Evaluating Choice of Sentiment Lexicon on Machine Learning Model Accuracy

Aidan Gresham (UID: 305919791), Thomas Blakely (UID: 406065192), Tian Qiu (UID: 405377477), Wesley Bian (UID: 605585753), Yaquan Wang (UID: 806037461), Yuzhi Yao (UID: 005541837)

Contents

1 Introduction

1.1 Background

Sentiment analysis is a subfield of Natural Language Processing (NLP) focused on teaching computers to understand and interpret the emotional tone of human language. While sentiment analysis has many applications in areas like market research and public policy, the field also has significant developmental challenges due to the inherently fluid nature of the human language. For instance, while the phrase "Oh, he's bad" may have a traditional interpretation that a person is mediocre at a task, the modern use of "bad" means the phrase could also be interpreted as saying a person is attractive.

The fluidity of the human language often leads to challenges in sentiment analysis, especially when using pre-defined opinion lexicons. Although these lexicons offer advantages in time efficiency and consistency, they often manifest challenges in the form of misclassification issues, leading to poor model accuracy.

In the film industry, it is crucial that models using sentiment analysis have high accuracy. Decisions made in this industry are infrequent and costly, and misinterpreting public sentiment can lead to decisions resulting in financial disaster. An example of this is Sony's re-release of the movie "Morbius" (2022) [1] due to social media buzz, which ended in financial failure as the film only grossed $280,000 opening weekend despite being re-released in over 1,000 theaters.

1.2 Research Problem and Goal

Gauging public sentiment, however, is a difficult task. Movies often receive skewed ratings concentrating on either a '1' or a '5.' Thus, alternative ways are needed to properly gauge public opinion on a particular movie. One way to do this is by analyzing written reviews. However, this is a difficult task as it is not always clear what people intend with their reviews, as movie reviews are often written semi-casually with confusing jargon, making interpreting reviews an almost domain-specific task.

Given these challenges, our research aims to compare the performance of two popular sentiment lexicons, the Valence Aware Dictionary and sEntiment Reasoner (VADER) sentiment lexicon, and Bing Liu's Opinion Lexicon, to assess the importance of contextualized-specific sentiment lexicon. By utilizing Logistic Regression, K-Nearest Neighbors (KNN), Decision Trees, Random Forest, Linear Discriminant Analysis (LDA), and Quadratic Discriminant Analysis (QDA), we will evaluate each lexicon's efficacy in classifying Internet Movie Database (IMDB) reviews as positive or negative. This study highlights the importance of context-specific lexicons for sentiment analysis and the effectiveness of compressing robust lexicons for binary classification.

2 Data Preprocessing

2.1 Dataset

2.1.1 IMDB

For our analysis, we utilized the IMDB [2] dataset containing two columns of 50,000 observations. The first column consists of text data, specifically the written IMDB reviews. The second column is a binary variable denoting if the sentiment of the corresponding IMDB review is "positive" or "negative." There are no missing values in this dataset.

2.1.2 Bing Liu's Opinion Lexicon

Bing Liu's Opinion Lexicon [3] is composed of two separate text files divided based on sentiment. The first file contains 2,006 words identified as having positive sentiment, while the second file contains 4,783 words classified as having negative sentiment. Altogether, the lexicon contains 6,789 words split between the two files.

2.1.3 VADER's Sentiment Lexicon

VADER's sentiment lexicon [4] is a text file containing four columns of 7,519 observations. The first column consists of the word, the second column the mean human sentiment score, the third column the standard deviation of the word assuming it follows the normal distribution, and the fourth column contains 10 individual ratings of the word's sentiment taken during an experiment. For lexicon comparison, the third and fourth column were not considered for this experiment. The second column was converted was converted so that mean scores greater than zero were labelled "positive", otherwise they were labelled as "negative". After conversion, 3,347 words were classified as having positive sentiment, 4,172 negative.

