PAO Report for NLP : sentiment analysis

Robin Niel

January 23, 2019

Contents

1 Introduction

1.1 Motivation

Natural Language Processing (**NLP** for short) has been a challenging task for a long time. This field concerns a lot of problem, from question answering to text classification and automated translation. Before, this was done by huge rules system which were very complex and hardly maintenable. Recently, a new approach called machine learning has helped greatly improve the performance of the systems in this field. We will mainly focus on text classification for now. Primerarly, this was done by using a machine learning model called Naive Bayes. After that, the use of SVMs also allowed some progress. Very recently, a new approach has taken over : the use of neural network[1]. In this report, I'll try to expose my attempts to implement a **general** sentiment analysis model by using neural networks.

1.2 Problem formalisation

The task I will try to tackle is the task of **sentiment analysis**. Sentiment analysis is a problem where given a sentence or a short text, the sentiment of the text is to be predicted. Here the word sentiment can have different meaning :

- Real sentiments like happiness, anger, etc ...
- The polarity : negative, positive or neutral
- The subjectivity of the text

In this work, I'm interested specifically in the polarity, that is to say predict if a text is **positive** or **negative** (and **neutral**). This part of sentiment analysis is usefull for websites which have a review or comment functionality. Since many websites allow to mark a review in addition to the text, most of the datasets available for this task are datasete composed of online reviews. The label of the data can take different forms :

1. $y \in \{0, 1\}$, $y \in \{-1, 1\}$ for binary values (positive/negative)
2. $y \in \{0, 1, 2\}$, $y \in \{-1, 0, 1\}$ when there is a neutral class
3. $y \in \{1, 2, 3, 4, ..., M\}$, $y \in \{0, 1, 2, 3, 4, ..., M-1\}$ where M is the maximum mark on the mark scale

We can note that in the (3) case, we can set up rules to transform the variable into (1) or (2).

1.3 Goal

For specific datasets, designing a very accurate model is possible but in this work I am interested in designing a general sentiment analysis model, which is a more difficult task. This can be explained by the change of vocabulary from one website to another or even the length of reviews. A classical sentiment analysis model (for the (1) case) can be summarized by the following figure :

[1] Proof of the greater adequacy of the neural networks over SVMs for this task can be found in my previous report available on gihub

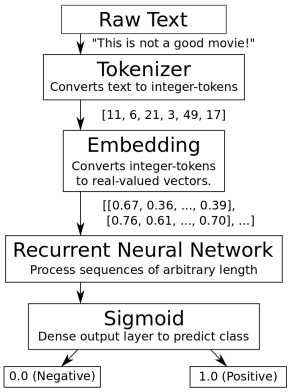

Figure 1: Classical sentiment analysis architecture (source)

The goal of this work will be to implement such a system. We will focus on improving the accuray of the system by mainly focusing on the Recurrent Neural Network part. It's important to note that most of the work around NLP uses recurrent networks because the text is a sequence where the order of the words carries a meaning.

2 Implementation

For this part, I will detail the steps of the schema in Figure 1.

The model is implemented in python using different librairies : gensim, scikit learn, keras and numpy.

2.1 Raw text

2.1.1 Training dataset

To train the different models on a small scale, I choose two datasets that I merged into one :

- Dataset 1
- Dataset 2

The two datasets together form a corpus containing ∼13 000 reviews.

On a bigger scale, I needed datasets with much more reviews so I took two categories of the Amazon review datasets : Electronics_5 and Movies_and_TV_5. The two datasets combine to ∼3.3 million reviews.

2.1.2 Test dataset

As I said earlier, the goal of this work is to develop a general sentiment analysis dataset. In order to make sure that is the case and that we are not doing overfitting on the training data, I tested the different models on an external dataset available here. It's important to note that this dataset is in fact composed of data coming from three websites : Amazon, Yelp and IMBD, respectively an online market, a site allowing to mark restaurants and a site allowing to mark movies and TV shows. The results are presented with these three sites separated.

2.2 Tokenizer and embedding

For this part, I will be using the gensim library. Concerning the embedding, I will be using a Word2Vec model. I will discuss the influence of the Word2Vec model in the next section.

2.3 Recurrent Neural Network

For the architecture of the neural network, I used the one proposed in this article. Following is a schema summarizing the architecture of the network.

