Assignement 2 - CS941

Edoardo Barp - u1406032

March 18, 2018

Abstract

Sentiment classification of tweets is a popular task in NLP given the amount, variety and complexity of the data. In this paper, three approaches to classifications are approached, two of which based solely on lexicons, and one including vocabulary and word embeddings, and a discussion is made on the various features extracted and computed.

1 Introduction

Twitter, being a public social network, contains tweets from all sorts of sources, making the language used very varied; the datasets given are further evidence of this, including a wide spectrum of tweets, from very formal tweets, such as news alerts, to very subjective and often slang-full tweets, probably coming from the younger audience. As such, many difficulties are faced when attempting to classify these tweets since the variety in vocabulary, format and style is great. For instance, it is sometimes even non-trivial for a human to decide whether a tweet can be considered neutral, or if it is biased enough to be attributed a non-neutral stance, and similarly whether a tweet is sarcastic or not, especially without context, since no information is known about the authors.

2 Features Extracted

2.1 Lexicons

The features extracted can be loosely separated into the following categories. A list of all features is available in Fig.1 below.

- **Generic features:** are the generic details of a tweet, including the number of words, the presence of user mentions, url etc. These don't explicitly carry a lot of information per se, but can be useful in case of ties to give a slight idea of context.
- **Vocabulary:** by far one of the most important, the presence of some words can be extremely useful in classifying the tweet. For instance, racial, religious, political or other forms of group slurs can be found to be highly correlated with negative sentiment, as one would expect. Similarly, the presence of *good/bad* words is be a fairly strong point.
- **Style:** includes features such as the use of punctuations, usually more associated with a neutral stance, as opposed to the use of elongated words, usually not associated with a neutral stance.
- **Statistical:** these are features meant to give a sentiment probability to each word, depending on which type of tweets it tends to appear, and are further discussed in the next section.

2.2 Lexicons Details

Let us explain further the features that aren't self-explanatory.

In the **statistical** column, the *negative*, *neutral* and *positive words* are part of a dictionary which, for every token in the training set, gives the probability of it being associated with one of the sentiments. Note that this probability is weighted to account for the fact that there are many less negative tweets as there are positive and neutral ones. Additionally, token below a count threshold[1] are filtered since these

[1] Given a token in the training set, it is checked whether this word appears less often than a threshold. The threshold was set to 5 through simple trial an error, and should really be studied further to find which threshold gives the best results.

Generic Features	Style	Statistical	Vocabulary
Length in words	Count of punctuation mark	Negative words	Count of insults/slurs
Count of user mentions	Count of exclamation mark	Neutral words	Count of positive words
Count of hashtags	Coutn of Interrogation mark	Positive words	Count of negative words
Count of URLs	Count of negations	Positive emojis	Count of religious words
	Count of laughter	Neutral emojis	
		Negative emojis	
		Positive synset	
		Negative synset	

Figure 1: List of all features extracted from tweets

are assumed to not appear enough to be used confidently, and similarly, words which are found to carry too little information are also removed. The latter is determined using the H_{10} entropy, defined as

$$H_{10}(\text{token}) = -\sum_{s \in \text{sentiment}} p(s|\text{token}) \log_{10} p(s|\text{token})$$

Given this metric of information carried, every token with H_{10}-value in the interval $(0.46, 0.54)$[2] are removed.

The measures for *positive, neutral and negative emojis* are found in a similar way, although no filtering is applied to them.

The *positive and negative synset* are taken from the default set included in `nltk.corpus.sentiwornet` and found through the function `senti_synset`.

Finally, given that a tweet is composed of several words, all the different features are summed over every word, as to obtain an overall tweet value, normalized by the tweet length

Moreover, the `twitter` specific features were replaced with the following keywords:

- user mentions → <user>
- hashtag → <hashtag>
- url link → <url>

Note that the counts for the various features under the **vocabulary** and part of the **style** columns come directly from datasets; notably, the positive/negative datasets are the ones from the 1^{st} assignement, while religious, negations and laughter were created, and the insults/slurs dataset was taken from an online source[1].

