Université Nice Sophia Antipolis

Data Science Master's Project

CNN text classification model using Word2Vec

Date: 16 - June - 2017

GALDO SEARA Luis
GONZÁLEZ HUESCA Juan Manuel

Supervisor: Profr. PRECIOSO Frederic

1. Introduction

When the project was presented on late October 2016 we saw it as a very good opportunity to go beyond regular lectures and do research with the aim of learning an important topic for our master's degree and also developing a model for a real life application; but it was even more relevant because we were going to be part of a team that was going to build a scientific observatory of french political discourses for the 2017 French National Elections, something related to the most important political topic of the year in the country we were living in.

The project represented the most important and relevant challenge we have faced in the master's degree so far, and we are very pleased that after many months of research and development we did achieve the initial project goals, besides all the knowledge that we have acquired.

Along the project we faced many difficulties and issues which were solved as they were appearing with a deeper research, following a heuristic process approach, and also very important, with the help of our supervisor and the members of the linguistic team.

It tooks us several weeks, even a few months to fully understand the word2vec model, this was in parallel with our lectures and understanding on neural networks. After, we dealt with python libraries and code related issues which were solved. But the most challenging part was the interpretation of the whole analysis, that is why along the project many testing approaches were used in order to find out the best one, with good results.

The report will start with a background of the state-of-the-art, then we will move to the analysis and design where the whole model will be outlined, next the implementation will be explained. After that we will move to the testing, results and to finalize: conclusion, evaluation and further work will be detailed.

2. Background

In a time where deep learning has become a key data analysis method due to the new computing technologies, there are new applications where it can be applied with impressive results. Some of them are in the field of Natural Language Processing (NLP) where methods of projections of words in a vector space have led to impressive results by grouping similar words with strong semantic relationships [1].

Taking advantage of all these new techniques, the goal is to classify political discourses and find semantic relationships between different candidates via Convolutional Neural Network (CNN) using a vector representation model of words (Word2Vec). The CNNs are responsible for all the major breakthroughs in Image Classification and Computer Vision and, it has been proved a simple CNN with little tuning achieves excellent results on multiple benchmarks of sentence-level classification tasks [2][3].

Ths project provided the opportunity to apply deep learning models within NLP in a real scenario, in this case, in the linguistic analysis of political discourses from candidates in the USA 2016 National elections and the France 2017 National elections. This work was done while working closely with a linguistic team, Laurent Vanni and Mélanie Ducoffe, from the Université Sophia-Nice Antipolis and with our supervisor Profr. Frédéric Precioso, from the Laboratoire d'Informatique, Signaux et Systèmes de Sophia-Antipolis (I3S) at Polytech Nice-Sophia.

3. Analysis and design

The project started at a low pace while researching about the word2vec model and applications. The vector representations of words learned by word2vec models have been shown to carry semantic meanings and are useful in various NLP tasks [4], that's why the first step was to understand the two word2vec models: continuous bag-of-word (CBOW) and skip-gram (SG), with their parameters and optimization techniques.

Based on our experience working with data mining and machine learning methods during the first year of the data science masters degree we were able to have a the big picture of the knowledge discovery process, indicated on figure 1, which helps us to outline all the further steps in the project.

