Frederico Lucas Marques da Silva – 99222 / Group 76
Pedro Agostinho da Cruz – 99297 / Group 76

Model

Count Vectorizer + Multinomial Naive Bayes

This model utilizes Count Vectorizer to transform raw text data into a vector of word frequencies. The Multinomial Naive Bayes (MNB) classifier uses these vectors to calculate the conditional probabilities of each word, given a particular sentiment label. These probabilities are then associated with the sentiment labels of the training data.

Count Vectorizer + Random Forest

In this approach, Count Vectorizer is again used for text vectorization. The classifier used here is Random Forest, an ensemble method that constructs multiple decision trees during training. The output class is determined by taking the mode of the classes predicted by the individual trees.

Support Vector Classifier + TF-IDF

For this model, raw text data is vectorized using the Term Frequency-Inverse Document Frequency (TF-IDF) algorithm. The Support Vector Classifier (SVC) then uses these vectors to find the optimal hyperplane that maximizes the distance between data points of different sentiment classes.

Pre-processing: Model-Based

For all the models mentioned above, the following text pre-processing steps were applied:

1) Lowercasing: Unifies capitalized and non-capitalized versions of words.
2) Removal of Stop Words: Eliminates words that lack significant meaning.
3) Lemmatization: Reduces words to their base or root form.
4) Stemming: Strips affixes to transform words to their root form.
5) Removal of Punctuation and Expressions: Reduces textual noise.
6) Using Placeholders: Retains meaningful expressions like, exclamation and question marks, three dots and emojis.

Transformer based models - Bert and Roberta:

Pre-processing: Transformer-Based

1) Tokenization: This involves breaking down the input text into tokens that are used in the sequences.
2) Embedding: Each token is mapped to a high-dimensional vector. These vectors, which are learned during training, encapsulate the semantic significance of each token.
3) Positional Encodings: These enable the model to understand the sequence in which tokens appear.

Transformer-based models are designed to process an entire sequence simultaneously. They utilize a stack of encoders and decoders to capture both the semantic and syntactic attributes of the input. The decoders then generate the output. Thanks to the concept of attention, these models can identify dependencies in lengthy phrases, which aids in data pre-processing.

We tested both the Bert and Robert model variants. Despite being pre-trained, the results were not as expected. We believe this was due to our training data being too small for the model to perform optimally.

Experimental Setup and Results

Evaluation Measures:

Cross-Validation: Utilized 10-fold cross-validation with accuracy as the scoring metric.

Train-Test Split: Employed an 80-20 split for training-testing data. The random state parameter was set to 1.

Dataset Used:

The dataset for this project was sourced exclusively from the provided training file. Given the absence of labels in the external test set (test_just_reviews.txt), we generated our own test sets from the training data for model evaluation.

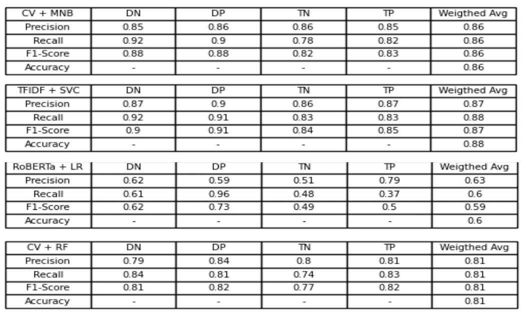
Parameter Tuning:

For hyper-parameter optimization, GridSearchCV was universally applied across all models.

Model Performance:

Tabulated Results:

Table 1: accuracies of sklearn function "classification_report". Performance metrics including F1-score, precision, recall, and weighted average were tabulated for each model.



CV + MNB	DN	DP	TN	TP	Weigthed Avg
Precision	0.85	0.86	0.86	0.85	0.86
Recall	0.92	0.9	0.78	0.82	0.86
F1-Score	0.88	0.88	0.82	0.83	0.86
Accuracy	-	-	-	-	0.86

TFIDF + SVC	DN	DP	TN	TP	Weigthed Avg
Precision	0.87	0.9	0.86	0.87	0.87
Recall	0.92	0.91	0.83	0.83	0.88
F1-Score	0.9	0.91	0.84	0.85	0.87
Accuracy	-	-	-	-	0.88

RoBERTa + LR	DN	DP	TN	TP	Weigthed Avg
Precision	0.62	0.59	0.51	0.79	0.63
Recall	0.61	0.96	0.48	0.37	0.6
F1-Score	0.62	0.73	0.49	0.5	0.59
Accuracy	-	-	-	-	0.6

CV + RF	DN	DP	TN	TP	Weigthed Avg
Precision	0.79	0.84	0.8	0.81	0.81
Recall	0.84	0.81	0.74	0.83	0.81
F1-Score	0.81	0.82	0.77	0.82	0.81
Accuracy	-	-	-	-	0.81