2.3 Vocabulary Embedding

By vocabulary embedding, it is intended the action of transforming *n-grams* of a tweet into a vectorial form. The simplest way is, using $1 - grams$, i.e. every token, to create a matrix $N \times V$ where N is the number of tweets, and V the vocabulary. Then every tweet can be written as a very sparse binary vector in the $N \times V$ space, where if the word corresponding to the i^{th} vocabulary word is present, then the i^{th} entry of the vectorized tweet will be 1. The idea is simply generalized to *n-grams* by considering a vocabulary made not only of every single token, but also every single *n-gram* appearing in the training set.

Once embedded, the massive albeit sparse matrix is reduced using a χ^2 test to find the 500 most important features[3].

2.4 Word Embedding

Another way of embedding which was used was through GloVe (Global Vectors). GloVe is a model which allows to transform word into vectors, in which similar words are assigned similar vectors. Given these vectors are found by training in an unsupervised way over massive dataset, which is not viable for this assignment, a pre-trained model was used[2], which embeds words as 200-dimensional vectors.

[2] This interval was found through quick trial and error on words that were expected to be filtered, such as "the", and words that weren't

[3] 500 was found through cross-validation

Tweets were then transformed by first embedding every word together, to create a $N \times D$ matrix, where N is the number of words and D the embedding dimension, which in the context of this paper was set to 200. Then, transformations are applied over the columns to obtain vectors of fixed length, independent on N. Five transformations were used in this assignement:

- Sum normalised by maximum of resulting vector as to not be length-dependent
- Average
- Maximum
- Minimum
- Median

Each of this transformation generated D-dimensional vectors, which can then be stacked together, to form a 1000-dimensional vector.

2.5 Important Data Note

The data provided for training was found to be heavily biased toward neutral and positive tweets, with respectively $18\%, 46\% and 36\%$ of negative, neutral and positive tweets. This was found to cause many problems in the various classifiers, as discussed further throughout the paper. To avoid this, the training data was balanced by removing all the extra neutral and positive tweets before being fed to models for training. An example of the change in performance can be later seen in Fig. 5. Therefore, it was decided to use the dev set as an extension of the training set, since no better use was found.

3 Classifiers

3.1 Classifier 1: Lexicon Based Ensemble

The first classifier only used the lexicon features as described earlier. Many classifiers were tested intensively using Cross-Validation to try and only keep the few best. It was interesting to note many of the classifiers commonly used in the literature didn't seem to produce good results, such as *Naive Bayes*, of which multiple subtypes were used, such as *GaussianNB, MultinomialNB and BernoulliNB.*

Many of the tested classifiers were dropped after the first few long cross-validations since they didn't seem to score well enough to be investigated further, and only a handful were kept: *RidgeClassifier, PassiveAggressiveClassifier (PAC), Bagging and SGDClassifier.*

These were heavily studied and cross-validated to achieve the best performing results. It was also found that applying a χ^2 to reduce the dimensions to 10 lead to an increase in $1-2\%$ over all test datasets.

The classifiers were then put together as a majority vote ensemble.

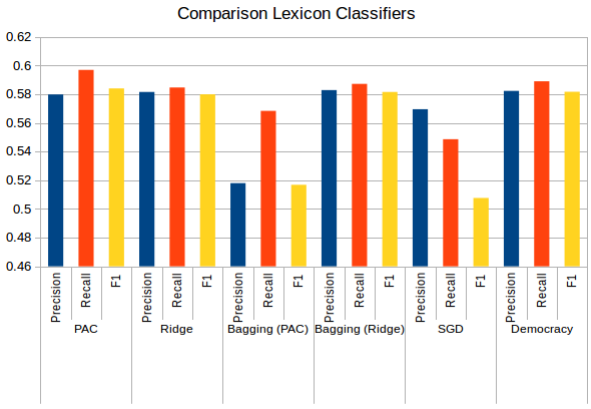
Model	Parameters	F1-score
Ridge	solver: auto, $\alpha = 1000$	0.580
PAC	$C = 0.0001$, loss: squared hinge	0.584
Bagging	estimator: Ridge, $n_{\text{estimators}} = 5$, max samples $= 0.2$	0.587
Democracy	—	0.582

Table 1: Final classifiers' scores. Although PAC seems to have a slight edge on the majority vote *(democracy)*, it was actually found to have more variance, and therefore the democratic vote was still preferred for the final submission.

