French Web Domain Classification Using Text and Graph Data

MAJDOUBI Ahmed Amine and **ABOU EL QASSIME Mehdi**
École Polytechnique
{ahmed.majdoubi, mehdi.abou-el-qassime}@polytechnique.edu

Abstract

Text classification can be either a supervised or an unsupervised learning task, which is considered nowadays as very important method for solving very challenging real-world problems such as sentiment analysis or topic modeling. These topics are determined by a set of training instances in the format of sequences of text. On the other hand, graph nodes classification is a very similar scope that uses the propagation of information through the graph with the association of external information for nodes classification or segmentation. In this paper, we are aiming to explore both text and graph data for the creation of a classification model for french web domains. We will start by giving the state of art for text and graph based classification tasks, then we will present the data that we will be using for the classification tasks. After that, we will present the results for the different models that we have tried. Finally, we will be presenting our final model and explain our approach.

1 Introduction

Web Domain classification is a major field of research for information retrieval such as scientific and technical watch. Most of the studies claim that domain classification is strongly correlated with its content but requires very appropriate descriptors. There exist a wide variety of ways dealing with text classification in general. All methods are based on text representation but only few are relevant to our problem since the objective of our classification task is rather related to lexical features of the texts.

We will also be using information extracted from graph network of the french domains. The main approaches used for node classification with graph data are based on the propagation of information and the association with meaningful external information. The majority of studies rely on the assumption that neighbour nodes have similar classes. This subclass of methods is denoted by graph machine learning. In this particular field we can distinguish three main approaches: graph convolution networks that learns the relation between the input of a node and its neighbors, output regularization requiring the neighboring nodes to be of similar classes, and finally node graph labels propagation that learns the optimal propagation and uses it for real valued vectors.

In this paper, we will present both basic and novel methods dealing with graph and text classification while making sure to adapt them for our use case. We start by presenting some working preliminaries aiming to make this paper easily understandable for every reader. After that, we will present the structure and some characteristics of our data set and then proceed by cleaning and reprocessing it. Finally we will present the approaches that we have followed in our study and the results obtained for each approach.

2 Related Work

Text classification is a supervised machine learning method used to classify sentences and text documents. Most recent text classification and document categorization systems are based on four main steps: feature extraction, dimension reduction, classifier selection, and evaluation as explained in [6]. The most important part among these four steps in text classification is the feature extraction step, which rely generally on text embedding. There are many old and novel techniques of feature extraction such as bag-of-words which is a simple representation of a text document based on each word frequency or TF-IDF features (Term Frequency-Inverse Document Frequency) proposed by [9] aiming to lessen the effect of implicitly common words in the corpus. There area also many vectorization methods including Word2Vec presented in [4] that aims to improve the word embedding architecture. Other recent techniques include the embedding of the text labels in the same document space as discussed in [5], and introduce an attention framework that measures the compatibility of embedding between text sequences and labels. There are also some more complex text classification techniques that are based on neural networks.

As for graph learning and node classification, there are three main approaches. The first one is based on graph partitioning which uses the eigenvalues of the Laplacian of a graph defined to be $L = D - A$ where D denotes a diagonal Matrix with the sum of each row in the diagonal and A the adjacency matrix (detailed approach available on [3]). There are other similar works that are based on the same idea, but are using smoothness and graph distance measures differently. The second method, known as diffusion labels, is based on the propagation of information through the graph. Other known

methods are based on this approach like DeepWalk [2] where a truncated random walk is performed on nodes, or by projecting nodes to minimize the distance of neighboured nodes [1].

