French web domain classification: Altegrad Challenge

Al Houceine Kilani , Naoumi Salmane
ENS Paris Saclay: Master MVA
houcinekila@gmail.com, salmane.naoumi@gmail.com,

Abstract

This project was done in the framework of the course ''ALTEGRAD'' taught by M. VAZIRIGIANNIS, during the first semester of the MVA master 2019-2020. Our team's name is **KILANI — NAOUMI**.

1 Introduction

The goal of this challenge is to solve a web domain classification problem. We were given a sub-graph of the French web 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. Furthermore, We were provided with the text extracted from all the pages of each domain. A subset of these domains were manually classified into 8 categories and split between a training set and a test set. Our task was to predict the categories to which the domains of the test set belong.

2 Data Description

In this challenge, we were given :

- A sub-graph of the French web graph where nodes correspond to domains. It has 28,002 vertices (domain ids) and 319,498 weighted, directed edges(Existence of hyperlinks from some pages of the source domain to some pages of the target).
- A file containing the text of all the web pages of each domain
- The training set corresponds to 1,994 labeled domain ids. One domain id and label per row. Also, the test set contains 560 domain ids of which the category must be predicted. One domain id per row and the multiple classes are ***business/finance, education/research, entertainment, health/medical, news/press, politics/government/law, sports, tech/science***

3 Data analysis

We dispose of a data set that incorporates both graph and text structures which makes the problem more challenging and interesting. Also, the first remark is that there is a class imbalance as illustrated in Figure 1. In fact, business and entertainment domains are representing more than 45% of the whole data set.

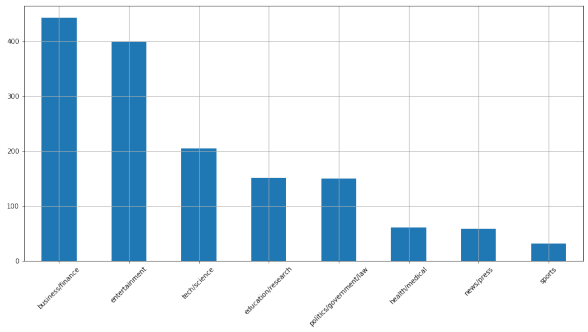

Figure 1: Class imbalance

Also, when analyzing the hyperlinks, it appears that there is a vast difference of number of edges for nodes in specific classes.

	number of edges
health/medical	1134
entertainment	10949
education/research	4325
tech/science	4162
politics/government/law	7684
news/press	3686
sports	280
business/finance	8233

Figure 2: hyperlinks analysis — Number of edges connected (in or our) to the nodes of each class

Finally, some of the main challenges is to classify isolated nodes or domains with empty texts or with texts such as the one in Figure 3 where it appears really difficult to classify it

since it can be judged to refer to multiple classes (politics, education...). Also, there exist some pages with different languages such as English, Chinese and Arabic.

```
['                                Forbidden\n',
 '\n',
 "   You don't have permission to access /robots.txt on this server.\n",
 '                                Forbidden\n',
 '\n',
 "   You don't have permission to access / on this server.\n",
 '                                Forbidden\n',
 '\n',
 "   You don't have permission to access /Contactez-nous on this server.\n",
 '                                Forbidden\n',
 '\n',
 "   You don't have permission to access\n",
 '   /Politiques-publiques/Jeunesse-sports-et-vie-associative/Vie-associativ\n',
 '   e/Aides-au-developpement-de-la-vie-associative/FDVA on this server.\n',
 '                                Forbidden\n',
 '\n',
 "   You don't have permission to access\n",
 '   /Actualites/Actualite-des-particuliers/Activation-du-champ-de-tir-tempo\n',
 '   raire-des-lacs-Robert-le-15-mars-2018 on this server.\n']
```

Figure 3: Example of an empty text

4 Tested approaches :

4.1 Processing :

Our pre-processing steps mainly focused on remove accents, transforming everything to lower case and removing stopwords as well as removing from the texts everything that may harm the prediction :

- Hyperlinks (split in multiple rows for instance)
- Left-overs of the scrapping (such as titles of the images recognized with their .jpg)
- Phone numbers
- emails

This list is not exhaustive and we added words gradually when seeing the errors in predictions in our local test step. This is what is done in the function 'process_text' in the provided IPython notebook.

Second step was to replace prices, dates and hours by some special tokens. The motivation behind that was to flag out the pages with so many of these special tokens (the news pages contain a lot of dates for instance whereas finance and business pages have a lot of EUR and prices in them).

