Défi IA 2021

Loïc Rakotoson, Dylan Monfret, Antoine Adam

Contents

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. Etudes existentes

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 [1]. 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) [2], le deuxième essayant de traiter le problème via du bagging [3]. 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

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

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.