Project: Sentiment Classification

Course: Machine Learning COMP-544DL, UNIC

Professor: Dr. Ioannis Katakis

Student: Zois Panagiotis Lanaras

Introduction

In this project, we were provided with a dataset containing 8,000 text entries, each labeled with one of three sentiment categories: positive, neutral, or negative. The primary objective was to develop a sentiment classification model capable of accurately predicting the sentiment of unseen text data. As the performance of our submissions will be ranked based on their classification accuracy on unseen data, this metric was chosen as the primary focus of experimentation and evaluation throughout the project. All presented results were derived from experiments conducted on 90% of the dataset, with the remaining 10% reserved as unseen data for final evaluation. Additionally, each grid search was performed with 10-fold cross-validation using stratified sampling to ensure balanced representation of sentiment classes across folds.

Preprocessing

In this section, we detail the steps taken to clean and prepare the text data for further analysis and modeling. The primary goal of preprocessing was to remove extra noise and standardize the text while preserving its semantic meaning, thereby improving the performance of the classification models.

The following steps were applied in this exact order:

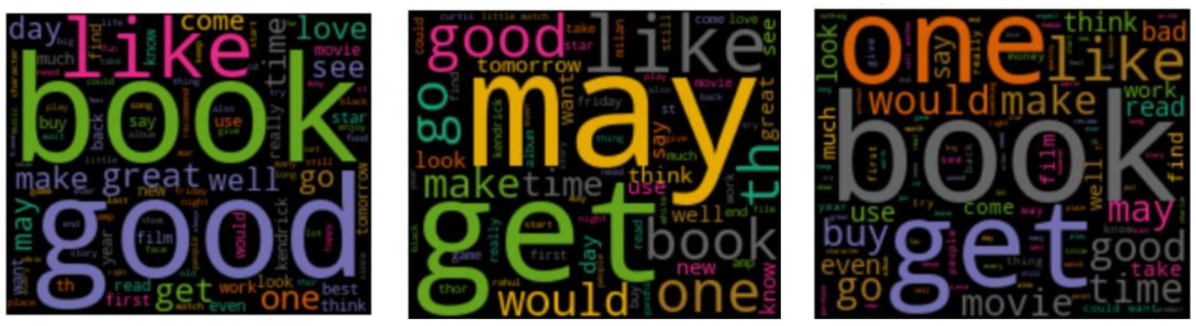
1. **Removing URLs**: removed URLs as they do not contribute to sentiment analysis.
2. **Handling Mentions and Hashtags**: removed mentions (e.g., @someone) and split combined words in hashtags (e.g., #SummerGreece → Summer Greece).
3. **Converting facial expression and Slangs to text**: Replaced common patterns that denote to facial expressions (e.g. **:)** to happy face) and slangs (e.g. **lol** to laughing out loud) to text.
4. **Basic Text Cleaning**: Converted text to lowercase, removed punctuation, and eliminated extra spaces.
5. **Translating Non-English Text** (30-40 texts where not in English)
6. **Stemming**: Applied Porter Stemmer to reduce words to their base form.

To validate whether the impact of certain preprocessing steps was positive, a grid search was conducted using the best performing classification algorithm. The results of this analysis are as follows:

- **Step 2**: Retaining the words after the hashtags improved performance.
- **Steps 3 and 4**: Replacing patterns for facial expressions and abbreviations.
- **Step 6**: Porter Stemming outperformed Lemmatization, and filtering stopwords was not beneficial.

Exploratory Data Analysis

Afterwards, exploratory data analysis was conducted to understand the dataset and its underlying patterns. The first observation was that the classes were **imbalanced,** with 38% of the total sentiments being positive, 35% neutral and 27% negative. Also, among the most frequent words in each sentiment class, many common words were observed, especially between neutral and positive texts, which raised concerns about whether the classification models will effectively differentiate between the sentiment classes given that they are also imbalanced.



(a) (b) (c)

Figure 1. World cloud of positive (a), neutral (b) and negative (c) sentiment texts.

