BTNLP Final Project: Twitter Sentiment Analysis

Tyler Lee and **Brisca Balthes**
Universität des Saarlandes

Abstract

This is our final project for Basic Tools for NLP where methods and concepts from the entire class come together.

1 Introduction

In this project we implement and compare different methods and tools for preprocessing a twitter dataset and for performing sentiment analysis on it. Specifically, we compare preprocessing with and without stopword removal, and we compare the sentiment analysis with NLTK/VADAR, SpaCy/Textblob, and a Hugging-Face pipeline. We find 1) that stopwords are of little relevance in our task and 2) that SpaCy/Textblob performs better than NLTK/VADAR, that both are outperformed by the pipeline, but that none of the analyses is satisfactory.

2 Dataset

The dataset is composed of 74681 tweets. Each tweet is accompanied by the four-digit ID number, the topic (e.g. "Borderlands") and sentiment ("Positive," "Neutral," "Negative" or "Irrelevant") of the tweet. Most tweets exist 2-5 times with slightly different wording. These are indicated by the same ID in the first column. Some of the tweets contain non-ASCII characters, one or more emojis and/or additional whitespace. Of the 74681 lines in the dataset, the tweet itself is missing in 686 lines. The topic and sentiment are present in all lines.

3 Preprocessing

At first, we each implemented preprocessing classes independently. Both version include removing rows with NaN values in the column of tweets, lowercasing the tweets, replacing non-ASCII characters with whitespace, removing additional whitespace and tokenizing the tweets into lists of strings. For the evaluation, we had to convert the continuous sentiment scores to the discrete numbers -1, 0, or 1, for which we chose a threshold for neutral sentiment of +/- 0.25.

Additionally, in Brisca's version rows with tweets shorter than 3 tokens are removed (removes empty tweets and e.g. single punctuation characters), as well as tweets with "Irrelevant" sentiment. The remaining sentiment labels "Positive", "Neutral", and "Negative" are converted to 1, 0, and -1, respectively, and emojis are converted to strings (before removing non-ASCII characters), such as 👍 -> ":thumbsup:".

In Tyler's version, the preprocessing extends to lowercasing and dropping NaNs in the column of topic/ entity. However, the greatest difference here is the removing of stopwords. After that, the tokens are re-joined into a single string.

4 Sentiment analysis results

4.1 NLTK/VADER sentiment analysis

After preprocessing the data set according to the process described above, we ran a VADER sentiment analysis on the tweets. We then compared the results to the gold sentiment.

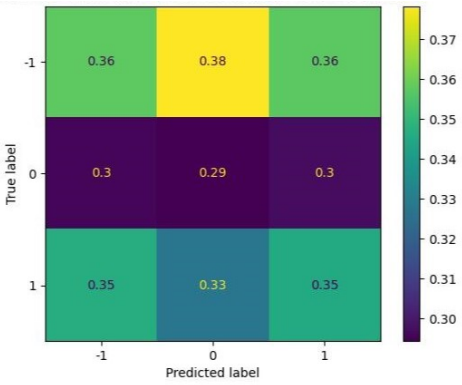
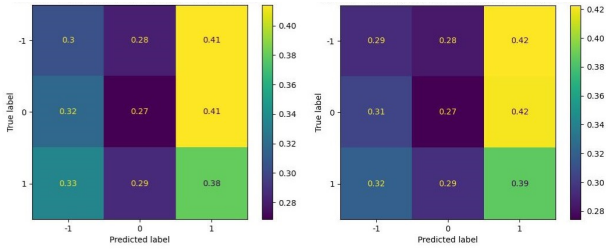


Figure 1: VADAR vs gold sentiment (col normalized)

As Figure 1 indicates, with our preprocessing,

VADER performs poorly when the gold sentiment is Neutral (0). In other words, it is inclined to classify tweets as Positive (1) or Negative (-1), even when the gold sentiment is Neutral.



(a) Stopwords included (b) Stopwords removed

Figure 2: VADER vs. gold sentiment (row normalized)

Perhaps surprisingly, the performance of VADER on this dataset does not benefit from removing stop words from the tweets. In fact, the performance is very slightly worse. Figures 2a and 2b illustrate the performance on VADER when stopwords are included and are removed, respectively. In both cases, non-ASCII characters and additional whitespace were still removed. One can see that when VADER predicts negative sentiment, it is slightly more accurate when stopwords are included than when removed. This is a quite interesting finding, and exploring it further on other datasets may provide more insight on the value of removing stopwords.

4.2 SpaCy/Textblob sentiment analysis

As a next step, we implemented sentiment analysis with SpaCy/Textblob and first compared the predictions, ignoring accuracy. Figure 3 shows that both analyzers have strongest agreement on which tweets to classify as neutral.

