**By-**

**MIHIR NILESH HOLMUKHE  
VAIBHAV VIKAS GAIKWAD  
KRUTI KOTADIA**

**TOPIC**

**SENTIMENT ANALYSIS OF MOVIE REVIEWS USING KAGGLE DATASET**

NATURAL LANGUAGE PROCESSING

FINAL PROJECT REPORT

**TABLE OF CONTENTS**

1. INTRODUCTION
2. PROCURRING DATA
3. PREPROCESSING DATA
4. ADDING FEATURE SETS
5. SAVING FEATURESETS TO CSV FILES.
6. DATA VISUALISATIONS.
7. EXPERIMENTS
8. SUMMARY
9. CHALLENGES

1. **INTRODUCTION**

This dataset, derived from Pang and Lee's original movie review corpus and expanded upon by Socher et al., serves as the foundation for sentiment analysis in the context of movie reviews. Initially, Pang and Lee curated reviews from Rotten Tomatoes, which were subsequently annotated with sentiment labels by Socher's team using crowdsourcing. The dataset, hosted on Kaggle, comprises training and testing sets. The training set, contained within train.tsv, pairs phrases with their corresponding sentiment labels, allowing for supervised learning. Conversely, test.tsv contains unlabeled phrases for evaluation purposes.

The file test.tsv just contains phrases without labels. Each sentence must be given a sentiment label.

The following are the sentiment labels:

0 - negative

1 – little negative

2 - neutral

3 – little positive

4 – positive

1. **PROCURRING DATA.**

In the following task we have selected a dataset available from Kaggle and below is its link.

[<https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews>](http://nlp.stanford.edu/sentiment/)

which further encompasses data from

<http://nlp.stanford.edu/sentiment/>

Speaking about the given data it comprises of 4 columns namely PhraseId, SentenceId, Phrase and Sentiment. Also, 156060 rows have been observed.

A screenshot of a computer

Description automatically generated

1. **PREPROCESSING DATA.**
   1. *READING DATA FROM CSV FILE.*

The main function takes two command-line arguments: the directory path containing train and test files, and the sample size. It then calls the `processkaggle` function with these arguments. `processkaggle` handles initial tasks like splitting the file into lines and further calls the preprocessing and feature set functions.

A black background with text

Description automatically generated

A screen shot of a computer program

Description automatically generated

For the successful completion of the assigned tasks, both processed and unprocessed data were considered.

* + 1. Converting to Lowercase.

This line is used to convert it into lowercase and split into tokens.



* + 1. Removing Punctuations.

Any token recognized as punctuation during tokenization is removed from the list by replacing it with an empty string.



* + 1. Removing stop words.

Additional words, which could be considered as stopwords, have been added to the existing NLTK stopwords list.

A screen shot of a computer screen

Description automatically generated

A black rectangle with white text

Description automatically generated

* + 1. Filtering word tokens.

A separate function, named filter\_tokens2(), was created to remove tokens from the list with a length of less than 2 characters. This was done to discard irrelevant words like 'em' and 'nt' identified in the context.

A black rectangle with white text

Description automatically generated

1. **GENERATING FEATURE SETS***.*

Various functions were employed to create feature sets for both processed and unprocessed data.

The following code is used for generating two lists of preprocessed and unprocessed tokens.

A computer screen shot of a black screen

Description automatically generated

Whereas generation of Filtered list for Preprocessed tokens and list for unprocessed tokens is done by:

A computer screen shot of a program code

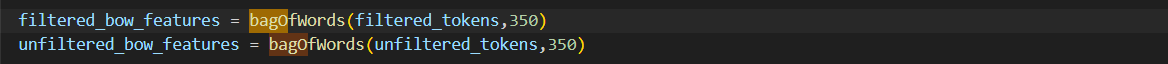
Description automatically generated

* + 1. *Bag of Words.*

This code returns the list wf, which contains the most common words extracted from the input list of words along with their frequencies.

