**Yelp Review Analysis With Natural Processing Tools**

**1. Introduction and Problem Statement**

Sentiment Analysis is a problem where AI’s analyze text data to see whether text is positive, or negative opinionated. Sentimental Analysis is difficult because of factors such as ambiguity where words can have multiple meanings, sarcasm and informal text such as emoticons and words not in a formal dictionary among other factors. To attempt to solve this problem, our Project is implemented with a pipeline that consists of a Preprocessor, a Vectorizer and a Classifier. We will implement methods such as lemmatization, negation detection, n-grams, and stop words with Tfidf and Count Vectors with multiple types of classifiers in an attempt to improve the classifier accuracy returned from cross validation and 5-fold cross validation.

Linear Support Vector Machine Classifier had the highest accuracy out of all the classifiers with an optimal combination. It had a 4-class classification accuracy of 75.5% and an positive or negative accuracy of 96.1%. In most cases negation detection increases accuracy rate, while removing stopwords and lemmatizing words lower the accuracy rate of the classifiers. Applying n-gram tends to increase accuracy until n is equal to 3. Tfidf vectors did better for Linear Support Vector Machines while Count vectorizers did better for both Multinomial Naïve Bayes and Linear Regression. 5-fold cross validation helped for Multinomial Naïve Bayes classifiers but didn’t do anything for the other two classifiers.

**2. Related Work:**

Sentiment analysis is a category of Natural Language Processing that utilizes data science techniques to determine the underlying sentiment of a document or a given piece of text. To train our own models to perform sentiment analysis, we relied on supervised learning algorithms. Supervised learning algorithms experience a dataset containing features, but each example is also associated with a label or target. Our code made use of implementations of machine learning algorithms in the scikit learn library and preprocessing modules from the nltk library. Multinomial naive Bayes text classification was proposed by Maron (1961) at the RAND Corporation for the task of assigning subject categories to journal abstracts. Naive Bayes was first applied to spam detection in Heckerman et al. (1998). In our project we used multinomial Naive Bayes as one of our algorithms to predict the star rating for the reviews in the testing set. Support Vector Machines has been applied for testing different domains of data sets from movie reviews, topics from computers, hotels or music and also about opinions from digital cameras. We have implemented this in our project to estimate the strength of positive and negative sentiment in short texts. Linear regression has been used in the past to determine the rating of textual book reviews from websites like amazon.com. We used linear regression to fit our training data into a linear model in our project.

**3. Data Sets**

The Data Set we are using for this project is the dataset provided by Yelp for Round 11 of the Yelp Dataset Challenge. The dataset can be obtained at [*https://www.yelp.com/dataset/challenge*](https://www.yelp.com/dataset/challenge) . We are using subset of 200,000 reviews from the Dataset that contains 2.2million Reviews. We have converted the data from the original JSON format to the CSV format with the convert.py source code provided with the Yelp Dataset. Each review contains the attributes ['business\_id', 'cool', 'date', 'funny', 'review\_id', 'stars', 'text', 'useful', 'user\_id']; however for our project we are only interested in the stars and text attributes. The stars attribute contains a numerical value between 1 and 5 which is the score the reviewer gave the business in their review stored as a ‘str’ type. For our project we are only concerned with positive and negative reviews so we have filtered out all the reviews of which have a value of ‘3’ in the stars attribute. The text attribute contains the review the user gave the business in a string format. This string can be composed of multiple sentences because it contains all the characters in the original review including the punctuation. We also have a “balanced” subset where it has the same number of reviews of each class which is sorted by the numerical value in the stars attribute. This is achieved by returning a numerical value that is based on the number of reviews in the subset of each class and setting the minimum value between the 4 classes as the number of reviews of each class we want to include in this “balanced” subset from the original subset of 200,000 reviews. This dataset could be thought of as a subset of the original subset which contains 16,557 reviews of each class for a total of 66,228 reviews. The original 200,000 review subset contains '5': 86,502, '4': 46,788, '1': 27,372, '2': 16,557 reviews for each respective class. The number of tokens or words in the 200,000 dataset is 22,383,959 with a vocabulary size of 209,956. The number of tokens or words in the “balanced” dataset is 9,517,919 with a vocabulary size of 125,735.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Dataset** | **# of Reviews** | **# of 5\* Reviews** | **# of 4\* Reviews** | **# of 2\* Reviews** | **# of 1\* Reviews** | **# of Tokens** | **Vocab Size** |
| **200,000 Subset** | **200,000** | **86,502** | **46,788** | **16,557** | **27,372** | **22,383,959** | **209,956** |
| **“Balanced Subset”** | **66,228** | **16,557** | **16,557** | **16,557** | **16,557** | **9,517,919** | **125,735** |

