Sentiment analysis on self-driving car Tweets using different NLP methods

Dharani Doppalapudi and Adam Ratzman

Abstract

Traditional approaches for modern researchers use sentiment analysis tools like VADER to analyze and predict text sentiment given an existing library of trained text. Instead, we attempted to create our own sentiment analysis models using 1) a bag of words approach, 2) converting tweets to a ‘positivity score’ using the MPQA subjectivity lexicon, 3) a bag of words approach as well as the MPQA subjectivity lexicon, and finally 4) using dependency triples. We used the NLTK and scikit-learn libraries to build these models in Python. We found that all four approaches ranged significantly in their effectiveness, though all ranged in their score between .55 and .64. We compared these scores with predictions using VADER and naive bayes, and found that all our models except the subjectivity lexicon did better. After this project, we can conclude that though our approach was not very effective, using a bag of words or dependency triples does extract enough information from Tweets to make more informed predictions.

Keywords:

NLTK, scikit-learn, machine learning, sentiment analysis, Twitter

Introduction

As avid social media users, we are interested in extracting information from user posts. Sentiment analysis is an increasingly important tool in data science, allowing us to see what people think about topics of interest over time. We were interested in how we could use what we learned in class, along with knowledge we picked up in our own research for this project, to perform sentiment analysis on existing Tweets.

We decided to use CrowdFlower's self-driving car sentiment dataset, which consisted of 7,156 Tweets labeled by their sentiment. We decided to choose this dataset because it did not have many existing public projects exploring its data, and additionally because we are ourselves interested in self-driving cars, so it was an interesting topic to explore for us.

Methods

We made a deliberate decision to avoid using any existing sentiment analysis tools and instead to use what we had learned in class to create our own models in scikit-learn. In addition, we decided to modify the sentiment scores. Originally, the dataset was given from 1-5, with 1 being the most negative sentiment in a Tweet, 3 being neutral, and 5 being the most positive, as well as -1 for not relevant. Given our relatively small sample size (only about 7,500), and the small proportion of tweets with sentiments of 1 and 5, we decided to condense 1-2 and 4-5 together, creating four sentiments: 1, a negative Tweet, 2, a neutral Tweet, 3, a positive Tweet, and -1, a not relevant Tweet.

We wanted first to explore a basic approach using a bag of words, with variable ngram ranges from just unigrams to unigrams, bigrams, and trigrams. Using a bag of words for textual analysis is a classic approach in machine learning, due to the properties of a bag of words. A bag of words disregards the placement, syntax, and semantics of the text, just counting the number of instances of words in the text (Brownlee, 2019). Because of this, it is simple to create a vocabulary of words using training data and be able to create a feature vector of instances of this vocabulary. Our bag of words, however, has some modifications compared to a traditional bag of words. Firstly, it uses nltk's part-of-speech tagger to create POS tags for all words in the Tweets. It then filters out all non-noun, adjective, or verbs. Afterwards, it uses nltk's lemmatization utility to use the lemmatized versions of these nouns, adjectives, and verbs in the bag of words. Lemmatization is the process of reducing stemmed or otherwise modified words into their base form, such as the first-person singular form of to be, am, into be (Manning, Raghavan, and Schütze, 2008). We then use scikit-learn's TdidfVectorizer, purpose-built for bag of words, to create feature vectors for the Tweets, and train a SVC model. We tried three ngram combinations: unigrams, unigrams and bigrams, and unigrams, bigrams, and trigrams, to see whether any of these combinations performed better than the traditional unigram.

However, short social media posts that can include hashtags, links, mentions, and other unique words may harm a bag of words approach, as the frequency of misspellings, slang, and other modifications to standard English are higher in these kinds of posts. Because of this, we decided to use the MPQA subjectivity lexicon in our second approach and see how it compared with the bag of words approach. The MPQA subjectivity lexicon is a list of words rated by their polarity (negative, positive, or neutral), which we used to create a positivity 'score' for each Tweet in our dataset (Wilson, Wiebe, and Hoffmann, 2005). We then used scikit-learn's DictVectorizer to vectorize this into feature vectors and train a SVC model.

We wanted to see whether combining these first two approaches would improve the classification accuracy. Because we were having difficulty with the traditional FeatureUnion/ColumnTransformer approach in scikit-learn to combine separate kinds of features, we ended up appending each instance of positive, negative, or neutral, to the lemmatized Tweet sentence. For example, if the Tweet "self-driving cars are amazing and cool" contained 2 positive words, we would turn this into "self-driving cars be amazing cool positive positive," and we hoped that some additional subjectivity value in the resulting TdidfVectorizer feature vector would be able to be extracted.