2.2 Data Cleaning

2.2.1 IMDB

Robust cleaning was necessary to convert the IMDB dataset into something usable for our Machine Learning (ML) modeling. Examples of issues include:

1. Line break elements from HTML scraping: "
"
2. Inconsistent use of punctuation.
3. Inconsistent word capitalization.

While it can be argued that the inclusion of break elements is the only true flaw in the dataset and that things like inconsistent punctuation and capitalization may have practical relevance to sentiment prediction, we cannot capture these nuances due to the construction of Bing Liu's Opinion Lexicon.

To prevent modeling issues, R was used to:

1. Convert all text to lowercase.
2. Remove stopwords as defined by the tm [5] package.
3. Remove all line break elements.
4. Remove all punctuation.
5. Remove all numbers.
6. Lemmatize all strings.

2.2.2 Lexicon Cleaning

While no cleaning is necessary for Bing Liu's lexicon, VADER's lexicon includes several instances of capitalized letters, which were converted to lowercase using R.

2.3 Feature Engineering: TF-IDF and Principal Component Analysis (PCA)

After cleaning the data in R, Python was used for feature engineering and experimental analysis. For these procedures, the same code was run twice, only differing in choice of sentiment lexicon. Other parameters were not changed to preserve uniformity and increase the likelihood that the observed differences between models are due to the lexicons themselves and not model configuration.

From the sklearn.feature_extraction.text [6] package, the TfidfVectorizer() function was used to weigh the importance of words based on their occurrence in a particular document and across the cleaned dataset using Term Frequency (TF) and Inverse Document Frequency (IDF). The min_df was set to 0.1, and the max_df was set to 0.7, meaning that words that appeared in fewer than 10% or more than 70% of the documents were excluded from the feature set.

After transforming the text data into numerical form, PCA was performed to reduce dimensionality using the PCA() function from the sklearn.decomposition [7] package. Different levels of n_components, the number of components to keep in the PCA() function, were then experimented with to test how increasing the number of components improved model accuracy. With both lexicons, the effect of increasing the number of components from 90 to 1,514 resulted in an approximate 2% increase in model accuracy across all models. However, the downside of using such a high number of components is that it is computationally expensive, making testing the models extremely time-consuming. Thus, n_components was set to 600, significantly reducing the overall computational workload and resulting in each model losing roughly 0.2% accuracy compared to when n_components was set to 1,514.

3 Experiment

3.1 Logistic Regression

Logistic regression was the first method used to compare lexicons. Due to the large model size, the dataset was split 70-30 instead of 50-50 as originally prepared. The higher training split allowed the model to capture a broader array of word sentiments, leading to higher model accuracy at the cost of an increased risk of overfitting and higher variance. However, as 30% still accounts for 15,000 test reviews, the model will likely still be able to generalize to other datasets.

To perform the Logistic Regression, the LogisticRegression() function from the sklearn.linear_model [8] package was used. By default, this method uses L2 regularization, which we decided to stick with as we already performed PCA and did not see a need to create further sparsity through L1 regularization. We also decided against tuning the hyperparameter, which is set to 1 as default in this function, in order to note inappropriately favor one lexicon over the other.

Bing Liu's Opinion Lexicon slightly outperformed VADER's Sentiment Lexicon in terms of both model accuracy (85.5%, 84.6%), CV mean accuracy (85%, 84%), and ROC-AUC (0.93, 0.92).