We tried inference on texts where we removed rows that contain a single word (The motivation behind it is that those words are mainly titles of sections in the website such as "Acceuil", "Menu", "Contact"...), but we tried this text only when we we working on solutions that are context based (BERT) and not token based (TF-IDF even if it serves the sake of decreasing the vocabulary size).

We also saw that many websites had some redundant content (problem with the scrapper maybe ? or the website is like that) so we removed duplicated rows.

4.2 Text Based Methods only :

Correlation of local score with LB score :

One major step to the success in this competition is to be faithful when validating our models. Indeed we had to make sure that the metric we saw on the local data set (validation) was similar or correlated to the one we get on the leaderboard. This way we know whether an approach will give good results on the LB or not.

So what we did was taking the training_hosts, split them into local train and local test data sets. Train the model and cross validate it only on the local train and then see the performance on the local set (the local set was then acting like a "mini test set"). But we had to keep in mind that each time we were looking at the performance on that set and regulating our model, there was a risk of over-fitting on that set (and thus it would no longer be an isolated test set but just another fold of validation data).

Standard features :

What we call here standard features are the tf-idf features of each text in the training set. Indeed after pre-processing the texts and applying the tf-idf followed by a simple logistic regression for the classification we improved the score on the leader-board of the Text baseline. That was our starting point and we got to see how our local validation score would be correlated to the leader-board score.

After that the approach that worked the best using these tf-idf features was to apply dimensionality reduction on them using a SVD decomposition and then fitting an SVM classifier.

After that, we tried the apply the same approach on the unprocessed raw data and it scored surprisingly better than on the processed text. This is when we began to doubt on the quality of our pre-processing that was too harsh and discarding a lot of precious information.

BERT features :

Using the CamemBERT version of BERT that is available using the HuggingFace Python library, we inferred the embeddings (768 dimensional vectors) of the texts using many approaches :

- **Approach 1** : For each text, we pass every row of it through BERT, and take the CLS token (first token) embedding of each row. And then we average them. This will be the representation of the text.
- **Approach 2** : We join the rows of each text. This yields us to have a very big context separated by dots. Since BERT cannot process input sentences that have more than 512 tokens (510 actually because the first and last are for the SOS and EOS special tokens), we subdivided it to form new sub-texts. For instance a sequence of 1000 tokens labeled A will give us to instances one of length 512 tokens and the other of length 498 tokens both labeled A. We perform a sort of data augmentation here. After this we do as for the **Approach 1** : take the embedding of the CLS tokens of each sub-corpus.
- **Approach 3** : Same as **Approach 2** but instead of performing the data augmentation, we average the CLS tokens of the sub-corpuses obtained.
- Approach hidden states : Instead of taking the CLS token in **Approach 3**, we average to last hidden states (output of the encoder) to be the representation of the text.

- Approach hidden states + LSTM : Same as before but we input the last hidden states to an LSTM (Bi-LSTM tried too).

It is worth mentioning that we experimented other Bert-based architectures like Flaubert and Multilingual Distillbert but those trials were not improving the score on our local test set (also confirmed on the LB.)

4.3 Text and Graph based methods :

We tried other approaches that consisted of using the texts along with the hyperlink relationships.

- **Approach 1** (Manual message passing) : first fit a text based model on the labeled nodes. Then for a given node $node_{testi}$ in the test set, infer the label of the neighbouring nodes. We thus obtain $n_{Neigh(node_{testi})}$ vectors of probabilities. We compute the weighted average (given the edge weights) of those probabilities. But then we perform a voting between this so called "information of neighbours" with the node's own probability (obtained by inferring the vector of probabilities from its text).
- **Approach 2** (training set augmentation): Augment the training data set by including some non labeled nodes. The conditions to include a node as labeled X are : 1) The node should point towards nodes that are all labeled X in the non augmented training set — 2) the weight of the weight of the link starting from that node should be more than a fixed threshold. We eventually add 3503 more rows in the training set if threshold = 0
- **Approach 3** (Manual message passing using true probability vector) :
  - Start from a labeled node (say of class 3)
  - Its probability vector will be set to $[0, 0, 0, 1, 0, 0, 0, 0]$
  - Propagate these vectors to their direct parents weighted by the corresponding edge weights
  - The probabilities vector of a given node in the testing set will be a weighted average of the probabilities of all its parents.

  We also tried a version of this approach where we aggregate this vector with a vector of probabilities obtained from its text inference with a text based model.
- **Approach 4** : apply a simple Node2Vec on the whole graph (takes a very long time to compute) : This approach launches a biased random walk on the whole graph then used Word2vec to output features of each node.