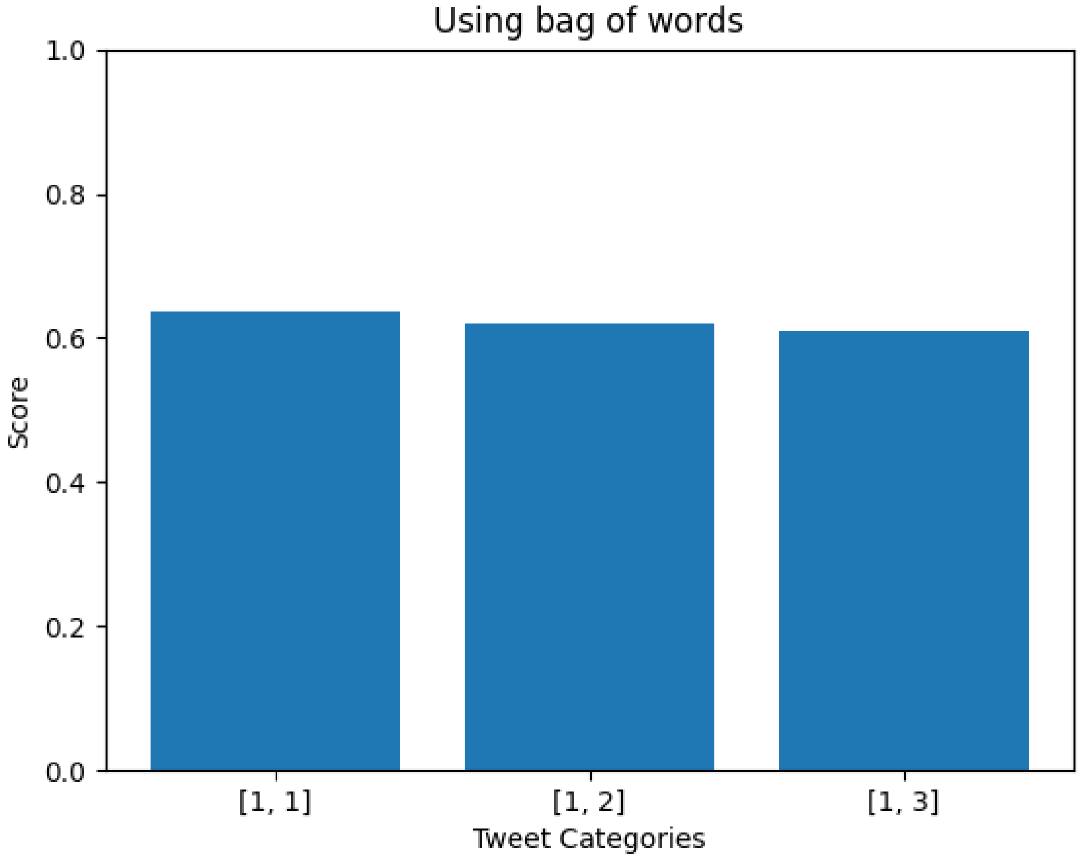
Lastly, we wanted to see whether dependency triples would be a better predictor of Tweet sentiment than a bag of words. We used MaltParser for this, passing in the raw Tweet text to nltk's MaltParser binding to retrieve dependency triples for each sentence in each Tweet (Nivre, Hall, and Nilsson, 2006). As MaltParser only accepts singular sentences, and Tweets can be multiple sentences, we had to convert each Tweet into its constituent sentences, run them through MaltParser, and then reconstruct the entire Tweet's dependency triples afterwards. We then create feature vectors from these dependency triples, and then train a SVC model.

Results

We compared all of our results to what the result would have been if we had used a Naive Bayes approach, as well as the python VADER library, which contains a sentiment analyzer (Hutto and Gilbert, 2014). The bag of words with subjectivity lexicon outperformed the other three approaches, followed by bag of words without subjectivity lexicon, and then dependency triples.

In comparison to the other three approaches, only using the subjectivity lexicon received a lower score, and was even lower than the Naive Bayes score. The subjectivity lexicon received a score of 0.549, which is lower than both the Naive Bayes (.550) and VADER score (.569). The dependency triples approach had a score of 0.584.

The other approaches, the bag of words with subjectivity lexicon, had a score of 0.648, and the bag of words without subjectivity lexicon had a score of 0.637 with unigrams, 0.620 with unigrams and bigrams, and 0.610 with unigrams, bigrams, and trigrams. These two approaches scored significantly higher than the other two and predicted almost two out of every three Tweets correctly. The bag of words scores are plotted below.



Caption: Comparison between bag of words performance with only unigrams [1,1], unigrams and bigrams [1,2], and unigrams, bigrams, and trigrams [1,3]

Discussion

We had low expectations about how our approaches would perform before implementing them, and so we were pleasantly surprised at the success of the bag of words model and the bag of words with subjectivity lexicon model. We expected that these would perform decently, though not great, and the bag of words with unigrams beat out the Naive Bayes and VADER approaches by more than 0.05, attesting to the strength of that approach. We were a little surprised that also adding bigrams and trigrams to the bag of words actually reduced the performance of the model; we were expecting an opposite result, and we would be interested to look more into why this is occurring.

It was initially surprising that the subjectivity lexicon performed very poorly, but after analyzing the Tweets, we realized that the kind of speech that was common among the Tweets was prone to have slang, misspellings, or newer words, which may have contributed to not many found words from the lexicon, which led to poor results in this model. We also did not expect the dependency triples approach to work very well, and felt vindicated when it slightly overperformed Naive Bayes.

Overall, we demonstrated that we could successfully train several kinds of models and use different features to try to complete our task. We ended up with a moderately successful model, and though there are several improvements that could be made (including more training data), we are happy with the results we obtained. The bag of words model with subjectivity lexicon is successful enough to be used to relatively accurately predict sentiment for Tweets on this topic.

References

Brownlee, J.. (2019, August 7). A gentle introduction to the bag-of-words model. Machine Learning Mastery. Retrieved December 5, 2021, from https://machinelearningmastery.com/gentle-introduction-bag-words-model/.

CrowdFlower. (2016, November 21). Sentiment self-driving cars - dataset by Crowdflower. data.world. Retrieved December 6, 2021, from https://data.world/crowdflower/sentiment-self-driving-cars.

Hutto, C.J. & Gilbert, E.E. (2014). VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014.

Manning, C., Raghavan, P., and Schütze, H. (2008). Introduction to Information Retrieval, Cambridge University Press.

Nivre, J., Hall, J., and Nilsson, J., (2006, May). Maltparser: A data-driven parser-generator for dependency parsing. In LREC (Vol. 6, pp. 2216-2219).

Wilson, T., Wiebe, J., and Hoffmann, P. (2005). Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. Proc. of HLT-EMNLP-2005.