Confusion Matrices:

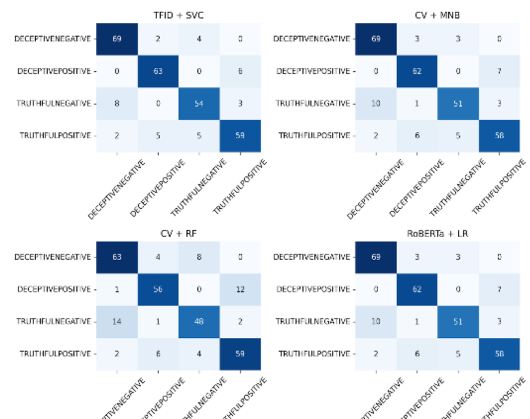


Figure 1: Confusion matrices of models. These were generated for each model, providing insights into true positives, true negatives, false positives, and false negatives.

Graphical Representation:

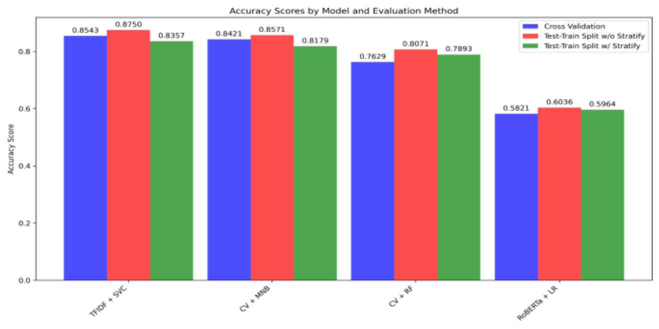


Figure 2: Accuracy Scores by Model Evaluation Method. A graph was plotted to illustrate the accuracy variances between Cross-Validation and Train-Test Split evaluation techniques.

After looking at all the results the TFID+SVC model was elected as the best model and will be used in the discussion.

Discussion

When looking at the mislabeled results, we decided to focus more on the mislabel categories which happened more frequently, as they would allow for a better understanding of the limitations of our model. To find possible patterns that happened within the mislabel data, we have taken the mislabeled data input and made a count of the number of both the positive and negative expressions we could find, here at the findings:

TRUTHFULPOSITIVE vs DECEPTIVEPOSITIVE and DECEPTIVEPOSITIVE vs TRUTHFULPOSITIVE.

For this case, we noticed that the consistent pattern in the mislabeled input was that only positive expressions were found without the finding of negative expressions. For example, in |train.txt - 443|, we identified 8 positive and 0 negative expressions, happening the mislabel of *TRUTHFULPOSITIVE* to *DECEPTIVEPOSITIVE*. In |train.txt - 893| we found 6 positives and 0 negatives happening the mislabel of *DECEPTIVEPOSITIVE*. to *TRUTHFULPOSITIVE*. Therefore, we believe that the lack of a contrasting expression, or an imbalance where the count of positive or negative expressions significantly outweighs the other, made our model had a hard time distinguishing between truthful and deceptive for positive labels.

In regards of the mislabeling of *TRUTHFULPOSITIVE* to *TRUTHFULNEGATIVE*, we noticed that although in the original phrases there were more positive expressions than negative expressions, after our text processing we often remove some expressions with positive tone. Look at |train.txt - 1192|, which includes the passage "*I believe I cannot describe how amazing was…",* after it was processed, our model transformed the positive toned expression "*amazing*" to "*azing*" which doesn't make sense neither has tone. This is significant in sentiment analysis, because the presence or absence of specific words can greatly influence the labeling of the data. Moreover, since this model is using bigrams, he could often identify expressions like "*be better*" as a negative expression. The discrepancy between the number of negative and positive reviews has led the model in this case, and cases like this one, into thinking the text was overall negative, classifying it as *TRUTHFULNEGATIVE*.

When it comes to the mislabel *TRUTHFULNEGATIVE* vs *DECEPTIVENEGATIVE*, we have noticed that despite having expressions with a very negative meaning on the phrase, if the model starts by receiving positive expressions, it will label the phrase as deceptive. Let's look at the example: |train.txt - 556|. In this case, the phrase starts by giving positive comments such as: "*was great*", "*was overall great*", "*luxury hotel*". This, in conjunction with the other negative comments, seems to lead the model into thinking that this could be a deceptive review when in reality is a truthful negative review. More cases like this were also encountered.

Future Work

In the future, we hope to better explorer transformative models like roBERTa and BERT but also XLNet by leveraging a bigger dataset for that purpose. Additionally, we would also like to have more time to fine-tune our models parameter using more options in GridSearchCV as well as explore different model approaches.