Défi IA 2021

Loïc Rakotoson, Dylan Monfret, Antoine Adam - WeTried
Master Mathématiques Appliquées, Statistique – Parcours Science des Données
Université Rennes 2, Université de Rennes 1, Agrocampus Ouest, ENSAI, INSA de Rennes
Promotion 2021

Défi IA 2020-2021 organisé par l'INSA Toulouse
Projet Science des données
Enseignants encadrants : Yann Soulard & Romain Tavenard, maîtres de conférences

Contents

CHAPTER 1

Introduction - Présentation du concours

Le présent rapport a été rédigé dans le cadre de la cinquième édition du Défi IA organisé par l'INSA Toulouse. Le défi, rassemblant un large panel d'universités de France et de la francophonie, confronte les équipes d'étudiants de ces universités autour de problématiques pouvant être résolues par des méthodes d'apprentissage machine.

Tout comme deux autres organismes universitaires de la région, l'Université Rennes II a pris part à ce concours inter-universitaire en y engageant les étudiants en deuxième année du Master de Mathématique Appliqué, Statistiques (MAS) dont nous faisons partie. Cet évènement intervient dans le processus de formation de l'université rennaise.

Antoine Adam, Dylan Monfret et Loïc Rakotoson formons l'équipe " *We Tried* ", avec l'appui des enseignants-chercheurs et encadrant pour ce projet Yann Soullard et Romain Tavenard, pour répondre à la problématique de cette année.

1. Présentation générale

Pour cette cinquième édition du Défi IA, l'objectif est d'assigner **automatiquement et le plus convenablement possible un métier à une description de métiers**. Dans le cadre de ce projet, nous dénombrons 28 métiers assignables. Les intitulés de métiers sont tous issues de données récoltées par Common Crawl, organisation à but non lucratif recueillant de nombreuses données trouvables sur le web pour les mettre à la disposition de la communauté scientifique. Ces données ont été utilisé lors de la conception du modèle de traitement du langage auto-régressif GPT-3 (plus précisément l'entraînement de celui-ci). Le Défi IA de cette année relève donc à la fois d'un problème de classification en classe multiple et de traitement naturel du langage.

Cette compétition inclus également quelques contraintes concernant la construction des classifieurs :

- 217 197 intitulés labellisés servent à la phase apprentissage, 54 300 non labellisés (sans métier associé) pour la phase validation de nos algorithmes et pour les soumissions.

- Seules les données fournies peuvent être utilisé dans le cadre du concours.
- Et par conséquent, l'ajustement de modèles pré-entraînés doit se faire uniquement avec les données mises à disposition.

Les équipes participantes sont classées dans un premier temps sur le F1-Score des prédictions effectué avec le jeu de validation ; puis les 10 meilleures équipes du classement précédent sont jugées sur l'impartialité de leur soumission vis-à-vis du genre. Et c'est donc ici qu'intervient une métrique quantifiant l'impartialité d'un algorithme : le Disparate Impact.

2. Variable sensible & biais de classifieur

2.1. Introduction du Disparate Impact.

L'équité à l'embauche au regard du genre est une problématique courante dans le monde du travail, et l'introduction de tri préliminaire par apprentissage machine dans le cadre d'embauche n'est pas encore parfaitement au point (comme par exemple le cas des photographies sur les CV pour les personnes de couleurs). Construire un classifieur juste est donc nécessaire.

Nous concernant, un classifieur parfaitement juste serait en mesure de classer les textes donnés sans prise en compte du genre. C'est évidement impossible puisque certains intitulés décrivent des individus, des hommes ou femmes, et sont donc genrés. Avec les professions qui connaissent de fortes disparités de genre (pour diverse raison d'ordre sociologique), il serait risqué de construire des classifieurs implicitement influencés par le genre. Il faut donc trouver un équilibre entre équité et réalité de chaque métier vis-à-vis du genre. C'est là qu'intervient le Disparate Impact.

Le **Disparate Impact** est une métrique qui mesure, comme son nom l'indique, la disparité d'un groupe par rapport à une variable sensible. Pour un groupe donné, le **DI** est égal au rapport entre la proportion d'individu d'un sous-groupe majoritaire sur la proportion d'un sous-groupe minoritaire. C'est une mesure qui oscille donc entre 1 et l'infini, en supposant que les sous-groupes sont de taille non-nuls (cette métrique n'aurait de toute manière pas d'intérêt dans le cas contraire). Par conséquent, le DI vaut 1 s'il dans les proportions des dits sous-groupes sont équivalents.

