Copenhagen Business School
HANDELSHØJSKOLEN

Sentiment Analysis with Emojis

Natural Language Processing and Text Analytics [KAN-CDSCO1002U]

Written Product for Exam

Group Members

Julia Caroline Louisa Hahn - juha22af@student.cbs.dk - 158369

August Kauffmann - auka15ac@student.cbs.dk - 103299

Aleksandra Sonia Durska - aldu22ae@student.cbs.dk - 158370

José Maria Elvas Ferrari Careto - jofe22ac@student.cbs.dk - 158288

Examiners

Daniel Hardt

Rajani Singh

Sine Zambach

Submission Details

Date of submission: 30-05-2023

Number of characters: 29.281

Number of pages: 14

Table of Contents

Abstract

Emojis and emoticons are commonly used to express meaning and emotions in written communication, yet they are often overlooked in natural language processing (NLP) tasks. This project investigates the potential benefits of including emojis in sentiment analysis using various machine learning models. The study utilises a corpus of 64.599 labelled tweets representing negative, neutral, and positive sentiments. Three distinct datasets are created, treating emojis in three different ways: unprocessed, translated to text, and stripped. A VADER model, a random forest model, and a RoBERTa model are trained, tuned, and compared for sentiment analysis on these dataset variations. The findings demonstrate that including either translated or unprocessed emojis significantly improves evaluation metrics across all three models. Based on these findings, the project emphasises the importance of emojis in sentiment analysis and argues that further research is needed on the processing of modelling of emojis.

1. Introduction

In the age of social media, where discussions on various topics are happening every minute, the way feelings are expressed has evolved beyond traditional text-based communication (Gnana Priya, 2019). Emojis, graphic representations of emotions, have become a popular way to enhance and even replace written expressions (Erle et al., 2022; Mei, 2019).

This emoticon trend has gained significant attention, with Statista revealing a continuous rise in the presence of emojis within tweets - from 14.52% in 2016 to 20.69% in 2021 (Dixon, 2021). Researchers have recognized the value of emojis as indicators of sentiment and have developed models that can classify the sentiment based on the presence of emojis. These models can be used for a wide array of tasks ranging from analysing voter sentiment during elections to sentiment on stock prices (Kralj Novak et al., 2015).

Despite the widespread adoption and significance of emojis in the digital age, their incorporation in sentiment analysis models still presents its own set of complications, e.g. when dealing with ambiguous text or when emojis are used ironically (Wang et al., 2020; Farías et al., 2016). Notably, the existing literature sheds light on the underrepresentation of emojis in labelled Twitter datasets, highlighting a knowledge gap about their impact on the performance of sentiment analysis models (McShane et al., 2021). This raises the following research question:

How does the presence of emojis affect the performance of machine learning models in sentiment analysis of tweets?

In order to investigate this question, we will deploy three different models and compare their performances on tweets with and without emojis.

2. Related work

Several past publications have focused on identifying sentiments of texts through the use of emojis and emoticons.

Novak et al. (2015) pioneered the creation of a lexicon to analyse the emotional content conveyed by emojis. Through a sentiment examination of the most recurrent emojis in Twitter posts, it was discovered that most emojis lean towards a positive emotional expression. The authors computed sentiment scores for various emojis based on how positive or negative the tweets they appeared in were. This study is also the original source of the dataset used in this project.

Another study conducted by Eisner et al. (2016) highlights the shortcomings of NLP applications for social media that solely rely on pre-learned word meanings and overlook the importance of emojis in their understanding. To address this issue, Eisner et al. (2016) developed vector representations of the

sentiments conveyed by emojis. Eisner et al. (2016) employed the Random Forest and linear Support Vector Machine (SVM) classifiers, both with and without the incorporation of the emoji2vec tool. They discovered that integrating emoji2vec significantly improved the overall accuracy of text classifications, particularly for texts containing emojis. When solely utilising Google News embeddings, the accuracy achieved was 46.0% with Random Forest and 47.1% with Linear SVM. However, when incorporating emoji2vec embeddings into the Google News embeddings, the accuracy increased to 54.4% accuracy using Random Forest and 59.2% using Linear SVM (Eisner et al., 2016).