**4. Description of Technical Approach**

Our Project is implemented with a pipeline that consists of a Preprocessor, a Vectorizer and a Classifier. Our classifiers are not interested in 3-star reviews, because 3-star reviews are considered neutral instead of either slightly or strongly, positive or negative. The Preprocessor will be responsible for filtering out all 3-star reviews from our dataset. This project aims to reach the highest possible classifier accuracy by applying a combination of methods such as creating a “Balanced” dataset from the original subset of 200,000 reviews, lemmatization, negation detection, n-grams, removing stop words, and comparing Tfidf Vectors to Count Vectors. We also implemented a method to remove punctuation but ended up not including this in the final version because this method hinders negation detection. Applying a combination of these methods allows us to test which combinations help or hinder each Classifier when it comes to accuracy, allowing us to apply and record the optimal combinations. The three different classifiers our Project will Implement are Multinomial Naïve Bayes, Linear Support Vector Machine and Linear Regression respectively. Our Project trained and tested classifiers with both an 80/20 training and testing split cross-validation and a 5-fold cross-validation. This results in the classifier yielding a prediction of either 1, 2, 4 or 5-stars for every review. The project initially scored and evaluated methods by testing the accuracy of our classifier with a four-class classification which only scores a prediction as correct if it matches with the stars rating, but have since added a binary classification which scores a prediction as correct if it’s correctly classified as positive or negative.

The Preprocessor is used to perform data manipulation on the dataset before passing the information to the vectorizer. The Preprocessor starts with opening the data file and creating a subset of 200,000 reviews from the Yelp Dataset Challenge 11. The Preprocessor will then filter out all reviews that are associated with a 3-star rating while creating the dataset. The Preprocessor will also be able to do a combination of data and text manipulation methods that will may help with classifier accuracy. This allows us to test which combinations of methods is best for training a Classifier and achieving the highest accuracy.

* **Additional Preprocessor Methods**
  + **Creating a “Balanced” dataset –** This option will make the Preprocessor create a subset that consists of same number of reviews of each class which is sorted by the numerical value in the stars attribute of each review. This is achieved by returning a numerical value that is based on the number of reviews in the subset of each class and setting the minimum value between the 4 classes as the number of reviews of each class we want to include in this “balanced” subset from the original subset of 200,000 reviews.
  + **Negation Detection –** This option will classify each word following negative contraction as “negative” until it reaches a form of sentence ending punctuation. This allows words such as “love” to be classified as negative instead of positive if it follows after a contraction like “don’t”.
  + **Lemmatization –** This option will reduce each word in the text data back to its dictionary base form. EX. fighting is lemmatized to fight.
  + **N-grams –** This option creates takes into account contiguous sequences of “n” number of tokens in the data. **EX.** A trigram will account for sequences of 3 tokens that directly follow after each other in the text data.
  + **Stop words -** This option removes words that are very common in the English language from the text. **Ex.** The, and.
  + **“Removed from Project” Removal of Punctuation –** Removes forms of punctuation from the text data. Removed, because it hinders Negation Detection.

Vectorization is used to extract features from a dataset and format it for training and testing in a classifier. We will have two different forms of vectors in our project.

* **Tfidf Vector –** stands for term frequency – inverse document frequency. It weighs the importance of each unique token proportionally to the number of times the token appears in the review and offsets it with the occurrence of the word in the dataset and corpus to account for the fact some words appear more frequently in natural language.
* **Count Vector** – is a feature vector that simply counts the number of occurrences of each token in the review and dataset without weighing.

Classifiers are used to train and test our dataset to accurately predict the class of the reviews in the test set. Our project has implemented Multinomial Naïve Bayes, Linear Support Vector Machine and Linear Regression. We are using an 80% training and 20% testing split for the data subset.