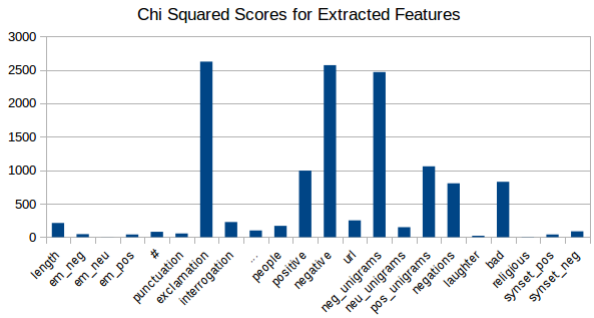
3.2 Classifier 2: Lexicon Based Neural Network

Knowing the capabilities of neural networks, and since classical classifiers seemed to be working pretty well, it was decided to attempt to classify tweets still only using lexicon features, but with a neural network as only model.
Many structures were tested, from very shallow single layer to deep and large networks, and it was quickly found that shallower networks performed much better. One can clearly see from Fig.3 that the shallower model, in green, is much more constant than the two deeper and more complex ones. This model's architecture was:

$$\text{Dense(15, tanh)} \rightarrow \text{Dense(20, tanh)} \rightarrow \text{Dense(30, tanh)} \rightarrow \text{Dense(10, relu)} \rightarrow \text{Dense(3, sigmoid)}$$



(a) Classifier 1: Results for final classifiers attempted



(b) χ^2 values for lexicons features

Categorical Cross-entropy was used as loss, together with *adam* optimizer. Note the use of the *sigmoid* as output activation, which was found to give more balanced results compared to *softmax* which would output very extreme probabilities.
It is also worth mentioning that the results obtained before balancing the datasets labels by removing many of the neutral and positive tweets were much worse, were the network would clearly become biased against the negative and would score as low as 0.4 F1-score.

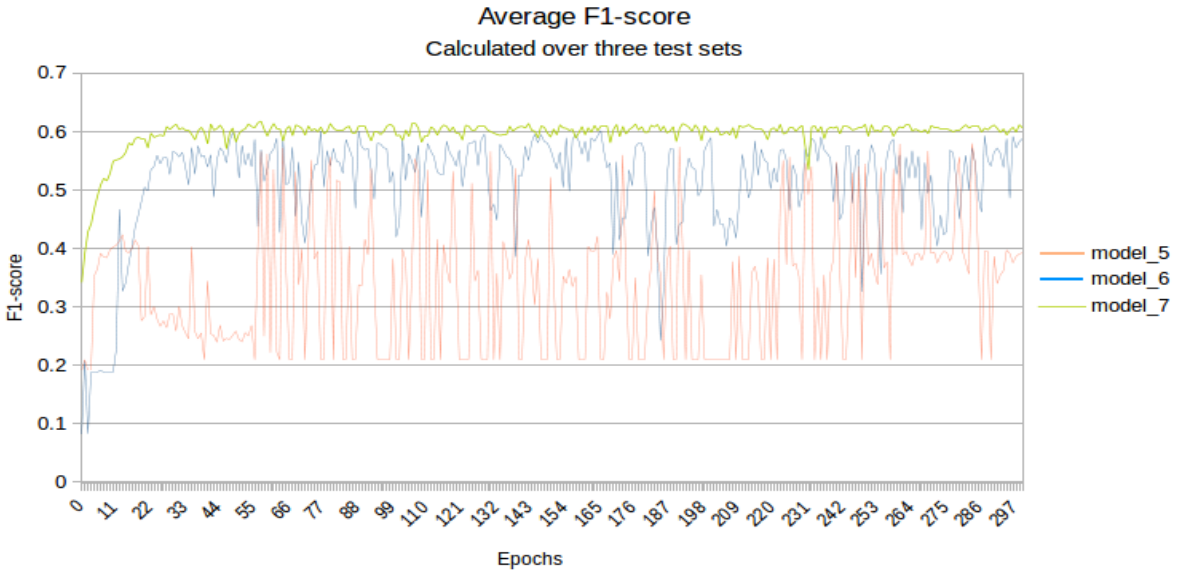


Figure 3: *Model 5&6* are deep networks with respective size 12 and 8, using mixture of *tanh* and *relu* activations with sigmoid output. One can clearly see how the increase in complexity correlates with increase in variance of performance.

3.3 Classifier 3: Fully Embedded Classifier

To try and further improve the performances, all the features were stacked together to form a 1522-dimensional vector, as can be seen from the visual representation in Fig. 4 below.