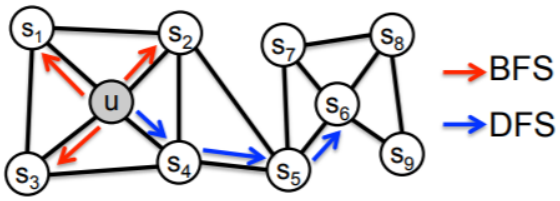
In this last part of reviewing related work regarding text and graph classification, we will discuss some methods that aim to combine between text and graph features. These methods are inspired from the success of spectral based convolutions. In the paper [8], they propose a simplification of this method using a two-layer approach, which can be summarized as: $softmax(AReLU(AXW1)W2)$ where A denotes a normalized adjency matrix of the graph and can be written as : $D^{\frac{-1}{2}}AD^{\frac{-1}{2}}$. This method uses also nodes features to learn the representation. Planetoid (Predicting Labels And Neighbors with Embeddings Transductively Or Inductively from Data) [10] is another similar approach that assumes node attributes(features) that takes into account both the class labels and the graph structure to learn node embeddings by trying to minimize simultaneously the supervised loss of labels prediction and unsupervised loss of graph context.

3 Preliminaries

In this section we aim to define briefly some technical notions and explanations of the some algorithms so as the paper is better understandable.

We will start by the notion of embedding, which is a relatively low-dimensional space into which we can translate high-dimensional vectors that are in our case text sequences extracted from web domains or nodes from the graph. Embeddings make it easier to apply some more complex models on large inputs like sparse vectors representing words. Ideally, an embedding captures some of the semantics of the input by placing semantically similar inputs close together in the embedding space and it can be learned and reused. We define some of the embedding techniques that we are going to use in our approach.

We will begin by trying Doc2vec, which is a very known embedding method, which may be very suitable for our case if we consider the domains as documents. The goal of Doc2Vec is to create a numeric representation of a document, regardless of its length. But unlike words, documents do not come in logical structures such as words, so instead of using just words to predict the next word, we also added another feature vector, which is document-unique and when training the word vectors W, the document vector D is trained as well, and in the end of training, it holds a numeric representation of the document (for more information about Doc2Vec, read [5]). We also defined another embedding method that is specific for our graph representation. The method is called Node2Vec [6], and it is an algorithm aiming to preserve network neighborhoods of nodes by finding a representation of the graph as a real vector, where nodes in the same neighborhoods have small euclidean distance between them and at the same time the node structure is preserved. In order to explore the graph, this method uses a second order random walk that aims to sample the graph that trades-off between BFS and DPS, between exploring the neighbors and exploring other neighborhoods, between micro-view and macro-view as shown in figure1. The trade-off is achieved by using hyper-parameters p and q to calculate the unnormalized transition probability. p — return rate , q — exploration rate.



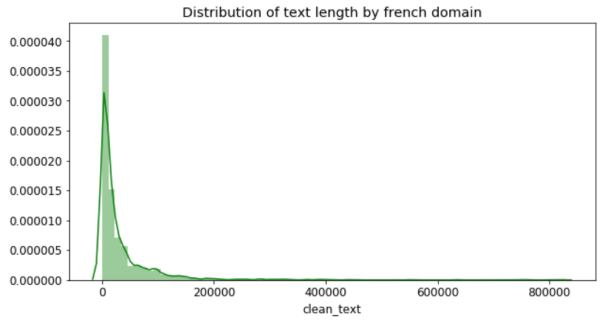
We will now give a brief explanation of the Graph neural network method. In a more general way, most graph neural network models have a somewhat universal architecture that is known as Graph Convolutional Networks, where filter parameters are typically shared over all locations in the graph. For these models, the goal is to learn a function of signals/features on a graph G=(V,E) which takes as input a feature description Xi for every node i; summarized in a $N \times D$ feature matrix X (N: number of nodes, D: number of input features), which is a representation of the graph structure in matrix form; typically in the form of an adjacency matrix A (or some function thereof), and produces a node-level output Z (an N feature matrix, where F is the number of output features per node). In particular, the mathematical representation of the GCN model is given by $f(A,H) = \sigma(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}HW)$ where D is the diagonal node degree matrix of A.

As for language models, there are now ubiquitous in Natural Language Processing. Most available models have either been trained on English data or on the concatenation of data in multiple languages. We propose here to explore [7] a pretrained model based on Bidirectional Encoder Representations from Transformers and best suitable for our language case(French) created by training a monolingual BERT paper11 model on the French language using recent large-scale corpora dataset. The evaluation of the model has been made on four downstream tasks (POS tagging, dependency parsing, NER and natural language inference NLI), achieving state-of-the-art results in most tasks, confirming the effectiveness of large BERT-based models for French data. An implementation of the model is available in for popular open-source libraries in BERT-implementationn.