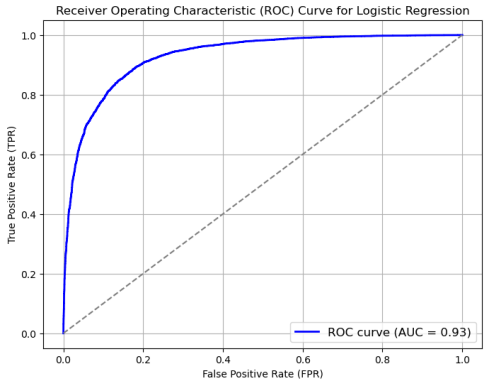

Figure 1: Bing Liu's Logistic Regression ROC

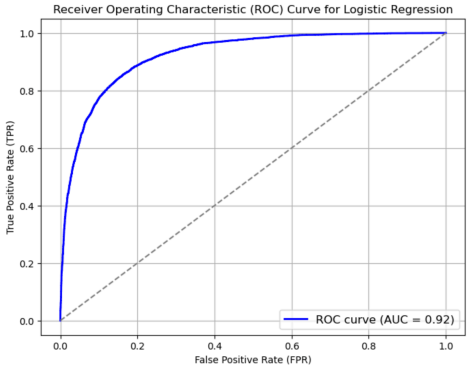

Figure 2: VADER's Logistic Regression ROC

3.2 K-Nearest Neighbor

K-Nearest Neighbors was the second method used to analyze our lexicons. Like with the Logistic Regression model, we decided it would be best to split our data 70-30 and continue with that split for the rest of the testing to maintain as much consistency as we can across different types of models and prevent overturning. To perform the K-Nearest Neighbors, the KNeighborsClassifier() function from the sklearn.neighbors [9] package is used.

To find the best performing K, the original method of choosing a range of 1 to 50 for K used more computational power than expected. Thus, we used a standard method of choosing a K equal to the square root of N, where N represents the number of samples in the training dataset. As we had 35,000 training samples, we chose a K = 187. While this is not the true optimal K in terms of model accuracy due to limited computation power, we believe this method struck the perfect balance between accuracy and computational efficiency and allowed for rigorous testing

Comparing Bing Liu's and VADER's lexicon, VADER's Sentiment Lexicon performs slightly better in terms of accuracy (78%, 79%), CV mean accuracy (77.5%, 77.7%), and ROC-AUC (0.86, 0.87).

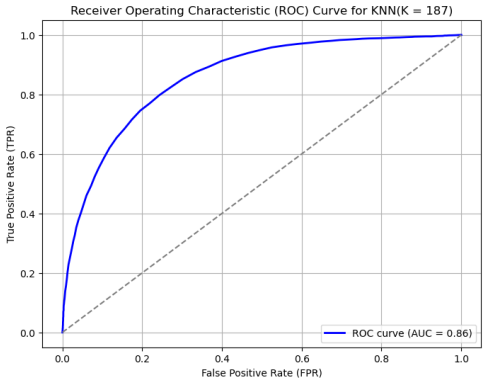

Figure 3: Bing Liu's K-NN ROC

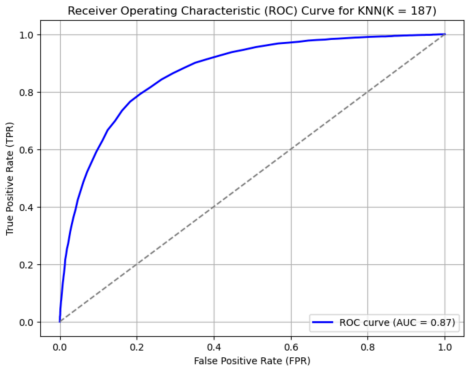

Figure 4: VADER's K-NN ROC

3.3 Decision Trees

To create Decision Tree models, we used the DecisionTreeClassifier() function from sklearn.tree [10] package. As this function can output different values on different instances of running the code, we set the random_state=0 for reproducibility.

Since this model is non-parametric, no explicit hyper-parameter tuning was used in the performance test.

Comparing Bing Liu's and VADER's lexicon, Bing Liu's Sentiment Lexicon performs slightly better in terms of accuracy (72%, 71%), and ROC-AUC (0.72, 0.71). CV mean accuracy was not computed for Decision Trees due to computational efficiency issues.