Also, we were able to identify some outlier texts based on their number of tokens as 75% of the texts had less than 34 words while 10% of them had more than 100 tokens.

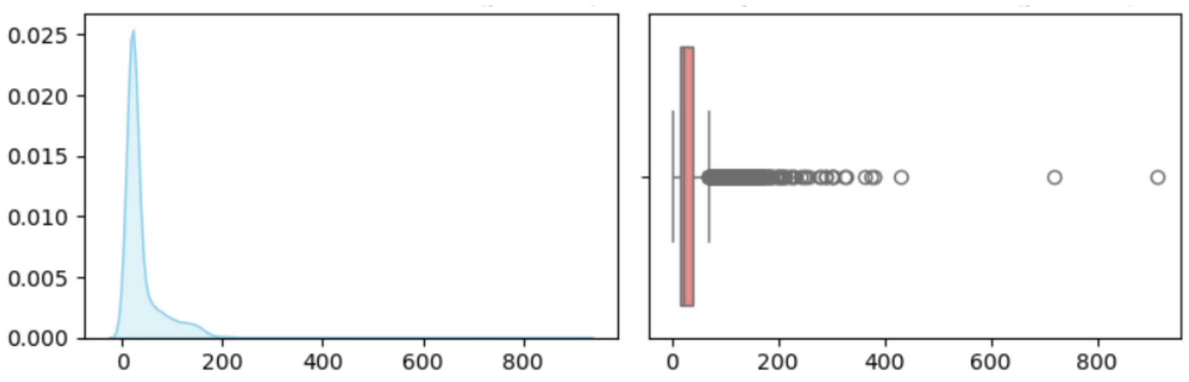


Figure 2. Distribution (a) and Boxplot (b) of total number of tokens for each text.

Based on these findings, **two more preprocessing steps** were applied to reduce extra noise on our data, **filtering words with length of 1** (artifacts from the earlier preprocessing steps) and **truncating texts to a maximum of 200 tokens** by retaining the first and last 100 tokens with the assumption that the most critical information is found at the beginning and end of each text. After applying all preprocessing steps, the **total number of unique tokens** was reduced from **32,500 to 17,500**, resulting in a more compact and manageable dataset.

Feature Extraction and Model Selection

The next action was to generate features from the texts, by creating a **document matrix** (Bag of Words) using **TF-IDF**, which evaluates word importance by considering its frequency in a document corpus and **CounterVetorizer**, which reprsesents the words as vectors based on the occurancies. The models that were selected for the sentiment classification task were:

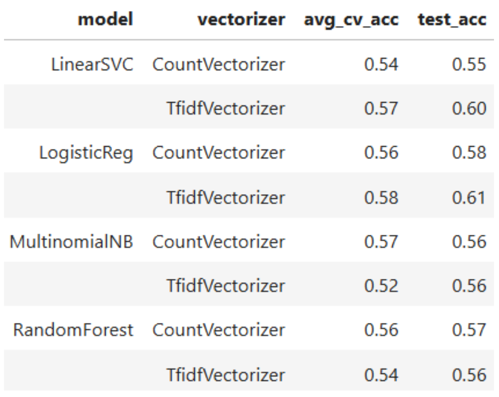
1.LinearSVC: SVM with a linear kernel, which separates data points of each class in the feature space by finding a hyperplane that maximizes the margin between classes.

2.Naive Bayes: A probabilistic classifier that calculates the likelihood of a word belonging to a particular class.

3.Random Forest: An ensemble method of decision trees.

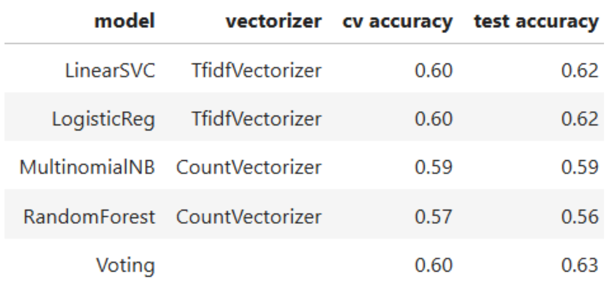
4. Logistic Regression: A regression-based classifier that predicts probabilities using a logistic function (e.g., sigmoid) and optimizes the classification boundary by minimizing log loss.