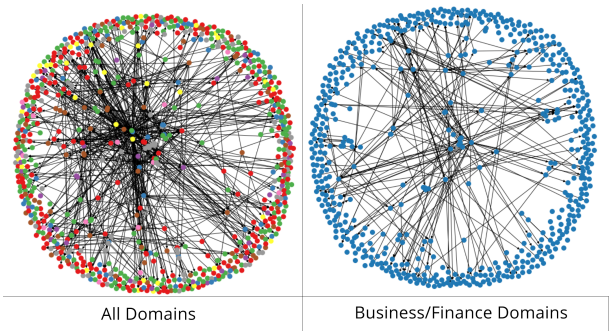
4 Data Exploration

Our input dataset consist of a directed graph, a list of texts and labels. First of all the list of texts represents the total text of all the web pages of the corresponding domain. We have a total of 2554 French domains with text data. The following image shows the distribution of total text length by domain. We can see that a lot of domains have little to no text, and that some domains contain a lot of text, so our classification model needs to be invariant to text length.

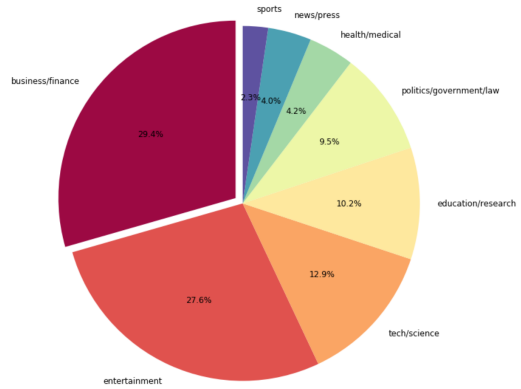
The next data is in the form of a directed graph. This graph contains 28002 vertices and 319498 directed weighted edges. Nodes correspond to domains and edges correspond to the total number of hyperlinks connecting two domains. Since we



only have the labels of 1994 domains, the goal of this graph to find hidden non direct connections between the 2554 domains. The following picture shows the directed graph for all 2554 domains, and the directed graph for Business/Finance domains.



In this subset of 2554 nodes, We can see that a lot of domains don't have any connections at all within the group, which means that we need to extract features based on the entire graph to gain useful classification features for our subgroup. The last figure shows a pie chart of the percentage of domains by label. We have 8 labels in total that we need to classify. We wan also note that there a huge imbalance between labels, which we should also take into consideration when making our model.



5 Experimentation

5.1 Text Pre-processing

As described before, all methods of language processing start with data prepossessing and cleaning in the case of text sequences input, we describe in this paragraph our approach of processing raw input web domain text keeping in mind the problem objective, we try to use simple and efficient steps of cleaning to only keep information that is most correlated to our labels, we define in the following the major steps of our data cleaning :

- **Step 1** : Tokenization, which is a pre-processing method which breaks a stream of text into words, phrases,symbols, or other meaningful elements called tokens. We first tokenize by sentence, then by word to ensure that punctuation is caught as it's own token.
- **Step2** : While processing with tokenization of words, we replace each " ' " by a space to separate a maximum of stop-words from information.
- **Step3** : Removing all tokens that contains non alphabetical special characters (with and without accents) and the sign " - ".
- **Step4** : keep only tokens that contain at least three letters, we do not expect tokens with less number of letters to hold a meaningful insight about our labels.
- **Step5** : Filter out french and English stop-words, as they don't contain important significance to be used in our particular classification task.
- **Step6** : rejoin the tokens by a space and save the text in a new column, to get full string feature extracted, and if needed we can simply tokenize it.