Given the successful results of the neural network on the lexicon, it was decided to try and implement another one on this newly structured data. After much trial and error varying structure, activation functions and batch sizes as for the 2^{nd} classifier, no neural network was found that could perform even close to as well. It was therefore decided to go back to the standard methods as used for the 1^{st} Classifier.

Cross-Validation was heavily made use of to finally find the test a variety of simpler classifiers and attaching them together as part of a democratic (majority vote) ensemble.
The final classifiers' results can be found in Table 2 below.

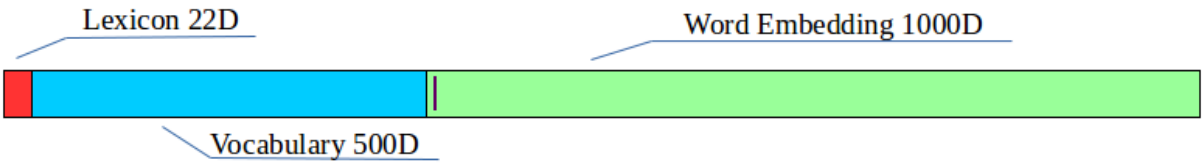
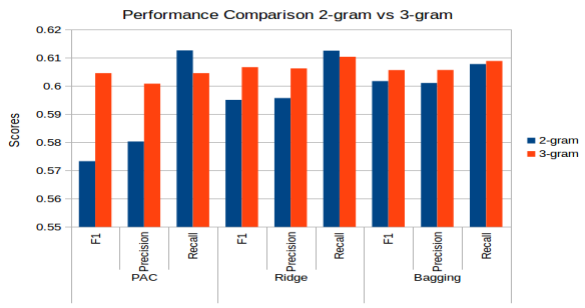
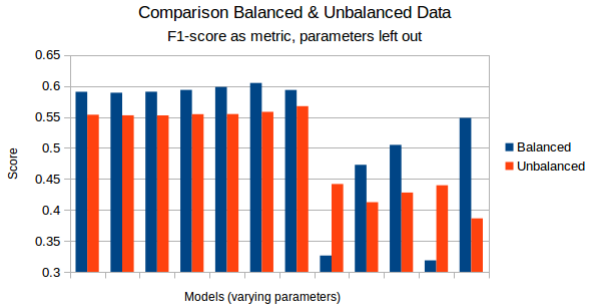


Figure 4: Visual representation of the final 1522 Dimensional vector. The numbers next to the descriptions are the dimensional sizes.



(a) Various classifiers best performances using range $1-2$ and $1-3$ *grams*



(b) Using balanced labels instead of unbalanced leads to increase in performances

Figure 5: Performance using different *n-grams* ranges (a) and balanced vs unbalanced data (b) is presented

A lot of the testing for this phase consisted in hyper-parameters, especially one that was found to be very important was the number of *n-grams* to chose in the vocabulary embedding. It was found that using the range of $1-3$ *grams* gave the best results, as can be seen from the Fig.5

Model	Parameters	F1-score
Ridge	solver: auto, $\alpha = 1000$	0.595
PAC	C=0.0001, loss: hinge	0.58
Bagging	estimator: Ridge, estimators=5, max samples=0.2	0.602
Bagging	estimator: PAC(C=0.0001), estimators=8, max samples=0.4	0.594
SGDClassifier	$\alpha = 0.01$, iter=100	0.604
Democracy	—	0.603

Table 2: Once again, democratic word was, overall, the most stable throughout the test sets, and was therefore chosen to represent the final classification.

4 Conclusion

The sentiment classification of tweets was shown to be far from a trivial task. A lot of analysis went into finding out the best features, and how to combine them together to get the best results. Many further development could be done in order to really get all the right classifiers and parameters, and combining neural network's "probabilistic" approach with classifiers in a weighted majority vote could probably allow for a few percentages increase in score. However it does appear like a boundary is reached with the current features, as can be seen from the amount of classifiers all converging to the same score up to few percents, therefore it appears further and improved feature extraction is the key to toward a better sentiment classifier.

Additionally, an honourable mention should be given to two other types of models tested, namely CNNs and LSTMs. Unfortunately I may not detail further the investigations, but the main problems seemed to be overfitting for the CNN and the computation intesive nature of LTSMs, together with its sensitivity to parameters.

References

[1] `www.cs.cmu.edu/ biglou/resources/bad-words.txt`

[2] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.