Avec les classifieurs, nous pouvons approcher cette métrique d'un point de vue probabiliste, et ainsi, pour un classifieur g, un ensemble de données X, une variable cible Y prenant k-modalités possibles $\{y_1, \ldots, y_k\}$ et S comme variable sensible prenant ici 2 modalités (0 pour le groupe majoritaire ou 1 pour le minoritaire), le DI se définit de la sorte :

- Pour le classifieur et une modalité y_i de Y :

$$DI(g, X, S) = \frac{\mathbb{P}(g(X) = y_i|S = 0)}{\mathbb{P}(g(X) = y_i|S = 1)}$$

- Sur les données réelles et une modalité y_i de Y (biais du jeu de données) :

$$DI(y_i, X, S) = \frac{\mathbb{P}(Y = y_i|S = 0)}{\mathbb{P}(Y = y_i|S = 1)}$$

Maintenant, puisque l'on traite près d'une trentaine de classe de Y pour notre problème, on s'intéresse plus au **macro DI**, soit la moyenne des DI pour toutes les modalités possibles $\{y_1, \ldots, y_k\}$:

$$MDI_g = \frac{1}{k} \sum_{i=1}^{k} DI(g, X, S)$$

$$MDI_Y = \frac{1}{k} \sum_{i=1}^{k} DI(y_i, X, S)$$

Mais comme dit précédemment, il y a un juste milieu à trouver entre l'équité et le biais "naturel" des données. Le biais du jeu de données ne doit pas être altérer. Donc l'objectif n'est pas d'arriver à $MDI_g = 1$ de disparité moyenne, mais de réduire au minimum la différence absolue entre MDI_g et MDI_Y. Le classifieur le plus juste serait donc celui minimisant la différence suivante :

$$\Delta_{MDI} = |MDI_g - MDI_Y|$$

2.2. Le Disparate Impact : un faux problème ?

Une quantité Δ_{MDI} qui tend vers 0 impliquerait donc un classifieur juste. Mais, nous verrons au cours de l'étude que l'impartialité n'est pas l'objectif immédiat dans la construction de nos classifieurs, nous dirons que c'est un critère que nous surveillons pour savoir si nos classifieurs sont beaucoup trop partiales. Il y la plusieurs raison à ce choix d'orientation :

- Premièrement, l'objectif du concours qu'est d'intégrer un TOP 10 basé sur le F1-Score des soumissions. La priorité est donc de pousser le F1-Score toujours plus proche de 1.
- Ensuite, dans le cadre du concours, il nous a été assuré par les organisateurs que les vrais disparités pour chaque métier ne sont pas sensiblement éloignés des disparités des données d'apprentissage. Cela signifie qu'en prédisant le plus justement possible sur le terrain du F1-Score, les DI obtenus pour chaque métier ne seront pas trop éloigné de la réalité lors de nous soumissions sur Kaggle.

Maintenant, le sujet des disparités de genre au sein de chaque métier n'as pas été totalement écarté. La disparité sera traitée comme **un problème de données déséquilibrées** avec une partie de **d'augmentation de données** expliquée plus en détail dans la section *Méthodologie*.

CHAPTER 2

Méthodologie

1. Travaux connexes

1.1. Classification supervisée de textes. L'architecture classique pour la classification des textes provenant de descriptions des postes est composée d'une couche d'embedding à la tête d'une couche de convolution [1] ou de couches de récurrentes suivi d'une couche de convolution [2]. Entraînés uniquement sur les données textuelles sur les postes, les modèles performent très bien en atteignant des F1 de 72.

L'architecture des transformers [3] compile, après l'embedding, plusieurs couches d'encodeurs et de décodeurs composés de couches récurrentes. Le modèle est ensuite entraîné sur des phrases où des mots sont masqués, et sur des textes où des phrases sont à deviner.