3. Methodology

3.1 Conceptual framework and Training Strategy

The figure below provides an illustration of the conceptual framework of the code and the project:

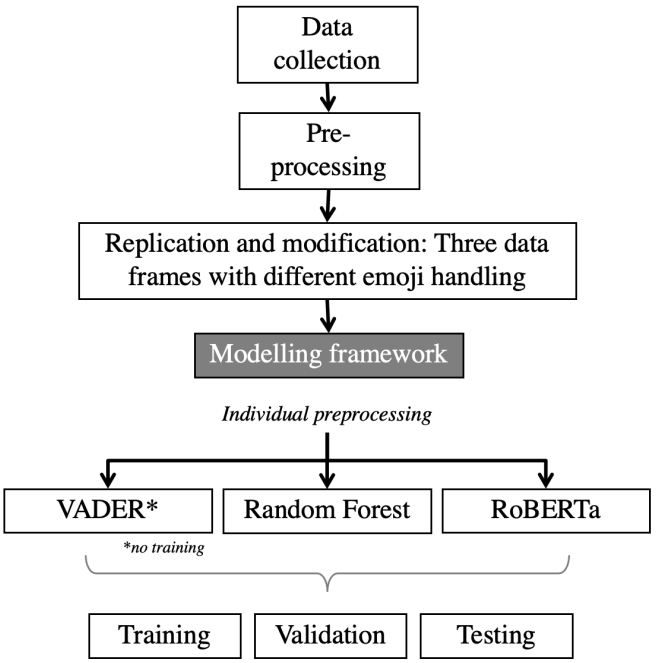


Figure A: Visualisation of the conceptual framework

A corpus of labelled tweets is first subjected to various pre-processing treatments. Next, three copies of the dataset are created, where emojis are processed in different ways. These three datasets are fed into three different models: VADER, Random Forest and RoBERTa.

We used a 70:15:15 training/validation/testing split for our modelling. The training set is used to train Random forest from scratch as well as further train the pre-trained RoBERTa model. VADER will not be trained as it is a rule-based model. The validation split is used to get initial results for the three models using default hyper parameters. Next, various hyper parameters are tuned for Random Forest and RoBERTa on the validation set. Finally, the three models will be compared in the test set.

4. Dataset Preparation

The following sections describe the exploratory data analysis (EDA), pre-processing, and various dataset transformations in order to effectively prepare the dataset for model training, tuning and testing.

4.1 Dataset Description

The dataset used in this study, titled Emoji2Vec, is publicly available on the open-source platform, GitHub (Eisner & Rocktäschel, 2016). The origins of this dataset trace back to the work of Kralj Novak et al. (2015) who manually labelling the data set with positive, neutral, or negative sentiment. The dataset consists of 87,429 rows and 3 columns: the tweet's unique ID, the text of the tweet itself, and the associated sentiment (Eisner & Rocktäschel, 2016). The dataset exclusively contains English tweets, and not all the tweets include emojis. Presented below are a few sample tweets extracted from the original dataset:

ID	Text	Sentiment
514382643602141184	RT @5SOS: Our bus is 100% rock n roll http://t...	Positive
512775998668955648	@alexsnowden_ what the hell snowballs	Negative
514447604982513664	Chelsea just fell off her chair 😂😂👏👏	Negative
512769161974382592	Bucs should be playing at 7 tomorrow night	Neutral

Figure B: Example tweets from the original dataset (DF)

For a detailed breakdown of the tweets classified under positive, negative, and neutral sentiments considering both the presence and absence of emojis, refer to Figure A.1 in the appendix.

