Sentiment Analysis of Movie Reviews

Emily N. Mech

1 Introduction

Sentiment analysis is a computational approach to classify a text into a category (e.g., negative, neutral, positive) according to the sentiment expressed within the text. For example, reviews of movies, restaurants, and products contain opinions that can be negative, neutral, or positive, and it is the goal of sentiment analysis to be able to correctly classify the opinion expressed in the review. While it may be a seemingly trivial task for a human to categorize a review as positive or negative, there are many challenges to create an automated system that can perform as well as humans. However, in spite of these challenges, the potential benefits of a performant sentiment analyzer are numerous. For example, it is important for companies to stay apprised of the polarity of their product reviews, and it would be beneficial to flag particularly positive reviews to use in marketing or to address negative reviews with action from the customer service team. An automated system would make this process more efficient and cost effective. Accordingly, developing state-of-the-art sentiment analysis is an important area of research in computational linguistics today with potentially widespread applicability in research and business settings.

2 Problem Definition

In computational linguistics, a sentiment analyzer is a program that receives a text as input and that outputs a category of opinion or sentiment that the text expresses. The exact categories with which to classify the text is based on the goals of the researcher. For example, it may only be necessary to classify movie reviews into positive, negative, or neutral categories to understand whether a movie is generally liked or disliked (Yessenov and Misailovic, 2009). However, if the goal is to understand the emotion expressed by an author of a text, it may be important to classify the text into several different emotion categories such as disgust, anger, and fear (Mohammad and Turney, 2010). With a supervised learning approach, the sentiment analyzer is trained on texts that have been classified according to human sentiment (or emotion) ratings, and it is tested on a test set that is not labeled. The performance of the analyzer on the test set demonstrates how well it is able to capture the features of the text that are important for classifying the sentiment within it. Unsupervised approaches can be used to see which types of texts cluster together and may contain language expressing similar sentiment.

3 Previous Work

Previous work in sentiment analysis has experimented not only with the classification categories used to analyze the sentiment in a given text, but has also tested different models, text representations, and features to better understand which representations are most beneficial for the analysis. For example, even constraining the input to movie reviews, there are several methods that researchers have used to classify sentiment.

Singh and colleagues (2013) classified movie reviews based on aspect level information. In their approach, they identified went through each review sentence by sentence and identified content that contained an opinion. Additionally, they explored words within a 5-gram range to find any adjective or adverb content which could provide additional sentiment information. They then looked up this content in the SentiWordNet library and assigned the words the corresponding score from the library. Their system achieved a best performance of 78% on their movie review database and was fast to implement (Singh et al., 2013).

Deep learning approaches to sentiment analysis have become increasingly popular as sentiment

classifiers become more sophisticated. There are many deep learning models that may be appropriate for sentiment analysis. However, Shirani-Mehr (2014) compared several algorithms and found that recurrent neural networks did not much outperform the baseline Naive Bayes algorithm. Recursive networks did better than recurrent networks, but the CNNs provided the best architecture with which to classify sentiment in text (Shirani-Mehr, 2014).

However, authors are going beyond even single architecture deep learning approaches in favor of hybrid models that take advantage of the strengths of different architectures. For example, Rehman and colleagues (2019) utilized word embeddings from Word2Vec and then used an LSTM to extract deeper lexical semantics and long distance dependencies and a CNN to further refine the word embeddings. This hybrid model achieved 91% classification accuracy on two benchmark movie reviews datasets and was able to harness the strengths of each of the model architectures to analyze sentiment in movie reviews (Rehman et al., 2019).

4 Approach

In the current experiment, a Logistic Regression, Decision Tree, and Multilayer Perceptron Model were tested with raw and cleaned input text, baseline, intermediate, and improved features, and differing feature vector representations. First, a baseline system was constructed in which a bag-of-words approach was implemented. The input to this system was either the raw review data that was wholly unprocessed or data that had been cleaned with the following steps:

1. Remove newline character
2. Lowercase all tokens
3. Remove double dashes (–)
4. Remove digits
5. Remove stand-alone punctuation
6. Remove stop words (but spare negation words)
7. Replace words with Porter stems.

While all of these steps were utilized in to create the clean text input to the model, there is functionality built in to the system to easily allow the user to specify whether cleaning steps 3-7 should be implemented. However, in the final models, all of these steps were implemented as they allow (to varying degrees) overlapping representations. For example, a lowercase and uppercase letter within the same word does not change the meaning of that word. Using a uniform capitalization scheme should help the model to learn the same representation for the same words regardless of how they are written. Additionally, stop words are typically considered to not carry a lot of semantic information - removing them should help remove some noise from the text. Finally, words with the same stem likely have a largely similar meaning. By employing stemming on the data, these same meanings should have the same stems in the model with the goal of improving the semantic representations in the model.

Several machine learning models are suitable for sentiment analysis. However, given that the movie review database used to train the model was labeled, I tested three different supervised learning algorithms. First, a logistic regression (i.e., logit) classifier was used that classified a review as positive or negative based on cutoff values along the logistic function. This classifier has built in regularization to which helps to prevent overfitting. Second, a decision tree classifier was tested. This classifier uses binary recursive partitioning to split data first into partitions and then into branches. The classifier learns simple decision rules which allow it to step through the branches and make a final classification. Finally, I implemented a multilayer perceptron neural network model to classify the reviews. This model consists of an input layer, hidden layers, and an output layer. The hidden layer allows the model to learn non-linear representations of the input to make a classification decision. Overall, the best fitting models were the multilayer perceptron models. These models likely fit the best as some non-linearities may have been present in the input which were important to classify the reviews as negative or positive.