1.2. Données déséquilibrées. L'algorithme SMOTE [4] est l'outil classique pour gérer les données déséquilibrées en effectuant un sous-échantillonnage des classes dominantes et un sur-échantillonnage des classes sous-représentées en s'aidant des méthodes à noyau. Cet algorithme ne peut s'employer que sur une représentation tabulaire des textes (TF-IDF, occurences, ...) mais ne tient pas compte du contexte. La version WEMOTE [5] applicable sur les embeddings effectue le sur-échantillonnage en prenant en compte la représentation du texte mais ne permet pas encore de faire de meilleures performances par rapport à SMOTE.

Les travaux sur la correction des biais basés sur le genre dans le traitement de language naturel [6] ont démontré que les corrections de logits après les prédictions permettent de réduire le biais certes mais réduisent beaucoup trop les performances. Les solutions en amont sont: la suppression de l'axe du genre dans les embeddings [7] si elle est détectée, l'augmentation des données [8] accompagnée ou non de l'inter-changement des genres en utilisant de la reconnaissance d'entité nommée (NER) pour les noms et les postes, ainsi que l'inversement des genres des mots en s'aidant de l'étiquetage morpho-syntaxique (PosTag).

1.3. Nettoyage des labels. L'algorithme t-SNE [9] permet de visualiser la proximité et la disparité des labels. La représentation des données permet ensuite de visualiser la distribution des labels sur les deux axes principales. L'algorithme WAR [10] permet une régularisation des données en séparant les labels en utilisant la distance de Wasserstein.

2. Travaux effectués

2.1. Pré-traitements.

Nettoyage des labels.

En examinant les données, nous avons relevé des descriptions ambiguës pour un même label. Il s'agit du poste "*architect*" qui comprend des descriptions pour les architectes en constructions et les architectes informatiques. Le premier étant proche du label "*interior designer*", alors que le second se rapproche plus de "*software engineer*".

> *architect*: "He runs a boutique design studio attending clients. His work explores the convergence of human arts and science to give shape to an ever evolving design practice."
>
> *architect*: "He focuses on cloud security, identity and access management, mobility security, and security for Microsoft platforms and solutions."

Pour séparer le label, nous avons opté pour le modèle de l'Allocation de Dirichlet latente (LDA) où le but est d'effectuer un clustering en regroupant les données de ce label par leur sujet, c'est-à-dire la fréquence des mots utilisés. Nous avons choisi la log-vraisemblance pour mesurer la qualité de nos clusters, en cherchant par GridSearchCV les paramètres optimaux pour le nombre de clusters et le learning decay. Nous avons gardé 2 clusters et un learning decay de 0.5 pour le label "*architect*", et avons donc ajouté le label 29 aux catégories pour "*data architect*".
L'opération étant assez coûteuse et n'ayant pas observé plus d'ambiguïtés pour le reste des labels, nous n'avons pas étendu le nettoyage des labels sur l'ensemble des données.

Augmentation de donnée.

Lors de ces travaux, nous avons traité le problème du biais sur le genre comme un problème de données déséquilibrées. En effet, les données sur un poste stéréotypé pour un genre sous représente l'autre genre, et crée donc un déséquilibre par rapport à la mesure sur cette variable.

Nous n'avons pas choisi les algorithmes SMOTE, WEMOTE et la suppression de l'axe du genre dans les embeddings; ces solutions étant dépendantes de la représentation du corpus. Nous avons opté pour la Beyond-Back Translation [11] qui consiste à créer des nuances de mots et de syntaxe en s'appuyant sur les différences d'expressions entre les langues.

La deuxième méthode est l'inversement du genre (GenderSwap[12] [13]). Cependant, nous n'avons pas utilisé les techniques habituelles avec la NER et les PosTag, mais plutôt des embeddings de Word2vec en appliquant à l'échelle d'une phrase l'opération vectorielle dont l'exemple classique est `king - male + female = queen`.
Pour sur-échantillonner le genre sous-représenté pour un label nous avons sélectionné aléatoirement la moitié des données du genre ainsi que le quart du genre opposé. Les traductions ont été faites en `Anglais - Français - Anglais` sur cet échantillon et le GenderSwaping a été appliqué en plus sur les données provenant du quart du genre opposé.

original: "Jane is the pastor's wife and she gets up every morning to light the candles in the church."

beyond back: "Jane is the priest's spouse and she wakes up every morning to light the candles of the chapel."

gender swap: "John is the husband of the priestess and he wakes up every morning to light the candles of the chapel."