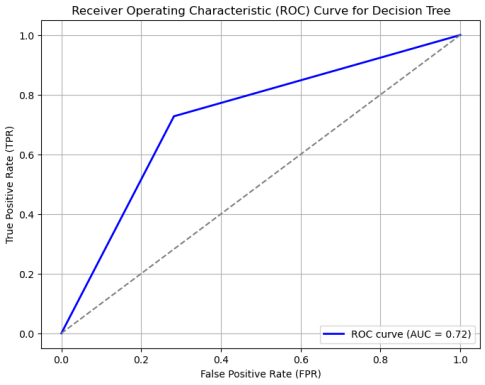

Figure 5: Bing Liu's Decision Tree ROC

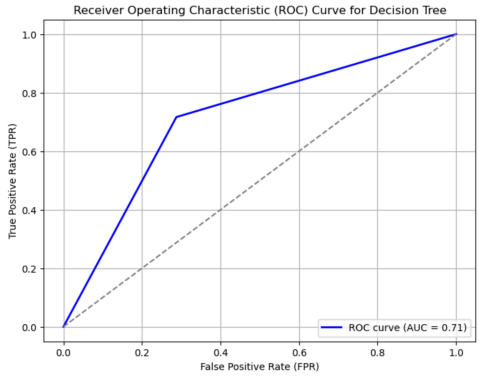

Figure 6: VADER's Decision Tree ROC

3.4 Random Forest

For the Random Forest models, the RandomForestClassifier() function from sklearn.ensemble [11] package used. Like with the Decision Tree models, a random_state=0 was used for reproducibility.

By default, the RandomForestClassifier() sets the number of trees to 100. Through experimentation, we tried setting trees to values such as with n = 200. Surprisingly, this did not increase our lexicons' accuracy and AUC. Thus, we decided to stick with our number of trees at 100 for generalizability, as it is a widely used baseline with Random Forest models, and increasing the number of trees didn't significantly affect our results. While it might be possible to increase accuracy and AUC by setting the number of trees to an even larger number than 200, considering how computationally expensive it was to create a model with 200 trees that did not result in higher accuracy or AUC, we did not view increasing the number of trees to be a worthwhile endeavor. No other explicit hyperparameter tuning was performed in the setting.

Comparing Bing Liu's and VADER's lexicon, Bing Liu's Sentiment Lexicon performs slightly better with an accuracy of (81%, 80%), and an ROC-AUC (0.89, 0.88). CV mean accuracy was not computed for Random Forest due to computational efficiency issues.

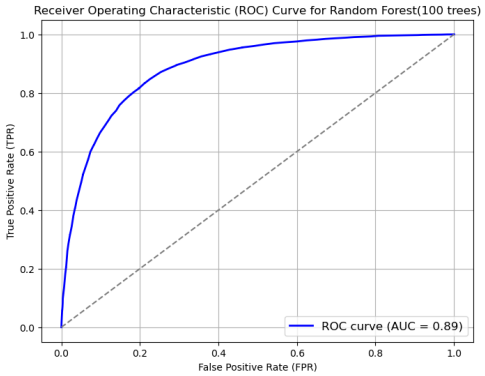

Figure 7: Bing Liu's Random Forest ROC

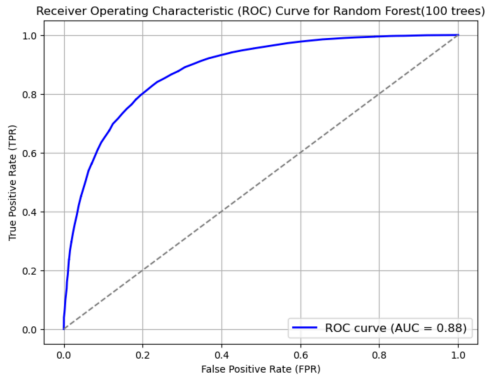

Figure 8: VADER's Random Forest ROC

3.5 Linear Discriminant Analysis

For our fifth model, we create a Linear Discriminant Analysis model using the LinearDiscriminantAnalysis() function from sklearn.discriminant_analysis [12] package was used.

No explicit hyper-parameter tuning was used in the performance test.

