Twitter Sentiment Analysis : A comparative approach

Mariem BOUHAHA Eden BELOUADAH

12 janvier 2018

1 Introduction

Text classification is a class of problems that finds application in many tasks, mainly sentiment analysis, predicting movie reviews as well as classifying emails as spam or not.

In recent literature, deep learning seems to be promising for text classification, achieving state-of-the-art results on a suite of standard benchmark problems.

In this project, we aim to compare and analyze the performance of several deep architectures that we use for sentiment analysis from tweets. For that, we studied the behavior of some deep neural networks as well as the influence of data representation on the models' performances.

In this intermediate report, we will present the data in hand as well as some statistics to describe it, then we will give a brief description of the models tested so far and finally present the preliminary results.

2 Presenting the data

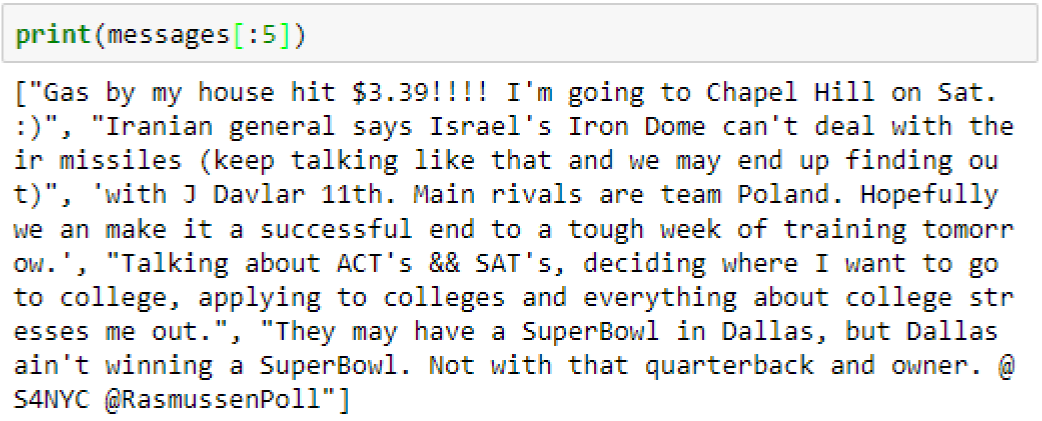
For this project, we used the B-task SemEval-2013 training and development data, which have the following characteristics before any preprocessing :

Set	Number of examples	Positive	Negative	Neutral
Train	5916	2179	816	2921
Development	876	324	158	394

Tableau 1. Classes distribution for trainset and devset

If we visualize an extract (the first 5 for example) of the tweets in the training set, we get the text shown in figure 1.

We noticed that, in addition to words, messages contain hashtags, URLs and user tags, which are less likely to be useful for sentiment prediction.



```
print(messages[:5])
```

```
["Gas by my house hit $3.39!!!! I'm going to Chapel Hill on Sat.
:)", "Iranian general says Israel's Iron Dome can't deal with the
ir missiles (keep talking like that and we may end up finding ou
t)", 'with J Davlar 11th. Main rivals are team Poland. Hopefully
we an make it a successful end to a tough week of training tomorr
ow.', "Talking about ACT's && SAT's, deciding where I want to go
to college, applying to colleges and everything about college str
esses me out.", "They may have a SuperBowl in Dallas, but Dallas
ain't winning a SuperBowl. Not with that quarterback and owner. @
S4NYC @RasmussenPoll"]
```

Figure 1. A small extract from Train

We also computed some statistics for the tweets, which we summarize in table 2.

Type	**Positive messages**			**Negative messages**			**Neutral messages**		
Length	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
Train	4	44	23	6	38	23	5	40	23
Dev	7	35	23	8	34	23	6	38	22

Tableau 2. Statstics about messages length for trainset and devset before data preprocessing

3 Data Preprocessing

Since we cannot use the "dirty" raw text tweets for classification, we did some preprocessing as following :

- Tokening messages using *NLTK Tweet Tokenizer* which is more appropriate for this type of messages,
- Using regular expressions to remove digits (24, 12.400$...), punctuations (?, :, ;,",#, @...), tags (@user) and every sequence of special characters. Howerver, the emoticons are kept,
- Using *NLTK Stop words* to remove all useless words,
- Preprocessing Hashtags : removing # and separating the hashtag parts (#VeryHappy -> Very + Happy)
- Transform all messages to lower case.