2.2. Modèles de classification.

Support Vector Machine Linéaire.
Pour les modèles classiques, nous avons opté pour la représentation TF-IDF du corpus, en sélectionnant ensuite les 133 000[1] meilleurs variables par un test de chi2.

Nous avons donc effectué un benchmark naïf sur les modèles classiques de classifications et nous avons observé `Linear SVC` et `Ridge Classifier` se démarquer. Ces deux modèles ont donc été optimisés chacun par la méthode *Nested GridSearchCV* sur 5 folds internes et 3 folds externes. Le modèle du machine à vecteur support est celui qui a le mieux performé en terme de F1-score et de Disparate Impact. Toutefois, le modèle tel quel ne produit pas de logits à la prédiction, nécessaire aux méthodes d'ensembles qu'on envisage pour la suite.
La méthode `Calibrated CV` donne les probabilités d'obtenir à une classe dans un problème à multi-classe en utilisant une méthode d'ensemble sur un estimateur de base déjà entraîné. Cette calibration a légèrement amélioré les performances de `Linear SVC` lors de la phase de test.

Roberta.

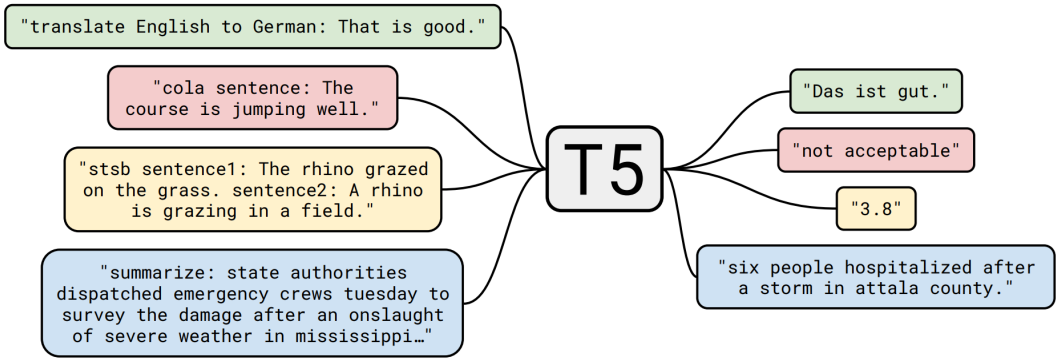
[1] limite imposée par nos machines

Le modèle [14] est un modèle de transformer basé sur BERT. L'architecture est la même, mais certains changements sur le prétraitement des données sont appliquées, comme le masquage dynamique, une taille de batch plus importante, et un encodage BPE plus large. Aussi, ce modèle est entrainé avec plus de passes, et sur un corpus plus important.
De manière générale, RoBERTa (ou d'autres transformers), après quelques epochs de fine-tuning, permettent d'arriver à un F1-score beaucoup plus grand, et surtout ces modèles (surtout ls plus larges avec 300M ou plus paramètres) semblent moins sensible au déséquilibre des classes cibles. Cependant, comme nous le présentons plus en détail plus loin, quelques problèmes d'instabilité commencent à apparaitre sur ces gros modèles.

Electra.

Le modèle [15] est un modèle lui aussi basé sur BERT, mais à la place du token [MASK], les auteurs proposent de remplacer certains tokens par des mots corrompus (créés par un générateur), et de préentrainer le modèle comme un discriminateur. Malheureusement nous avons commencer à nous intéresser à celui là un peu tard. Et puisque le fine-tuning était très long, nous n'avons pas pu l'expérimenter beaucoup même si les résultats étaient prometteurs.

Le modèle générateur T5.
Le modèle T5 est un modèle de transformer encoder-decoder créé par GoogleAI[16] dont le but est d'unifier toutes les tâches de NLP. La motivation principale réside dans le fait que tous les problèmes de NLP peuvent être transformé en problème de Text-to-Text.