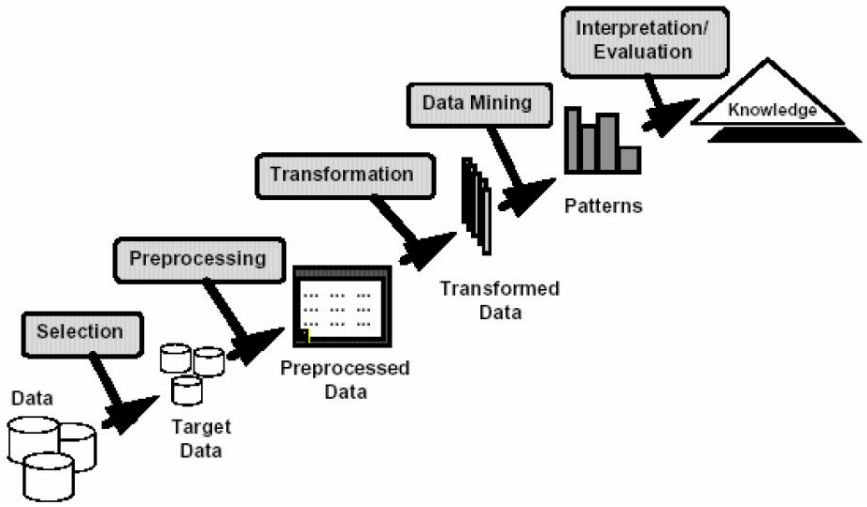

Figure 1. Knowledge discovery process.

With the knowledge discovery process in mind and the experience gathered while working on other NLP tasks in other lectures, it was possible to draw a high level diagram of the project implementation as shown on figure 2.

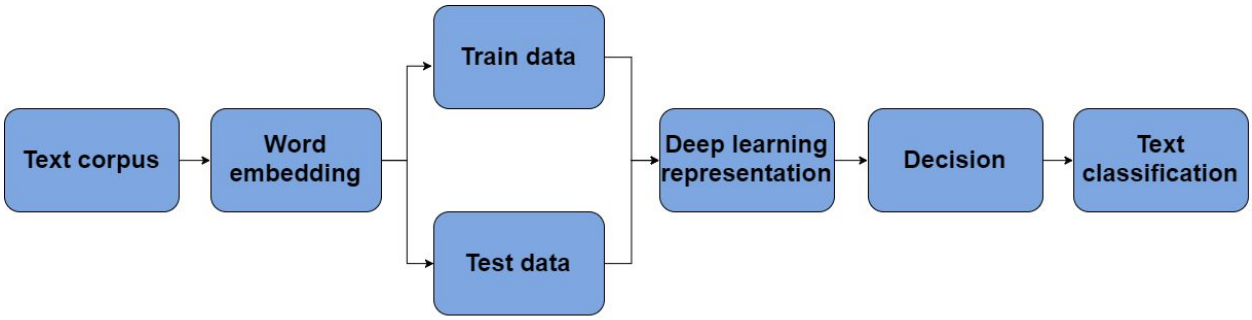

Figure 2. Implementation of the CNN text classification model using word2vec project

The preprocessing of data was the first step in the process, so at the beginning the speeches went through a text cleaning phase where the discourses were divided in sentences using the tokenizer library from the Natural Language Toolkit platform (NLTK); then it was a lemmatization step with an annotating text tool called TreeTagger that aims to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. To finalize the text cleaning process, the stopwords were removed as they don't provide any useful information to find strong semantic relationships in the sentences.

Once the text corpus is cleaned, the word embedding method word2vec is used to represents (embed) words in a continuous vector space where semantically similar words are mapped to nearby points ('are embedded nearby each other') [5]. The word2vec is a computationally efficient predictive model for learning word embeddings from raw text. As described above, it can be used with two different architectures (figure 3): the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model. Algorithmically, these models are similar, except that CBOW predicts target words (e.g. 'mat') from source context words ('the cat sits on the'), while the skip-gram does the inverse and predicts source context-words from the target words.

