French web domain classification

Saif Eddine Ghribi [1] , **Mohamed Skander Hellal**[1] , **Ramzi Charradi**[2]
[1]M2 Datascience
[2]M2 MVA
{saif.ghribi, mohamed.hellal, ramzi.charradi}@telecom-paristech.fr,

Abstract

Like every year the Advanced learning for text and graph data class is concluded by a data challenge that should allow to use in practice what was taught during the class. This year, the goal of this competition is to predict the categories of a subset of french web domains given both the text of the web page and a graph representing relations between the web pages.

1 Team name and members

Our team name for this challenge is **EPT**. This team is made by **Saif Eddine Ghribi**, **Mohamed Skander Hellal** who are both following the M2 Datascience Master at Ecole Polytechnique and **Ramzi Charradi** who is following the MVA master at ENS Cachan.

2 Introduction

The goal of this challenge is to solve a web domain classification problem. In other words, the goal is to assign to a web page a class among eight possible classes of web domains given two types of data :

- Graph 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.
- Text extracted from all the pages of each domain.

A subset of these domains were manually classified, this might have induced some errors in annotation. Furthermore, We see a number of text files that were empty, this maybe was an error in the data extraction process. Thus, the graph contains some isolated nodes with no links and corresponding to an empty content. All of this make the classification a non trivial and challenging task.

In this report, we explain the method we proposed to tackle the above problems, starting from an extensive preprocessing on the texts until using some recent and state-of-the art machine learning models dedicated for graph and text classification tasks.
We also report the results of the methods we tried using Multi-class Loss and accuracy of the test set (25% of the provided data split randomly). Finally, we give the best configuration and method that yielded the best results on the kaggle leaderboard.

3 Text preprocessing

The preprocessing of text is done through 3 steps:

- Tokenization : Since the input text is read as a string ,tokenization is mandatory to recognize words.A token will be assigned to every word.
- Filtering Stopwords : Given a set of french stopwords provided by [1] ,these words will be deleted from the text as they have no significant information.
- Keeping French words only : Using the enchant python library ,only french words recognized by this library will be kept.Thus stemming won't be used in preprocessing.

We have used stemming but the performance dropped drastically.We can explain this decrease by the fact that the shift of meaning of two words with the same stem could be important depending on the subject.

As the labels represent websites themes,we have used k-core decomposition in order to extract the keywords per text. However, following this approach the performance obtained was poor.
This poor performance can be explained by two reasons :

- The keywords extracted do not match the label theme (For example a text with label health has most the keywords with entertainment theme).
- K-core method does not take into account semantic relationships between the words.

4 Improving the text baseline model

The text baseline model was a starting point to discover the text data.TF-IDF was used to determine the vocabulary and logistic regression was used to predict the labels of test data.
The idea of this section is to use random search in order to find the best hyperparameters and to predict better the test labels ie lower test loss. The Random search was applied on the following hyperparameters :

- Logistic regression hyperparameters

- C : Inverse of regularization strength
- penalty : L1 or L2 penalties
- max_iter : maximum iterations of the logistic regression

- TF-IDF hyperparameters
  - max_features
  - min_df
  - max_df

```
Best estimator:
LogisticRegression(C=1.4696969696969697, class_weight=None, dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=30000, multi_class='auto', n_jobs=None,
                   penalty='l2', random_state=None, solver='lbfgs', tol=0.0001,
                   verbose=0, warm_start=False)

Best normalized gini score for 2-fold search with 5 parameter combinations:
-3.7689934340204365

Best hyperparameters:
{'penalty': 'l2', 'max_iter': 30000, 'C': 1.4696969696969697}
```

Figure 1: Best hyperparamters of the text baseline model

As for TF-IDF,the best hyperparameters were max_features = 3000 ,max_df = 1000 and min_df=10.Thanks to these new hyperprameters , the test loss dropped to 1.15.

