**Classification and Sentiment Analysis of Rotten Tomatoes Movie Reviews**

Eunice Kang

*Claremont Graduate University*

*eunice.kang@cgu.edu*

# Abstract

*For the classification of sentiment reviews, feature engineering generally consists of creating n-gram features for the classification models. However, n-grams do not incorporate sentiment information. In this paper, a sentiment score (or Sentiscore) is derived from the SentiWordNet lexicon as a feature to attempt to mollify this issue. A corpus consisting of Rotten Tomatoes movie reviews is used to train two classifier models. The models are created using different feature sets: one of solely n-gram vectors, n-gram vectors with the Sentiscore, and finally, n-gram vectors with the binned Sentiscore. The results further validate that using n-grams as features is a good methodology for sentiment classification as it creates models with good classification accuracy. In addition, the results also demonstrate that feature engineering is a challenging task and great deliberation goes into creating significant features during the feature engineering process.*

**Keywords:** natural language processing, sentiment analysis, classification, feature engineering

# Introduction

Sentiment analysis is the process of identifying and extracting opinions from natural language using natural language processing tools and techniques [1]. It is also known as opinion mining. There are two types of sentiment analysis: subjectivity classification, which is classifying whether the text is objective or subjective, and polarity classification, which is classifying whether the text is positive, negative, or neutral.

Sentiment analysis is helpful in marketing analysis, such as product feedback and customer satisfaction data, because organizations can analyze and use the sentiment of their customers to improve their products [1] [2].

Furthermore, there are two broad approaches to sentiment analysis [2]. First, is a lexicon based approach, which utilizes sentiment lexicons, such as SentiWordNet, SenticNet, or VADER (Valence Aware Dictionary for sEntiment Reasoning) to extract the polarity from the text [2]. The second approach is a machine learning based approach, in which a classifier model, such as a Logistic Regression model, Naïve Bayes, or Support Vector Machine (SVM), is trained on features extracted from the text in order to classify the sentiment.

In order to extract the input parameters or features for the machine learning based approach, feature engineering needs to take place. Feature engineering is the process of generating or deriving attributes from raw data or a corpus [3]. The challenge of feature engineering is that creating manual features is time consuming, and it is difficult to derive significant features. Many methods propose using n-grams as features using a vectorizer, such as TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer or Count Vectorizer [2]. An n-gram is a sequence of *N* number of words; however, using n-grams as features does not include sentiment information. Thus, features can also be derived from sentiment lexicons to incorporate sentiment information.

This paper discusses a project conducted on classifying the polarity sentiment of 480,000 labeled Rotten Tomatoes reviews using the machine based approach mentioned above. First, two classifier models, Logistic Regression and Support Vector Machine (SVM) models, are trained solely using n-gram features using the TF-IDF Vectorizer. Next, a feature derived from SentiWordNet, a sentiment score, is added to the n-gram features and used to train the two classifier models. Finally, the results of the two parts are then compared: the classifiers solely trained on n-gram features and the classifiers trained on n-grams with the sentiment score.

# Literature Review

As mentioned in the above section, vectorizers are generally used to create n-gram features for text classification problems [2]; however, the issue is that n-grams only include the context of words, but leave out sentiment information. To bypass this problem, several researchers have come up with solutions to include sentiment information into their classification models.

For example, Dey et al. [2] proposes multiplying the TF-IDF rating for each feature (n-gram) by a sentiment score derived from multiple sentiment lexicons: VADER and SO-CAL (Semantic Orientation CALculator). This essentially incorporates a weight to each n-gram based on its sentiment information and be used to classify text from cross-domains.

Another group of researchers, Tang et al. [4] addresses this issue by applying Sentiment-Specific Word Embedding (SSWE), which encodes the sentiment information in the vector representation of the text. They build three neural networks to learn SSWE to classify the polarity sentiment of Twitter tweets.