Parameter tuning was performed on the best performed vectorization method for each model. Parameters such as **min_df** and **max_df** (which exclude words appearing in fewer or more than min_df or max_df documents respectively) were tested to address potential overfitting caused by unique typographical errors in the text and to exclude highly frequent words that appeared across all sentiment classes on the EDA section.



model	vectorizer	avg_cv_acc	test_acc
LinearSVC	CountVectorizer	0.54	0.55
	TfidfVectorizer	0.57	0.60
LogisticReg	CountVectorizer	0.56	0.58
	TfidfVectorizer	0.58	0.61
MultinomialNB	CountVectorizer	0.57	0.56
	TfidfVectorizer	0.52	0.56
RandomForest	CountVectorizer	0.56	0.57
	TfidfVectorizer	0.54	0.56

(a)



model	vectorizer	cv accuracy	test accuracy
LinearSVC	TfidfVectorizer	0.60	0.62
LogisticReg	TfidfVectorizer	0.60	0.62
MultinomialNB	CountVectorizer	0.59	0.59
RandomForest	CountVectorizer	0.57	0.56
Voting		0.60	0.63

(b)

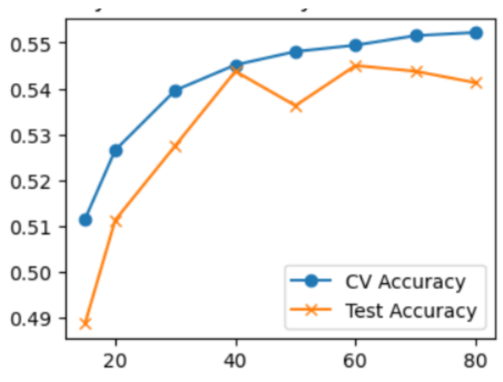
Figure 3. Average cross validation accuracy for each model with tf-idf and CounterVectorizer (a) and after parameter tuning the selected vectorizer (b). Random Forest was excluded from voting.

It is also noteworthy that all models performed better when **bigrams** were included in the feature extraction process, as this introduced some **word dependency**.

Parameter Tuning the models

The next phase was to parameter tuning the models by selecting a few key parameters to avoid overfitting. Additionaly, evaluation metrics such as f1 and precision were investigated.

The results of the cross validation with the best parameters that resulted from the Grid Search of each model were almost **identical** to those show on **Figure 3. (b).** Although Random Forest slightly improved after setting limits on some key parameters to avoid overfitting it was abandoned as the other models outperformed it.



