Advanced Machine Learning for Text and Graph Data Data Challenge 2019/2020: French Web Domain Classification

ABBES Siwar, ABED Dina, KARRAY Elyes
M2 Data Science, Université Paris-Saclay
Team name: SDE
{elyes.karray, dina.abed, siwar.abbes }@polytechnique.edu

Abstract

Text based methods and Graph based methods on web pages document for classification tasks are of many studies.
In this report, we implement different prepossessing strategies to deal with our data and we explore different methods of machine learning and deep learning to solve a web domain classification problem. We present all the methods deployed and the best scored method.

1 Introduction

The work is about a multi-task classification of the French web domain. We have 8 different categories: business/finance, education/research, entertainment, health/medical, news/press, politics/government/law, sports and tech/science. We are provided of two types of data: the graph data where nodes correspond to domains. A directed edge between two nodes indicates that there is a hyperlink from at least one page of the source domain to at least one page of the target domain and a text data in which we have the extracted text of the pages with their label.

We first used different methods to get the encoding and add new features for both our two types of data and then we used, in particular, three methods of classification: XGBOOST, CNN and CamemBERT.

2 Preprocessing

To deal with our data (text data and graph data) and make them ready for the models, we did a whole work of cleaning, feature extraction and embedding.

2.1 Text feature extraction

Focusing on the text data to predict the category, we try to extract more explanatory variables of our texts. The added features are the following:

- Number of hashtags in the text
- Number of users tagged
- Number of numerical values
- Numbers of emojis
- Number of words in a text
- average length of the words of the text

2.2 Text cleaning

Always dealing with the text data, cleaning it with all the "noisy" characters is necessary. To do, we tried to : Lowercasing all words, Removing non ASCII characters, Removing all the single characters hanging between two spaces, Removing all the punctuation, Removing stop-words, Removing numerical characters,Removing other non alphabetic characters.

2.3 Lemmatization

Lemmatization designates a lexical treatment given to a text for its analysis. This processing consists in applying to the occurrences of lexemes subject to inflection (in French, verbs, nouns, adjectives) a coding referring to their common lexical entry ("canonical form" recorded in the dictionaries of the language, most commonly), which is designated by the term lemma.
To do so, we used the spaCy package of python, which is an open-source library for advanced Natural Language Processing (NLP) in Python.

2.4 Word cloud

In this section, we present the word cloud that we had for text after preprocessing from both categories: health/medical and sports.

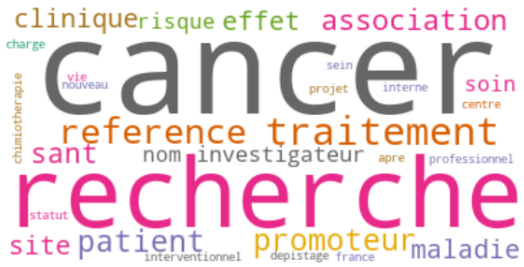

Figure 1: Word cloud for a health/medical text

Figure 2: Word cloud for sport text

2.5 Word Embedding

For word embedding, we used French word embeddings models. It is about pre-trained word2vec models for French. Their format is the same initial binary format proposed with word2vec v0.1c. [Fauconnier, 2015] We used the embeddings trained on the French web (raw, no lemmatization/POS tagging) of dimensionality 200 and a count cutoff of 100.

2.6 Document Embedding

To obtain the document embedding, we trained a TF-IDF model on the corpus of texts to obtain the frequency terms for our vocabulary. We then compute the document embeddings as averages of the word embeddings weighted by the frequencies of each term in the document.

2.7 Graph features

We tried to incorporate features obtained from the graph using two approaches:

- For each node, we computed the inverse distances between it and each labeled node from the training set. We then grouped these distances by label and computed their mean and minimum, hoping that this would give us meaningful information about the proximity of the node to each group.
- We used DeepWalk to get node embeddings [Perozzi *et al.*, 2014].

None of these approaches improved the overall performance of our first model.

3 Models Used

To solve our classification problem we tested three main models. one simple machine learning algorithm: XGBOOST and two deep Learning approaches: CNN and CAMEMBERT.

3.1 XGBOOST

Simply, XGBoost (like eXtreme Gradient Boosting) is an optimized open source implementation of the gradient boost tree algorithm.
We are speaking of a model aggregation method. The idea is therefore simple: instead of using a single model, the algorithm will use several which will then be combined to obtain a single result.[Tianqi Chen, 2016]
the algorithm works sequentially. Unlike for example the Random Forest. this will make it slower of course but it will especially allow the algorithm to improve by capitalization compared to previous executions. He therefore begins by building a first model which he will of course evaluate (we are of course supervised learning). From this first evaluation, each individual will then be weighted according to the performance of the prediction. This what we call self-improvement. We used this algorithm on the previously obtained document embeddings. Optimal hyperparameters were found with cross-validation.