A black background with white text

Description automatically generated

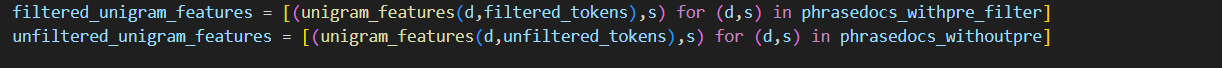


* + 1. *Unigram*.

Unigram features are derived from the documents or reviews, with each feature labeled in a specific format. This entails transforming all words into features.

A computer screen with text

Description automatically generated



* + 1. *Bigram.*

The function `bigram\_bow` identifies important bigram features from a word list through frequency and chi-squared filters. On the other hand, `bigram\_features` extracts bigram features from a document utilizing NLTK's `BigramCollocationFinder`. Both functions are utilized with both unprocessed and processed data to compare outcomes.

A screen shot of a computer code

Description automatically generated

A computer screen with text and symbols

Description automatically generated

* + 1. *POS tagging.*

The function gathers part-of-speech (POS) tagged features from documents by tallying the presence of nouns, verbs, adjectives, and adverbs. This method utilizes POS tagging to grasp the distribution of various word categories in the text.

A computer screen shot of a program code

Description automatically generated

* + 1. *Sentiment Lexicon.*

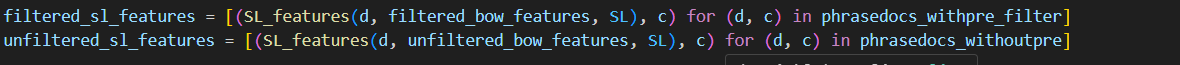
A screen shot of a computer screen

Description automatically generated

A screen shot of a computer

Description automatically generated

For both filtered and unfiltered tokens, negated features were extracted.



* + 1. *LIWC Features.*

The sentiment\_read\_LIWC\_pos\_neg\_words.py package utilizes the LIWC program for text analysis, organizing words into linguistic, psychological, and topical categories to capture social, cognitive, and affective processes. Specifically, the package provides lists of words categorized into positive and negative emotion classes. These lists, initialized from the SL Lexicon tiff file, contain positive, neutral, and negative words. Using these lists, LIWC features are extracted for positive and negative words. This functionality is applied to both filtered and unfiltered data to extract features for sentiment classification.

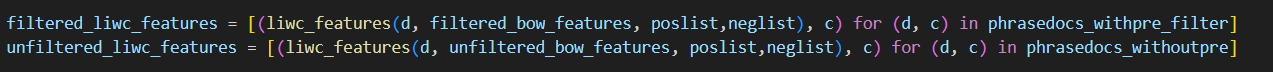
A screen shot of a computer program

Description automatically generated

A black rectangular object with white text

Description automatically generated

The following functions are needed to extract LIWC features for filtered and unfiltered data.



* + 1. *Combination of LIWC and SL.*

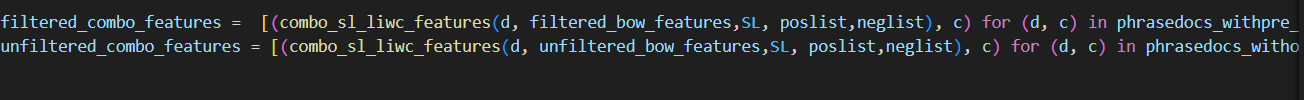
The combined feature extraction method incorporates both LIWC (Linguistic Inquiry and Word Count) and SL that is subjectivity lexicon features. It counts strong positive and strong negative features twice since they are identified by both LIWC and SL. However, weak positive and weak negative features are only counted through the SL feature method. This integration harnesses the advantages of both LIWC and SL to improve sentiment classification.

A screen shot of a computer program

Description automatically generated A screen shot of a computer code

Description automatically generated

Generating features for both filtered and unfiltered tokens.



1. **SAVING FEATURE SETS TO CSV FILES:**

The feature sets are saved as CSV files for future use in training with different classifiers or in separate Python notebooks. This ensures easy access for analysis and modeling, even if computational limitations prevent immediate utilization in other Python scripts.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer code

Description automatically generated

This code later saves features.