* **Multinomial Naïve Bayes** will represent the conditional probability of each word along with the frequency of occurrence for each rating level 1 to 5. The classifier will then assign the review a rating according to the highest conditional probability.
* **Linear Support Vector Machines** will clearly separate each review according to the positive or negative classification. The classifier will then classify the review depending on where it falls on the separation.
* **Linear Regression** fits a linear function to the input and output data from the training dataset.

**4. Software**The programming language used for this project is Python 3.6. The following external libraries were used in this project: Csv, Nltk, Collections, Sklearn and the convert.py code provided by the Yelp Dataset Challenge. The Csv library was used to read csv files. The NLTK library is used to preprocess the text applying methods such as Lemmatization, Word Tokenizing, Mark Negation / Negation Detection, and Stopwords. The Collections library was used for the counter method to create the “balanced” subset. The Sklearn library was used to Implement the Multinomial Naïve Bayes, Linear Support Vector Machine and Linear Regression Classifiers. The Sklearn library was also used to implement Tfidf Vectors and Count Vectors to train the Classifiers with. The convert.py code was used to convert the JSON files from the Yelp Dataset Challenge to the CSV files.

**Publicly Available Code: Python 3.6, Yelp Dataset Challenge convert.py**

|  |  |
| --- | --- |
| **Library** | **Methods / Class** |
| **csv** | **reader -reads the csv file** |
| **collections** | **Counter – implements a counter** |
| **nltk.stem.wordnet** | **WordNetLemmatizer – lemmatizes words to dictionary base form.** |
| **nltk.sentiment.util** | **mark\_negation – implements negation detection on the text.** |
| **nltk.corpus** | **stopwords – removes stopwords from the text.** |
| **nltk.tokenize** | **word\_tokenize – tokenizes the words in text.** |
| **sklearn.feature\_extraction.text** | **TfidfVectorizer, CountVectorizer – vectorizes the input data for classifiers.** |
| **sklearn.model\_selection** | **train\_test\_split – splits the data into a training and testing set.** |
| **sklearn.naive\_bayes** | **MultinomialNB – creates Multinomial Naïve Bayers classifier.** |
| **sklearn.svm** | **LinearSVC – creates Linear Support Vector Machine classifier.** |
| **sklearn** | **metrics – compares the accuracy of the predictions to the test set.** |
| **sklearn.model\_selection** | **Kfold – splits the data into folds of training and testing data.** |
| **sklearn.linear\_model** | **LinearRegression – creates a Linear Regression classifier.** |

**Code Written by Self:**

We have implemented code to create a both the 200,000 review subset and the “balanced subset”, and used the public methods to create ways to preprocess the text and keep it in the correct format for things such as lemmatization, and counting the number of tokens and the vocabulary size of the subset. We have also written code using the methods provided by sklearn to Create and test the classifier by creating a vectorizer, fitting the vectorizer, splitting the training and testing data, training and testing a classifier, and evaluating its accuracy. This has been implemented on the Multinomial Naïve Bayes, Linear Support Vector Machine and Linear Regression Classifiers. We have written a script that evaluates the classifier with the 4-class evaluation method using the metrics library and have also created our own method to evaluate the accuracy of the classifier using the binary classification method that only cares if it’s positive or negative.

**5. Experiments and Evaluation**

We evaluated our Classifiers mainly by testing the accuracy of its’ predictions with a four-class classification which only scores a prediction as correct if it matches with the stars rating, but have since added a binary classification which scores a prediction as correct if it’s correctly classified as positive or negative. Because the binary classification evaluation method was added later we have less experimental data with it, but results show that for Linear SVM and Linear Regression it follows the exact same trend as the 4-class classification either increasing when the method increases the 4-class classification and decreasing when the 4-class classification decreases. The increase and decrease in accuracy are lower than for 4-class classification, because the base accuracy had less room for improvement with the binary classification method. However; for Multinomial NB it seems that binary classification accuracy is not significantly affected by the additional methods.

Our experiments were mainly set up with an 80% training and 20% testing data split of our dataset; but we also conducted 5-fold cross validation so every fold has a chance to be the test data in some experiments. It seems that in most cases 5-fold cross validation doesn’t do better than 80% training and 20% testing data split.

