COMP30027 Assignment 2: Report

Anonymous

1 Introduction

The dataset provided contains two lists of Twitter posts (tweets) made on the platform prior to 2017 (Rosenthal et al., 2017). Each tweet is an instance in the dataset. The two files enclosed in the dataset are a `Train.csv` for training and a `Test.csv` for testing. For each Tweet, included is the text and its ID. Included in the training file is also a column containing the sentiments of the tweets. Tweets can either have a "positive", "neutral" or "negative" sentiment. The goal was to develop and train an accurate classifer model using the dataset and apply it to the testing set.

2 Methodology

2.1 Raw Data Analysis

First, the dataset was looked at based on the raw data as it came. The training set contains 21802 labelled instances and the testing set contains 6099 instances. The raw tweets are strings of lowercase text and contain misspelled words, non-alphanumeric characters, Twitter-specific features (mentions and hashtags), and foreign language.

The distribution of the sentiment labels across the training set was also considered, displaying a clear majority of neutral tweets 1.

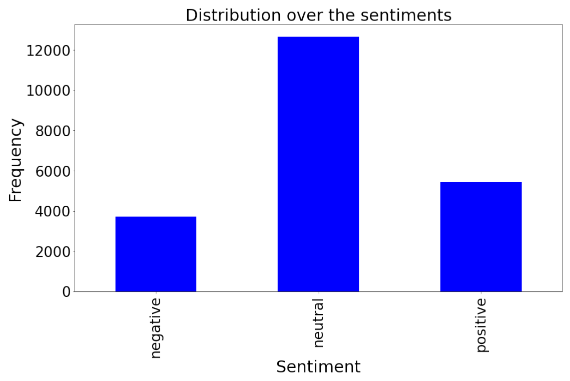

Figure 1: Distribution of the sentiments

2.2 Preprocessing

2.2.1 Cleaning

Some features required the text instances be cleaned before extraction. Cleaning involves pruning the tweet of certain characters and features to varying degrees. Usually, this means isolating only the alphabetic characters in the text, and performing simple casefolding. Since all instances in the dataset are lowercase, only removal of unwanted characters was necessary. A type of cleaning considered was the removal of repeated consecutive characters. An example where this may have been useful is when encountering words such as `hey`, which can also appear as `heyyy`. One issue with this method of avoiding suffixing is that it can change the meanings of certain words. For example, `good` becomes `god`. Extreme suffixing can also be circumvented through stemming or lemmatization, which are discussed in Sections 2.2.2 and 2.2.3.

2.2.2 Stems

Using the Porter stemmer algorithm, the roots of words can be found by removing common english stems. This was considered as an alternative to tokenising by words. However, this may be less effective with words that aren't in english (as they may use different suffixes). This method is tested ...

2.2.3 Lemmas

Lemmatization finds the true root of words based on their language definitions. Implementing lemmatization required a vast corpus of different languages, with the possibility of overlapping lemmas. The corpus recommended in `NLTK` contains only the english language. Therefore it was not used to generate results. This suffers from the same drawback as stemming, relying the text being in english.

2.2.4 Stopwords

Initially, the dataset was analyzed to see how different words are distributed throughout it.

This yielded a word cloud over both the training and testing sets, with the most common words being considered for the stopword list (Figure 2).

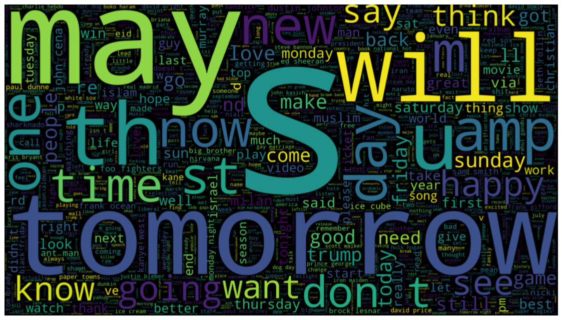

Figure 2: Word cloud over all tweets

The construction of the list was contextual, as some words that appear frequently may still be useful for sentiment analysis. To get a better picture of these common, but useful words, three more word clouds were constructed over the word lists per sentiment (Figures 3, 4, and 5).