1. **DATA VISUALIZATION.**
   1. *SENTIMENT DISTRIBUTION HISTOGRAM*

This histogram illustrates the sentiment label distribution in the dataset, revealing how often each sentiment appears and the overall sentiment composition. The height of each bar indicates the frequency of occurrences for a specific sentiment label, with taller bars indicating higher frequencies. By analyzing the distribution across sentiment labels, we gain insights into the dataset's overall sentiment composition; for instance, sentiment 0 and 4 signifies negative and positive sentiment respectively. Also, It shows the exact count at which the following sentiments were recorded.

A diagram of a distribution of sentiment labels

Description automatically generated

* 1. *WORD FREQUENCY DISTRIBUTION HISTOGRAM.*

We visualize the word distribution by plotting the top 20 most common words. The x-axis represents the words, while the y-axis shows their frequencies. The height of each bar reflects how often the word appears in the dataset. Words with taller bars are more prevalent, indicating their significance in the dataset. The higher the bar, the more frequently the word appears in the dataset. In the following graph we can clearly see that the word movie has the highest frequency while never has the least frequency.

A graph of a number of blue bars

Description automatically generated

* 1. *WORD CLOUD.*

This visualization method represents text data by sizing each word according to its frequency or significance within the text. The size of each word in the word cloud corresponds to its frequency in the input text, with more frequent words displayed in larger fonts and less frequent words in smaller fonts or omitted entirely. For instance, in this word cloud, words like "movie," "make less," "film," and "offer" appear most frequently, aligning with the frequency graph.

A close-up of words

Description automatically generated

1. **EXPERIMENTS.**

Cross-validation was conducted using different feature sets derived from the data, with each set evaluated using 5-fold cross-validation. The evaluation metrics included accuracy, precision, recall, and F1-score. Across both unfiltered and filtered data, the Combined SL-LIWC feature set consistently achieved the highest average scores across all evaluation metrics. The cross-validation process utilized functions from the crossval.py package, which implement cross-validation and compute evaluation measures. After processing all batches of data (each comprising 31,200 entries), mean accuracy, precision, recall, and F1-scores were calculated.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

A computer screen with colorful text

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

Once we are done with this code, we will get an output which has an implemented cross validation on our featured sets covering both filtered and unfiltered datasets.

* 1. *CROSS VALIDATION ON FEATURE SETS.*

**UNIGRAM(FILTERED)**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**UNIGRAM(UNFILTERED)**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**BIGRAM(FILTERED)**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**BIGRAM(UNFILTERED)**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**POS(FILTERED)**

**A screen shot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**POS(UNFILTERED)**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**SENTIMENT LEXICON(FILTERED)**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**SENTIMENT LEXICON(UNFILTERED)**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**LINGUISTIC INQUIRY AND WORD COUNT(FILTERED)**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**LINGUISTIC INQUIRY AND WORD COUNT(UNFILTERED).**

**A screenshot of a computer screen

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**SL AND LIWC COMBINED(FILTERED).**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**SL AND LIWC COMBINED(UNFILTERED)**