5 Transfer Learning

5.1 Transformers : Bert, XLM

Transformers are a type of neural network architecture that have been gaining popularity. Transformers were recently used by OpenAI in their language models, and also used recently by DeepMind for AlphaStar. Thay are defining the state-of-the art in many natural language processing tasks. This was mainly the motive for us to use transformers for the classification.

We used BERT (Bidirectional Encoder Representations from Transformers) [3] and XLM (cross-lingual language models) [2] with the huggingface PyTorch library **Transformers** to quickly and efficiently fine-tune a model for classification. The reason why we chose specifically those two models among a bunch of other strong models like XLnet and GPT is because hugging face provides some pretrained models on 102 languages in the case of Bert and on the concatenation of English and French wikipedia for XLM. Knowing that our web pages are mainly in french and contain also some English, we judged that these two models would most likely be the best.

The process is the same for the two models. We take the pretrained model, add an untrained layer of neurons on the end, and train the new model for our classification task. However, the text should be prepared in a specific way beforehand.

First, The text must be split into tokens, then these tokens must be mapped to their index in the tokenizer vocabulary. We precise that the tokenization must be performed by the tokenizer included with the corresponding model in transformers library. This is because each model has a specific fixed vocabulary and a particular way of handling out-of-vocabulary words. Second, we add special tokens to the start and end of each sentence and Pad and truncate all sentences to a single constant length. One limitation of the Bert and XLM models of transformers library is that there is a maximal supported length for each sentence before tokenization which is equal to 512. This pushed us to think about how we could extract the most useful words from the text in order to satisfy this condition of 512 words. Thus, we tested taking the top 512 words based on their frequency in the text, and we tested taking the words from the top core using the k-core method, The results between these two methods is nearly the same with a small advantage in the favor of the first method.

Afer padding, we must choose how much words would we feed to the network from the tokenized sequences. This is an important hyperparameter that affects both the speed of the training and the performance of the model. The length we used is 100 words which yielded the best results.

The final processing step is computing the attention masks. The masks simply makes it explicit which tokens are actual words versus which are padding. Now that the input data is properly formatted we feed it to the model.

	loss	accuracy
XLM	1.32	0.59
Bert	1.07	0.62

Table 1: Performance of the models on the test set

Th following table resumes the performance of Bert and XLM on the test set.

Both models were trained for 4 epochs, with a batch size of 16, categorical cross entropy as loss, and adam as optimizer.The learning rate was set initially at $1e-5$ and we used a cosine strategy ie. We create a schedule with a learning rate that decreases following the values of the cosine function with several hard restarts, after a warmup period during which it increases linearly between 0 and 1. This strategy which is provided by the transformers library can be described by Figure 2. It helps avoiding some local minimas to get the best out of the moodel.

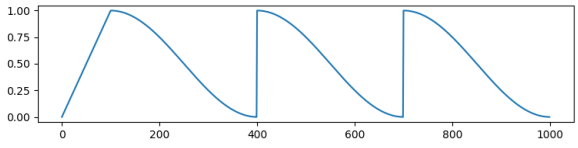

Figure 2: Evolution of the learning rate (taken from the documentation of huggingface)

5.2 FastText model

FastText is a project that provides word vectors(representations) of many languages.Fortunately ,the french words representations are also available("cc.fr.300.bin").Our input is now the representation of a text using words representation given by FastText.Once again the preprocessed text of training and test data are used.Thanks to the function get_sentence_vector provided by FastText ,the input is now a matrix where rows are the