5.2 Text-Based Models

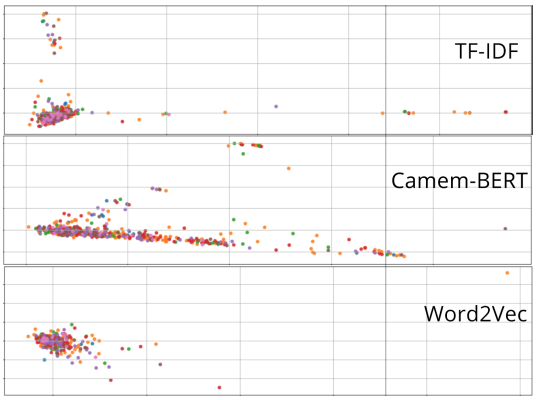
In this section of the paper, we will be working with text classification algorithms that seem relevant to our case. We applied the Tf-Idf proposed in [9] to generate a feature vector of length 18000 for each French web domain. We then perform a dimension reduction using principal component analysis to get a 1800 feature vector size with a 99,4 % explained variance. We apply to these features multiple classification models that are better suitable for the case of text classification with a relatively low number of observation. After using a logistic regression model and fine-tuning its hyper-parameters, we got a loss of 1,25.

We also used the Doc2Vec vectorization technique, expecting it to give better results by considering each web domain as a document. using a pretrained Doc2Vec model, we generated feature vectors of size 64, and by training the same models used before, we found that the best performance is done by a random Forest classifier with number of 100 estimators and max-depth 8. This models gave a loss of 1,7, which may be due to the fact that Doc2Vec can't extract semantic information from the domains.

The third and final embedding method that we have tried is the French BERT pretrained model named camemBERT [7], which achieves state of the art performances. This embedding generate features of size 120, a size fixed taking into

account all the input sequences. Using those features as an input to our models gave us a loss of 2,1. This method also did not work for our case, and we think the problem could be fault of implementation or the incompatibility between our preprocessed text and the pretrained model.

After getting these unexpected results using the methods we aimed to give significant performance, we decided to project each of the features extracted in a 2-dimensional space so we can have information about the separability of data for each of the used embedding methods which leads to a comparison between all embedding techniques used.



5.3 Graph-Based Models

In this section, we will be using only using graph data to perform the classification task. The main idea is to learn information from the distribution of the network in order to represent each of the nodes in a low-dimensional space to be able to use traditional classification models.

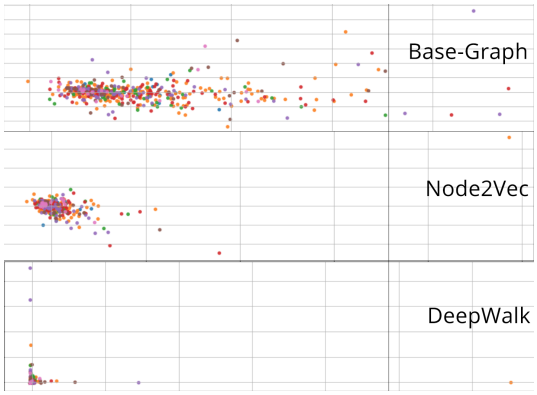
We started by using a Node2Vec by fitting it to the graph representation to generate feature vectors of size 100 for each node in the subset of the labeled nodes of the graph. This gives us a loss of 1,5 using a RandomForest classifier which improves slightly the performance of basic graph input features.

The second model that we have tried is the DeepWalk model [2], which have proven successful in multiple Natural Language Processing tasks. Basically, the DeepWalk model learns a representation of the graph's vertices through its random walks. We trained the model on our graph using the implementation seen in the lab6 within the ALTEGRAD course, through the minimization of the log of conditional probabilities regarding the embedding vectors. We used the features extracted from this embedding to classify the nodes, resulting in a loss of 1,74 by means of Random Forest Classifier.

As before we present the projection of each of the nodes embedding method in a 2-dimensional space to better visualize the separability of the features.

5.4 Mix Graph-text based model

In this section we will be using text and graph data at the same time to classify the French web domains. This can be done by ensembling the results of the different models created before, or by using novel methods that include both structures of data at the same time.