4.4 Comments on these approaches :

We confirmed that the tf-idf approaches are very adequate for this kind of tasks (because a human would look for keywords in the website text to infer it), but the main drawback was it was giving a very large vocabulary (10k+ features). The reason that pushed us towards two directions : apply more preprocessing or apply a dimensionality reduction before classifying. But again we did not expect these approaches to perform well (being simple is an advantage here as well as a major drawback if we wanted to climb the ladder on Kaggle).

The idea behind averaging the CLS tokens of BERT is to simulate an approach that a human would do to categorize a page. If he begins by reading a first group of sentences, try to summarize what it was about and then jump to another group and so on to finally use all this summarized information to infer in which category the text falls, it is exactly what the **Approach 3** does. From that you can infer what were the motivations of the other BERT approaches.

Surprisingly, no processing at all worked the best with these BERT approaches. We realised it after recalling that it was the same for the standard features approach but we thought wrongly that since BERT gives attention weights to tokens, it will give less weights to the "noisy" tokens and that it would not increase the score by a lot. This indeed confirms the fact that our pre-processing was too brute and too violent and that we should have reconsidered it a bit earlier in the competition.

As what concerns the graph approaches, given the obtained scores locally we realised that forcing the labels (for data augmentation approach) or the manual message passing techniques will not give good results because the hyperlink structure can be so unpredictable. For instance, a news/blog page will have a hyperlink leading to a business/finance page that commercialise the product the blog was talking about. Plus, even if a node is of probability 1 for a given class, another annotator would not have classified it the same (actually we saw that for sport pages that lead to website where you can purchase sport equipment, so are they sport or business ?).

5 Best scoring approach :

5.1 Processing Features extracted :

As said before, no processing at all worked best for the BERT based approaches. Thus since the **Approach 3** using the average to CLS token embedding of the sub-corpuses of each text was the best scoring approach, we moved on using it to infer the class probabilities. We tried reducing the dimensions on top of that and concatenating these features with tf-idf features of each text but ended up always scoring less than when we exclusively used only them.

5.2 Classifier :

We tried first a logistic regression, but then we got some extra ranks in the ladder by switching to a simple NN (actually we wanted to confirm if a logistic regression was equivalent to a fully connected 1 layer NN but ended up having a better score - we explain this by the optimizer used during the fitting of the neural network). We used the validation set to see when we should perform early stopping that was around 450 epochs roughly (all the details are in the provided IPython notebook).

5.3 Empty texts handling :

When building the BERT features, if we ever encounter any issue (the only issue is trying to tokenize empty texts), we set the features to the 768-dimensional vector of zeros. Also, we noticed that some test texts where empty or just useless(

	Raw text	text processed	no single word	text no dupl	text processed 2
tf-idf + SVD + SVM	X	1.29 — 1.13	X	X	1.28 — 1.14
tf-idf + SVD + SVM + **Approach 1** Graph	X	X	X	X	1.31 — X
tf-idf + SVD + SVM + **Approach 2** Graph	X	X	X	X	1.26 — 1.16
tf-idf + SVD + SVM + **Approach 3** Graph	X	X	X	X	1.27 —1.14
Approach 1 BERT + SVM	X	X	1.80 — X	X	X
Approach 2 BERT + LogReg	X	X	1.30— 1.15	X	1.30 — X
Approach 3 BERT + LogReg	1.21 — 1.01	X	1.26 — 1.11	1.28 — 1.08 (*)	1.28 — 1.08
Approach 3 BERT + SVM	1.21 — X	X	X	X	X
Approach 3 BERT concat SVD SVM + LogReg	1.19 —1.03	X	X	X	X
Approach 3 BERT + NN	1.18 — 0.99	X	X	X	X
Approach 3 BERT + LogReg + **Approach 1** Graph	X	X	X	1.28 — 1.08	X
Approach 3 BERT + SVM+ **Approach 2** Graph	X	X	X	1.30 — 1.13	X
Approach BERT Hidden states + LogReg	X	X	X	1.28 — 1.11	X
Approach BERT Hidden states + LSTM + FC	1.26 — X	X	X	X	X
Approach 4 Graph	1.28 — 1.14 (**)	X	X	X	X
Approach 3 BERT + NN + **Approach 4** (***)	X — 0.94	X	X	X	X

Table 1: Obtained scores (local — LB) — X means that we did not try it or submit it — (*) : Maybe a mistake submitting the same predictions —(**): nothing related to the raw texts since it is applied on the whole graph — (***) : **Approach 4** only on the empty texts (given a threshold on the number of characters)—

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_32 (InputLayer)        (None, 768)               0
_________________________________________________________________
dense_162 (Dense)            (None, 8)                 6152
=================================================================
Total params: 6,152
Trainable params: 6,152
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: Caption

Moved Permanently web-pages for example). This was a motivation to consider graph structure to classify domains with no useful textual information.

