# NLP: Yelp Sentiment Analysis

Team Members: Ushna Arshid and Rabindra Yadav

## 1. Introduction

Quantifying a thought or an opinion has always been a challenge for modern applications. Most websites today allow customers to provide product or service feedback via text and a five-star rating system. The star rating system, as wonderful as it is, isn't perfect for determining how someone genuinely feels about anything. A four-star rating may indicate that a product is exceptional to some, while it may indicate that it is simply above ordinary to others. There's a reason why so many websites utilize it: it's basic and straightforward for both people and programs. The full-text review should be read instead to receive a more accurate rendition of these reviewers' viewpoints. This, however, is not an option for the average person. What if we forced the programs to read the entire text instead?

Sentiment analysis emphasizes the advantage of text-based data. It is easy for a program to read thousands of lines of code but it's different in case of unstructured text. About 80% of the data available on the internet are text-based and unstructured. The main challenge in sentimental analysis is that it is really hard to increase the accuracy of the model you try to build.

Users of Yelp.com can rate restaurants and other establishments using a five-star rating system as well as an open-ended text review. The focus of this study will be primarily on text-based reviews.

## 2. Data Collection

Dataset we are using is directly accessed from the yelp website. Since the dataset size was very large, we have chunked the datasets to 90k entries. So, the main datasets we are using are:

- Yelp_review
- Yelp_business

## 3. Exploratory Data Analysis (EDA)

Since the data used in this project is primarily based on text, the main purpose of EDA will be to explore the relationship between different variables in the dataset.In this instance, we also want to apply basic natural language processing techniques to see how this data will be used for the prediction model. Because this data is primarily text-based, the first few steps were:

- Remove stop words:  Stop words are not useful in sentiment analysis. Removing stop words will increase memory space and processing power.

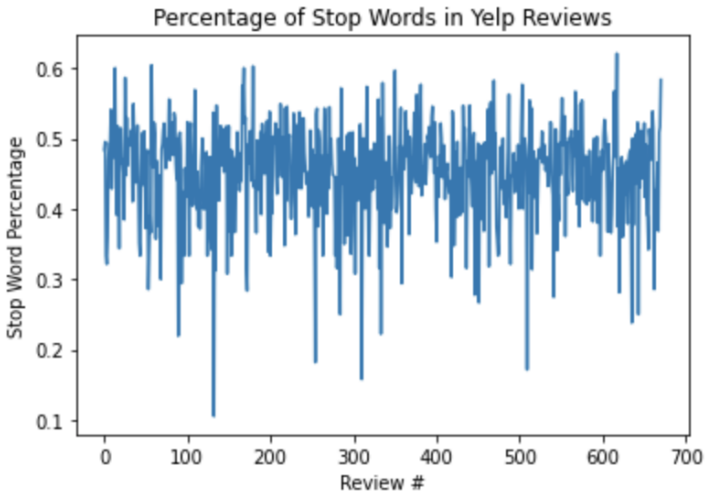


**Fig 1. Percentage of Stop Words for Each Review**

As you can see, most yelp reviews are about 50% stop words on average - removing them will save significant time and computing power.

- Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word. This is another way to save time and computing power when working with large amounts of text data.
- Converting words into vectors using Word2Vec Library. Word2Vec stores each unique word occurrence as a variable in a series. As we add multiple lines for multiple reviews, this becomes a matrix where the columns are each unique word and the rows are the total number of times that word appears.
- To find out if a word or phrase is positive or not, we can calculate its Polarity value based on the TextBlob library's positivity index. Polarity ranges from +1 to -1. When the polarity is +1, the word is 100% positive and when the polarity -1, the word implies 100% negative.
- The last relationship that was looked into was the percentage of positive words in each review. If there's a noticeable increase in the percentage of positive words in a review that also has a high polarity and vice versa for negative reviews, it will make the model even more accurate. The TextBlob function assigns a polarity score to each word vector in the review before assigning a score to the entire review. This allows us to categorize each review and evaluate sentiment without having to rely on the star ranking system.

