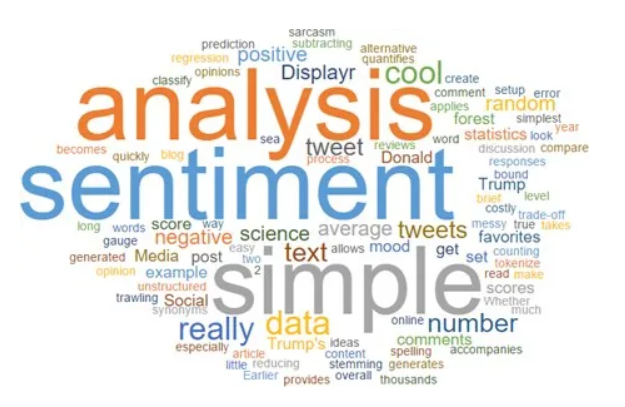
***Short Report***

***BSc Computer Science***

****

**Author Name:** Dhyan Nilesh Patel

**Assignment:** 6COM2000 Advanced AI NLP Mini Project Specification

**Student ID:** 21008521

**Professor:**[Olga Tveretina](https://herts.instructure.com/courses/107838/users/149)

The overarching goal is to develop a robust binary review classification model for a restaurant's customer reviews. The binary distinction entails categorizing reviews as either positive or negative, with positive reviews signifying customer appreciation and negative reviews indicating constructive criticism. The motivation behind this endeavour lies in the restaurant's intention to set up an in-house customer support team. This team aims to address and resolve issues raised in negative feedback, offering discounts as a gesture of goodwill to encourage customers to revisit. To accomplish this, I downloaded the dataset from Kaggle (<https://www.kaggle.com/datasets/d4rklucif3r/restaurant-reviews/data>), containing customer reviews along with labelled sentiments. The dataset is structured with two primary columns: "Review" and "Liked." The "Liked" column provides a summary of the sentiment distribution, showing that there are 500 positive reviews (Liked=1) and 500 negative reviews (Liked=0). This balanced distribution is conducive to training a sentiment analysis model that is representative of both positive and negative sentiments (1).

To gain a visual understanding of the most frequent words in positive and negative reviews, word clouds are generated as shown in Figure 1. The code combines positive and negative reviews separately into single strings, and then uses the Word Cloud library to create visual representations. These word clouds offer a quick and intuitive glimpse into the prominent words associated with each sentiment. Positive and negative sentiment word clouds are then displayed side by side for comparative analysis (2).

A close-up of words

Description automatically generated

Figure 1

The first step involves preparing the raw textual data for analysis. The code utilizes the Natural Language Toolkit (NLTK) to perform data preprocessing. This includes a series of operations to clean and standardize the text. The **clean\_data** function applies regular expressions to remove non-alphabetic characters, URLs, punctuation, numbers, and extraneous white spaces. Additionally, it converts the text to lowercase, tokenizes it into individual words, and applies stemming and lemmatization to reduce words to their root form. This comprehensive preprocessing ensures that the text data is in a consistent and suitable format for subsequent analysis. Next, we applied **clean\_data** function to the dataset and store it in the variable named cleaned\_corpus as a list (3).



Figure 2.1: Top 5 Reviews before data preprocessing



Figure 2.1 1: Top 5 Reviews after data preprocessing

Then we use the **word\_tokenize** function from the Natural Language Toolkit to break down each cleaned review into its constituent words. Next, we go into Tokenization and Preprocessing for Machine Learning. This is a bit technical, but essentially, we turn each review into a sequence of numbers. It's like giving each word a unique code. This makes it easier for a computer to understand and work with the text. We use a tool from the Keras library to do this.



Figure 2.1 2: Top 5 Reviews after applying word\_tokenize function

Moving forward, we will talk about Bag-of-Words Representation. Here, we turn our textual data into numbers. We create a matrix, which is like a table, where each row is a review, and each column is a unique word in all the reviews. The numbers in the matrix show how many times each word appears in each review. The conversion of the tokenized corpus into a bag-of-representation uses the technique CountVectorizer (4). This transformation simplifies the text data for machine learning models, providing a foundation for training a Naive Bayes classifier.