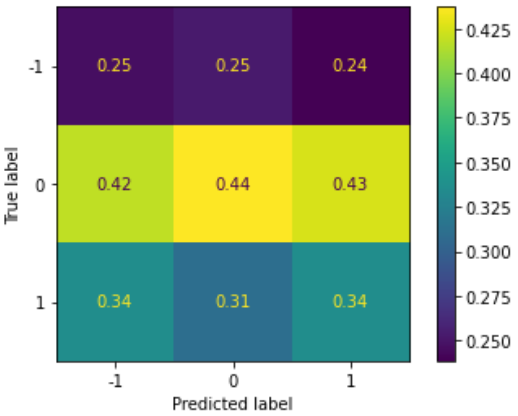


Figure 3: Agreement of NLTK/VADAR and SpaCy/Textblob predictions (col normalized)

Otherwise, we see no correlation between the predictions of both models as there is no distinct diagonal. Instead, each model has a very stable distribution of scores, irrespective of the other model's prediction. For example, NLTK will predict negative sentiment for around 25% of all tweets and neutral sentiment for around 43% of all tweets regardless of Textblob's prediction. This indicates that one or both models are not very good.

The poor, close to chance level performance of NLTK/VADAR was illustrated above, but SpaCy/Textblob actually performs much better, as can be seen from Figure 4. While it also tends to predict neutral sentiment more often, it is better at correctly predicting positive (42%) and neutral sentiment (65%) than the NLTK model. It also predicts opposite sentiment only rarely (<10%). However, it performs bad on tweets with negative sentiment, which it mostly classifies as neutral (64%). Again, we evaluated the role of removing stopwords but found only negligible differences in the numbers.

Out of the two sentiment analyzers, the Textblob model should be preferred, despite not being great. We want to point out that the threshold we set to convert the continuous predictions to discrete labels may also influence the results.

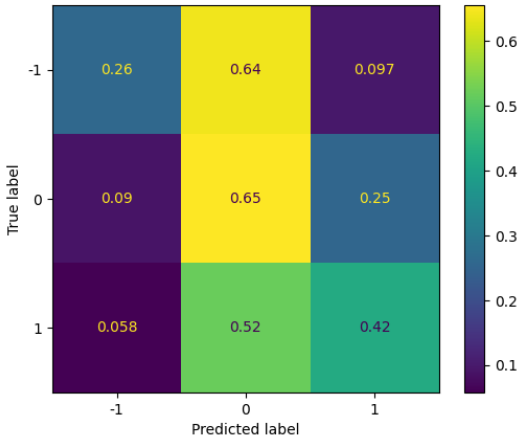


Figure 4: Textblob vs gold sentiment (row normalized)

4.3 Hugging-Face/ pipeline sentiment analysis

Finally, we implemented sentiment analysis with a Hugging-Face pipeline. The Hugging-Face model has a disadvantage in that it can only predict positive or negative, but not neutral sentiment. However, on positive and negative tweets, it performs much better than the other two models, with 71% and 85% accuracy, respectively, as illustrated in Figure 5.

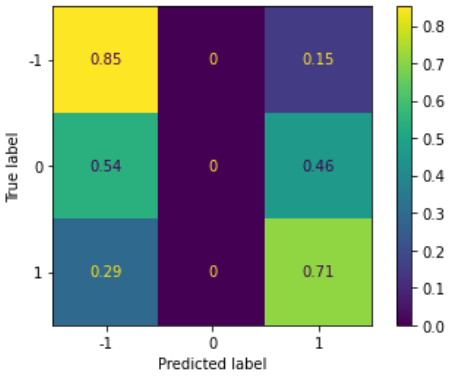


Figure 5: Hugging-Face pipeline vs gold sentiment (row normalized)

5 Comparison of the methods

NLTK provides VADER, a sentiment analyzer oriented toward social media in particular (Mogyorosi, 2021). Although its performance is better on social media, it still generalizes reasonably well to other domains (Hutto and Gilbert, 2014). It uses a sentiment lexicon, combined with five heuristics, to assess the sentiment of text (paper). To use the VADER sentiment analyzer with NLTK, one must create a SentimentIntensityAnalyzer object and call its method .polarity_scores, passing the target text as a string (Mogyorosi, 2021). This returns a dictionary whose four keys are 'neg' (negative), 'neu' (neutral), 'pos' (positive)' and 'comp' (compound) and whose values are the valence scores are those polarities. The compound score are the summed, adjusted and normalized valence scores. Therefore, the compound score is a composite of the valence scores (Hutto, 2021).