Comparing Bing Liu's and VADER's lexicon, Bing Liu's Sentiment Lexicon performed slightly better with an accuracy (84.9%, 84%), CV mean accuracy (84.2%, 83.3%) and an ROC-AUC (0.93, 0.92). The accuracy and AUC numbers are quite high, especially compared to K-NN, which usually dominates LDA and Logistic Regression when the decision boundary is non-linear. However, after comparing the results of our Logistic Regression,

K-NN, and LDA and seeing both Logistic Regression and LDA outperforming K-NN, we assume the decision boundary is linear or close to it.

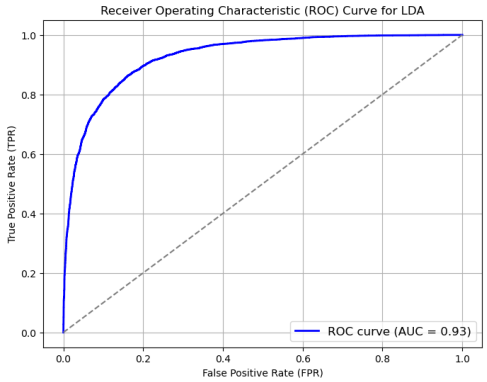

Figure 9: Bing Liu's LDA ROC

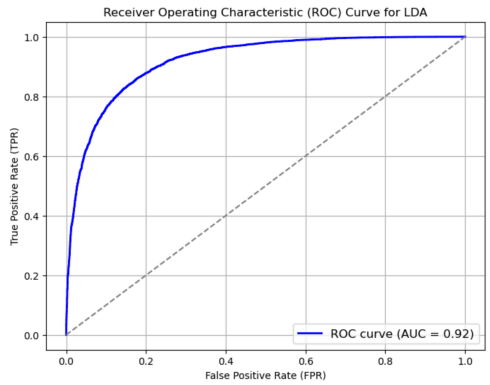

Figure 10: VADER's LDA ROC

3.6 Quadratic Discriminant Analysis

The last model we created was a Quadratic Discriminant Analysis model using the QuadraticDiscriminantAnalysis() function from the sklearn.discriminant_analysis [13] package.

No explicit hyper-parameter tuning was used in the performance test.

Comparing Bing Liu's and VADER's lexicon, Bing Liu's Sentiment Lexicon performs slightly higher by accuracy (79.8%, 79.2%), CV mean accuracy (79.3%, 78.6%), but the two had an equal ROC-AUC value at (0.85, 0.85).

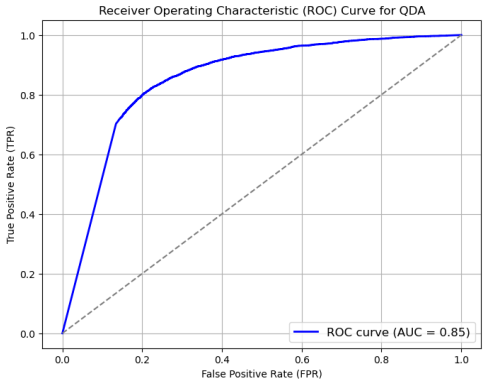

Figure 11: Bing Liu's QDA ROC

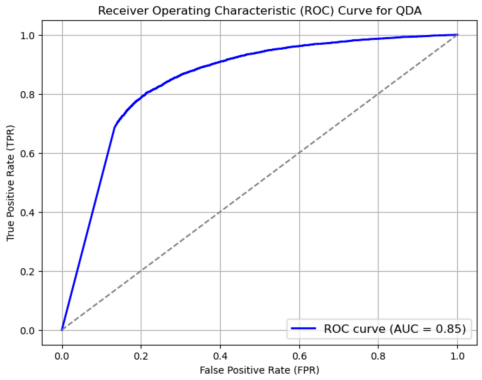

Figure 12: VADER's QDA ROC

4 Conclusion

4.1 Initial Findings

This study aimed to analyze the effectiveness of using contextually-tuned lexicons over more generic ones. To do this, we utilized VADER's sentiment lexicon, a lexicon tuned for social-media analysis, and Bing Liu's Opinion Lexicon, a more generalized lexicon. We then analyzed the accuracy and AUC of these lexicons on IMDB reviews using several machine learning models, hypothesizing that the VADER's lexicon would outperform Bing Liu due to the more casual nature of IMDB reviews.