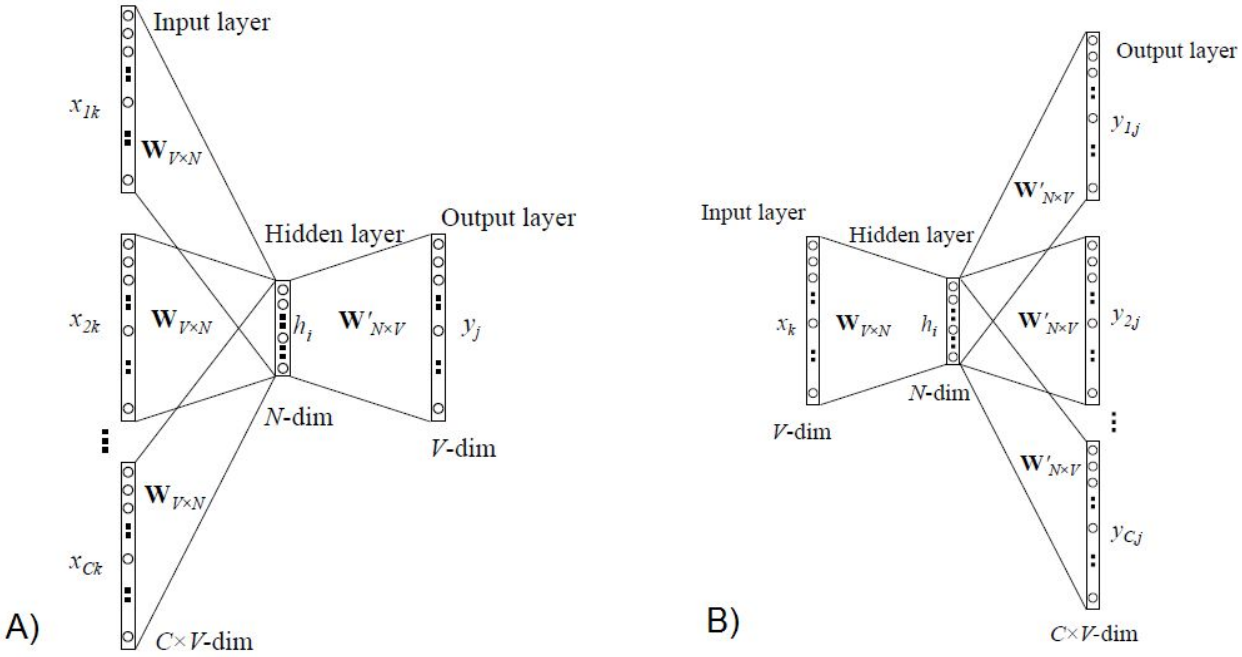

Figure 3. Word2vec models: A) Continuous bag-of-word model , B) The skip-gram model

After this, data from each candidate was separated in two groups. Around 80% of the data was used for training, and the 20% left for testing. In some cases, only one speech of a candidate was part of the data set, it was used for training.

In the next step, an artificial neural network is trained with the proper data (train data set). Instead of a traditional feedforward neural network, a Convolutional Neural Network (CNN) was

implemented because it has been proved it is possible to achieve impressive results by using convolutions over the input layer to compute the output. It has different layers:

- Embedding
- Dropout
- Convolutional block
- Concatenate
- Dropout
- Activation function

In the embedding layer, weights from the Word2Vec model are applied in the neural network. Dropout helps to avoid overfitting by setting some random weights to zero. Afterwards, a convolutional block comes. This block is executed several times, and it includes three layers:

- Convolution1D
- Max pooling
- Flatten

The first one, Convolution1D, creates a convolution kernel (filter or feature detector) that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs. It has different parameters that specify the following:

- Kernel Size: An integer or tuple/list of a single integer, specifying the length of the 1D convolution window.
- Number of filters: Integer, the dimensionality of the output space (i.e. the number output of filters in the convolution).
- Rectified Linear Unit (ReLU) - nonlinear activation function [2].

This convolutional block is executed in parallel. Then, the results are concatenated and dropout is applied again. To finalize the process, an activation function is applied, in this case, softmax, which reduces the number of neurons to the number of possible outputs in a way that the addition of the values of these outputs is 1, so they can be used as a probability to classify the sentences.