output of the get_sentence_vector of every train and data preprocessed text. The shape of the sentence representation is 300 features.A neural network of a CNN layer ,dropout layer, and an 8-neuron last layer with softmax as activation ,is used as output.

```
Model: "sequential_93"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_93 (Conv1D)           (None, 297, 300)          1500
_________________________________________________________________
dropout_108 (Dropout)        (None, 297, 300)          0
_________________________________________________________________
flatten_93 (Flatten)         (None, 89100)             0
_________________________________________________________________
dense_109 (Dense)            (None, 8)                 712808
=================================================================
Total params: 714,308
Trainable params: 714,308
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: FastText model neural network architecture

Once again ,a set of hyperparamters must be tuned which are the number of kernels and the size of kernel of the CNN layer ,the dropout coefficient and the activation functions. Using the random search to find the best model led to reduce the test loss to 1.22.However, blending models of different hyperparameters ie taking the mean prediction of these models reduced the loss test loss to 1.07.

6 Graph Based Models

6.1 Graph Convolutional Networks

Graph Convolutional Networks [5] are a variant of Convolutional Networks that operate on graph structured data.Graph Convolutional Networks can be performed on text and graph at the same time.
As the dataset is composed of text and graph, our inputs will be defined as follows :

- Adjacency matrix : It is an N $\times$ N matrix where N is the number of nodes.
- Text feature matrix : It is an N $\times$ D matrix where N is the number of nodes and D is the number of features.
- Labels matrix : It is an N $\times$ E matrix with N is the number of nodes and E is the label vector dimension.

In order to train the model,we defined a subgraph formed only by the training nodes.From the subgraph obtained, we have derived the adjacency matrix. The Text feature matrix is extracted using TF-IDF with max features equal to 3000.

6.2 Graph Attention Networks

Graph Attention Networks [**at**] are a variant of Neural Networks that can operate on graph structured Data.The main idea of Graph Attention Networks is based on stacking different layers in which the nodes are able to take in hand their neighborhoods features.
In order to use Graph Attention Networks, we are going to keep the same inputs defined in subsection 8.1 using subgraph of training data and TF-IDF to extract Text feature matrix :

- Adjacency matrix : It is an N $\times$ N matrix where N is the number of nodes.
- Text feature matrix : It is an N $\times$ D matrix where N is the number of nodes and D is the number of features.
- Labels matrix: It is an N $\times$ E matrix with N is the number of nodes and E is the label vector dimension.

6.3 Neural Message Passing

Neural Message Passing is a propagation message schemes that is used for semi-supervised classification tasks based on graph structured data.Neural Message Passing defines a propagation scheme that can be used with Neural Networks to obtain predictions from nodes features. This method is based on Graph Convolutional Networks and personalized Pagerank.
Graph Convolutional Networks and PageRank combination can define a personalized propagation scheme called APPNP (Approximation of personalized propagation of neural predictions) [4].
This algorithm ensures the possibility of teleporting to the root node,so encoding the local neighborhood for every root node.Therefore, the algorithm ensures the locality towards the root node and the possibility of extracting information from the neighborhood via the propagation scheme.

