Altegrad Challenge Report

Théo Viel
Logistic Regression Baseline
MVA, ENPC
theo.viel.enpc@gmail.com

Abstract

We aim at classifying nodes of a sub-graph of the French web graph. These nodes correspond to domains, and directed edges between two nodes indicate hyperlinks between them. Nodes come along with the text scrapped from the pages. My approach is two fold : I first use state of the art transformers to extract powerful features on the text, and then reuse these features alongside with the graph structure of our data to generate the final predictions. The documented code will be made publicly available on GitHub[1].

1 Introduction

The problem studied here is a text classification one, in which each text corresponds to a node of a graph. In fact, we are using a sub-graph of the French web graph where nodes correspond to domains, and directed edges between two nodes indicate hyperlink between domain pages. Therefore, the problem is adapted to both graph approaches and NLP approaches. I chose to use transformers to extract features on the texts, and then use them as node features to build a classifier base of the features of a node and of those of its nearest neighbours.

Therefore, we tackle an 8-class text classification problem, the classes and their distribution is represented figure 1. We have 2125 labelled texts for training, and the class distribution is really unbalanced. The metric we were asked to optimise is the multi-class Loss and 560 texts are used for testing.

Recently, transformers [Vaswani *et al.*, 2017] have been dominating most natural language processing benchmarks. Results shown in the paper introducing BERT [Devlin *et al.*, 2018] greatly overcame those of previously existing approaches, and it is therefore natural to leverage such architectures for the task. Pretrained models are easily usable and made available by *HuggingFace*'s transformer library, enabling to use transfer learning for tasks such as the one here. As our texts here are mostly in French, it is better to use models pretrained on French data. The two best solutions then are CamemBert [Martin *et al.*, 2019] and FlauBert [Le *et al.*, 2019].

Camembert showed more convincing results, so more focus was put on this model. It is based on the RoBerta [Liu *et al.*, 2019] architecture and pretrained on OSCAR [Ortiz Suárez *et al.*, 2019], a multilingual corpus scrapped by the *ALMAnaCH Inria* team. It achieved state of the art in French part-of-speech tagging, French named entity recognition and French natural language inference.

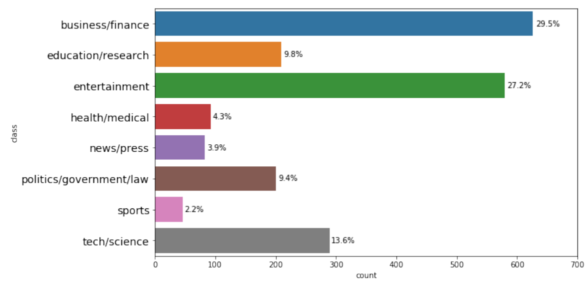

Figure 1: Class distribution of the training data.

2 Modelling

2.1 Data Preparation

I decided to go for a quite aggressive cleaning strategy, as our texts were scrapped from the web. The following steps were applied :

- Removing links
- Cleaning numbers
- Cleaning apostrophes
- Cleaning consecutive characters
- Removing the "alternate" word which appeared a lot at the beginning of some texts
- Removing too long words ($>$ 25 letters)
- Cleaning spaces and removing backslashes special characters

It also appeared that some texts were not in French, I used the *Google Translate API* to translate the texts in the training and testing data to French. Although I did not use this step for all of my models.

[1]https://github.com/TheoViel

2.2 Input Data

CamemBert already has its own word embeddings, and the tokenizer provided by *HuggingFace* does almost all the job. Given a sentence, its tokenized version is represented figure 2. The tokens are then converted to their ids in the vocabulary and fed to the embedding. CamemBert allows for a maximum sentence length of 512, which is the one we use.

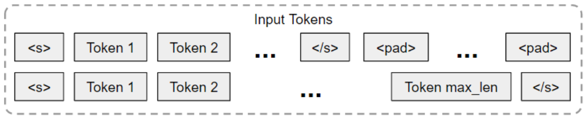

Figure 2: Tokenized sentence.

To perform augmentation, we can choose not to use the first word of the text as the first token, but one further in the text. Usually, most of the information is at the beginning of the text. However, given that our data has few labelled samples, I used this augmentation at train time for some of my models : A text of length l will be considered starting from token i with a probability $\frac{l-i}{L}$ where L is the normalisation term : $L = \sum_{i=0}^{l} l - i = \frac{l(l+1)}{2}$.
This augmentation worked at train time, and added diversity to the models, but did not work at test time.