A black and white image of a number

Description automatically generated with medium confidence

Figure 3: Bag of Words Representation

The Naive Bayes Classifier is chosen for sentiment analysis due to its simplicity and effectiveness in text classification tasks. The code defines a custom Naïve Bayes Classifier class, which is trained on the Bag-of-Words representation of the reviews. The class calculates class probabilities and feature probabilities, incorporating Laplacian smoothing to handle unseen words. The Laplacian smoothing helps manage unseen words, preventing probabilities from becoming zero. During training, it learns from labelled data, calculating probabilities for each class and each feature (word). When making predictions, it considers the logarithms of probabilities to avoid small number issues (5). This classifier is then trained on specific data to predict sentiments effectively for the computer to learn patterns and train a model to understand whether a review is positive or negative. The balanced dataset allows the model to learn the conditional probabilities more accurately for both positive and negative classes. This balance ensures that the classifier does not become skewed toward one class, preventing biases that might occur in imbalanced datasets.

In reviewing how well the model performed, it achieved an accuracy of about 50.75% on the training data, meaning it got slightly more than half of the predictions right. This accuracy, though not super high, makes sense given the challenges of analysing sentiments in text. On the test data, the accuracy improved a bit, reaching around 56%. This suggests the model is doing a decent job with new, unseen examples. However, it's crucial to consider metrics beyond accuracy, especially given the balanced distribution of positive and negative reviews in the dataset. The overall report card, called the classification report, highlights a balanced performance with a score of around 53%. For positive reviews, the model is about 54% precise, meaning 54% of the positive predictions are correct. A higher recall of around 83% shows the model is good at catching a large portion of actual positive sentiments. However, specificity, which checks how well the model identifies negative reviews, is at 29%. This suggests there's room to get better at accurately identifying negative sentiments. In short, the model does well with positive reviews but can improve in recognizing negative ones (6).

A screenshot of a computer

Description automatically generated

Figure 4: Classification Report

A detailed breakdown from the confusion matrix reveals the model's performance intricacies, with 83 true positives, 71 false positives, 29 true negatives, and 17 false negatives. These values contribute comprehensively to the overall assessment, showcasing areas of proficiency and potential improvement for the model.

A screenshot of a computer

Description automatically generatedA green squares with white text

Description automatically generated

Figure 5: Confusion Matrix

The model is applied to unseen reviews from an external dataset. The unseen reviews are converted into cleaned reviews by using the **clean\_data** function. The cleaned reviews are transformed into a bag-of-words representation, and the Naive Bayes model predicts the sentiment. The predictions can aid the restaurant in proactively addressing potential issues raised by customers. While the implemented solution provides valuable insights, potential improvements could involve experimenting with different text preprocessing techniques, exploring more advanced machine learning models, and fine-tuning hyperparameters (7). Additionally, incorporating sentiment analysis libraries or pre-trained models could enhance the accuracy and efficiency of the classification process. However, the chosen approach strikes a balance between simplicity and effectiveness for the current problem statement.

# References

1. *Restaurant reviews* (2021). <https://www.kaggle.com/datasets/d4rklucif3r/restaurant-reviews>.
2. Wang, M. (2022) 'WordClouds with Python - MLearning.ai - Medium,' *Medium*, 21 September.<https://medium.com/mlearning-ai/wordclouds-with-python-c287887acc8b>.
3. Rastogi, K. (2022) *Text cleaning methods in NLP*.

<https://www.analyticsvidhya.com/blog/2022/01/text-cleaning-methods-in-nlp/>.

1. Great Learning Team (2022) *An Introduction to Bag of Words in NLP using Python | What is BoW?* <https://www.mygreatlearning.com/blog/bag-of-words/>.
2. Chaudhuri, K.D. (2023) *Building Naive Bayes Classifier from Scratch to Perform Sentiment Analysis*. <https://www.analyticsvidhya.com/blog/2022/03/building-naive-bayes-classifier-from-scratch-to-perform-sentiment-analysis/>.
3. *Compute classification report and confusion matrics in Python* (no date). <https://www.tutorialspoint.com/compute-classification-report-and-confusion-matrics-in-python>.
4. Paul, S. (2021) *Python Sentiment Analysis Tutorial*. <https://www.datacamp.com/tutorial/simplifying-sentiment-analysis-python>.