4.2 Exploratory Data Analysis (EDA)

We used various EDA techniques to determine key characteristics of the dataset and gain valuable insights for pre-processing. This involved examining the number of instances, attributes, data types, missing values, duplicates and class distributions. Furthermore, the ID column is dropped as it does not contribute any meaningful value to our sentiment analysis of tweets. There were no duplicates in the dataset, but 22,830 rows had either no tweet or no sentiment assigned. These rows were dropped resulting in a final dataset of 64.599 tweets.

While examining the distribution of sentiments, an imbalance was observed in the data with nearly half (46%) of the tweets categorised as neutral, while positive and negative sentiments accounted for only 28% and 26% respectively (see Appendix A.2). We considered up-sampling or down-sampling the dataset in order to make the models learn more from positive/ negative tweets and less from neutral tweets. However, we decided not to address the class imbalance as it would result in our dataset not being representative of real-world Twitter comments, which in turn could harm the generalizability of the models.

4.3 Pre-Processing

Additionally, prevalent abbreviations used within the "Text" column, for instance, "lol" (laughs out loud) have been transformed into their corresponding full forms by using the Abbreviations.csv, sourced from a public set available at Kaggle (Mbaye, 2020). This transformation was achieved by implementing a function called "convert_abbrev" which utilises regular expression patterns to replace these abbreviations with their complete expressions.

4.3.1 Main preprocessing

When handling text data, thorough analysis is a prerequisite before using it in modelling. Regular expressions are an essential tool in this process and offer a powerful and flexible method for pattern matching and text manipulation (Jurafsky et al., 2008) and are used in the following functions.