2.3 Architecture

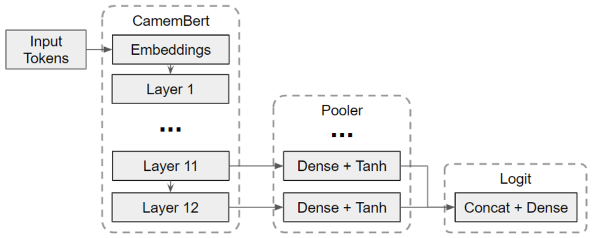

Figure 3: Scheme of the implemented architecture.

As mentioned, I use the pretrained CamemBert model, which I fine-tune on our task. To do so, I plug a pooler on the n last layers of the transformer. The Pooler works the following way :

- For each layer, either average the extracted sentences, or use the representation of the first token.
- Feed them to a dense layer of c units with a *tanh* activation.

Each of the output of features the dense layer is concatenated. We then obtain a vector of size $n \times c$, which is fed to the logit, i.e. a dense layer of size the number of classes (8). In the end, I had better results using $c = 768$, as 768 is the orignial CamemBert hidden size, and using $n > 1$ did not really help. Selecting either the first vector and the averaging them yield good results though.

2.4 Training Pipeline

The optimiser used is AdamW [Loshchilov and Hutter, 2017], with $\beta = (0.5, 0.999)$ and no weight decay. Naturally, I went for the cross-entropy loss.

The training in split into two steps. We first train layers that are not pretrained, i.e. the pooler and logits, using a relatively high learning rate (10^{-3}) with a linear scheduling for two epochs. As the transformer is frozen, we can use a batch size of 64. Then, we fine-tune the whole model for 2 epochs with a batch size of 6, using an advanced learning rate policy described bellow. The hardware used is a single *Nvidia RTX 2080 Ti*, for a maximum sentence length of 512, it cannot fit bigger batches than that, and using gradient accumulation did not really help.

Overall, run times are reasonable : about 4 minutes per fold. However, extracting text features for the whole graph is a bit long.

2.5 Learning rate

For both training steps, we use a linear scheduling, decreasing the learning rate to 0. The mentioned learning rate is the one we start from.

In fact, transformers are very sensitive to the learning rate, the idea is to use a lower learning rate for layers closer to the output, during the second part of the training. The last CamemBert layer is given a learning rate of 3×10^{-5}, and we decay this learning rate by 0.95 at each layer. Considering the scheme 3, layer i has a learning rate of $3 \times 10^{-5} \times 0.95^{12-i}$, with the embeddings being $i = 0$. The pooler and logit blocks are given a learning rate of 10^{-4}.

2.6 Leveraging the graph structure

The idea is straight-forward, use the features extracted by the transformer to embed the nodes. I directly used the 8-dimension vector corresponding to the probability of each class, as the 768-sized features appeared to be too large. For each node, it is fed with the k-nearest neighbours for both incoming and outgoing edges. We then obtain a feature vector of size $8 + 2 \times k \times 8$. This feature is fed to a gradient boosting algorithm, to obtain the final label probabilities. Better performances were obtained with $k = 1$ or $k = 2$.

3 Results

3.1 CamemBert Models

We report table 1 the local cross-validation scores. A stratified 5-fold scheme was used. 5 models were produced, as I played with 3 parameters :

- Whether to translate or not texts that were not in French
- Whether to apply data augmentation or not
- Whether to use the average of the sentence features or only the first vector

CamemBert significantly outperformed FlauBert and Multilingual BERT in my early experiments, so I decided to focus on this architecture only. Single models achieve approximately 0.97 on the public leaderboard, with a quite high instability. They perform reasonably well as it took me two months to outperform the original 0.96270 I achieved using my second submission, and this submission was still first on public leaderboard 3 days before the deadline.

Id	Trad.	Augment.	Pooling	Loss
1	False	False	First Token	1.151
2	False	True	First Token	1.151
3	True	False	First Token	1.137
4	True	True	First Token	1.144
5	True	False	Average	1.136

Table 1: Performances of the transformers for different settings.

3.2 Graph structure

We report table 2 the benefits of leveraging the graph structure on our results. This strategy gives a 0.03 boost on average. This improvement shows that indeed the problem benefits from using the underlying graph, and has more value than one simply consisting of fine-tuning deep learning models on the texts.

Id	Before	$k = 1$	$k = 2$
1	1.151	1.119	1.117
2	1.151	1.099	1.095
3	1.137	1.095	1.093
4	1.144	1.112	1.110
5	1.136	1.102	1.102
Blend	1.123	1.092	1.093

Table 2: Performances after leveraging the graph structure.

