Report - Text Mining

Lia Furtado and Hugo Vinson
Professor: Julien Velcin

Université Lumière Lyon 2, France
lia.furtado@univ-lyon2.com
hugo.vinsion@univ-lyon2.com

1 Introduction

This project was made for the Text Mining Class in the Université Lyon 2 Data Mining Masters. The main goal was to apply and learn text mining techniques in a dataset.

Our work was separated into the following phases:

- Data Retrieval
- Text Cleaning and Data Pre-processing
- Exploratory data analysis
- Text Vectorization
- Text Clustering and Visualization
- Search Engine

In the following sections we will explain our implementation for each of these steps.

2 Data Retrieval

The dataset chosen was of scientific articles published in Computer Science and stored in the database DBLP (https: //dblp.uni-trier.de). This database had four *json* files with each one having one million of samples. This data had the articles information such as title, abstract, authors, conference and year of publication. For this work we only took publications from the year 2016 to limit the volume of data. This subset of data was about 11.261 samples after the pre-processing stage.

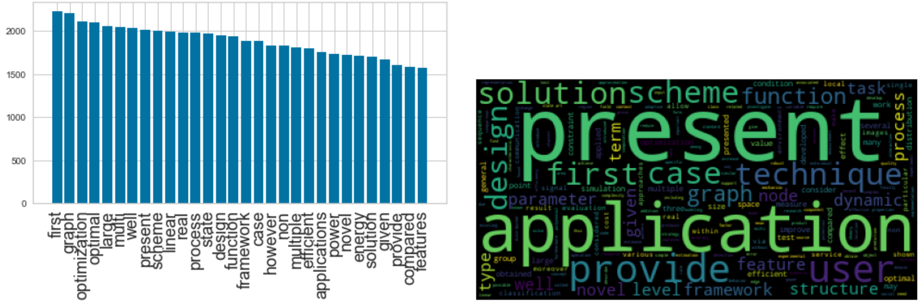
3 Text Cleaning and Data Pre-processing

First, we concatenated the title and the abstract to make a text representation for each article. Afterwards we cleaned each text by removing punctuation, non latin characters, digits and stop words. Additionally, we transformed the text into a list of individual tokens/words. Later on we removed some non-English texts and the 40 most used words.

We also cleaned and pre-processed the authors and venues (conference that the article was published). These informations will be used later on in the search engine so we cleaned this words by removing punctuation, non-latin letters and did other necessary improvements and finally turned them into tokens. We separated the words by a dash to better compare the authors and venues.

4 Exploratory data analysis

This section consisted in understanding the data and visualizing its relevant information. First we plotted the top 30 most common words and also a wordcloud (See Figure 4).



(a) Top 30 words that appear the most in the text

(b) WordCloud that represents the most used words

We can see that the words are really generic like present, application, first, solution etc.

Afterwards we got the top 10 authors and venues of the dataset.

Lastly we filtered the 20 authors that had written more articles and we did a network were the nodes were the authors and the edges were the articles linking the co-authors based on their articles in common. The network has the main clusters centered in each of the 20 authors.

In the bellow example in Figure 2 we can see some clusters of authors linked by the articles. In the left cluster we have an example of Benny Sudakov an author in the center of a cluster that is connected to several other co-authors of the "Saturation in random graphs".

5 Text Vectorization

This is a crucial part of this work because it defines how to transform the textual data into a numerical form. We vectorized the texts by applying three techniques: the tf-idf, Word2Vec and BERT. In the following sections we will explain how we did the transformation for each one.

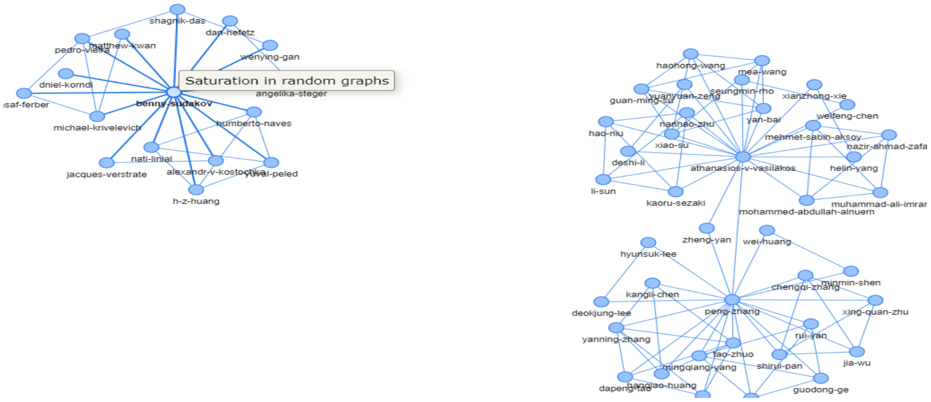


Fig. 2: Network of co-authors in articles

5.1 Term Frequency Inverse Document Frequency Vectorizer

Tfidf is equals to the number times a word appears in a document multiplied by the inverse document frequency of the word across the set of documents. This vectorizer considers in the overall documents the weight of words. We applied this Vectorizer and filtered for words that appear more then two times. The main disadvantages of this approach is that it creates a very big sparse matrix filled with a lot of zeros, in our example the vectors had a 18793 dimension.

5.2 Doc2Vec

This Doc2Vec is called an embedding technique because it transforms the text data into a vectorial representation by using the N-grams model. This is derived from the Word2Vec but here it create a vectorial representation of the whole document not just the word. In order to apply this to our dataset we first have to train the model with our text data, in the main parameters we choose to build vectors with 100 dimension. This technique allows us to create an embedding with only the words available in our corpus.

