Défi IA 2021

Loïc Rakotoson, Dylan Monfret, Antoine Adam

Contents

Chapter 1. Sample Chapter 4
1. Sample Section 4
1.1. Sample SubSection 4

Chapter 2. Sample Chapter 5
1. Sample Section 5
1.1. Sample SubSection 5

Chapter 3. 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. Ce que nous avons fait 7
3. Ensemble de Roberta 7
3.1. Pourquoi utiliser plusieurs fois la même architecture ? 7
3.2. Impact de l'initialisation et de l'ordre des données 7
3.3. Ensemble de prédiction après chaque epoch 8

Chapter 4. Sample Chapter 9
1. Sample Section 9
1.1. Sample SubSection 9

Chapter 5. Sample Chapter 10
1. Sample Section 10
1.1. Sample SubSection 10

Références 11

CHAPTER 1

Sample Chapter

1. Sample Section

1.1. Sample SubSection.

CHAPTER 2

Sample Chapter

1. Sample Section

1.1. Sample SubSection.

CHAPTER 3

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 pour les noms et les postes, ainsi que l'inversement des genres des mots en s'aidant de l'étiquetage morpho-syntaxique.

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 un régulatisation des données en séparant les labels en utilisant la distance de Wasserstein.

2. Ce que nous avons fait

3. Ensemble de Roberta

3.1. Pourquoi utiliser plusieurs fois la même architecture ? 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 grand, comme on le voit sur la figure ci contre...

Afficher le boxplo des 15 macro f1-score (en cours de calcul)

3.2. Impact de l'initialisation et de l'ordre des données. 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 [11]. 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) [12], le deuxième essayant de traiter le problème via du bagging [13]. 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.

Comme nous le voyons sur la figure, les résultats sont assez éparpillés, la seule différence dans chaque étant la graine aléatoire utilisée pour initialiser les poids (et pour l'ordre d'arrivée des données durant l'entrainement). 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). Le résultats de plusieurs essais est sur la figure ci contre.

Afficher la figure mdr

3.3. Ensemble de prédiction après chaque epoch. Les résultats

work in progress, un graphique avec l'écolution du score n fonction des epochs

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

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

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