3.3 Submission

For the final submission, we report table 3 the performance on the public leaderboard using the blends of the 5 models, i.e. taking the the average of the predictions. The two strategies consist of blending before or after applying the graph based model. Blending improves the results by a small margin, but greatly improves stability of the models. Considering that the test set only consists of 500 samples, it is easier to maintain a high score on both public and private test sets with a blend that with a single model.

Strategy	k	CV	LB
Blend before	1	1.092	0.937
Blend before	2	1.093	0.948
Blend after	1	1.083	0.938
Blend after	2	1.082	0.954
Blend after	2	1.082	0.954
Blend after *	1	**1.081**	**0.935**

Table 3: Performances after leveraging the graph structure.

This table highlights the low correlation between the public leaderboard and my cross-validation scheme. Models used with $k = 2$ neighbours perform worse on the leaderboard despite having an approximately as good cv.

To further boost the results, it is possible to ignore texts that were not correctly parsed they are quite easy to detect as their length is relatively low, and they contain keywords such as ”error”, ”denied”, ”404” and so on. Removing these texts when looking for nearest neighbours resulted in the best local validation score and best leaderboard, using $k = 1$, as reported on the last line of the table 3. Finally, the confusion matrix of the best model is reported 4.

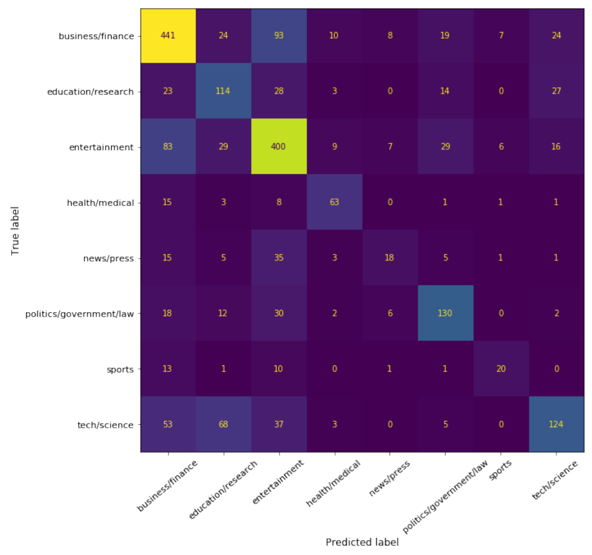

Figure 4: Confusion matrix of the best obtained model.

4 Exploration tracks

Provided that we actually have a lot of text data, this leaves a lot of room for unsupervised approaches. Although I never tried, it is actually possible to fine-tune the language model on a given corpus, using the same training scheme as the authors did. This enables the model to adapt to the context of the french websites, which I believe would definitely help given that we have few labelled samples. Pseudo-labelling can also be looked into, although I am not sure it will perform that well for my models are not good enough to produce accurate pseudo labels. Still, such labels could be used for further pretraining the models. Once again, the idea is to help the transformer adapt to our corpus before moving to the fine-tuning on the training data.

A flaw of my approach is that the training is not end-to-end. Indeed, it would be better (although more complicated) to directly use the graph knowledge when training the transformers. I also expect my hardware not to be enough to back propagate several texts in a transformer architecture.

5 Conclusion

I presented my approach that achieved a 0.935 (rank #1) loss on the public leaderboard. It consists of using transformers to extract powerful text features alongside with a rather simple use of the graph structure to predict the final class probabilities. Ultimately, I consider my approach to be rather straightforward, most of my work consisting of correctly fine-tuning CamemBert. There is still room for improvement as my use of the graph structure is rather simple.

References

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[Le *et al.*, 2019] Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoît Crabbé, Laurent Besacier, and Didier Schwab. Flaubert: Unsupervised language model pre-training for french, 2019.

[Liu *et al.*, 2019] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[Loshchilov and Hutter, 2017] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

[Martin *et al.*, 2019] Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoît Sagot. CamemBERT: a Tasty French Language Model. *arXiv e-prints*, page arXiv:1911.03894, Nov 2019.

[Ortiz Suárez *et al.*, 2019] Pedro Javier Ortiz Suárez, Benoît Sagot, and Laurent Romary. Asynchronous Pipeline for Processing Huge Corpora on Medium to Low Resource Infrastructures. In Piotr Bański, Adrien Barbaresi, Hanno Biber, Evelyn Breiteneder, Simon Clematide, Marc Kupietz, Harald Lüngen, and Caroline Iliadi, editors, *7th Workshop on the Challenges in the Management of Large Corpora (CMLC-7)*, Cardiff, United Kingdom, July 2019. Leibniz-Institut für Deutsche Sprache.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.