Dans notre cas, l'encoder reçoit la description, une séquence d'environ 190 tokens, et le décode en nom de poste, soit un séquence de 4 tokens pour notre problème. L'avantage par rapport aux

modèles de classification précédents réside dans le fait que le modèle obtient aussi des informations provenant de la représentation du label lors de l'entraînement, au lieu d'un unique chiffre. Le revers, c'est l'espace de génération du modèle qui s'étend théoriquement sur tous les mots de la langue anglaise, ce qui donne des labels qui n'existent pas dans nos données mais qui s'en rapprochent. Nous avions par exemples, pour "*yoga teacher*", les dénominations "*pilates coach*" et "*meditation teacher*". Pour re-calibrer les données, les scores de similarités basés sur les synonymes avec `wordnet` aux labels sont calculés sur les sorties générées. Dans notre cas, les labels sont assez disparates et les différentes dénominations des sorties sont rares.

La méthode d'ensemble.

Lors de nos premiers essais avec des architectures transformers, qui était avec des petits modèles, de l'ordre d'une cinquantaine de millions de paramètres, rien n'était à signaler, les résultats était stables d'un essai à l'autre, même avec exactement les mêmes hyper-paramètres. Ce n'est que lorsque nous commençons à utiliser des modèles pré-entrainés beaucoup plus large (quelques centaine de millions paramètres) que les résultats deviennent moins stables. Tout d'abord de temps à autre, durant certains tests, la perte d'entrainement se mettait à remonter d'un coup, avec l'accuracy pouvant passer de 80% à presque 0 en quelques batchs seulement. Une autre instabilité survient dans les résultats, les différences dans le macro f1-score estimé deviennent beaucoup plus grande, entre 0.80 et 0.83.

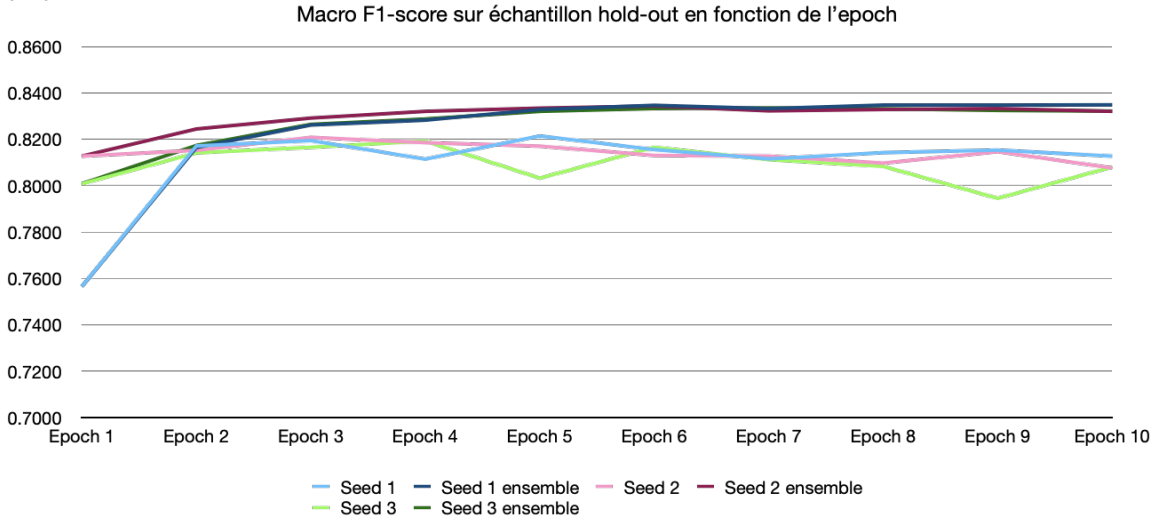
L'une des ressources les plus utile pour cette compétition à été les précédentes compétitions de traitement du language sur Kaggle, notamment Jigsaw Multilingual Toxic Comment Classification et les discussions et présentations des différents membres du top10. Surtout, c'est en lisant les solution de l'équipe gagnante que nous en sommes venu à tester les techniques présentées dans ce chapitre [17]. Cette équipe présente notamment deux articles traitant de l'instabilité des transformers, le premier s'interessant à l'impact de l'initialisation des poids de la dernière couche (de classification) [18], le deuxième essayant de traiter le problème via du bagging [19]. Par manque de temps, mais aussi puisque la solution du premier article apporte de bon résultats, nous n'avons pas tester le bagging de transformers.
Dans cet article, les tests sont fait en monitorant en continu la qualité des graines, c'est à dire en laissant un échantillon de coté. Puis ils choisissent de prendre un ensemble (soft voting) des dix meilleures graines parmi trente. Cependant pour nous, cela n'est pas forcément satisfaisant, car nous n'avons pas envie de nous séparer d'un partie des données. En effet, pour repérer les "bonnes" graines, nous aurions besoin d'un échantillon de validation, car les bonnes graines dépendent des données.