To address the issue of including sentiment information into the modeling process, this project creates a sentiment score from SentiWordNet and includes it as a part of the feature set for two classifier models.

# Experiment

The pipeline for this project is as follows: (1) obtain the Rotten Tomatoes movie reviews corpus, (2) preprocess the corpus to be suitable for analysis, (3) feature engineering, (4) build the classifier models, and (5) evaluate the models.

## Corpus

The corpus for this project is obtained from the Kaggle website [5] as a .csv file format. It consists of 480,000 labeled reviews from the Rotten Tomatoes website. Each review is a snippet from a full review written by an accredited critic and can be as short as a single sentence or as long as a short paragraph. Half of the reviews are also labeled as “Fresh” (i.e. positive) and the other half as “Rotten” (i.e. negative).

The table below shows a sample of what the data looks like:

|  |  |
| --- | --- |
| **Freshness** | **Review** |
| rotten | Lensed with skill to no particular end. |
| fresh | As you watch "Fences," there's never a doubt that these lives matter, and that's a good and noble thing, but you're also aware (maybe too aware) of how much the movie itself wants to matter. |

**Table 1***:* Rotten Tomatoes movie reviews corpus sample

To preprocess the corpus, all of the text is converted to lowercase and punctuation and symbols are removed using regular expressions. In addition, several stopwords are specified for removal. The specified stopwords are: ‘in’, ‘at’, ‘of’, ‘a’, ‘an’, ‘the’, ‘and’, and ‘it’. The Natural Language Toolkit (NLTK) [6] stopwords corpus is not used for the stopword removal, because some of the defined stopwords in the nltk corpus might show significance in polarity classification, especially negation words, such as ‘no’ or ‘not’. The specified stopwords are fed into the TF-IDF Vectorizer [7] as a parameter.

Furthermore, the TF-IDF Vectorizer is also given the WordNet Lemmatizer as a tokenizer. The WordNet Lemmatizer tokenizes and lemmatizes each review before vectorization.

## Feature Engineering

To create features for the models, TF-IDF Vectorizer [7] from scikit-learn [8] is used to create n-gram features from unigrams to trigrams. For each feature, the vectorizer gives a score of how important the feature is to the document in relation to the entire corpus [9]. Rather than taking the simple frequency of each feature for each review, the TF-IDF Vectorizer rescales the frequency by the frequency across the entire corpus. As mentioned previously, the WordNet Lemmatizer and specified list of stopwords are given as parameters during the initialization of the TF-IDF Vectorizer to preprocess the data before vectorization. The TF-IDF Vectorizer returns a sparse document-term matrix [7].

The SentiWordNet corpus from NLTK [6] [10] is used to create a feature, a sentiment score or Sentiscore, to incorporate sentiment information into the modeling phase. A method is created which takes a single review, tokenizes it, and tags each word with its corresponding parts-of-speech (POS) tag using the default NLTK POS tagger, which uses the Penn Treebank tagset [6]. The method then iterates through each word in the review, converting the Penn Treebank tag to its equivalent WordNet POS tag using a helper method. This is done because SentiWordNet uses WordNet POS tags instead of the Penn Treebank tagset. The word is then lemmatized using the WordNet Lemmatizer and is looked up in the SentiWordNet lexicon, which returns a SentiSynset if the word is found. Each SentiSynset contains a positive score, negative score, and an objectivity score for each word, which are positive floats. For the purpose of this project, only the positive score and negative score was used to create a sentiment net score by subtracting the total negative scores from the total positive scores of each review.

For example, the sentence *The movie is bad* has a net sentiment score of -0.5. *Movie* has a 0.0 for both the positive score and negative score, *is* has a positive score of 0.25 and a negative score of 0.125, and *bad* has a positive score of 0.0 and a negative score of 0.625. *The* is not contained in the SentiWordNet lexicon because is it a stopword. Thus, the net score equates to -0.5 (= 0.25 - (0.125+0.625)).