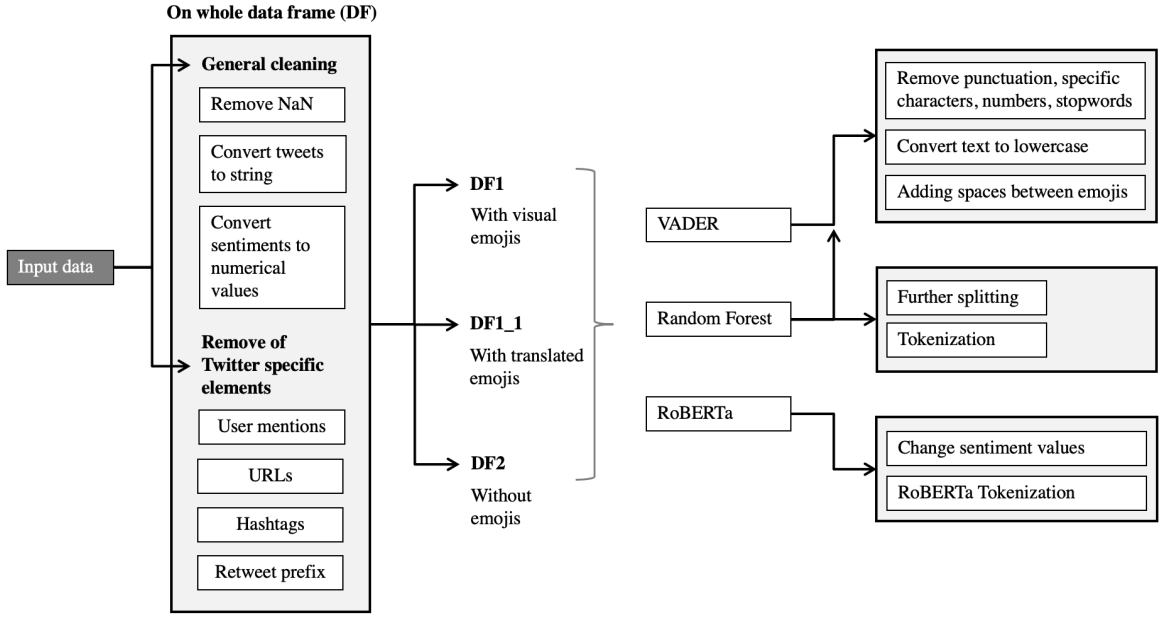


Figure D: Visualisation of preprocessing workflow

The preprocess_text function is applied to all three dataframes. Within this function, several steps are taken to enhance the data quality and ensure consistency. Firstly, any missing or invalid data is removed, and all values in the "Text" column are converted to strings for uniformity across the dataset. To maintain focus on the textual content, specific Twitter-related elements are eliminated, including user mentions starting with the @ symbol, URLs beginning with http, hashtags, and the "RT" (retweet) prefix found in shared tweets. Additionally, multiple consecutive spaces are removed. Sentiment values expressed as words (Positive, Neutral, Negative) are translated into their corresponding numeric representations (1, 0, -1) for optimal utilisation of the NLP models at a later stage of this research. Lastly, the add_space_between_emojis function is called which inserts spaces between the emojis in every row of the "Text" column to allow for better separation and distinction

between individual emojis and the text. For instance, the text “Great👍🌞” would be transformed into “Great 👍 🌞”.

After this cleanup, the dataset consisted of a total of 64,599 entries available for further analysis. An overview of the dataset structure after the main preprocessing can be seen in Figure E.

Text	Sentiment
Our bus is 100% rock n roll	1
what the hell snowballs	-1
Chelsea just fell off her chair face with tea	-1
Bucs should be playing at 7 tomorrow night	0

Figure E: Dataset structure based on DF1_1 after preprocessing

4.3.2 Dataset replication

Once the crucial steps of data preprocessing were completed, the dataset was replicated into three distinct dataframes - DF1, DF1_1, and DF2 - to effectively address the research question. The first dataframe, DF1, preserves the original dataset as it is, including all emojis represented as visual images. The second dataframe, DF1_1, replicates the contents of DF1 but translates the emojis into textual descriptions. For example, an emoji represented as “😂” in DF1 will appear as a corresponding text description like “:face_with_tears_of_joy:”, the underscores and colons are deleted from the translation in DF1_1. In contrast, the third dataframe, DF2, is a modified version where all emojis have been completely removed, leaving only the textual data for analysis, without any emoji representation. This is also represented in the table below:

Data frame	Characteristics	Example
DF1	With visual emojis	😂
DF1_1	With translated emojis	face with tears of joy
DF2	Without emojis	

Figure C: Overview of the characteristics of each dataset

4.3.3 Individual preprocessing

In order to translate the visual emojis to their respective textual description the emoji.demojize function from the emoji library is used on DF1_1. Specifically designed for use with DF2, the remove_emojis function ensures that the text columns in the DF2 are cleaned of all emoji characters.

For all datasets the data is split before the model's implementation . The split_data function divides the data into train, validation and test set, based on the previously mentioned fractions. The datasets are globally divided to ensure the most comparable accuracy for all of the models which is received by inputting the same Text columns into all models.

4.3 Data Transformations for NLP Models

Once the primary preprocessing steps have been completed, additional model-specific steps are applied for proper and optimal utilisation of the models. To optimise the data for VADER, further processing is applied using the pre_process_text2 function. This function removes punctuation and specific characters, converts the text to lowercase, eliminates numbers, and removes common English stopwords. Notably, tokenization is intentionally avoided as it hurts the model.

For Random Forest, besides applying pre_process_text2, the Text column is tokenised. Furthermore, it has been determined through testing that applying lemmatization negatively impacts the model's performance and is, therefore, not used. Moreover, for this particular model, the data splits are processed even further by dividing the data frames into two series based on the column name.

In the case of RoBERTa, as little preprocessing as possible was done. Punctuation is removed and tokenization is applied, however, this time as part of the model, as RoBERTa has its own tokenizer.