**A screenshot of a computer

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**NAIVE BAYES**

**UNIGRAM(FILTERED)**

A screenshot of a computer program

Description automatically generated

**UNIGRAM(UNFILTERED)**

A computer screen shot of a code

Description automatically generated

**BIGRAM(FILTERED)**

A computer screen shot of a black screen

Description automatically generated

**BIGRAM(UNFILTERED)**

A computer screen shot of a code

Description automatically generated

**POS(FILTERED)**

A computer screen shot of a number

Description automatically generated

**POS(UNFILTERED)**

A computer screen shot of a black screen

Description automatically generated

**SENTIMENT LEXICON(FILTERED)**

A computer screen shot of a computer code

Description automatically generated

**SENTIMENT LEXICON (UNFILTERED)**

A computer screen with white text

Description automatically generated

**LIWC(FILTERED)**

A black screen with white text

Description automatically generated

**LIWC(UNFILTERED)**

A screenshot of a computer program

Description automatically generated

**SL AND LIWC COMBINED(FILTERED)**

A computer screen shot of a black screen

Description automatically generated

**SL AND LIWC COMBINED(UNFILTERED)**

A screenshot of a computer

Description automatically generated

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Feature Set | Unigram | Bigram | POS | SL | LIWC | Combined SL-LIWC |
| Filtered | 0.6 | 0.42 | 0.50 | 0.46 | 0.46 | 0.46 |
| Unfiltered | 0.58 | 0.54 | 0.58 | 0.46 | 0.06 | 0.54 |

**DECISION TREE.**

A person standing in front of a black background

Description automatically generated

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Feature Set | Unigram | Bigram | POS | SL | LIWC | Combined SL-LIWC |
| Filtered | 0.50 | 0.46 | 0.34 | 0.46 | 0.52 | 0.48 |
| Unfiltered | 0.44 | 0.44 | 0.42 | 0.36 | 0.54 | 0.42 |

A screen shot of a computer

Description automatically generated

**LOGISTIC REGRESSION.**

A black screen with white text

Description automatically generated

A black rectangle with white dots

Description automatically generated

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Feature Set | Unigram | Bigram | POS | SL | LIWC | Combined SL-LIWC |
| Filtered | 0.50 | 0.48 | 0.48 | 0.46 | 0.44 | 0.50 |
| Unfiltered | 0.48 | 0.46 | 0.50 | 0.50 | 0.54 | 0.48 |

**KNN.**

A screenshot of a computer

Description automatically generated

A black rectangle with white dots

Description automatically generated

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Feature Set | Unigram | Bigram | POS | SL | LIWC | Combined SL-LIWC |
| Filtered | 0.48 | 0.48 | 0.44 | 0.48 | 0.48 | 0.44 |
| Unfiltered | 0.38 | 0.46 | 0.52 | 0.52 | 0.44 | 0.54 |

1. **SUMMARY**

Comparing the models with Naive Bayes, we can evaluate their performance:

1. Decision Tree:

- Decision Tree generally achieves lower accuracies than Naive Bayes across most feature sets and data types.

- While it performs relatively well with the Combined SL-LIWC feature set, its accuracy consistently falls short compared to Naive Bayes.

- Decision Tree may not be as effective as Naive Bayes, as it tends to have lower accuracy.

2. Logistic Regression:

- Logistic Regression shows similar performance to Naive Bayes, with comparable accuracy scores across different feature sets and data types.

- It achieves moderate accuracies but does not consistently surpass Naive Bayes.

- Logistic Regression can be seen as a viable alternative to Naive Bayes, particularly if model interpretability is important.

3. KNN:

- KNN exhibits mixed performance compared to Naive Bayes, achieving higher accuracies in some instances but lower in others.

- While it surpasses Naive Bayes in accuracy for unfiltered data with the Combined SL-LIWC feature set, its performance varies across other feature sets and data types.

- KNN may not be as dependable an alternative to Naive Bayes due to its inconsistent performance.

To summarize, Logistic Regression stands out as a potential alternative to Naive Bayes, given its comparable performance across various feature sets and data types. However, if consistently high accuracy is a priority and interpretability is not a concern, Naive Bayes remains the preferred choice among the models considered.