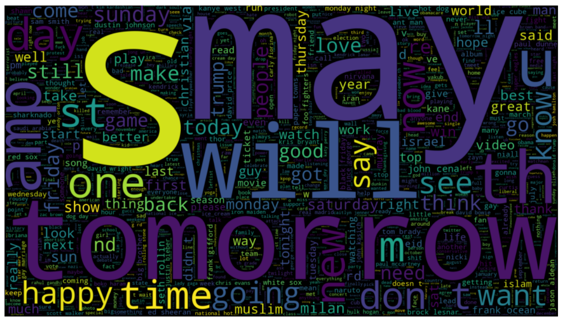

Figure 3: Word cloud over positive training tweets

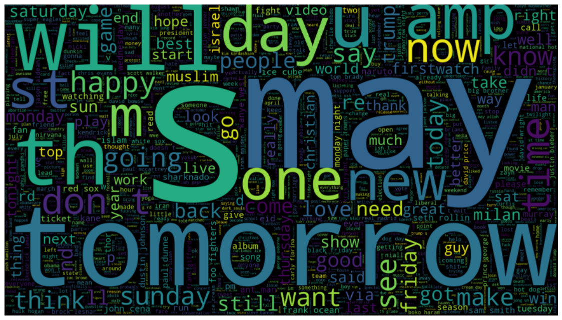

Figure 4: Word cloud over neutral training tweets

However this method did not consider stopwords in other languages (since english tweets are an overwhelming majority in the dataset). Since manual stopword list construction was not

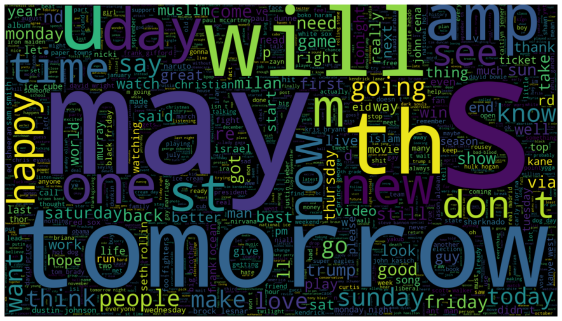

Figure 5: Word cloud over negative training tweets

as exhaustive as the data required, the Python `NLTK` module's stopword corpus is used where necessary (Bird et al., 2009).

2.3 Vectorization

Classifiers in the `SciKit-learn` Python library require that all data be vectorized to a real-valued space. The following vectorizers were used.

2.3.1 Count

Vectorising the tweet into term counts can highlight terms which appear more often in tweets for certain sentiments. This is implemented using `SciKit-learn`'s `CountVectorizer` (Pedregosa et al., 2011).

2.3.2 TF-IDF

This measure of relative word frequency provides more insight, as it measures words that appeach more often in one tweet relative to their overall frequency. This is implemented using `SciKit-learn`'s `TfidfVectorizer` (Pedregosa et al., 2011).

2.3.3 Metrics

The final type of vectorization that will occur is in the form of metrics. Certain metrics such as world length may be distributed differently based on the sentiment of the tweet. This is implemented by creating a list of mappings to metrics, then vectorizing with `SciKit-learn`'s `DictVectorizer` (Pedregosa et al., 2011).

2.3.4 Why not hashing?

The `SciKit-learn` library suggests another text feature extractor in the form of hashing. This is not used here as the distinct advantage this vectorizer has over others is saving on space and time (Pedregosa et al., 2011). While the dataset contains more than 20000 tweets, using the other vectorizers did not present such issues in practice.

2.4 Features

While the data is given as a raw text format, there are multiple features which can be extracted for the purpose of sentiment analysis.

2.4.1 N-grams

This includes extraction of individual words or characters (1-grams of each) and word pairings (2-grams). If $N > 1$, important orderings/configurations of words can be identified, but at the cost of exponentially increasing the possible number of features. With a vocabulary of w unique words, there can be as many as w^N distinct N-grams. Therefore, the N-grams used for model construction were 1-grams on words, 1-grams on characters, and 2-grams on words. 2-grams on characters were not considered

This method will requires that tweets are cleaned, but may not need stopword removal.

2.4.2 Word Lengths

A tokenization where the tokens generated are the lengths of the words in the tweet.

2.4.3 Character Frequencies

2.4.4 Links

2.4.5 Hashtags

2.4.6 Mentions

2.4.7 Emoticons

2.4.8 Simple Metrics

2.4.9 Phonetic Frequencies

2.4.10 Poetic Phonetics

2.5 Model Selection

2.5.1 Feature Selection

2.5.2 Classifiers

2.5.3 Evaluation Metrics

3 Results

4 Analysis

5 Conclusions

References

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada, August. Association for Computational Linguistics.