In fact, the Bert based method ignores the relationships between the domain and we might believe that the hyperlinks provide additional information about the subject of a website as one would expect that a page is referring to target pages with the same subject or the same context. But, the problem was to find a proper method to exploit such information from the hyperlinks. One approach would be to use manual feature engineering to generate graph-related node features (as it's the case for the baseline). For example, calculating quantitative values of the structural position of a node in the graph (or a sub-graph) such as the number of neighbours (nodes degree) and other useful structural node features like PageRank or other various centrality measures. Also, Recent methods based on Graph neural networks with convolutional or attention mechanisms show improvement of results over the traditional models. However, using multiple methods for node classification, Node2Vec approach over the whole graph was the one performing well on the LB. Actually, we performed a biased random walk (which is a random walk that is biased in one direction, leading to a net drift on average of particles in one specific direction) to generate some trajectories and then we train a Word2Vec model to extract node features from the graph. Then, we fit a logistic regression model and we predicted the probabilities of the test set. Finally, the best approach we found in order to take into account text and graph features was to predict in one hand texts with rich content with our Bert model and in the other hand the probabilities of empty texts by the probabilities that we found using the Node2Vec approach.

5.4 Comment on our best approach :

Again the motivation behind this approach is as mentioned before : Simulate what a human would do ==¿ Summarize the information we get when reading the content of the web page sequentially (here the sequence if the maximum length that can be processed by BERT). When we have an empty text, or more precisely a text with no useful information, we go look at what the graph tells us about the node being processed.

6 Conclusion and further improvements :

A path that should definitely be explored more in depth is the pre-processing. Indeed as said before, our approach was too harsh and discarded very important information for classification. When imputing the texts for BERT tokenizer, he still managed to output good CLS tokens as he gave them low attention weights. Maybe by getting those attention weights out from the encoder and analyzing what are those tokens, we could clean adequately the text and leave only important information for the encoder to look at.

Also, we noticed the existence of some pages with full languages completely different from French (e.g, Chinese, Arabic) but with content that could facilitate its classification. However, as we treat only french words, we discard a lot of useful information. A suitable path would be to incorporate other language models trained on multilingual data or to be able then to detect the language of a text and to classify it with a corresponding language classification model.

Another good approach would be to try another GraphNN approach by initializing the nodes features by the BERT's features (actually we did try this at the beginning of the competition with the tf-idf features but it did not give good results because of two reasons : 1) the quality of the features, 2) we were only looking at the sub-graph of the nodes that interest us). Since it is very costly to process all the texts through BERT, we did not try it on the whole graph (and did not try on sub-graph either because we did not have much time left). But this might be interesting to see what it gives.

Other things we tried were :

- Pseudo labelling where we train the model on the labeled nodes, predict the label of unlabeled nodes and then include them in the training set to retrain the whole nodes. But this did not give outstanding results (but allows to gain some extra points on the ladder : -0.002 points).
- Handle the class imbalance : but this did not improve the performance as we do not care about the classification accuracy here but rather the confidence of our predictions. But it is not a reason to discard this path totally. Indeed, class imbalance is harmful for modelling and might need to be compensated by using for example a weighted cross-entropy loss in model training (such as for NN), with class weights inversely proportional to class support or even fine tuned with a Grid search.

Futher work and leads may try to address these two aspects of : using the big amount of unlabeled texts of increase accuracy (other than graph based approaches) and the tuning of the hyperparameters (sample weights, class weights, fine tuning of additional the two classifiers used : NN (on BERT features) and LogReg (on Node2vec)).

7 Details for reproducibility :

Major bottlenecks in the provided script are :

- The BERT features creation
- The Node2Vec features creation

Plus if you intend to run the .ipynb on Google Colab :

- You may face a problem when uploading the whole data (text/text being too big)
- Even if you manage to upload the entire files, Colab kind of disconnects and says that the files are inexisting even if they have been successfully uploaded.

To overcome these issues :

- BERT features : we ran them once on colab and saved them into .npy objects
- The Node2vec features : we ran the script once and saved them into a .csv
- Train/Test hosts : we ran them locally and saved them into .csv for further uses

You can find these files in the following **Github repository** One just needs to download them and put them in the ./data folder.

If those files are downloaded, read the comments in the .ipynb to see what are the lines that should be uncommented and ran and those that should be commented as there is no need to run them.