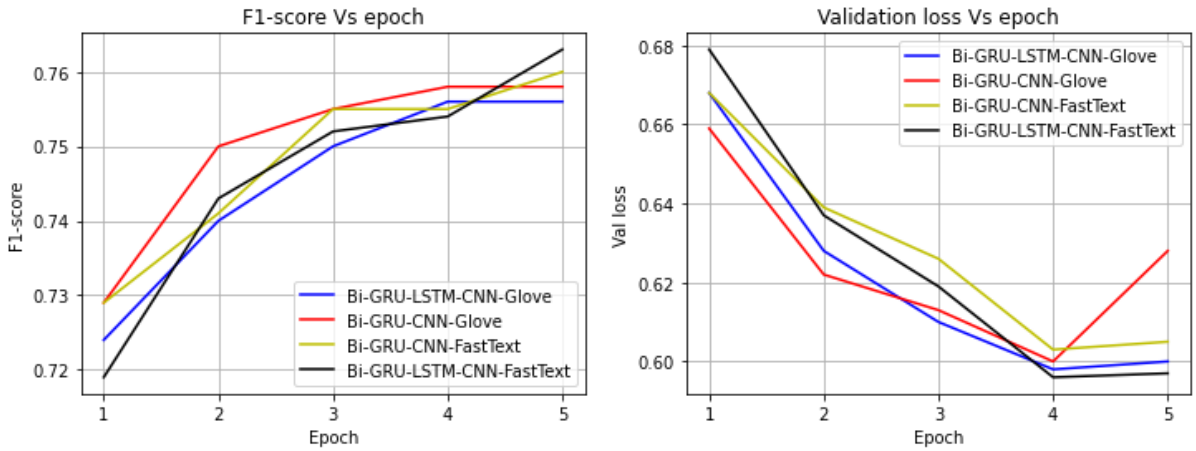

Figure 7 : F1-score on test data and validation loss for different models

Bien qu'il ne soit pas le meilleur modèle sur l'ensemble de tests, le modèle Bi-GRU-CNN combiné avec le modèle pré-entrainés d'Embedding tels que Glove et FastText a également donné des résultats assez positifs. La précision de ces modèles est supérieure à 81 % et le F1-score au-dessus de 75 %.

Enfin, le modèle d'ensemble que nous avons proposé a obtenu les meilleurs résultats et était beaucoup plus élevé que les autres modèles, à savoir 82,85 % pour l'accuracy et 77,10 % pour le F1-score. On peut constater que la méthode d'ensemble donne les meilleurs résultats, mais qu'elle n'est pas vraiment stable.

3. Implémentation du vote majoritaire final

L'objectif ici est d'améliorer le f1_score global de nos prédictions. Sachant que la précision de notre modèle est inférieure à 77 %, nous ne pouvons pas nous fier à un seul modèle. Nous avons donc agrégé les prévisions et donné la priorité au meilleur modèle parmi les trois, à savoir le modèle BERT initial.
Il est à noter que durant la rédaction de ce rapport, la méthode d'ensemble appliquée sur les modèles Bi-GRU-LSTM-CNN et Bi-GRU-CNN combinés au embeding Glove et FastText a donné un F1-score de 77.10% et une accuracy de 82.85% sur l'ensemble de test.

Conclusion

L’objectif de notre étude était de proposer un modèle de classification supervisée des catégories d’emploi à partir des descriptions d’emploi et du sexe. Nous avons implémenté trois différentes grandes méthodes de classification (SVC, BERT, Bi-GRU-LSTM-CNN) et appliqué une méthode d’ensemble de vote majoritaire, qui nous a donné des résultats satisfaisants.

Sur la page Kaggle du Défi, les dernières performances que nous avions étaient :

Score public : 0,78049
Score privé : 0,78013

Sur la base de ces résultats, nous pouvons conclure que notre modèle final n'était pas trop adapté aux données des tests.
Ce défi nous a permis d'explorer les différentes techniques de prétraitement des données textuels, les différents modèles de classification textuels et de se familiariser des outils tels que Keras, TensorFlow ou encore Pytorch.

En guise de perspective, nous aurons pu chercher à rééquilibrer le jeu de données afin d’éviter le possible transfert de biais dans les prédictions. Nous aurions pu également faire une sélection (tuning) de certains paramètres comme la profondeur des couches convolutionnelles dans le modèle Bi-GRU-LSTM-CNN

References

[1] Y. Kim, “Convolutional neural networks for sentence classification,” arXiv Prepr. arXiv1408.5882, 2014
[2] B. Gamback and U. K. Sikdar, “Using convolutional neural networks to ¨classify hate-speech,” in Proceedings of the first workshop on abusive language online, 2017, pp. 85–90.
[3] P. Wang, B. Xu, J. Xu, G. Tian, C.-L. Liu, and H. Hao, “Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification,” Neurocomputing, vol. 174, pp. 806–814, 2016
[4] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” IEEE Trans. Signal Process., vol. 45, no. 11, pp. 2673–2681, 1997
[5] C. Li, G. Zhan, and Z. Li, “News Text Classification Based on Improved Bi-LSTM-CNN,” in 2018 9th International Conference on Information Technology in Medicine and Education (ITME), 2018, pp. 890–893
[6] Z. Wang, S. Sugaya, and D. P. T. Nguyen, “Salary Prediction using Bidirectional-GRU-CNN Model,” Assoc. Nat. Lang. Process., 2019.
[7] T. Van Huynh, V. D. Nguyen, K. Van Nguyen, N. L.-T. Nguyen, and A. G.-T. Nguyen, “Hate Speech Detection on Vietnamese Social Media Text using the Bi-GRU-LSTM-CNN Model,” arXiv Prepr. arXiv1911.03644, 2019.
[8] T. Van Huynh, K. Van Nguyen, N. L. Nguyen and A. G. Nguyen, "Job Prediction: From Deep Neural Network Models to Applications," 2020 RIVF International Conference on Computing and Communication Technologies (RIVF), Ho Chi Minh, Vietnam, 2020, pp. 1-6, doi: 10.1109/RIVF48685.2020.9140760.
[9] A. Onan, S. Korukoglu, and H. Bulut, “A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification,” Expert Syst. Appl., vol. 62, pp. 1–16, 2016