We chose to re-implement one of the novel methods of graphs learning representation, which is called the GCN model [8]. This models aims to merge in the learning process between text embedding and nodes relations with the network representation. Then we feed the GCN Model in the entry with an N by N Adjency matrix extracted from the graph and an N by D matrix representing the features of each node and a binary matrix for labels based on the implementation on the following Link. The limitation we encountered using such a model is that it should be fed with a matrix of labels which we do not have access to since not all the nodes are labeled, so we are obliged to only use a subset of the initial graph which makes the graph loses its important connectivity resulting with a mediocre result of 2,3 loss.

Since GCN didn't work, we opted for a more classic approach where we will do a voting type method to merge the prediction of different models. At the end, we chose TF-IDF, CamemBERT and Node2Vec for features extraction, and Logistic Regression and XGBoostClassifier models for classification. We tuned the hyperparameters of the Logstic Regression to give the best score on TF-IDF features, and tuned the hyperparameters of the xgboost model to give the best score for the rest of the features. We then used a VotingClassifier to get the final predictions. The final logloss score of the model on the test set was 1,1.

5.5 Models summary

Here we present our results using different techniques of embedding with multiple conventional classification models, for text and graph data learning, we could say that the each has a particular strength performing the feature extraction and a mix of the embedding of the two data representation may lead to better performance. the following present a summary of the loss of each of the used methods on a local subset test:

- **TF-IDF**: 1,32 with Logistic Regression on a local subset test, and a 1,21 on Kaggle test set with logistic regression tuned parameters and l2 penalty.
- **Doc2Vec**: 1,54 with Random Forest on a local subset test.
- **Camem-BERT**: 1,5 with Random Forest on a local subset test.
- **Node2Vec**: 1,54 with Random Forest on a local subset test.

- **DeepWalk**: 1,7 with Logistic Regression on a local subset test.
- **GCN**: 2,3 on a local subset test.
- **Voting classifier**: 1,1 on Kaggle test set using ensembling models.

6 Conclusion

The classification task of web domains is one of the most challenging problems in machine learning. As text and graph data sets proliferate, the development and documentation of supervised learning algorithms becomes an imperative issue, especially for text classification combined with network representations. Getting a better web domain categorization tools requires discerning algorithms first, but also a well designed features extractor that could better combine text and graph data. In this paper we have shown different implementations of methods that could help classify french web domain, and presented a summary of each of the models, but still some improvement can be done by better extracting useful information from our data.

References

[1] Jure Leskovec Aditya Grover. Node2vec. In *node2vec: Scalable Feature Learning for Networks*, 2016.

[2] Steven Skiena Bryan Perozzi, Rami Al-Rfou. Deepwalk. In *DeepWalk: Online Learning of Social Representations*, 2014.

[3] Guan Dhillon. Weighted graph cuts. In *Weighted graph cuts without eigenvectors a multilevel approach.*, 2007.

[4] Mikolov et .al. Word2vec. In *Efficient estimation of word representations in vector space*, 2013.

[5] Wenlin Wang Yizhe Zhang Dinghan Shen Xinyuan Zhang Ricardo Henao Lawrence Carin Guoyin Wang, Chunyuan Li. Joint embedding of words and labels for text classification. In *Joint Embedding of Words and Labels for Text Classification*, 2018.

[6] Kiana Jafari Meimandi Kamran Kowsari and Sanjana Mendu Mojtaba Heidarysafa. Attention is all you need. In *Text Classification Algorithms: A Survey*, pages 1–60, 2019.

[7] Pedro Javier Ortiz Suárez Yoann Dupont Laurent Romary Éric Villemonte de la Clergerie Djamé Seddah Benoît Sagot Louis Martin, Benjamin Muller. Camembert. In *CamemBERT: a Tasty French Language Model*, 2019.

[8] Peter Bloem Rianne van den Berg Ivan Titov Max Welling Michael Schlichtkrull, Thomas N. Kipf. Gcn. In *Modeling Relational Data with Graph Convolutional Networks*, 2016.

[9] K. A Sparck Jones. Term frequency-inverse document frequency. In *statistical interpretation of term specificity and its application in retrieval*, pages 11–21.

[10] Ruslan Salakhutdinov Zhilin Yang, William W. Cohen. Platenoid. In *Revisiting Semi-Supervised Learning with Graph Embeddings*, 2016.