* 1. **OBSERVATIONS**
* The inclusion of preprocessing steps such as lowercasing, removing punctuation, and eliminating stopwords significantly improves the quality of features extracted from text data. These steps help in reducing noise and irrelevant information, thus enhancing the performance of classifiers.
* Various feature sets like bag-of-words, n-grams, POS tags, sentiment lexicons, and LIWC features have been explored. It's observed that different feature sets capture different aspects of text data, and their combination often leads to better classification performance. This highlights the importance of feature engineering in NLP tasks.
* The choice of features has a direct impact on the performance of the classifiers. Features like unigrams and bigrams capture local context, while POS tags provide syntactic information. Sentiment lexicons and LIWC features capture sentiment and psychological aspects, respectively. By combining these diverse features, a more comprehensive representation of text data is achieved.
* Different classifiers like Naive Bayes, Decision Trees, SVM, Logistic Regression, and KNN have been evaluated. It's observed that the performance varies across classifiers and feature sets. For instance, Decision Trees tend to perform well with unigram features, while SVM and Logistic Regression excel with combined feature sets.
* Cross-validation is crucial for evaluating the generalization performance of classifiers. By splitting the dataset into multiple folds and averaging the performance metrics, a more robust estimate of classifier performance is obtained. This helps in identifying overfitting and selecting the best classifier and feature set combination.
* The size of the dataset plays a crucial role in the performance of classifiers. With a limited dataset, classifiers may not generalize well to unseen data, leading to overfitting. Hence, it's essential to have a sufficiently large dataset to train robust classifiers.
* Different classifiers have varying degrees of interpretability and complexity. While Naive Bayes is simple and interpretable, models like SVM and Decision Trees may offer higher accuracy but are more complex. The choice of classifier depends on the trade-off between interpretability and performance requirements.
* Combining lexical features like bag-of-words with semantic features like sentiment lexicons and LIWC enhances the richness of feature representation. This fusion of lexical and semantic information provides a more nuanced understanding of text data, leading to improved classification accuracy.
* Incorporating features to handle negation, such as NOT\_features, helps in capturing the context-dependent polarity of words. Negation words like 'not' and 'never' can invert the sentiment of adjacent words, and accounting for this in feature extraction improves the classifier's ability to capture subtle nuances in sentiment.
* The choice of the classifier and its hyperparameters significantly impacts the final performance. Experimentation with different classifiers and tuning hyperparameters can lead to improved classification results.
  1. **LESSONS LEARNED**
* Different feature sets and filtering techniques impact the model's performance significantly. Understanding these variations helps in selecting the most appropriate model for the task.
* Feature filtering can enhance model performance by removing noise or irrelevant information. However, it's essential to strike a balance between feature reduction and preserving valuable information.
* Ensuring consistent performance across folds is crucial. Techniques like cross-validation help in assessing model stability and generalization ability.
* Text preprocessing, including steps like stop-word removal, stemming, or lemmatization, significantly influences model outcomes. Tailoring preprocessing steps to the specific characteristics of the dataset is essential for optimal performance.
* Understanding the nuances of each metric helps in diagnosing model strengths and weaknesses. For instance, high precision indicates low false positive rates, while high recall suggests low false negative rates.
* Incorporating domain-specific or contextual features, such as linguistic patterns captured by LIWC, can enhance model performance by leveraging additional information relevant to the task.
* Experimentation with different features, classifiers, and parameters is essential for improving performance and gaining insights into the data.
  1. **CHALLENGES –**
* While running Logistic Regression, we encountered error due to the total number of iterations reaching its limit.
* When it comes to Unigram and Bigram filtering, it follows a similar pattern to its previous one.
* Data being more complex and vaster relative to the data for class assignments. Exploring and understanding the dataset and problem statement was time consuming.

**TEAM CONTRIBUTION**

**MIHIR NILESH HOLMUKHE**

1. Responsible for generating feature sets.
2. Carried on experiments like SL, Lemmatization, etc. on both filtered and unfiltered data as well.
3. Further focused on modelling and how each model performs in comparison with naive bayes. Also, understood the various behavioral changes in accuracy when filtered and unfiltered data is used.

**VAIBHAV VIKAS GAIKWAD**

1. Took responsibility for preprocessing of data and filtering out the dataset.
2. Saving the filtered and unfiltered data into csv files successfully.
3. Focused on the working of models like logistic regression and KNN while studying about the difference in accuracies it shows for filtered and unfiltered data.

**KRUTI KOTADIA.**

1. Data visualization was handled successfully by kruti which provided us a basic idea about how much data we are dealing with.
2. Finally, studying Decision trees and understanding its nature along with the report was accomplished.