4. Implementation

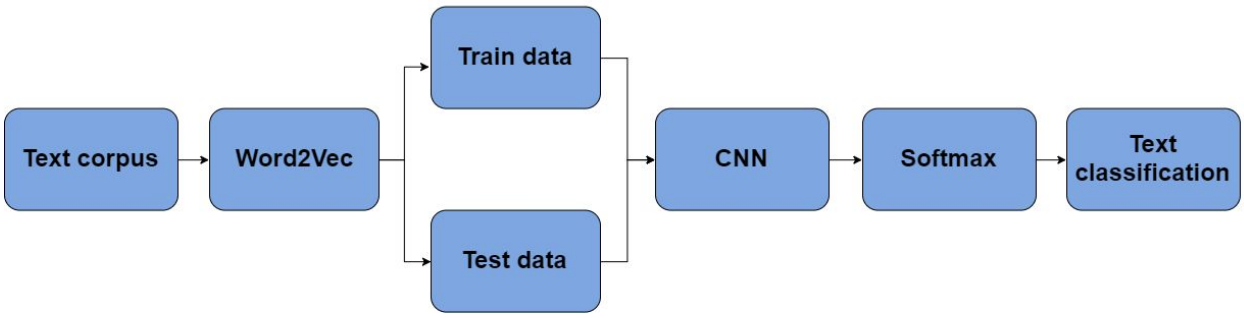

Figure 4. Implementation followed in the project.

Two different codes were implemented. One following the original paper [2], and a different one to adapt to colleagues, which were applying deconvolution to analyze the words that are important at the moment of classification. During the following description, the parameters used in Word2Vec and CNN are specified.

4.1. Word2Vec

In Word2Vec, as explained above, the speeches are divided into sentences so that the vectors can be generated. All the parameter specifications were followed as indicated in the main paper[2]. Figure 5 shows there are five main parameters to tune:

- Sentences: input data.
- Workers: 2. Number of threads to run in parallel. The value influences in the time the model takes to be trained.
- Size: 300. Specifies the size of the vector that will represent each word.
- Min_count: 1. Minimum number of appearances of a word to be taken into account to train the model. Because it is 1, all the words are taken into account.
- Window: 8. Number of words before and after each target word that are taken into account to train the model.
- Sample: 1e-3.

```
# Initialize and train the model
print('Training Word2Vec model...')
sentences = [[vocabulary_inv[w] for w in s] for s in sentence_matrix]
embedding_model = word2vec.Word2Vec(sentences, workers=num_workers,
                                    size=num_features, min_count=min_word_count,
                                    window=context, sample=downsampling)
```

Figure 5. Word2vec parameters setting

4.2. CNN

The recommended CNN implementation [2] was followed in order to follow a standard configuration and from there analyse the results for further tuning. In this case, there are three convolutional layers (Convolution1D) with 100 numbers of filter and with sizes 3, 4 and 5 respectively. Then, maxPooling1D is applied with a pool size of 2, in a way that the size of the matrix after applying Convolution1D is half the original size. Then all the results are flatten so that they can be concatenated, to apply “Softmax” as an activation function.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 30)	0	
embedding (Embedding)	(None, 30, 300)	3653400	input_1[0][0]
dropout_1 (Dropout)	(None, 30, 300)	0	embedding[0][0]
conv1d_1 (Conv1D)	(None, 28, 100)	90100	dropout_1[0][0]
conv1d_2 (Conv1D)	(None, 27, 100)	120100	dropout_1[0][0]
conv1d_3 (Conv1D)	(None, 26, 100)	150100	dropout_1[0][0]
max_pooling1d_1 (MaxPooling1D)	(None, 14, 100)	0	conv1d_1[0][0]
max_pooling1d_2 (MaxPooling1D)	(None, 13, 100)	0	conv1d_2[0][0]
max_pooling1d_3 (MaxPooling1D)	(None, 13, 100)	0	conv1d_3[0][0]
flatten_1 (Flatten)	(None, 1400)	0	max_pooling1d_1[0][0]
flatten_2 (Flatten)	(None, 1300)	0	max_pooling1d_2[0][0]
flatten_3 (Flatten)	(None, 1300)	0	max_pooling1d_3[0][0]
concatenate_1 (Concatenate)	(None, 4000)	0	flatten_1[0][0] flatten_2[0][0] flatten_3[0][0]
dropout_2 (Dropout)	(None, 4000)	0	concatenate_1[0][0]
dense_1 (Dense)	(None, 50)	200050	dropout_2[0][0]
dense_2 (Dense)	(None, 6)	306	dense_1[0][0]