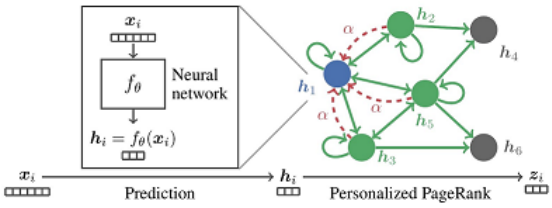

Figure 4: Illustration of Approximation of personalized propagation of neural predictions

As the method is based on Graph Convolutional Networks and PageRank, our inputs will be defined as follows:

- Subgraph : Graph formed by the training texts as nodes
- Text feature matrix : It is an N $\times$ D matrix where N is the number of nodes and D is the number of features.

The Text feature matrix is extracted using TF-IDF.

6.4 Results

Graph Convolutional Networks, Graph Attention Networks and Approximation of personalized propagation of neural predictions have given the following results:

	loss	accuracy
GCN	2.22	0.36
GAT	2.13	0.37
APPNP	1.89	0.3

Table 2: Performance on validation data

The results obtained show that the graph based models are not able to capture the information.
The poor performance obtained can be explained:

- A node can be linked with nodes with different labels.

- Using TF-IDF as text feature extractor is not suitable for our classification.

Further improvements can be performed on these models more precisely the text representation, for example we can use Bert or FastText embeddings as text embeddings before training the model.

7 Best model

7.1 Toward best model : Post Processing

The preprocessing function returned errors for 30 rows in the test data ,these 30 websites are whether empty or have text like Figure 4(websites like 18965,12970,8759,26547,...)

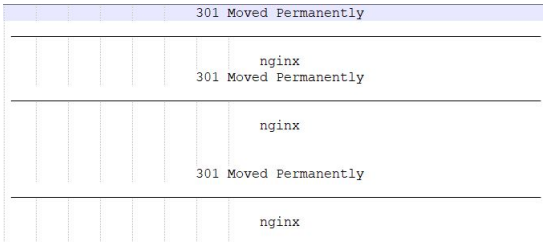

Figure 5: Example of empty websites

The model will assign randomly the classes to these websites.A solution of this problem would be to bring the predictions from the graph model or replace these random predictions with the frequency vector of classes since the test data are randomly taken and they have the same distribution as the training data.

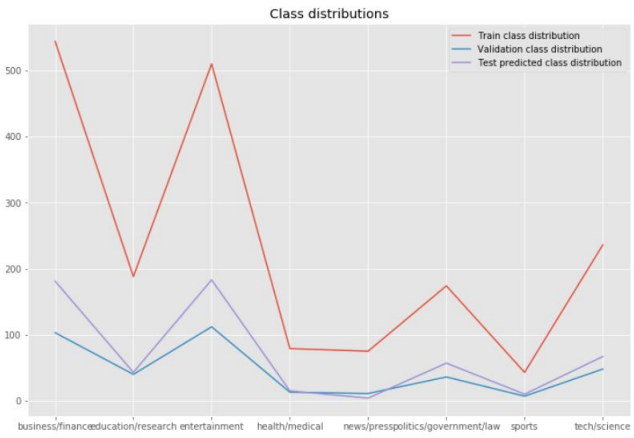

Figure 6: Train and test data class distribution

Figure 5 shows that the predicted distribution of fastText model and the train data is similar.Thus replacing predictions of empty websites with the class frequency vector dropped the test loss from 1.04 to 1.03 .

7.2 Blending Models

Finally, the final file submitted to kaggle that yielded the best results was obtained using ensembling. We took the best three models so far, namely : Bert, Fasttext and enhanced tfidf and we assemble them to generate the final predictions. We used the formula : $pred = 0.25 * Bert + 0.7 * Fasttext + 0.05 * tfidf$. You can generate this file using the notebook called $classification\_text$, However, this may require installing some libraries like transformers, a sufficient space on the disk for the french vectors of Fasttext and finally GPU in order to run successfully the Bert model. We mention also that we run the above methods separately, so when you run the whole notebook, you may encounter some problems regarding the RAM space.Thus, we recommend reducing the batch size of the Bert model or to run separately each method.

8 Conclusion

This contest was an interesting opportunity to apply various methods and algorithms on text and graph data.There are also very promising ideas to improve the results especially working on a model that can understand both French and English efficiently since many websites have English words Also, a further study on graph especially tuning graph deep learning methods would be very interesting.

References

[1] "French NLTK". In: Github link.

[2] Alexis Conneau. Guillaume Lample. "Cross-lingual Lan-guage Model Pretraining". In: (2019).

[3] Kenton Lee Jacob Devlin Ming-Wei Chang. "Pre-training of Deep Bidirectional Transformers for Language Understanding," in: (2018).

[4] StephanG unnemann. Johannes Klicpera Aleksandar Bojchevski. "Graph Neural Net-works meet Personalized PageRank". In: (2018).

[5] Max Welling. Thomas N. Kipf. "Semi-Supervised Classification with Graph Convolutional Networks". In: (2016).