

## **Défi IA 2021**

Loïc Rakotoson, Dylan Monfret, Antoine Adam



# Contents

Chapter 1. Introduction - Présentation du concours	3
1. Présentation générale	3
2. Variable sensible & biais de classifieur : introduction du Disparate Impact	4
Chapter 2. Méthodologie	6
1. Travaux connexes	6
1.1. Classification supervisée de textes	6
1.2. Données déséquilibrées	6
1.3. Nettoyage des labels	6
2. Travaux effectués	7
2.1. Pré-traitements	7
2.2. Modèles de classification	8
Chapter 3. Sample Chapter	12
1. Sample Section	12
1.1. Sample SubSection	12
Chapter 4. Sample Chapter	13
1. Sample Section	13
1.1. Sample SubSection	13
Chapter 5. Sample Chapter	14
1. Sample Section	14
1.1. Sample SubSection	14
Références	15

## 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 événement 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 de nos algorithmes :

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

- Seules les données fournies peuvent être utilisées 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 sont classées dans un premier temps selon le F1 score des prédictions sur le jeu de validation, puis, parmi les 10 meilleures équipes de ce premier classement, sur l'impartialité du classifieur vis-à-vis du genre. C'est ici qu'intervient une métrique quantifiant l'impartialité d'un algorithme : le Disparate Impact.

## 2. Variable sensible & biais de classifieur : 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'a pas encore amélioré les choses (notamment concernant les photographies sur les CV pour les personnes de couleurs). Il faut donc construire un classifieur juste. Le classifieur parfaitement juste idéal serait à même de classer les textes donnés sans prise en compte du genre. Ce qui est impossible puisque certaines descriptions décrivent des personnes, et ainsi ces descriptions peuvent être intrinsèquement liées au genre. Et puisque certaines professions connaissent de fortes disparités de genre (pour diverse raison d'ordre sociologique), il serait alors 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 mesurant 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 minoritaire sur la proportion d'un sous-groupe majoritaire. C'est une mesure qui oscille entre 1 et l'infini, en supposant que les sous-groupes sont de taille non-nuls. 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  $k$  valeurs possible  $\{y_1, \dots, y_k\}$  et  $S$  comme variable sensible prenant ici 2 modalités (0 majoritaire ou 1 minoritaire), le DI se définit de la sorte :

- Pour le classifieur et une modalité  $y_i$  de  $Y$  :

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

- Sur les données réelles et une modalité  $y_i$  de  $Y$  :

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

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

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

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

Mais comme dit précédemment, il y a un juste milieu à trouver entre l'équité et la réalité des métiers, ou du moins 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 :

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

## 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 <sup>1</sup> ou de couches de récurrentes suivi d'une couche de convolution <sup>2</sup>. Entraînés uniquement sur les données textuelles sur les postes, les modèles performant très bien en atteignant des F1 de 72.

L'architecture des transformers <sup>3</sup> 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 <sup>4</sup> 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, occurrences, ...) mais ne tient pas compte du contexte. La version WEMOTE <sup>5</sup> 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 langage naturel <sup>6</sup> 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 <sup>7</sup> si elle est détectée, l'augmentation des données <sup>8</sup> 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 <sup>9</sup> 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 <sup>10</sup> 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 <sup>11</sup> 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<sup>12 13</sup>). 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 au hasard 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 GenderSwapping a été appliqué 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<sup>1</sup> 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.*

---

<sup>1</sup>limite imposée par nos machines

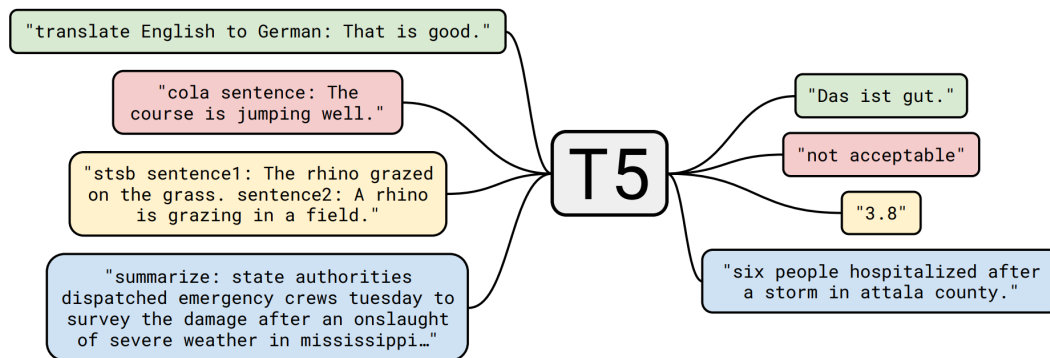
Le modèle <sup>14</sup> est un modèle de transformer basé sur BERT. L'architecture est la même, mais certains changements sur le prétraitement des données sont appliquées, comme le masquage dynamique, une taille de batch plus importante, et un encodage BPE plus large. Aussi, ce modèle est entraîné avec 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 les 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 apparaître sur ces gros modèles.