After confirming that more positive words correlated with a higher polarity score, the dataset was ready to move on to preprocessing for eventual model creation.

## 4. PRE-MODEL DATA PROCESSING

To develop a model capable of reliably assigning each review to a sentiment class, the model must be able to analyze two types of data:

- Text-based, like word matrices or Term Frequency Inverse Document Frequency vectors
- Number based like positive word percentage or classification indices

To prepare the dataset to be converted into a train/test split, we need to select the right features in order to not under or overfit the model. Currently, the sentiment distribution looks like this:


**Fig 2. Review Classifications for Training Data**

In order to convert the text from each review into a number that the model can then interpret, the text needs to be converted to a matrix using either CountVectorizer or TFIDFVectorizer. To ensure the best accuracy, we'll be testing both to see which one performs the best. In addition to the two vectorizers, it is also important to determine the ideal number of ngrams (token or token pairs) and minimum/maximum document frequency for the TFIDFVectorizer.

To test each feature, a function was written to run tests on a simple logistic regression to see which set of features returned the highest accuracy score. The testing pipeline tested TFIDF vs. Countvectorizer first, then minimum document frequency, and finally, the number of ngrams.

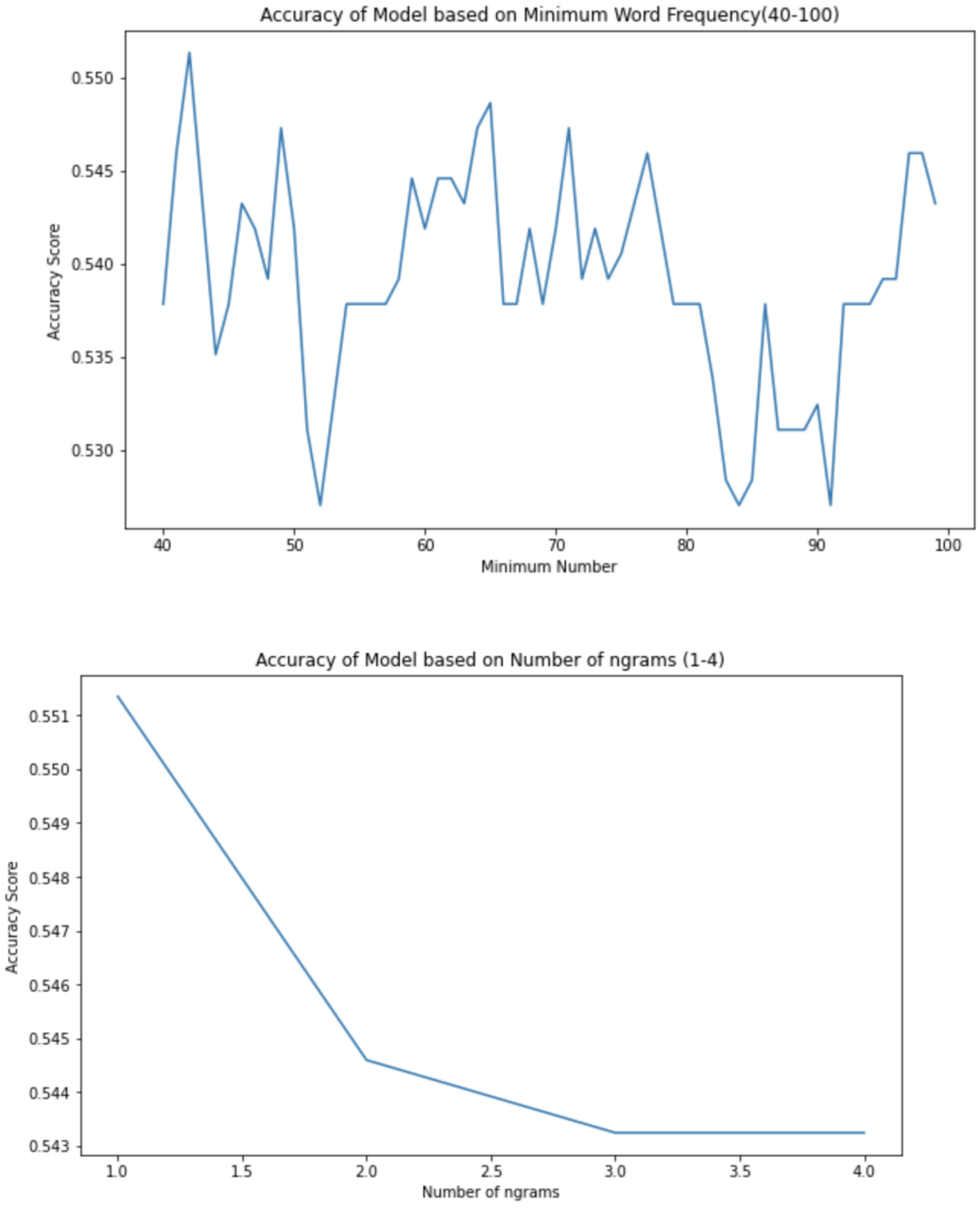


**Fig 3. Text-Based Feature Selection Performance**

In the end, the best result was a TFIDF Vectorizer with a minimum document frequency of 42 and 1 ngrams for each token. This means that the model will only take a word that

has appeared 42 or more times throughout all 10,000+ reviews. This helps with overfitting and saves a lot of time and processing power. With this in mind, we can export a processed dataset that has the percentage of positive words and a TFIDF vector for each review. We can finally move on to creating a model.