5. Models

5.1 Choice of Models

Three models have been chosen in this study. These three models were chosen as they have very different architectures, complexities and training strategies. The rule-based model VADER (Valence Aware Dictionary and Sentiment Reasoner) was chosen as a baseline model as it has a rather simple architecture, is designed for sentiment analysis and is effective at capturing short texts and emoticons (Hutto & Gilbert, 2014). Random Forest was chosen as it is useful for multi-classification, has an intermediary architecture complexity and can be trained from scratch. RoBERTa (Robustly Optimised BERT) is a pre-trained model and was chosen for its ability to understand context as well as its impressive performance on various NLP benchmarks (Liu et al., 2019; "RoBERTa", 2019).

5.2 VADER

VADER is a model that utilises an intensity score ranging from -1 (very negative) to +1 (very positive) to determine the sentiment of texts. The score is determined by combining the valence score, which assigns positive or negative sentiment to each word, with context awareness that encompasses syntactical considerations and word intensifiers like “very” or “slightly”. If the score exceeds 0.05, it is viewed as positive, whereas a score below -0.05 is seen as negative. Anything in between is considered neutral. (Hutto & Gilbert, 2014).

5.3 Random Forest

Random Forest is a machine learning model that works on multiclass classification and accepts emojis in their natural form. Random Forest is an ensemble method that combines multiple decision trees to generate accurate predictions (Breiman, 2001). For this project, the RandomForestClassifier from Scikit-learn is used. All the parameters are set to their default values except n_jobs and warm start.

n_jobs has been specified to -1, which allows the model to train trees in parallel by using all cores available, thus ensuring faster training speed. The warm start parameter allows the model to reuse existing fitted models when fitting new ones with additional trees. This speeds up model training significantly.

5.4 RoBERTa

The RoBERTa model was designed and trained by Facebook AI and builds on the same basic architecture as the BERT (Bidirectional Encoder Representations from Transformers) model. Compared to BERT, RoBERTa uses a more extensive masking strategy, has more parameters, and is trained on a much larger text corpus. The logic behind using a pretrained model, such as RoBERTa, is that the model learns general language skills from the transfer learning, and then gets fine tuned for the specific task of identifying tweet sentiment on our dataset.

In our code, we define a function for training the RoBERTa model. We set the max length of the input size to 140 to match Twitter's character limit in 2015. This step is important as input size has a large impact on GPU memory.The tweets are fed into a custom class CustomDataset, which prepares them for training with pytorch. We use the AdamW optimizer and a batch size of 32 for training as is typical for BERT-based models. We initially set the number of epochs to 1 and used a learning rate of 0.0006, which gradually is reduced using a cosine function. After model training, the trained model and tokenizer are saved locally. In order to save GPU memory we deleted all tensors as well as the tokenizer and the model after saving them locally.

6. Results

Please note that the results presented in the following may slightly vary compared to the ones uploaded in the jupyter notebook file due to random variations upon each execution.

6.1 Initial results

Throughout the assignment, it is mainly focused on the accuracy measure when evaluating the performance of the different models and datasets, however more detailed results, including the F1, recall, precision, as well as confusion and classification matrix are presented in the appendix. An overview of the initial results based on the accuracy measure of all models on the validation set can be seen below.

Model/DF	VADER	Random Forest	RoBERTa
Without emojis (DF2)	52.31%	59.58%	69.06%
Translated emojis (DF1_1)	53.88%	61.76%	70.45%
Visual emojis (DF1)	52.56%	61.97%	70.18%

Figure F: Model's Accuracy on Validation sets

The baseline model, VADER, has an accuracy of 52.31% on the dataset without emojis. The inclusion of translated emojis slightly increases the accuracy to 53.88%, but it decreases to 52.56% for the dataset with translated emojis. For Random Forest, the lowest accuracy was also observed for the dataset excluding emojis 59.85%. Both datasets with emojis performed better than the one without emojis, with the model using visual emojis outperforming the one with translated emojis with 61.97% and 61,76% respectively. In comparison, RoBERTa performed significantly better than the other two models across all three datasets with the highest performance of 70.18% on the dataset including visual emojis and the lowest on the dataset excluding emojis (69.06%).

6.2 Tuning

Next, we will propose the tuning decisions for both the Random Forest model and the RoBERTa model.