Accuracy metrics refute our hypothesis, however. As shown in the table below, of the six models used, VADER's lexicon outperformed Bing Liu only with the K-NN model, achieving an accuracy of 79% compared to 78%.

Somewhat similar results can be seen in the cumulative ROC charts between Bing Liu's and VADER's lexicon. As seen in the ROC curves below, in the six models, VADER had a higher AUC than Bing Liu's

in only one model, the K-NN model. VADER's, however, also tied the AUC of Bing Liu's with the QDA model.

Method	Accuracy
LR	0.8545
KNN	0.7800
DT	0.7200
RF	0.8100
LDA	0.8498
QDA	0.7987

Method	Accuracy
LR	0.8456
KNN	0.7900
DT	0.7100
RF	0.8000
LDA	0.8404
QDA	0.7924

Table 1: Bing Liu and VADER Accuracy Comparison

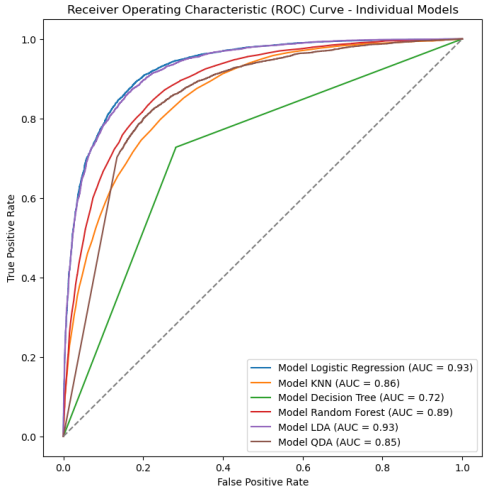

Figure 13: Bing Liu's ROC Curve for All Model

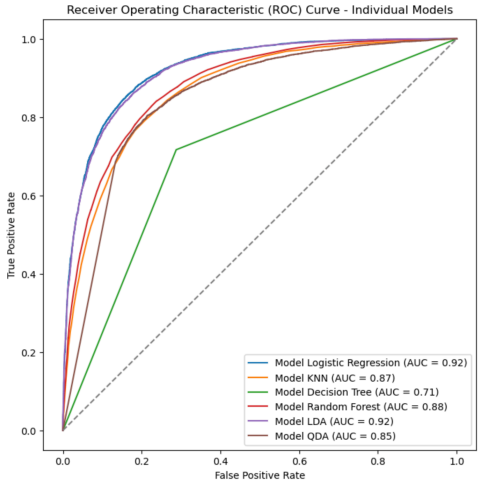

Figure 14: VADER's ROC Curve for All Models

4.2 Analysis and Limitations

Had VADER's lexicon demonstrated some edge over Bing Liu's, like having the model with the highest accuracy, the highest AUC, or outperformed Bing Liu's in most models, there may have been something to support our initial hypothesis. As it stands, however, Bing Liu's lexicon appears to have been the more effective lexicon at predicting sentiment from the IMDB dataset. This caused us to question why this might have been the case.

Our first assumption concerns our reduction technique. To fairly compare the two models, during the data-cleaning process of the VADER lexicon, we standardized it by removing the third and fourth columns. This led us to think that our standardization process could have impacted our results. However, checking the source code of the VADER lexicon, it appears that this is how the functions in the VADER package operate. Specifically, there is this line of code in the package: `(word, measure) = line.strip().split('\t')[0:2]`. As such, we do not think our process of removing the third and fourth columns affected our results.