Total params: 4,214,056
Trainable params: 4,214,056
Non-trainable params: 0

Figure 6. CNN recommended parameters configuration [2].

It can be seen on figure 7 how the implementation for our colleagues looks like. In this case, there are only two convolutional1D layers, with 300 number of filters and kernel size 3. Also,

padding is applied to increase the length of the sentences in two. It is done this way so that deconvolution can be applied over the network.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 32)	0	
embedding (Embedding)	(None, 32, 300)	1526700	input_1[0][0]
dropout_1 (Dropout)	(None, 32, 300)	0	embedding[0][0]
conv1d_1 (Conv1D)	(None, 30, 300)	270300	dropout_1[0][0]
conv1d_2 (Conv1D)	(None, 30, 300)	270300	dropout_1[0][0]
max_pooling1d_1 (MaxPooling1D)	(None, 15, 300)	0	conv1d_1[0][0]
max_pooling1d_2 (MaxPooling1D)	(None, 15, 300)	0	conv1d_2[0][0]
flatten_1 (Flatten)	(None, 4500)	0	max_pooling1d_1[0][0]
flatten_2 (Flatten)	(None, 4500)	0	max_pooling1d_2[0][0]
concatenate_1 (Concatenate)	(None, 9000)	0	flatten_1[0][0] flatten_2[0][0]
dropout_2 (Dropout)	(None, 9000)	0	concatenate_1[0][0]
dense_1 (Dense)	(None, 50)	450050	dropout_2[0][0]
dense_2 (Dense)	(None, 3)	153	dense_1[0][0]

Total params: 2,517,503
Trainable params: 2,517,503
Non-trainable params: 0

Figure 7. CNN parameters configuration for colleague's deconvolution project

5. Testing

5.1. Testing Approaches

Along the project, different approaches to solve the main problem were implemented.

5.1.1. Word2Vec model per candidate

At an early stage, due to the little knowledge there was about Word2Vec and its use, a model was created for each candidate (using their speeches as training data) with the intention of then checking which model was the one giving the best output when a new speech was passed as an input. This implementation had too many difficulties since it is hard to acknowledge when an output might be right or wrong if there is only one possible outcome.

5.1.2. Word classification

The next implementation was based in classifying each word. Instead of using sentences as training data, words were used. Of course, some words are impossible to classify as belonging to one candidate or another, i.e. “hello”,”bye-bye”, etc. Because of this reason and the bad results, this approach was discarded.

5.1.3. Sentence classification

After some research was done[2], sentence classification was chosen as the implementation to follow. Inside sentence classification several ways of training the data exist:

5.1.3.1. Dynamic size of sentences

One way was breaking down the speeches into sentences, then convert the sentences into list of words (whose values are calculated in the Word2Vec implementation). After this point, the sentences would be sent to the CNN model without any modifications. It was discovered that when a Convolutional Neural Network is trained with a 3 dimensional array (sentences, words per sentence, Word2Vec vector) at least two dimensions have to be fixed. Because of this reason, it was not possible to compile the model and to train it.

5.1.3.2. Static size of sentences

Due to the problem mentioned above. All sentences were limited to a given length (20 or 30 words depending on the test data). In case a sentence was longer than the limit, it would be cut. And in case it was shorter, it would be filled with zeros until the limit. This implementation was

based in the paper Convolutional Neural Networks for Sentence Classification [2]. In this paper they describe two possible options:

- CNN-static
    - It does not use Word2Vec, so we are not interested.
- CNN-non-static
    - This was one was the used approach, and it is explained in the section of Analysis and Design.