To improve my baseline model, I first considered changing the vector representation of the words in the model. I experimented with either count vectorization or term frequency-inverse document frequency (TFIDF) vectorization to represent the data. Count vectorization gives a lexical representation based simply on frequencies of the features in the input. However, TFIDF vectorization seeks to develop lexical representation based on the over-

all document level. In essence, TFIDF vectorization penalizes the most frequent words and allows potentially important but less frequent words to have greater weight. Testing this representation with both the raw and cleaned text allows us to see whether important semantics can come from lower frequency words in the reviews.

Additionally, I added in 3 features to the text representation based on the NRC Valence, Arousal, and Dominance lexicon (VAD lexicon) (Mohammad, 2018). Specifically, for each movie review, I looked up the valence of all of the words in the text. I then counted the number of positively valenced words (operationalized as a rating greater than .5), the number of negatively valenced words (a rating less than .5), and I calculated the average valence rating of the review. These features were designed to give an index of the overall amount of positive and negative vocabulary in the review, as well as the balance of negative to positive language in the review. The goal of creating these features was to identify particularly valenced terminology within the review, assign it's polarity, and measure the balance of the valenced terms in the model. The baseline model does not contain any of these features, the intermediate model simply includes the counts of the positive and negatively valenced words in each review, and the improved model contains the counts as well as the average valence ratings of each review.

The size of the various feature sets interacted with the textual representations to influence the performance of the sentiment classification. For example, the best performing model that used TDIDF vectorization on cleaned text did the best on the baseline set of features (i.e., no added features from the VAD lexicon). This model may have done better than it's equivalent model with intermediate/improved features because the TFIDF vectorization may have been better at highlighting the important words with valenced context than the simple look-up counts. However, the best overall performing model used count vectorization and an intermediate feature representation - features representing the counts of positive and negative words in the reviews. In this case, the model learned frequency information for all of it's features, and was potentially able to use the extra features as another source of informative frequencies with which to classify the text. It's possible that the average rating feature did not improve the best performing model as it was a different type of information (not a count) than the rest of the features. However, the equivalent model with count vectorization and baseline features performed more poorly than the model with improved features.

5 Results

The best fitting model for this dataset was the multilayer perceptron model that used cleaned input text, count vectorization, and positive and negative valence counts as additional features. It achieved an F1 score of 87%, a precision score of 57%, a recall score of 48%, and an accuracy score of 79% (for a full report of model scores, see results.csv). Additionally, looking at the pattern of results across all models, it seems that the multilayer perceptron model performed very well with cleaned text input, and the logistic regression classifier did particularly well with count vectorization. The decision tree model did not do particularly well, but did outperform the logistic regression models and multilayer perceptron models that had input or representations that were ill suited for the algorithms (i.e., TFIDF for logistic regression, raw input for multilayer perceptron model).

It is likely the case that the best fitting model was particularly adept at learning frequencies in the input and finding many possible interactions with frequency that supported successful classification. This may be why the cleaned input text was important for the best fitting multilayer perceptron models - it allowed words with similar meanings that may have had slightly different textual representations to be considered the same and to add to the frequency counts. Additionally, while POS representations were not specifically given to the model, adding the counts of the positive and negatively valenced words may have served an additional purpose of highlighting how much "content" information was present in a given review. This may have helped the model classify the reviews. Finally, it is likely that the decision tree models did not do as well as any given feature in this experiment was not particularly influential. Therefore, the probabilities and interactions among features were likely important for classification, leading to better performance in the logistic regression and multilayer perceptron models.

6 Discussion and Conclusions

Sentiment analysis is an exciting area for research to improve textual and feature representations to be able to recover the intended sentiment within a text. In this experiment, I tested different combinations of textual and feature representations by adding features and comparing models with and without the additions. I found that the best model was the multilayer perceptron model that was trained on clean text with a count vectorization representation and with additional features from the VAD lexicon. However, it is likely that this model could be further improved by pre-training it on word embeddings from a model such as Word2Vec, and by adding and improving features. For example, negation could be more explicitly represented by a feature as could long distance dependency information. Further, it is clear that the input and textual representations can be tailored to best fit the model's algorithm. In future research, more exploration of the best pre-processing approaches would be important to gain the biggest increases in performance. Finally, combining different deep learning algorithms may give the biggest benefit to classification as it allows one to take advantages of the different strengths of different models.

References

Saif Mohammad. 2018. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 174–184.

Saif Mohammad and Peter Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 26–34.

Anwar Ur Rehman, Ahmad Kamran Malik, Basit Raza, and Waqar Ali. 2019. A hybrid cnn-lstm model for improving accuracy of movie reviews sentiment analysis. *Multimedia Tools and Applications*, 78(18):26597–26613.

Houshmand Shirani-Mehr. 2014. Applications of deep learning to sentiment analysis of movie reviews. *Technical report*.

Vivek Kumar Singh, Rajesh Piryani, Ashraf Uddin, and Pranav Waila. 2013. Sentiment analysis of movie reviews: A new feature-based heuristic for aspect-level sentiment classification. In *2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*, pages 712–717. IEEE.

Kuat Yessenov and Saša Misailovic. 2009. Sentiment analysis of movie review comments. *Methodology*, 17:1–7.