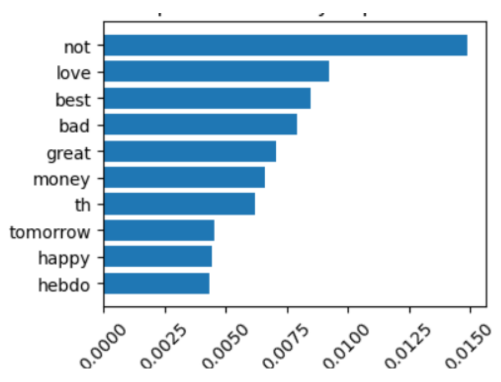
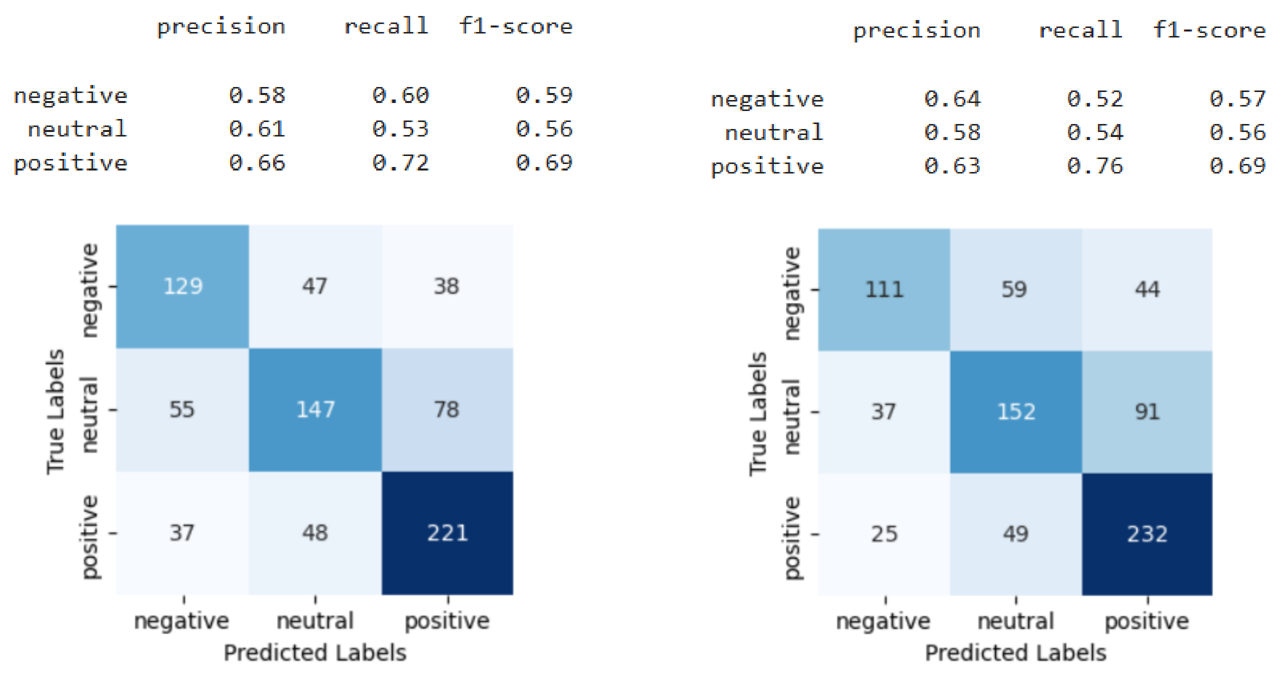


Figure 4. Test vs CV accuracy for different number of max_depth (left) and top 10 important features (right) of Random Forest. Note, from the EDA section, the word **th** was the 8th most frequent word on neutral texts often denoting random dates.

The most interesting observations were that, apart from LinearSVC, no other model effectively utilized class weights despite the class imbalance. This adjustment enabled LinearSVC to achieve the highest recall but the lowest precision for negative sentiments (as they constitute 27% of the total sentiments). Additionally, all models tended to overpredict positive sentiments.



```
          precision    recall  f1-score

negative       0.58      0.60      0.59
 neutral       0.61      0.53      0.56
positive       0.66      0.72      0.69
```

```
          precision    recall  f1-score

negative       0.64      0.52      0.57
 neutral       0.58      0.54      0.56
positive       0.63      0.76      0.69
```

Figure 5. Classification report and confusion matrix of LinearSVC (left) and Logistic Regression (right).

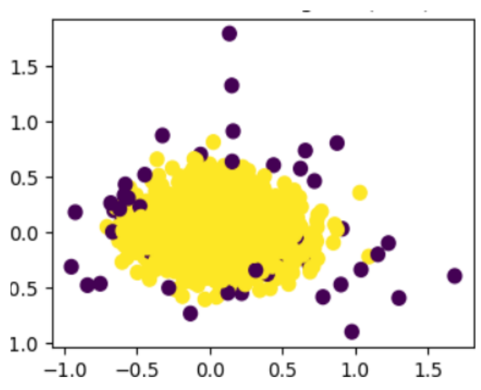
Investigating the poor performance of the models