The Sentiscore is also binned for comparison. To bin the Sentiscore, the overall net Sentiscore for the review is divided by the number of tokens (or words) which contributed to the Sentiscore. A score after division within the range of -0.33 and 0.33 is considered a neutral review and is given a value of 0. Any score below a -0.33 is deemed a negative review and is given a value of -1. A positive review is a score above 0.33 and is given a value of 1.

Since the Sentiscore method applies to a single review and returns a single Sentiscore, scikit-learn’s Pipeline, FeatureUnion, BaseEstimator and TransformerMixin are used to apply the Sentiscore to the entire corpus and combine the results with the results from the TF-IDF Vectorizer. First, a custom transformer is created using BaseEstimator [11] and TransformerMixin [12], which applies the Sentiscore method to each review in the dataset. Since this transformer returns a list, another transformer is created to cast the list as a transposed matrix array. Then, these two transformers are added together in a single pipeline using scikit-learn’s Pipeline [13], which will return the Sentiscore feature in the desired format that will allow it to be joined together with the result from the TF-IDF Vectorizer using FeatureUnion [14]. The final result after the FeatureUnion is a matrix that contains the n-gram vectors as well as the Sentiscores.

## Modeling Building and Results

Two classification models used to classify the Rotten Tomatoes movie review corpus are the Logistic Regression model [15] and Support Vector Machine (SVM) model (specifically, Linear Support Vector Classification) [16] from scikit-learn.

Before training the models, the corpus is split in training and test sets: 90% training (432,000 reviews) and 10% test set (48,000). Furthermore, to determine the best C value, the inverse regularization parameter, which is a penalty against complexity, the training set was further split into training and validation sets: 78% training (336,960) and 22% validation set (95,040). The C parameter can improve the performance of the model on new data, which is why different C values are tested on the validation set. The C value which results in the highest accuracy on the validation set is determined to be the best C. After determining the best C value, the full 90% training set (432,000) is then used to train the models.

For the feature set, the models are first created using only the n-gram vectors from the TF-IDF Vectorizer, from which the results will be used as a base comparison. Next, the net Sentiscore feature is added to the feature set. Lastly, the models are created using the n-gram vectors and the binned Sentiscore.

The results of the models are summarized in the table below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **n-grams only** | | **n-grams + Sentiscore** | | **n-grams + binned Sentiscore** | |
| **Acc.** | **F1** | **Acc.** | **F1** | **Acc.** | **F1** |
| **LogReg** | **0.867** | **0.866** | 0.864 | 0.863 | **0.867** | **0.866** |
| **SVM** | **0.914** | **0.913** | 0.913 | 0.913 | 0.912 | 0.912 |

**Table 2**: Results from modeling (reran the models since the presentation)

As seen from the results of the modeling phase, the models built on solely the n-grams features result in the highest scores for both the Logistic Regression and SVM models, overall. Although binning the Sentiscore results in the same accuracy score for the Logistic Regression Model, the accuracy score for SVM is still less than solely using the n-gram vectors. Keeping time complexity and memory resources in mind, adding the Sentiscore feature adds too much time to the model training phase. Overall, the performance measure scores do not significantly improve and time complexity increases with the addition of the Sentiscore feature; thus, using the n-gram vectors as features for the models is adequate to create a good performing classification model for sentiment classification.

|  |  |  |
| --- | --- | --- |
| **Unigram** | **Bigram** | **Trigram** |
| entertaining | not only | doesn’t have to |
| enjoyable | never boring | never less than |
| fun | even when | never fails to |
| refreshing | never dull | is more than |
| delightful | no less | so much fun |

**Table 3**: Top N-grams for Positive Reviews

|  |  |  |
| --- | --- | --- |
| **Unigram** | **Bigram** | **Trigram** |
| unfortunately | not very | little more than |
| fails | not enough | right place but |
| dull | doesn’t work | not good way |
| worst | not funny | not very good |
| disappointing | not good | how not to |