Nous choisissons plutôt d'expérimenter avec un protocole que nous pourrions répéter sur l'échantillon d'apprentissage en entier dans le cas où les résultats sont concluant. Plutôt que de choisir des graines, nous essayons de voir si le fait de prendre l'ensemble de plusieurs graines au hasard, est meilleur qu'une graine toute seule prise au hasard. Ainsi, notre classifier devient :

$$\hat{y} = \arg\max_j \sum_{i=1}^{n} p_{i,j}$$

où $p_{i,j}$ est la probabilité pour la classe j prédite par le i-ème classifier (parmi n graines/classifiers).

Une autre méthode pour essayer de stabiliser les résultats survient durant l'entrainement, plutôt que de choisir une epoch optimale via un echantillon de validation, en surveillant la val-loss, il est possible de faire un prédiction après chaque epoch, et de faire un ensemble de toutes les prédictions (via un soft-voting de nouveau), le résultats en terme de F1-score est bluffant. Sur le graphique si dessus, les série de couleur claires sont les scores des prédictions unique après chaque epoch, les séries de couleur foncées sont les scores de l'ensemble des prédictions des epoch, allant de la 1ère à la i-ème.



Finalement, en prenant ces deux méthodes en même temps, notre ensemble peut-être vu de la façon suivante :

$$\hat{y} = \arg\max_j \sum_{i=1}^{n} p_{i,j,e}$$

où $p_{i,j,e}$ est la probabilité pour la classe j prédite par le i-ème classifier (parmi n graines/classifiers) et à la e-ème epoch.

CHAPTER 3

Expérimentations

1. Entraînement des modèles

Le score `macro-F1` et la différence de Disparate Impact δ`DI` sont les métriques utilisées pour classer les performances des équipes dans la compétition. C'est tout naturellement que nous choisissons donc le score `F1` pour évaluer les performances de nos modèles.

Cependant, la métrique utilisé lors de l'entraînement est l'`accuracy`, ou plus précisément la `SparseCategoricalAccuracy` implémentée par Tensorflow. En effet, la mesure du score `F1`, même avec un callback personnalisé, ne s'effectue que sur le dernier batch vu lors de l'epoch, ce qui donne une information incomplète de la performance du modèle.
Ainsi , pour les modèles avec les couches transformers, nous observons les performances sur l'échantillon de validation par rapport à la fonction de perte `SparseCategoricalCrossEntropy` et gardons l'information donnée par l'`accuracy`.

En particulier, pour DistilBERT et RoBERTa, nous expérimentons rapidement, en particulier sur la learning rate (nous avons tenté de jouer sur la taille des batchs, mais nous étions rapidement limité par la mémoire GPU ou TPU disponible). Pour la learning-rate, il a fallu prendre un peu de temps pour trouver une valeur qui apporte de bons résultats des la première epoch (pour la méthode d'ensemble) car nous voulions être sur de pouvoir sommer dès la première epoch. Finalement pour RoBERTa, nous retenons une learning rate de 1.5e-5. Puisque nous avons une limite de temps d'utilisation des TPUs sur Kaggle, nous ne pouvons pas vraiment faire beaucoup d'expérimentations. Mais nous remarquons rapidement que les résultats pour le modèle large avec 350M de paramètres tournent autour de 0.82, cependant ils sont assez instables. Nous avons donc tenté les deux techniques d'ensemble (sur les epochs et sur les graines aléatoire) sur seulement 5 graines aléatoire différentes et 5 epochs pour chacune de ces graines. Ce qui fait en tout 25 prédictions différentes. Cependant, même avec aussi peut le résultat semble assez spectaculaire par rapport au précédents essais : le fait de prédire après chaque epoch permet de toujours être au dessus de 0.83 de F1-score, et le fait de le faire avec des graines aléatoires différentes permet de se rapprocher de 0.84.

