Sentiment Analysis of Movie Reviews Using the Word2Vec Algorithm

Emily Prewett and Lily Eckhardt

Natural Language Processing (NLP) is a form of artificial intelligence that attempts to allow computers to parse and communicate human language (Stryker). The most well-known and popular application of NLP is generative AI such as ChatGPT, where large amounts of language data is used to train a model to be able to respond to prompts and come up with human sounding text. There are also many other applications of NLP, with the focus of our project being on predicting if the overall sentiment of a given movie review is positive or negative over a large dataset. This is an example of "sentiment analysis", which will be explained further in the motivation section.

Our project is a solution to the Kaggle Competition "Bag of Words Meets Bags of Popcorn", and since it is a "Getting Started" challenge there are many known ways to approach performing sentiment analysis on the dataset provided. Since we both came into this project with no previous experience in NLP applications, we decided to pursue a fairly simple approach. As will be further described in the methodology section, we used Google's Word2Vec algorithm to encode meanings to words in large vectors and used the average values of these vectors to train a random forest model which ultimately predicted the review's overall sentiment. We then performed these steps on all of the 25,000 movie reviews in the provided test set, and submitted our predictions to the competition page to get a final accuracy score.

To get started, it’s important to understand the components of this competition. Sentiment analysis is an application of NLP that in our case seeks to take in movie review text and return a prediction for whether the review has an overall positive or negative "sentiment" (De La Cruz). Determining if a movie review is overall positive or negative in its assessment of a film is a very simple task for a human to perform across a relatively small number of reviews. However, if a data science approach can automate this process with a high accuracy, it can be easily used in all sorts of other applications. It is important to note that applying sentiment analysis in this way to movie reviews has relatively little value in most real world cases, since movie reviews tend to come with some numeric rating that accurately encodes sentiment. To give some examples of where an approach similar to ours could be valuable, however, consider a case where a political party wants to gauge public opinion on a specific candidate. Since it is relatively simple to scrape large numbers of posts from many social media sites, sentiment analysis could be used to give an idea of how popular a given political candidate is.

In the description of the Kaggle competition, a deep learning algorithm called Word2Vec is prominently featured as a possible solution to the problem. We decided to focus on project on implementing Word2Vec in this problem instead of using other methods like bag-of-words because we thought it would be interesting to see what the cutting edge deep learning approaches to NLP were like back in 2013. Word2Vec is an algorithm developed by Google that "provides an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representations of words" (Google). The continuous bag-of-words architecture is an approach where a single word is predicted based on the surrounding context words, and the skip-gram architecture is where the context words are instead predicted based on a single word (Trivedi). We tried using both architectures in our testing, but we decided that the differences between them on our dataset were small enough to just pick one which ended up being continuous bag-of-words. Since we spent most of our work focused on Word2Vec, our decisions for transforming the Word2Vec output vectors into usable input for a classifier and the choice of Random Forest itself were simply motivated by being simple solutions to fit our constrained timeframe.

For our project we were provided with 3 datasets, “labeledTrainData.tsv”, “unlabeledTrainData.tsv”, and “testData.tsv”. After running the necessary Python libraries, these data sets are then loaded in for data reprocessing. labeledTrainData.tsv is a file that contains 25,000 rows of a movie’s ID, text review, and binary sentiment rating (0 for a negative sentiment, 1 for a positive sentiment). unlabeledTrainData.tsv is an extra training data set that contains 50,000 additional unlabeled movie reviews. Finally, testData.tsv is a testing data set that contains 25,000 rows of a movie’s ID and text review. This set does not include the sentiment rating, as we aim to predict that value and will receive a Kaggle accuracy rating upon submission.

To process the data, the HTML tags and special characters are removed and replaced with spaces. Then, all of the text is transformed to lowercase and split by individual word into tokens. To wrap everything up, stopwords (e.g., “the”, “or”, “and”) are removed to reduce noise and focus processing on significant words.

One we have individualized words, they can go through our Word2Vec model, which analyzes the relationships between words. Each word is represented as a 100-dimensional vector and trained using the Continuous Bag of Words algorithm with a context window of size 10. As a result, a value is assigned to each word. After this, the values of every word in a review are averaged to generate a single value for that review.

Following this, we implemented a Random Forest Classifier that is trained on these reviews to predict future sentiment. The data is split into an 80% training set and 20% validating set, run through our classifier, and the model adapts based on its accuracy. The accuracy of this model is printed to examine how well the model works. In our case, the accuracy model continuously reported a value in the range of 82%-86%, which changes as a result of how our data is split.

With this code, we were able to process our testing data set. The testing data set is similarly cleaned, tokenized, and averaged. Then, it runs through the Random Forest Classifier model to get predictions for the review’s sentiments and formatted to be a csv file.

The csv we obtained was saved in the GitHub as “kaggle_submission.csv”. It is a csv file that contain the ID for every movie in the testing data set and their predicted sentiment value. When submitting this to Kaggle, the sentiment values are compared to the actual sentiment for that review and measured for accuracy. We received a final accuracy rating of 82.75%.

Kaggle Project Link: https://www.kaggle.com/competitions/word2vec-nlp-tutorial/overview/description

De La Cruz, Robert. "Sentiment Analysis Using Natural Language Processing (NLP) | by Robert De La Cruz | Medium." Medium, Medium, 18 Dec. 2023, https://medium.com/@robdelacruz/sentiment-analysis-using-natural-language-processing-nlp-3c12b77a73ec.

Google. "Word2vec." Google Code, https://code.google.com/archive/p/word2vec/. Accessed 26 Feb. 2025.

Stryker, Cole, and Jim Holdsworth. "What Is NLP (Natural Language Processing)? ." IBM, https://www.ibm.com/think/topics/natural-language-processing. Accessed 26 Feb. 2025.

Trivedi, Ayushi. "Understanding Continuous Bag of Words (CBOW)." Analytics Vidhya, https://www.facebook.com/AnalyticsVidhya/, 29 Nov. 2024, https://www.analyticsvidhya.com/blog/2024/11/continuous-bag-of-words-cbow/.

Otten, Neri Van. "Top 3 Easy Ways to Remove Stop Word in Python with SpaCy, NLTK & Gensim." *Spot Intelligence*, 10 Dec. 2022, spotintelligence.com/2022/12/10/stop-words-removal/.