**5. Model Selection**

At this point in the analysis, the baseline logistic regression with ideal feature parameters has an accuracy of 55%. We see that there is room for improvement using other modeling techniques.

To model the data, we have used 7 different models. The models will be used for analysis in the following way:

- Select a model type
- Define a pipeline and use GridSearch to find the  best parameters
- Save the model, accuracy, and f1 scores
- Compare the results with confusion matrices

In order to get a good variety of results, we tested 7 different types of models including Logistic Regression, Random Forest,Support Vector Classification, Naive Bayes with CountVectorizer, Naive Bayes with dense and sparse text matrices, stacked model and Naive Bayes with TFIDF. The stacked model was the most accurate,with an accuracy of 76%; this was achieved by combining the probability of the Naive Bayes model with the feature performance of the Gradient Boosted Classifier.

This process involved converting the full review text into a sparse matrix and then into a dense matrix. Matrices that contain mostly zero values are called sparse, distinct from matrices where most of the values are non-zero, called dense. Using the dense matrix as an input and the classifications as an output, the Naive Bayes model outputs a probability value based on the content of each review. This number can then be added to the training dataset that was created in pre-processing to make an improved training dataset.

This new combined training dataset was then tested using a Gradient Boosted Classifier because that is the model that performed best using the numerical and text-based training data. This stacked model combines the best text-based model with the best numerical model to improve our baseline accuracy by 21%.

| Model | Accuracy | F1_Macro | F1_Micro | F1_Weighted |
|---|---|---|---|---|
| Stacked Model | 0.766 | 0.724 | 0.746 | 0.782 |
| SVC | 0.598 | 0.582 | 0.598 | 0.596 |
| Logistic Regression | 0.584 | 0.567 | 0.584 | 0.582 |
| NB_CV | 0.576 | 0.577 | 0.576 | 0.575 |
| Random Forest | 0.558 | 0.534 | 0.558 | 0.553 |
| NB_TF | 0.527 | 0.577 | 0.576 | 0.575 |
| NB Probability Dense/Sparse | 0.521 | 0.521 | 0.521 | 0.511 |