These operations left us with a training vocabulary of size 17284, where preprocessed tweets now have the following statistics :

Type	**Positive messages**			**Negative messages**			**Neutral messages**		
Length	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
Train	3	22	12	3	20	13	3	22	12

Tableau 3. Statstics about messages length for trainset and devset after data preprocessing

As we notice, tweets have different lengths. To handle this, we take the size of the longest message and use it as a reference. The tweets having the same number of words are kept as they are, and shorter ones are replicated until their size become equal to that of the longest tweet.

In figure 2, we show the distribution of word frequencies in our training set.

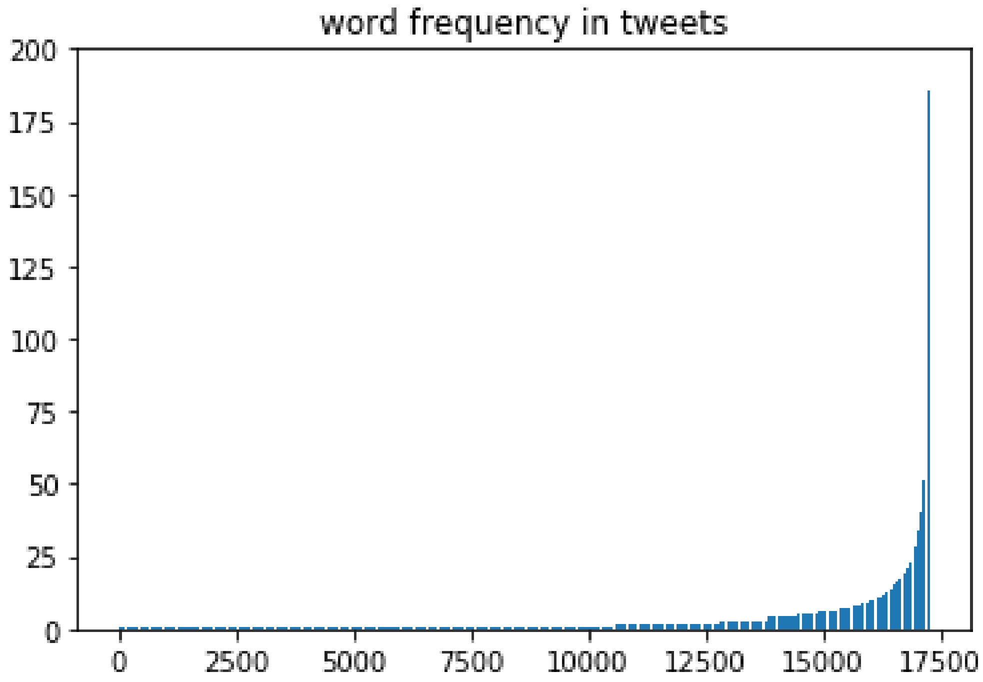


Figure 2. Word frequency

4 Models and results

4.1 Long Short Term Memory (LSTM)

The first reflex that we had when handling text data is to use recurrent neural networks. LSTM is one of the most used types of language models.

In this first work, we used batch mode. To be able to train LSTM model, we had to prepare the input matrix which has the dimensions (batch size X sequence length X embeddings length), where :

- Batch size = number of tweets in trainset,
- Sequence length = length of the longest tweet,
- Embedding length : size of embeddings fixed to 100.

Words embeddings are calculated using *word2vec* model trained on out training set.

The architecture of our model is as following :

- Number of layers : 1,
- Number of hidden units : 150,
- Knowing that we have three classes in our classification problem, the LSTM layer is connected to a linear layer of size (number of hidden units X3).
- Optimizer : We used Adam optimizer instead of smple SGD since the latter is less robust against local optima.
- Loss : We used the Cross Entropy Loss.