We then thought more about our data-cleaning process. In it, we removed punctuation for ease of analysis. On reflection, we believe this might have been a significant mistake that, had we done it, could have led to VADER's lexicon outperforming Bing Liu's. In the social media space, people frequently use emoticons to convey their feelings. Emoticons like :D or :P typically convey a positive sentiment, whereas emoticons like D: or)': convey a negative sentiment. These emoticons are accounted for in VADER's lexicon but are not in Bing Liu's. As such, their removal likely led to VADER's lexicon having slightly lower predictive power.

Furthermore, limitations on computational power put constraints on our study. Due to the limited computational power of our machines, we were restricted in how far we could tune our parameters and compose our scores of cross-validation in each method. In turn, more computationally expensive machine learning methods, like K-NN, Decision Trees, and Random Forests, were likely hindered and could not display their fullest capabilities. This, in turn, opens up the possibility that VADER could have outperformed Bing Liu's if we had more computation power. However, we would need further resources to test that assumption. Granted, the cost of optimization and performance is a balancing act. However, since our results aim to aid people in the Film Industry in making multi-million dollar decisions, it could be argued that having a more optimized model is beneficial.

Lastly, as feature engineering techniques were performed uniformly, there is a possibility that VADER's lexicon could outperform Bing Liu's if they were both optimized to their respective fullest extent. As we used two different lexicons, there will inherently be some level of favorability to different methods between the two lexicons. This, however, would come at the cost of us having a harder time extending the external validity of our findings to other models. As such, we note this as a potential limitation but not an area we think should be as heavily explored as our other stated concerns.

4.3 Recommendations for Further Research

For further research on the effectiveness of contextual-based lexicons, we recommend research focusing on the importance of punctuation marks for basic sentiment classification. As sentiment analysis is often used in informal writing, and from our findings, we surmise that the exclusion of punctuation marks affected model performance, we believe that further research into this field could lead to the development of packages that could accommodate punctuation usage, allowing models using basic lexicons to have higher performance.

Second, it would be of particular relevance if a study doing the inverse of what we did were performed in terms of lexicon selection. In other words, instead of using VADER's lexicon to analyze IMDB reviews, it would be of particular interest to analyze IMDB reviews using a lexicon more tuned for formal writing. Through this research, we would likely have a better idea of the importance of contextual tuning on sentiment accuracy in the context of basic classification.

5 References

[1] Morbius. (2022). In Wikipedia. https://en.wikipedia.org/wiki/Morbius_(film)

[2] Npathi, L. (2018). IMDB Dataset of 50K Movie Reviews. Kaggle. https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

[3] NLTK Data. (2017). Opinion Lexicon. Kaggle. https://www.kaggle.com/datasets/nltkdata/opinion-lexicon

[4] 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.

[5] Feinerer, I., & Hornik, K. (2023). tm: Text Mining Package. R package version [0.7-11]. https://rdrr.io/rforge/tm/man/stopwords.html

[6] Scikit-Learn Developers. (n.d.). sklearn.feature_extraction.text Scikit-Learn. Python package version [1.3.2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

[7] Scikit-Learn Developers. (n.d.). sklearn.decomposition.PCA. Scikit-Learn. Python package version [1.3.2] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[8] Scikit-Learn Developers. (n.d.). sklearn.linear_model.LogisticR Scikit-Learn. Python package version [1.3.2] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[9] Scikit-Learn Developers. (n.d.). sklearn.neighbors.KNeighborsC Scikit-Learn. Python package version [1.3.2] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[10] Scikit-Learn Developers. (n.d.). sklearn.tree.DecisionTreeClas Scikit-Learn. Python package version [1.3.2] https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

[11] Scikit-Learn Developers. (n.d.). sklearn.ensemble.RandomFore Scikit-Learn. Python package version [1.3.2] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[12] Scikit-Learn Developers. (n.d.). sklearn.discriminant_analysis Scikit-Learn. Python package version [1.3.2] https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html

[13] Scikit-Learn Developers. (n.d.). sklearn.discriminant_analysis Scikit-Learn. Python package version [1.3.2] https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis.html