Pour le modèle T5, l'`accuracy` est plutôt calculé par `SparseTopKCategoricalCrossEntropy` qui calcule la fréquence d'apparition d'un token dans les k prédictions, où $k = 5$.

Seul la `Calibrated Linear SVC` a été optimisé en observant le score `F1`

Pour la phase de test, δ`DI` a été introduite dans la `classification report`. Pour comparer les différents modèles, nous observons donc le score `F1` et δ`DI`

2. Evaluation des performances

Les résultats sont données dans la table 1, où les 5 premières mesures sont calculées sur l'échantillon de test et les 2 dernières sur la soumission pour la compétition.

Notre premier modèle, `Calibrated Linear SVC` performe +2.9 points par rapport à la baseline et réduit de 4 fois le δ`DI` par rapport à la régression logistique de la démonstration. Il s'agit du seul modèle qui tourne rapidement sur CPU et est le moins lourd.

Le modèle `TextCNN` et le réseau *Full-connected* qui suivent ont été proposés par Luiza Sayfullina et al.[20] et utilisé ensuite par Tin Van Huynh et al. en enlevant `FastText` pour prédire à partir des descriptions. Nous avons ajouté une couche Bi-LSTM après la couche d'Embedding, ce qui a amélioré le score de de 10 points même si les performances ne dépassent pas celles du `Calibrated Linear SVC`.

Le modèle `RoBERTa NLI` suit la même architecture que `RoBERTa` mais entraîné sur *Adversarial NLI*[21] par Facebook. En pratique, la gestion des mots absents du dictionnaire et les fautes d'orthographes est améliorée. On y retrouve les mêmes avantages que `Fast Text`. Entraîné sur les données augmentées, le modèle baisse le score de -0.2 mais possède le δ`DI` le plus bas.
Le modèle `Aug Ensemble RoBERTa` est l'application du modèle de *soft voting* sur 20 modèles de `RoBERTa` et 24 de `RoBERTa NLI` entraînés sur les données augmentées. Nous nottons que si le score `DI` n'a pas baissé comme attendu, le score `F1` a été amélioré de +0.1 point.

modèle	**accuracy**	**precission**	**recall**	**F1-score**	**δDI**	**score**	**DI**
baseline	-	-	-	72.56	-	-	-
naive Logistic Regression*	78.88	80.57	66.01	71.51	2.11	-	-
Calibrated Linear SVC	80.09	77.64	73.41	75.16	0.51	75.38	4.26
LSTM TextCNN	77.10	69.30	65.02	66.72	1.22	-	-
Sequential	78.68	76.78	70.07	72.93	0.63	-	-
DistilBERT	82.42	79.59	74.40	76.63	0.43	79.12	**3.94**
RoBERTa + EP	86.42	83.29	82.08	82.54	0.66	-	-
RoBERTa + Ensemble	**87.27**	83.98	83.12	83.44	0.65	83.79	4.00
RoBERTa NLI + EP	87.02	83.64	82.77	83.13	0.77	-	-
RoBERTa NLI + Ensemble	87.18	83.71	83.06	83.34	0.53	-	-
Aug RoBERTa NLI	86.96	83.28	82.87	82.98	**0.34**	-	-
Aug Ensemble RoBERTa	87.12	**84.06**	**83.65**	**83.85**	0.61	**83.89**	4.01
Electra + EP	86.60	83.77	82.11	82.86	-	-	-
T5 + EP	85.31	82.54	78.18	80.31	0.50	-	-

TABLE 1. Résultats des modèles

CHAPTER 4

Sample Chapter

1. Sample Section

1.1. Sample SubSection.

CHAPTER 5

Résumé

Dans ce rapport, nous présentons nos essais et résultats pour le Défi IA 2021 organisé par l'INSA Toulouse. Le problème consiste en une classification de textes parmi 28 classes, les classes étant des métiers. Les solutions sont jugées selon un critère de performance par rapport à la classification (le F1-score) mais aussi par rapport à la *fairness, mesuré par le disparate impact qui vise à mesuré à quelle point une solution est biaisé vis à vis du genre.*