5.2. Testing Data

The system was implemented using two different data sets:

5.2.1. USA candidates

The data set is composed of 30 speeches taken from “The American Presidency Project” [6], 10 from each candidate (Donald Trump, Hillary Clinton and Bernie Sanders). The total data from each candidate is approximately the same.

5.2.2. France candidates

The data set is composed of 33 speeches. There are 6 candidates and each of them has between 1 and 11 speeches. This data set is not even, which causes problems during training, as explained in the Results section.

6. Results

On table 1 there is a summary about the results of the different tests that were run.

	Candidates	**Training speeches**	**Testing speeches**	**Results**
USA	3	24	6	57.92 %
France	6	27	6	72.45 %
France 2.0	4	*	*	55.72 %

Table 1. Classification accuracy results of USA and France national election speeches (*Speeches were divided into sentences and then the model was trained).

In the case of USA, 8 speeches of each candidate were used for training, and 2 speeches of each candidate were used for testing. The accuracy obtained with the testing data was 57.92 %. In the case of France, two different approaches were followed:

- In the first case, the data division was similar to the one in the USA approach. In general, all the speeches of a candidate expect one or two (depending on the number of speeches) were used for training, while the remainings were used for testing. Two candidates, Hamon and Jadot, only had one speech, and it was used in training. The accuracy obtained from the testing data was 72.45%. The main problem of this results is that more than 60% of the speeches of the training and testing data belong to Melenchon, so the system is trained in a way that most of the times a sentence is classified as Melenchon.
- To avoid the problem mentioned above, the same number of sentences from each candidate were used for training. Because there was not enough data from Hamon and Jadot, their speeches were not used. From the other candidates, all the speeches were divided into sentences, and only those sentences containing more than 8 words (after the cleaning process) were used for training or testing. 200 sentences from each candidate were used for training, and the remaining for testing. The accuracy obtained was 51.94%.

7. Conclusion, Evaluation and Further Work

7.1. Conclusion

The project was completed, as the two main objectives were achieved:

- Word2Vec implementation to classify political discourses.
- Building a model that our colleagues could use for deconvolution.

From a personal point of view, we gained knowledge in Word2Vec, in neural networks and in Python. We were able to learn different techniques inside Word2Vec to represent words with vectors, and we learnt how to used the data generated by a Word2Vec model in a neural network.

7.2. Evaluation

The expected results were higher than the ones achieved, but we have built several reasons of why this might have happened:

- The training data was not enough, making it hard for the neural network to learn enough so that the classification was correct.
- Sentence classification might not be the best approach, even if it was the best one given the quantity of data we had. We believe that paragraph classification or speech classification (if enough data was available) would have given better results.

7.3. Further work

To keep developing this project, there exist several options:

- Gather more data so that the neural network has enough information to learn.
- Paragraph or speech classification might give back better results.
- Even though we consider the two previous ones as the best options to improve this project, some other improvements could be possibly made in the parameters of Word2Vec and the parameters of the Neural Network. Different options were tried during the project without achieving better results, but it is not discarded that it might be possible to improve the accuracy modifying these parameters.

Bibliography

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality.
https://arxiv.org/abs/1310.4546

[2] Yoon Kim. Convolutional Neural Networks for Sentence Classification.
https://arxiv.org/abs/1408.5882

[3] Understanding Convolutional Neural Networks for NLP.
http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

[4] Xin Rong. word2vec Parameter Learning Explained
https://arxiv.org/abs/1411.2738

[5] Vector Representations of Words
https://www.tensorflow.org/tutorials/word2vec

[6] The American Presidency Project
http://www.presidency.ucsb.edu/2016_election.php