**Table 4**: Top N-grams for Negative Reviews

# Conclusion

In this paper, two classification models are created to classify Rotten Tomatoes movie reviews using the n-gram vectors created by the TF-IDF Vectorizer and a sentiment information score, Sentiscore, created using the SentiWordNet lexicon. The project experiments by creating the models using solely the n-gram vectors then adding the Sentiscore later for comparison. The Sentiscore is also binned to see if there is a significance. The results show that using n-grams as features for sentiment analysis classification is a good methodology, validating the well-known method in the NLP community. Adding the Sentiscore as a feature did not add improvement to the models; therefore, the Sentiscore method is not a significant enough feature to increase accuracy of classification of sentiment polarity. Perhaps a different use of the SentiWordNet lexicon needs to be derived. In conclusion, feature engineering is a difficult and time-consuming task. Given a set of natural language text, deriving features that will be significant enough to influence a model is a challenging task. To improve this project, further deliberation will have to go into deriving a feature that can add sentiment information into the models and is also significant enough to improve the performance of said models.

1. References

|  |  |
| --- | --- |
| [1] | "Sentiment Analysis: Nearly Everything You Need to Know," MonkeyLearn Inc., 2019. [Online]. Available: https://monkeylearn.com/sentiment-analysis/. [Accessed 27 June 2019]. |
| [2] | A. Dey, M. Jenamani and J. J. Thakkar, "Lexical TF-IDF: An n-gram Feature Space for Cross-Domain Classification of Sentiment Reviews," *Pattern Recognition and Machine Intelligence,* pp. 380-386, 2017. |
| [3] | J. Thanaki, "Feature Engineering and NLP Algorithms," in *Python Natural Language Processing*, Birminghan, UK, Packt Publishing, 2017, pp. 102-172. |
| [4] | D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu and B. Qin, "Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, Maryland, 2014. |
| [5] | N. Gervais, "Rotten Tomatoes: 480,000 Labeled Critic Reviews," Kaggle, 26 March 2019. [Online]. Available: https://www.kaggle.com/nicolasgervais/rotten-tomatoes-480000-labeled-critic-reviews. [Accessed 21 May 2019]. |
| [6] | S. Bird, E. Klein and E. Loper, Natural Language Processing with Python, Sebastopol, CA: O'Reilly Media, Inc., 2009. |
| [7] | scikit-learn developers, "sklearn.feature\_extraction.text.TfidfVectorizer," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature\_extraction.text.TfidfVectorizer.html. [Accessed 2 July 2019]. |
| [8] | 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, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011. |
| [9] | P. Pandey, "Building a Simple Chatbot from Scratch in Python (using NLTK)," A Medium Corporation, 16 September 2018. [Online]. Available: https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e. [Accessed 1 July 2019]. |
| [10] | "SentiWordNet Interface," NLTK, [Online]. Available: http://www.nltk.org/howto/sentiwordnet.html. [Accessed 2 July 2019]. |
| [11] | scikit-learn developers, "sklearn.base.BaseEstimator," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.base.BaseEstimator.html. [Accessed 3 July 2019]. |
| [12] | scikit-learn developers, "sklearn.base.TransformerMixin," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.base.TransformerMixin.html. [Accessed 3 July 2019]. |
| [13] | scikit-learn developers, "sklearn.pipeline.Pipeline," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html. [Accessed 3 July 2019]. |
| [14] | scikit-learn developers, "sklearn.pipeline.FeatureUnion," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.FeatureUnion.html. [Accessed 3 July 2019]. |
| [15] | scikit-learn developers, "sklearn.linear\_model.LogisticRegression," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LogisticRegression.html#sklearn.linear\_model.LogisticRegression.fit. [Accessed 3 July 2019]. |
| [16] | scikit-learn developers, "sklearn.svm.LinearSVC," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC. [Accessed 3 July 2019]. |