Nous exposons tout d'abord la problématique du concours, ainsi que les métriques sur lesquelles nous sommes jugées, puis passons à la description de toutes les méthodes que nous avons essayé. Nous commençons bien entendu par des méthodes statistiques classiques, après avoir vectorisé nos textes (que ce soit via une procédure tf-idf, ou un simple de comptage des mots). Le résultat en terme de score se situe à 0.75 en utilisant une machine à support vecteur. Cependant un problème apparaît dans le fait que certaines classes soit beaucoup moins représentées que d'autres, et que nous arrivons moins bien à les prédire. Nous commençons donc à ce moment à nous interesser à des méthodes de rééquilibrage des données comme SMOTE. Cependant, dans le même temps, nous commençons à tester des méthodes de transfer-learning via les architectures transformers, les résultats sont tout de suite meilleurs, mais plus le modèle pré-entrainé devint gros, plus le résultat est instable. C'est pourquoi, après nous être renseigné à propos de ce problème, que nous choisissons de faire un ensemble de plusieurs prédictions, à travers un soft-voting classifier. Ces prédictions sont obtenu avec le même modèle pré-entrainé (RoBERTa large) que nous fine-tunons plusieurs fois avec différentes graines aléatoires. Enfin, nous essayons d'améliorer le disparate impact via RoBERTa NLI, car ce modèle était celui qui avait le disparate impact le plus bas lors de nos tests (tout en baissant un petit peu le F1-score). Malheureusement, lors de la soumission, l'effet inverse ce produit.

Références

1. *Real-Time Resume Classification System Using LinkedIn Profile Descriptions*, S Ramraj and V. Sivakumar and Kaushik Ramnath G

2. *Job Prediction: From Deep Neural Network Models to Applications*, Tin Van Huynh and Kiet Van Nguyen and Ngan Luu-Thuy Nguyen and Anh Gia-Tuan Nguyen

3. *Attention Is All You Need*, Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin

4. *SMOTE: Synthetic Minority over-Sampling Technique*, Chawla, Nitesh V. and Bowyer, Kevin W. and Hall, Lawrence O. and Kegelmeyer, W. Philip

5. *WEMOTE - Word Embedding based Minority Oversampling Technique for Imbalanced Emotion and Sentiment Classification*, Tao Chen and Qin Lu and R. Xu and Bin Liu and J. Xu

6. *Mitigating Gender Bias in Natural Language Processing: Literature Review*, Sun, Tony and Gaut and al.

7. *Gender Bias in Contextualized Word Embeddings*, Jieyu Zhao and Tianlu Wang and Mark Yatskar and Ryan Cotterell and Vicente Ordonez and Kai-Wei Chang

8. *Improving Neural Machine Translation Robustness via Data Augmentation: Beyond Back-Translation*, Li, Zhenhao and Specia, Lucia

9. *Visualizing Data using t-SNE*, Geoffrey Hinton and Laurens van der Maaten

10. *Wasserstein Adversarial Regularization (WAR) on label noise*, Bharath Bhushan Damodaran and Kilian Fatras and Sylvain Lobry and Rémi Flamary and Devis Tuia and Nicolas Courty

11. *Improving Neural Machine Translation Robustness via Data Augmentation: Beyond Back-Translation*, Li, Zhenhao and Specia, Lucia

12. *Gender Bias in Coreference Resolution: Evaluation and Debiasing Methods*, Zhao, Jieyu and Wang, Tianlu and Yatskar, Mark and Ordonez, Vicente and Chang, Kai-Wei

13. *Gender Swap*, https://github.com/valleyviolet

14. *RoBERTa*, Yinhan Liu et al. 2019

15. *Electra*, Kevin Clark et al. 2020

16. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*, *Colin Raffel and Noam Shazeer and Adam Roberts and Katherine Lee and Sharan Narang and Michael Matena and Yanqi Zhou and Wei Li and Peter J. Liu*

17. *1st place solution overview*, *Chun Ming Lee et rafiko1*

18. *Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping*, *J. Dodge et al. 2020*

19. *Bagging BERT Models for Robust Aggression Identification*, *Julian Risch and Ralf Krestel 2020*

20. *Domain Adaptation for Resume Classification Using Convolutional Neural Networks*, *Luiza Sayfullina and Eric Malmi and Yiping Liao and Alex Jung*

21. *Adversarial NLI: A New Benchmark for Natural Language*, *Yixin Nie and Adina Williams and Emily Dinan and Mohit Bansal and Jason Weston and Douwe Kiela*