After peforming forward propagation, we backward the error and update the parameters of the model.

The program shows real time plots for accuracy and the Cross Entropy Loss.

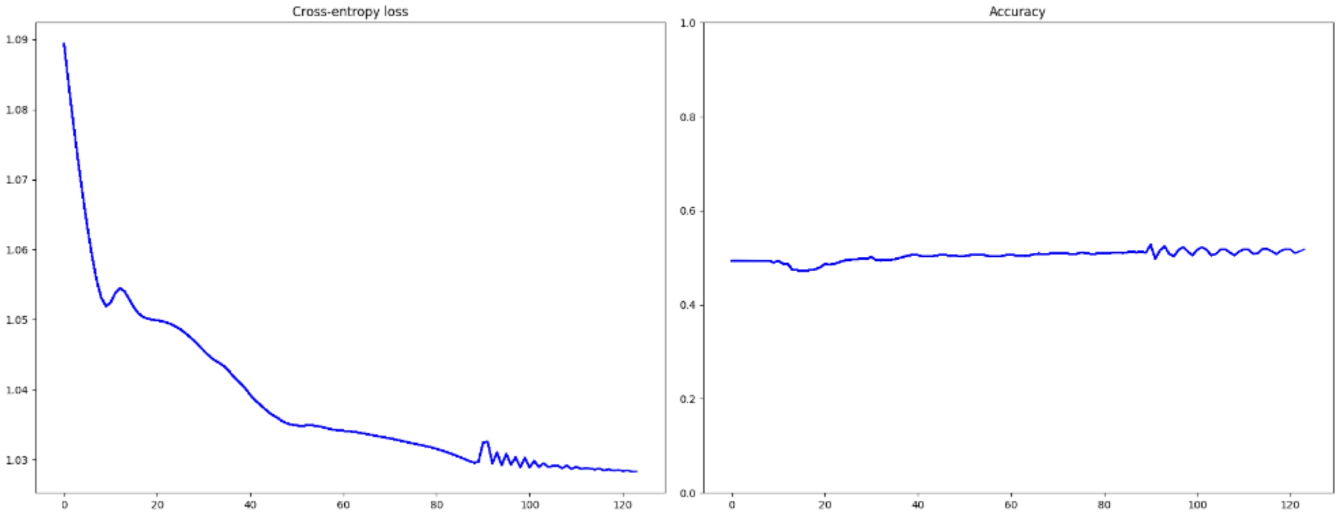


Figure 3. Accuracy and Loss evolution for LSTM baseline

Unfortunately, The maximum of accuracy that our model reached is 52%. We think about changing again the architecture of the model, probably change the data preprocessing and why not add dropout techniques.

4.2 Multi-Layer Perceptron (MLP)

We also wanted to test the performance of the MLP classifier if we train it on a word embedding-based representation of our tweets.

For that we used the Word2Vec embeddings for our Vocabulary and represented each tweet by the sum of the embeddings of its constituents. That is, if a tweet is "Happy New Year", the value of this observation would be the some of the embeddings of "Happy", "New" and "Year". And we do this for all the elements in the training set. Thus, using an embedding dimension of 100, the shape of our training set now becomes (5916, 100).

We then train the MLP classifier with one hidden layer of size 100 neurons, using stratified 10-fold cross-validation. We obtain the following results :

Fold	0	1	2	3	4	5	6	7	8	9
Accuracy	0.513	0.540	0.513	0.522	0.5	0.530	0.521	0.496	0.487	0.493

Tableau 4. Accuracy obtained for each fold

The mean accuracy is 0.51.

As we can see, the MLP performance is slightly better than a random classifier which would give us an accuracy of 33%. However, we can still optimize the model and/or the data representation. Deeper thoughts would be developped in our final report.

5 Conclusion

In this intermediate report we showed data preprocessing and our baseline results. In addition to making the models give a higher accuracy and a lower prediction error, we aim at analysing the behaviour of each of the models tested for the different representations of the training data to come up with an insight regarding the specificities of this particular classification task.