5.3 BERT

BERT is a transformers model pretrained on a large corpus of English data. In our solution we use the *bert-base-uncased* model from the Hugging Face repository. We used this model to build our vectorial representation of words. Each vector had a dimension of 768. Computing the representation of our texts with this tecnique had a very high execution time, so we decided to save the representation in a pickle format to speed up the process.

6 Text Clustering and Visualization

After we had the vectorial representation of the texts we experimented with making clusters of this data by applying the Kmeans algorithm. Initially we had to choose an ideal number of clusters, so we calculated the Silhouette coefficient from a range of 2 to 10 and got the highest value when k=3.

We wanted to compare our three different vectorial representations: the tf-idf vector with 18793 dimension, the doc2vectors with 100 dimension and the vectors made by the Bert transformation with 768 dimension. This way, we trained the Kmeans algorithm in each one of these representations and visualized our results by using Isomap a dimensionality reduction technique. In Figure 3 we have the results of the clustering for each representation.

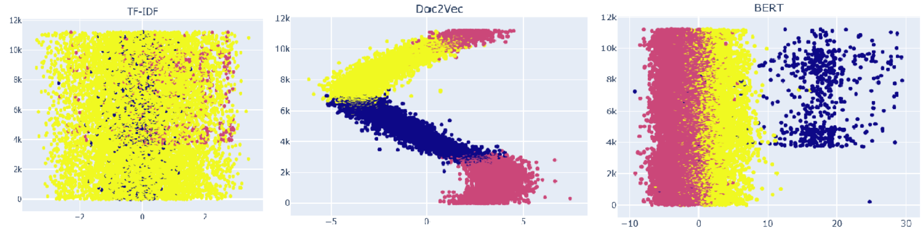


Fig. 3: Isomap visualization of the results of a the Kmeans clusters in each embedding technique

We can see by these results that with the tf-idf representation there is no clear separation of clusters, and moreover this was the visualization that took more time to process. This is probably explained by the very high dimension of data we have in this case. Comparably, with the Doc2vec and BERT representations we have some distinct clusters that represent that the articles are separated into groups maybe because of a difference in content and genre.

7 Search Engine

We will use process our texts to build two search engines. The first one will be based on key-words, so the user will choose some query words and the system will find similar articles with these words. The second one will be a search by articles titles, a title will be selected and the similar articles to this search will be returned.

To result of the search is the documents that are more similar to the words from the query. This similarity was measured by the cosine similarity metric. This measure is made to see the distance of two vectors in a space and is calculated by getting the cosine of the angle between them, that is, the dot product of the vectors divided by the product of their lengths.

The pipeline of the key-words search engine is:

1. Transform the queried words into vector representation (by any of the three methods)
2. Iterate through each one of the articles of the dataset
3. Compare the vectorial representation of the query with the one of an article by calculating the cosine similarity
4. The article has the highest similarity with the query is returned as the result of our search engine

7.1 Search by key-words

We choose to compare the results of our search engine by using our three vectorial representations of texts made by the previously described techniques. In table 1 we can see the results for the search of the query [”graph”, ”image”, ”points”],we have the corresponding cosine similarity and the execution time of each process.

Method	Article	Cosine Similarity	Execution Time
Query [”graph”, ”image”, ”points”]			
Tf-idf	graph plays crucial role success many graph representations handling structured especially emerging field graph signal processing however meaningful graph always readily available easy define depending application domain particular often desirable graph signal processing applications graph chosen admit certain ...	0.4389	87.07
Doc2Vec	graphical influential points cluster quality reliability engineering international early view online version record published inclusion issue	0.5968	11.15
BERT	connectivity independent let edge connected graph vertices proved alpha le either contracted graphs graphs fig	0.6219	200.78

Table 1: Comparing several different document embedding techniques in Keyword Search Engine

We can see that the tf-idf result looked for articles that had the exacts words of the query in their description, so it looked for articles that had multiple times the word ””graph”. The Doc2vec and BERT embeddings however can get information like synonyms of the words and find articles that are closely related to these queries in the space.

Doc2Vec found as similar the word graphical to graph, for example. Additionally, doc2vec was the fastest process compared to the other.

For the case of BERT, its main advantage is that it introduces the pre-trained knowledge getting the information of similar words even if we query words that are not in our original corpus. For example, the words ”fig” it understood that was related to the word ”image” in our query. However BERT takes a significant more time to compute do the vectorial transformation compared to the others, even in a case where the vectors were already previously saved.

We conclude that for a Search engine from key-words that you expect to find articles with those exact words that we specified the best representation is tf-idf. However, if you want a more powerful model that understands beyond your corpus BERT is a good solution, but there is a trade off between effectiveness and time.

7.2 Search by title

The second search feature is the title search. It allows to retrieve the title of the articles with the highest cosine similarity. It also allows to see if the selected articles have authors in common or if they have been published in the same venue. In this part we used Doc2vec embedding because it is a good compromise between time and similarity 1. The results show that often the most similar articles do not have authors in common. This is surprising since the authors usually work on a particular theme.

8 Conclusion

In this project we try to follow a common text mining method. We start to clean data, make a first exploratory and propose some visualisation. Then we proposed different embedding methods, compared them and applied clustering methods on these different representations. We are also interested in the relationships between authors and have shown in graph form examples of groups of authors. Finally we build a search engine based on cosine similarity. To conclude, Text Mining is a complex discipline. There are many possible treatments and we have implemented some of them. The main disadvantage of working with data such as this is the processing time, and the variety of studies that can be conducted requires a significant time commitment. But the results of the studies are often significant and the methods used interesting.