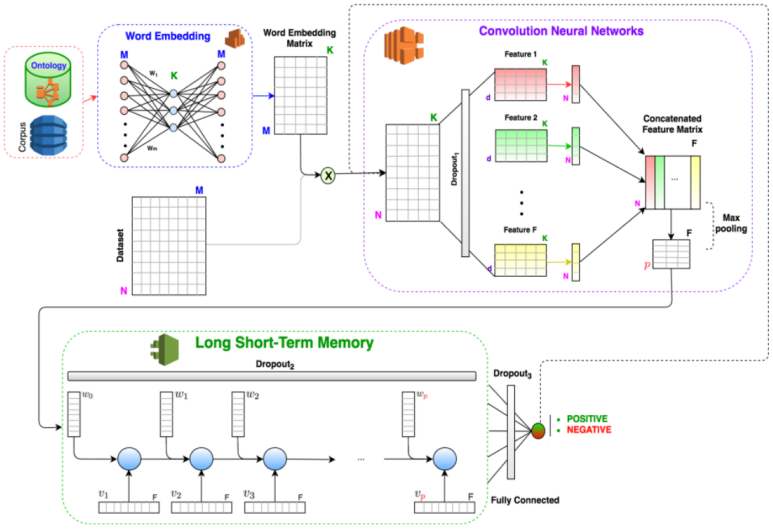

Figure 2: RNN architecture

The problem is that the article did not give the different hyperparameters such as the number of layers, the size of the kernel for convolutionnal layers, etc ... So I had to run experiments to determine them. The different architectures will be detailled in the next section.

3 Experiments and results

3.1 Word2Vec model and negation propagation

3.1.1 Word2Vec

Word2Vec is an embedding technique which allows to transform words into float vectors based on the context they were used[2]. The main advantage of that technique is that it can capture sementical relations between words. The example often taken is the king-man + woman = queen equation, but here are different examples :

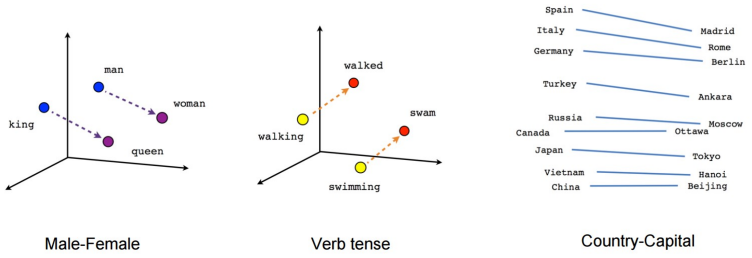

Figure 3: Word2Vec examples

Based on that, that is the method I chose to use for my model. The gensim library offers a module to use pre-trained Word2Vec model or train your own model. For my project I used a model made avaible by Google[3]. It's trained on Google News corpus and project words into a 300 dimensions vector space.

The problem with this model is that it canno't be trained more on another corpus. Moreover, the corpus on which it is trained is quite different of the one we will be training on. For those reasons, I also chose to train my own model on all the datasets I gathered.

3.1.2 Negation propagation

A problem often encountered when doing text analysis is the negation. Indeed, some sentences could be misinterpreted, for example : "It was not bad" could be interpreted as bad because of the presence of "bad" in the sentence. But here the "not" change the meaning of the "bad" word. To counter that effect, a proposition has been to use negation propagation. This method is based on a simple principle : when a "not" is encountered in a sentence, we delete it and then propagate it to every words after until we find a punctuation, that is to say {",",".","!","?",":"}.

Here is a quick example of the negation propagation :

[2] Original article

[3] Weights are avaible here

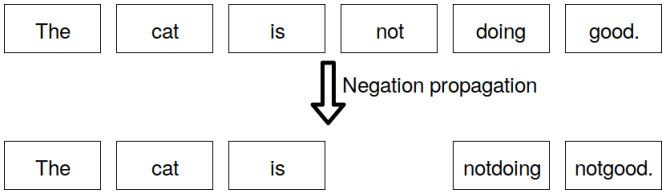

Figure 4: Negation propagation

This will form new words such as "notbad". The problem with that is that the GoogleNews word2vec model will not have those new words in its vocabulary. So to use negation propagation, we will need to train a new Word2Vec model on the Amazon datasets. We will also train another one without negation propagation in order to compare them.

3.1.3 External and internal embedding

Concerning the embedding, there is a choice to make : wether or not the embedding is included in the neural network. That is to say, is the words already transformed before being fed to the network or does the neural network has a layer allowing it to do the transformation. The advantage of the latter is that the network will be able to change the layer weight and potentially increase its performance compared to an external embedding. So I will add to the experiments a comparaison between internal and external embedding.