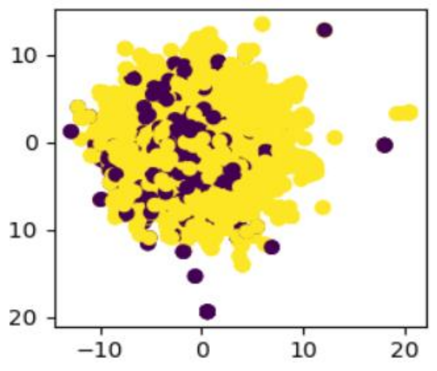
Since every model had poor performance before and after parameter tuning, an extra investigation was conducted based on the concerns that were raised from observations and results of the previous steps.

The first step was to modify the prediction logic of SVC by setting a probability threshold between 0.35 and 0.4 on the positive sentiments predictions to enhance the recall on the neutral sentiment, as each model was overpredicting positive and confusing neutral with positive sentiments. The second step was to apply dimensionality reduction using decomposition as SVC tries to separate data points of each class in the feature space. Since the initial unique words were 17,500 the introduction of bigrams increased the total features to 150,000 making the matrix even **sparser**. The results of both approaches were not promising.

Afterwards, an examination of potential outliers using DBSCAN clustering was conducted with two approaches. The first approach used GloVe embeddings, representing each word in a 50-dimensional space. Each text was represented as the average of its token embeddings, reducing the total features from 17,500 to 50. DBSCAN identified 2 clusters, with 60 potential outliers. Interestingly, some of these outliers had **incorrect sentiment labels**, though this observation was unrelated to the clustering process itself. However, excluding these outliers resulted in no improvement in cross-validation accuracy using LinearSVC.

components	avg_accuracy
200	0.56
500	0.58
1000	0.59
1500	0.59
2000	0.58
5000	0.56





(a) **(b)** GloVe with PCA **(c)** TF-IDF with SVD using t-SNE

Figure 6. Average cross validation accuracy of LinearSVC for different number of components of TruncatedSVD (a), DBSCAN clustering using GloVe embeddings (a) and TF-IDF (b).

The second approach was representing the data points to space using TF-IDF before applying DBSCAN. The combinations of the parameters epsilon and minPts that were tested were always leading to a high population of cluster of outliers (more than 50-60% of the total observations) due to the sparsity of our data points. Thus, Decomposition was applied again, and the clustering results were 2 clusters, with the outliers being 23% of the total observations. The exclusion of the potential outliers slightly enhanced the cross-validation accuracy of LinearSVC from 59.6% to 60.4%. This implies that either many observations on this 23% sample are true outliers or that the models start to overfit after a number of training samples.

Final model selection

Apart from the preprocessing and configuration of the feature extraction methods, no significant performance improvements were observed in the remaining experiments. The Voting Classifier was chosen as the final submission model with the exclusion of Random Forest. This decision was motivated by the observation that the remaining models, despite having identical results of 60% cross-validation accuracy, utilized different configurations of TF-IDF and CountVectorizer, leading to slightly different features and vocabularies. Thus, the Voting Classifier mitigates the risk of encountering out-of-vocabulary issues, particularly for texts with few tokens.

References

[1] An ensemble method with Sentiment Features and Clustering support, Nguyen Huy Tien, Nguyen Minh Le, https://aclanthology.org/I17-1065/

[2] Building a Custom Estimator for Scikit-learn: A Comprehensive Guide, https://www.geeksforgeeks.org/building-a-custom-estimator-for-scikit-learn-a-comprehensive-guide/

[3] PCA Explained Variance Concepts with Python, https://vitalflux.com/pca-explained-variance-concept-python-example/

[4] Up-to-date list of Slangs for Text Preprocessing, https://www.kaggle.com/code/nmaguette/up-to-date-list-of-slangs-for-text-preprocessing/notebook

[5] How to Translate Languages in Python, https://thepythoncode.com/article/translate-text-in-python