**Changes in 4-class Classification Accuracy on Average**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Classifier | Tfidf Vectorizer | Count Vectorizer | (1,3) n-grams | Negation Detection | Lemmatize | Remove stopwords | 5-fold Cross Validation | “Balanced” Dataset |
| Linear Support Vector Machine | 4% Increase | 4% Decrease | 2-2.5% Increase | 0.7-1% increase | 0.5% decrease | 0.5-1% decrease | negligible change | 3.7% decrease |
| Multinomial Naïve Bayes | 8% increase | 8% Decrease | 3% increase with Count vectorizer  8% decrease with tfidf vectorizer | 2% increase with Count Vectorizer | 1% decrease with Count Vectorizer  16% decrease with tfidf Vectorizer | 2% decrease | 4% increase if used instead of the “Balanced” dataset | 11% increase if used with tfidf vectorizer and not with 5-fold cross validation. |
| Linear Regression | 0% Increase | No test data, takes too long to run. | 7% increase | 0.2% increase |  | 1.5% decrease | negligible change |  |

Based on our experiments it seems the optimal combination for the Linear Support Vector Machines classifier is training the classifier with a tfidf vectorizer on the original 200,000 data subset applying (1,3) range n-grams and negation detection resulting in a 4-class classifier accuracy of 75.5% and a binary accuracy rate of 96.1%. The optimal combination for the Multinomial Naïve Bayes classifier is training the classifier with a count vectorizer on the original data subset applying a (1,3) range n-grams, negation detection and a 5-fold cross validation resulting in a 4-class classifier accuracy of 69.35% and a binary accuracy rate of 90.3%. The optimal combination for the Linear Regression classifier training the classifier with the original data subset applying (1,3) range n-grams and negation detection resulting in a 4-class classifier accuracy of 56% and a binary accuracy rate of 87.5%.

Our results suggest that in general n-grams and negation detection seem to help accuracy, while removing stopwords and lemmatizing words seem to lower the accuracy of the classifiers. Tfidf Vectorizers seem to work better for both the Linear SVM and Linear Regression classifiers while Count Vectorizers seem to work better for the Multinomial Naïve Bayes classifier. The balanced dataset with less reviews generally lowers the accuracy of the classifiers. 5-fold Cross validation had no significant impact on the results, except in some of Multinomial Naïve Bayes test cases. The best classifier overall was the Linear SVM while the Multinomial NB classifier seemed to be extremely volatile in increases and decrease to the accuracy depending on what methods were used to preprocess the data before training.

A link for the Full experimental results can be found in the appendix.

**6. Discussion and Conclusion [at least ½ a page]**

*We learned that with our current implementations Linear Support Vector Machines is the classifier with the highest accuracy, followed by Multinomial Naïve Bayes and Linear Regression. These results agreed with the expectations we got from looking at past implementations of Sentiment Analysis. 5-fold Cross validation not making a significant difference in most cases met our expectations because out data set was already substantially large. Lemmatization and removing stop words lowering accuracy didn’t agree with our expectations, because we expected it to remove “noise” and increase accuracy, but it lowered accuracy instead. We also expected tfidf vectorizer to perform better in all cases but this didn’t happen with Multinomial Naïve Bayes. Major limitations towards our current approach is how long it takes for a Count Vectorizer to work on a Linear Regression classifier.*

*If we were running a research laboratory, and we had more resources we would probably increase the dataset and use different classifiers like Neural Networks to apply Sentiment analysis. Deep Learning with Neural network would take a far longer time to train, but it may yield better results. Other things we could do is find other ways to increase the accuracy through the Preprocessor or find different ways to implement the neutral data into the classifier.*

***7. Appendix***

*Link for full experimental results –* <https://docs.google.com/document/d/1s8ljX-cWaAVYYit-9Tfk1sxbOrqML4kkCelYRhy2kP4/edit?usp=sharing>

***References***

Jurafsky, Daniel, and James H Martin. *Naive Bayes and Sentiment Classification*. web.stanford.edu/~jurafsky/slp3/6.pdf.

Zainuddin, Nurulhuda, and Ali Selamat. “Sentiment Analysis Using Support Vector Machine.”*Research Gate*, [www.researchgate.net/publication/286583940\_Sentiment\_analysis\_using\_Support\_Vector\_Machine](http://www.researchgate.net/publication/286583940_Sentiment_analysis_using_Support_Vector_Machine)

Ginosar, Shiry, and Avital Steinitz. “Sentiment Analysis Using Linear Regression.” pdfs.semanticscholar.org/7742/7e471142c6aa62de389ab699dc7d9aeb2c1e.pdf..