3.1.4 Results

In this experiment, I chose to compare the different architectures defined earlier, that is to say :

Name	Architecture		
Model 1	Word2Vec Model	Negation propagation	Embedding
	Google News	No	External
Model 2	Word2Vec Model	Negation propagation	Embedding
	Amazon trained	No	External
Model 3	Word2Vec Model	Negation propagation	Embedding
	Amazon trained	Yes	External
Model 4	Word2Vec Model	Negation propagation	Embedding
	Amazon trained	Yes	Internal

For the neural network, I will use a network with an input layer with a number of neurons equal to the maximum length of sentences followed by 50 LSTMs followed by 2 dense layers of 250 neurones each and an output layer with a softmax activation function. For the model 4 an extra embedding layer is added as the input layer.

Here are the obtained results :

Model	Test set (accuracy)	IMDB (movie) dataset	Amazon dataset	Yelp dataset	Mean
Sentences containing "not"	*0.6 %*	*8,3 %*	*10,7 %*	*12 %*	*10,3 %*
Model 1	0.7928	0.832	**0.825**	0.825	**0.827**
Model 2	0.7804	0.837	0.775	0.795	0.802
Model 3	0.7837	0.809	0.834	0.773	0.805
Model 4	0.7771	0.816	**0.852**	0.78	**0.816**

The first thing we can say is that our goal of having a general model is achieved because the accuracy on the datasets never seen by the model is higher than the test accuracy (the accuracy mesured on the test part of the training dataset). In term of the influence of the Word2Vec model, it is clear it's important because with the same network, we achieve very different accuracies.

The negation propagation doesn't seem to have a big influence on the final accuracy. Indeed we increased our accuracy on only one dataset in the three unknown ones. Moreover, the increase doesn't seem to be related to the percentage of sentences containing a "not". Further work is needed to conclude wether the not propagation is usefull in that context.

As a global conclusion, we can see that the best Word2Vec model to use is the GoogleNews one. But we can also see that the model 4 is also performing good and even beating the model 1 for the Amazon dataset. The low test accuracy might be linked to the small size of training set (∼10000 data).

3.2 Best models comparaison

In order to have a better idea about which model between model 1 and 4 is the best, but also if it is possible to improve the performance of the model 4 by training it on more data, we are gonna train both model on much more data. For that we will use the two Amazon review datasets. As I said, it represents 3.3 million reviews. The only difference is that for this dataset, $y \in \{0, 1, 2, 3, 4\}$. We need to transform it to adapt it to our architecture. So the 0 and 1 stars reviews will be classified as "negative" and the 3 and 4 stars reviews will be classified as "positive". The 3 stars reviews will be ignored but they can be used later as the "neutral" class we talked about earlier.

Here are the results :

Model	Test set (accuracy)	IMDB (movie) dataset	Amazon dataset	Yelp dataset	Mean
Model 1	NA	0.5	0.5	0.5	0.5
Model 1	*0.7928*	*0.832*	*0.825*	*0.825*	*0.827*
Model 4	*0.7771*	*0.816*	*0.852*	*0.78*	*0.816*
Model 4	NA	**0.84**	**0.866**	0.753	**0.822**

Here the models written in italic are the ones from the array in the previous part.

The GoogleNews model had a training issue that I could not solve, but we can see that the model 4 has indeed gain in performance on the IMDB and Amazon dataset. For the IMDB, the difference is quite important. We can see in red that the model has lost accuracy on the Yelp dataset. This might be caused by the over-representation of Amazon reviews in the corpus. But in general, the gain in accuracy is there. The model 4 accuracy after the training on a bigger dataset is almost equal to the model 1 in the previous part.

4 Conclusion

To conclude, I would say that the goal of having a general sentiment analysis model is achieved because the obtained accuracy is quite high. Moreover, the missclassified data is containing sentences that contains irony, which is impossible for this model to pick up, and/or specific vocabulary such as restaurant names or movie title. It is possible to implement several improvment of the model presented here. First, it is possible to change the embedding model by using, for example, GloVe or FastText and study the influence of such a change on the performance. It is also possible to study the influence of the negation propagation in order to decide wether to keep it in the architecture or not. Then solving the training issue for the model 1 would allow a better comparaison between the models. Finally, as usual in machine learning, training the model on more data would surely improve the results.

On a more general note, this work allowed me to improve my knowledge about neural networks and NLP. It also allowed me to see the flaws in my work method thanks to M. Gaüzère.