*Electra.*

*Le modèle générateur T5.*

Le modèle T5 est un modèle de transformer encoder-decoder créé par GoogleAI<sup>15</sup> 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 une 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 avons 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 étaient 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-entraînés beaucoup plus large (quelques centaines de millions de paramètres) que les résultats deviennent moins stables. Tout d'abord de temps à autre, durant certains tests, la perte d'entraînement 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 utiles pour cette compétition a été les précédentes compétitions de traitement du langage sur Kaggle, notamment [Jigsaw Multilingual Toxic Comment Classification](#) et les discussions et présentations des différents membres du top10. Surtout, c'est en lisant la solution de l'équipe gagnante que nous en sommes venu à tester les techniques présentées dans ce chapitre <sup>16</sup>. Cette équipe présente notamment deux articles traitant de l'instabilité des transformers, le premier s'intéressant à l'impact de l'initialisation des poids de la dernière couche (de classification) <sup>17</sup>, le deuxième essayant de traiter le problème via du bagging <sup>18</sup>. Par manque de temps, mais aussi puisque la solution du premier article apporte de bons résultats, nous n'avons pas testé le bagging de transformers.

Dans cet article, les tests sont faits en monitorant en continu la qualité des graines, c'est à dire en laissant un échantillon de côté. 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'une 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.

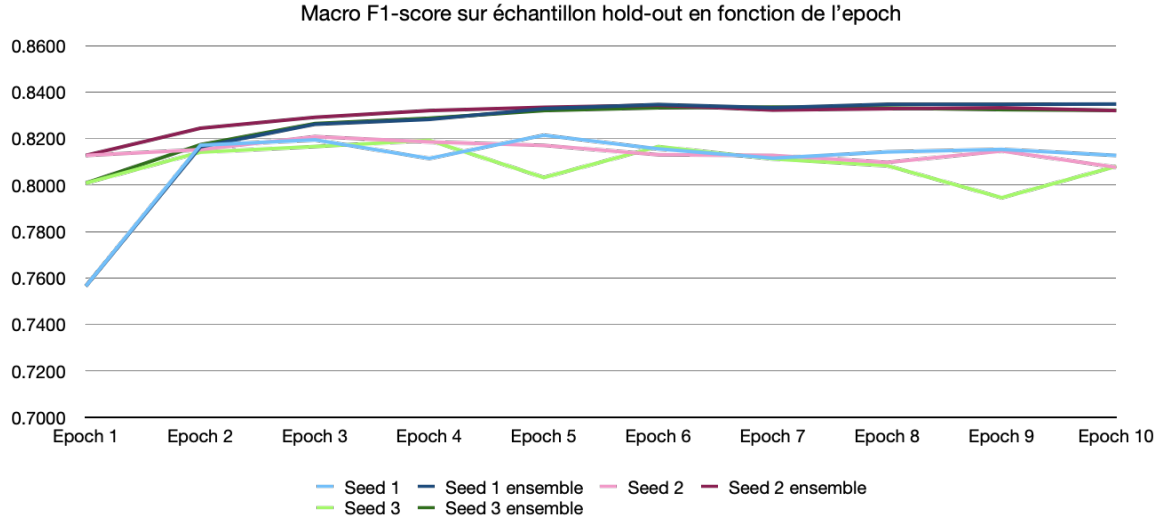
Nous choisissons plutôt d'expérimenter avec un protocole que nous pourrions répéter sur l'échantillon d'apprentissage en entier dans le cas où les résultats sont concluants. Plutôt que de choisir des graines, nous essayons de voir si le fait de prendre l'ensemble de plusieurs graines au hasard, est

meilleur qu'une graine toute seule prise au hasard. Ainsi, notre classifieur 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 classifieur (parmi  $n$  graines/classifieurs).

Une autre méthode pour essayer de stabiliser les résultats survient durant l'entraînement, plutôt que de choisir une epoch optimale via un échantillon de validation, en surveillant la val-loss, il est possible de faire une prédiction après chaque epoch, et de faire un ensemble de toutes les prédictions (via un soft-voting de nouveau), le résultat en terme de F1-score est bluffant. Sur le graphique ci-dessus, les séries 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 classifieur (parmi  $n$  graines/classifieurs) et à la  $e$ -ème epoch.

## CHAPTER 3

# Sample Chapter

### 1. Sample Section

#### 1.1. Sample SubSection.

## CHAPTER 4

# Sample Chapter

### 1. Sample Section

#### 1.1. Sample SubSection.

## CHAPTER 5

# Sample Chapter

### 1. Sample Section

#### 1.1. Sample SubSection.

## 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. **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



16. [1st place solution overview](#), Chun Ming Lee et rafiko1
17. [Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping](#), J. Dodge et al. 2020
18. [Bagging BERT Models for Robust Aggression Identification](#), Julian Risch and Ralf Krestel 2020