**Fig 4. Accuracy and F1 Scores for Each Model**

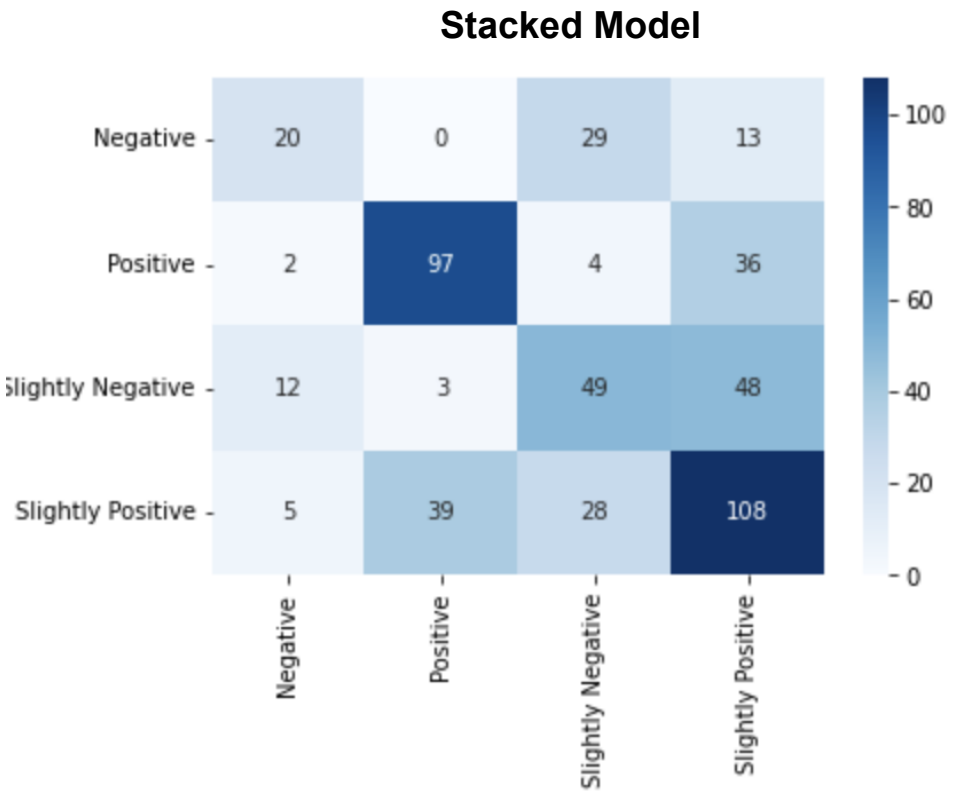


**Fig 5. Final Confusion Matrix for Stacked Model**

The confusion matrix shows the model's outputs for each of the four classes, and as you can see the distribution in Fig 5 is very similar to the distribution in Fig 2. The main drawback of the stacked model is that the data will have to be processed two times in order to first calculate the probability and then the classification result. We think that our accuracy of 76% is still quite low.

## 6. Conclusion

Now that this model has been trained, it is ready to be deployed and to predict new review data. The benefit of using a sentiment analysis model over a five-star rating system is that the model can output useful data in addition to the sentiment classification number

Some key takeaways from this project is that data cleaning plays a huge role in model accuracy. Handling missing values, feature engineering, feature selection, and adjusting proper hyperparameters can significantly improve machine learning models. As long as you have good data and the right model, you can really do anything with machine learning. NLP has many applications and this was just scratching the surface of what was possible.

**References:**


https://plotly.com/python/scatter-plots-on-maps/

https://towardsdatascience.com/how-i-consistently-improve-my-machine-learning-models-from-80-to-over-90-accuracy-6097063e1c9a

https://www.dataideology.com/data-modeling-key-to-success/

https://www.ics.uci.edu/~vpsaini/

https://medium.com/analytics-vidhya/performing-sentiment-analysis-on-yelp-restaurant-reviews-962334d6336d