SpaCy and Textblob are both powerful NLP libraries. SpaCy offers a variety of preprocessing functionality (e.g. tokenizing and pos tagging) and seems to be frequently used in combination with Textblob for sentiment analysis. However, SpaCy itself does not seem to have built-in sentiment analysis functionality.

Textblob itself heavily depends on the NLTK and pattern libraries. In order to use Textblob, NLTK corpora need to be downloaded with textblob.download_corpora (Loria, 2018).

A Textblob object is created by instantiating a TextBlob() class object and passing a string (a sentence or an entire document) as the argument. Calling the Textblob's property 'sentiment' returns a tuple of the shape Sentiment(polarity, subjectivity). Alternatively, polarity and subjectivity can be directly accessed by calling .sentiment.polarity on the textblob object. Polarity is a float from [-1, 1] with positive numbers representing positive polarity. Subjectivity is a float from [0, 1] with a higher value representing higher subjectivity (Pappas, 2022). If passing an entire document, the above method returns one value for polarity and subjectivity each for the entire document. Alternatively, one can iterate over the individual sentences using the property .sentences of the blob object (Loria, 2018).

The inner workings of Textblob depend on the class instantiation. The parameter analyzer defaults to PatternAnalyzer(), but can also be set to NaiveBayesAnalyzer(). NaiveBayesAnalyzer returns a different tuple: Sentiment(polarity, probability_pos_polarity, probability_neg_polarity) without subjectivity scores (Loria, 2018).

The default PatternAnalyzer() is based on a dictionary of words with pos tags, disambiguated senses, and scores for polarity and subjectivity (partly hand-tagged, partly inferred) for each word. There are additional rules, such as that adverbs can intensify the score of the following word and that negation inverses polarity of the following word. Then the average value over a string of words is returned (Linguistics and Psycholinguistics Research Center).

NLTK/VADER and SpaCy/Textblob offer slightly different information for sentiment analysis. VADER provides a more detailed description of valence in addition to the composite, normalized sentiment score, whereas Textblob only offers a single polarity metric, but also provides a subjectivity metric. We imagined both of these to be useful for our final project, but alsot thought that VADER may be preferable since it is specifically oriented toward social media and tweets in particular, while the subjectivity measure provided by SpaCy was not useful for our purposes.

6 Discussion & Conclusion

Our preprocessing took some basic measures such as lowercasing the tweets, removing non-ASCII characters, etc. Since we previously found that some tweets were missing from the data set, removing lines with missing values was an important step for preprocessing. Additionally, one version of our preprocessors also eliminated excessively short tweets. Additionally, although emojis can certainly indicate sentiment, we opted to remove

them for our preprocessing. Lastly, an analysis of the effects of preprocessing revealed some surprising results. In particular, it was not expected that removing stopwords would have no positive effect on accuracy. This finding reveals the need to test our assumptions about what steps are wise for preprocessing.

VADER offers a more nuanced picture of sentiment. This is quite valuable because communication is not always categorically positive or negative and also frequently incorporates both positive and negative sentiment. Unfortunately, we were unable to evaluate VADER's performance in breaking down positive, negative and neutral sentiment due to the fact that our gold sentiment was only categorically positive, negative or neutral. Nonetheless, it is intuitively reasonable to want to break down sentiment into more than just a single label, and future investigation could help us understand better how to perform accurate sentiment analysis using VADER.

Regarding the performance of the sentiment analyzers, we found that our implementation of NLTK/VADAR with our dataset was not very useful as it performed around chance level on all categories. SpaCy/Textblob performed significantly better, though also poorly with accuracies between 26-65% depending on the category. The HuggingFace pipeline, despite only differentiating between positive and negative sentiment, outperformed both prior analyzers. However, the inability to label tweets as neutral is a major shortcoming as it forces the algorithm to decide for a false and extreme category, positive or negative. If the pipeline could be implemented in a way to include the option neutral or to output a continuous score that could be converted to -1, 0, or 1, this could be an extremely valuable tool.

These differences in the performance are likely due at least two factors: Our preprocessing choices as well as the implementation of the libraries themselves. This underlines the importance of wise and thorough preprocessing and of investigating the inner workings of off-the-shelf tools.

References

C. Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the International AAAI Conference on Web and Social Media*, 8(1):216–225.

C.J. Hutto. 2021. vadersentiment.

Computational Linguistics and University of Antwerp (CLiPS) Psycholinguistics Research Center. pattern.

Steven Loria. 2018. textblob documentation. *Release 0.15*, 2.

Marius Mogyorosi. 2021. Sentiment analysis: First steps with python's nltk library.

Alexandros Pappas. 2022. How to perform sentiment analysis in python using textblob.