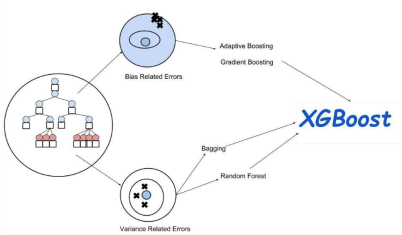

Figure 3: XGBoost architecture

3.2 CNN

A CNN uses a system comparable to a multilayer perceptron, but designed to reduce the number of calculations. The structure of a CNN consists of a succession of layers: an input layer, an output layer and a hidden layer composed of numerous convolutional layers, grouping layers, fully connected layers and normalization layers.

For our model, we used a an architecture very similar to Lab 3 of the course. We initialized the token embedding layer with the embeddings of [Fauconnier, 2015]. The optimal number of epochs was obtained with cross-validation.

3.3 CamemBERT

Nowadays, pretrained language models are super useful when dealing with text data. However, most of them are trained with english data or with data in multiple languages which make hard to use them on other lnguages.
Camemebert is a French version of the Bi-directional Encoders for Transformers (BERT). [Louis Martin, 2019]
CamemBERT was pre-trained on a French-speaking corpus and with different hyper-parameters discovered and tested for the first time by the Facebook team:

- CamemBERT chooses the words to predict dynamically, that is to say, not during the pre-processing of the input data, but during the forward pass, by randomly masking certain words in a sequence.
- It uses a different batch size: 8000 against 256 in the case of BERT.
- CamemBERT has only one pre-training objective: prediction of the ”hidden words” of a sequence.

We used the implementation of CamemBERT in the `simpletransformers` library as it proposes a simple pipeline for classification.

3.4 Other tricks

To overcome the dirtiness of our text database (some pages contained little to no text, others contained error pages) and

make sure that the two latter models don't get overconfident, we used label smoothing, as presented in [Müller *et al.*, 2019]. Instead of using hard labels (represented by one-hot encodings), we use soft probability vectors, where the probability of the true class is 0.9 and the others are equiprobable. This slightly increases our cross-validation and test scores for the final models.
A last trick we used before submission is to stack the predictions of the three models. It does improve the score because the loss is a convex function of the logits. This was done manually and improved our score from 1.2 to 1.13.

4 Results

Below is a tabular containing the results that we had for testing the different models described in the previous section.

Model	XGBoost	CamemBERT	CNN	stacked
Cross val	1.32	1.31	1.29	-
Test	1.21	1.20	1.19	1.13

5 Code description

This project is composed mainly in three directories:

- `models`: where we can find the different models we used.
- `utils`: where we find util functions for data loading and preprocessing.

We made scripts to make our script more usable. You could run the following steps after preparing the `data` folder:

- `install_deps.sh` to install dependencies using this command: `$ sh install_deps.sh`
- `cross_val.py` in order to run cross-validation (change model in cross_val.py) using this command: `$ python cross_val.py`
- `write_submission.py` to write submissions with the CNN model using this command: `$ python write_submission.py`
- `write_submission_stacked.py` in order to write submissions with the stacked models using this command: `$ python write_submission_stacked.py`

6 Conclusion

To sum up, this project was very interesting both on the informative side and the way we address data science problems side.
On an informative point of view, this project was interesting since we were able to discover lots of new algorithms such as XGBOOST, CNN and CamemBERT.
On an operational point of view, this project was really helpful and it though us to try several models but we not to try to use fancy models that we do not know that well. The project was a bit blurred at first, but later on, when we started to try several methods and we started to have a better understanding of the project, the changed our approach to deal with the problem and we had good results.

References

[Fauconnier, 2015] Jean-Philippe Fauconnier. French word embeddings, 2015.

[Louis Martin, 2019] Pedro Javier Ortiz Suárez Yoann Dupont Laurent Romary Éric Villemonte de la Clergerie Djamé Seddah Benoît Sagot Louis Martin, Benjamin Muller. Camembert: a tasty french language model. *arXiv:1911.03894v1e*, November 2019.

[Müller *et al.*, 2019] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? *CoRR*, abs/1906.02629, 2019.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652, 2014.

[Tianqi Chen, 2016] Carlos Guestrin Tianqi Chen. Xgboost: A scalable tree boosting system. *arXiv:1603.02754v3*, March–June 2016.