6.2.1 Random Forest

The tuning of the Random Forest was divided into two separate sections. First, following the directions of Thorn (2021), we have tested a different recommended criterion parameter for this model. According to the author, who specifically describes hyperparameter adjustments for tweet processing, we have tested our model on Entropy. The Gini criterion is a default, and the log-loss is disregarded. Based on the accuracy of both we have decided to proceed with Entropy, but the difference was marginal.

As a second step in the processing, we have decided to do a parameter grid search over the number of estimators and the maximum features parameters. We have decided to check both sqrt and log2 and disregard None for efficiency reasons. We have also decided to analyse seven different possibilities of trees. Based on Thorn's (2021) graph it can be seen that the error rate for the predictions is falling drastically with the number of trees with the peak point being very close to zero and then declining further at a lower rate. Consequently, based on Figure K we have chosen to check the numbers that are very close, mainly the first hundred and two of them being greater than that. Before setting the grid search, we also checked manually the different number of observations to determine where the cut off point was. We have tested values 300, 500 and 1000. The accuracy values have not improved at all with more than 300 trees, which led us to a final seven possibilities to be considered; it is also in line with the graph presented below. The grid search that was performed to find the best parameters

iterates over each of the maximum features and number of estimators, checking 14 combinations of possible improvements in the model. Based on the results of this grid search, we apply the best features being: Entropy criterion, 100 decision trees and square root to the test sample. It is worth mentioning that due to computability and time constraints we only applied the grid search to the model that includes the visual representation of emojis (DF1).

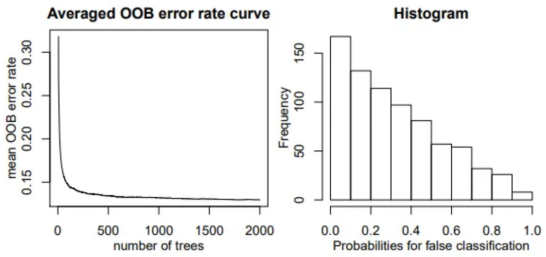


Figure K: Impact of the number of trees on the performance of the model (Thorn (2021))

6.2.2 RoBERTa

When tuning the trained RoBERTa model, we considered three hyper-parameters: learning rate, batch size and number of epochs. We performed univariate tuning due to training time constraints. Also, initial experimentation showed that the optimal hyper parameters did not change much based on the values of the other hyper parameters. First, we tuned the learning rate which resulted in the following accuracies on the validation set:

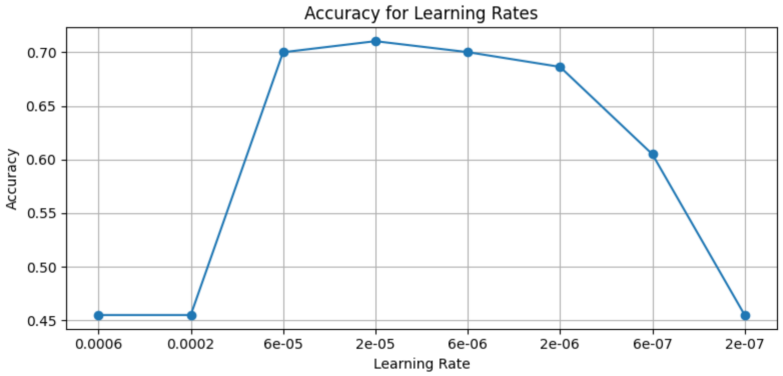


Figure G: Accuracy for different learning rates

When the learning rate gets very small, the accuracy gradually drops. This can be explained by slow convergence, the model's inability to escape local minima during the descent, or overfitting. For high learning rates, the accuracy is very poor, which can be explained by the model "forgetting" the outputs from earlier layers. As training time remained consistent across the different learning rates (see Appendix B.3), we picked the learning rate (0.0002) with the highest accuracy (71.11%).

Next, we tuned the training batch size:

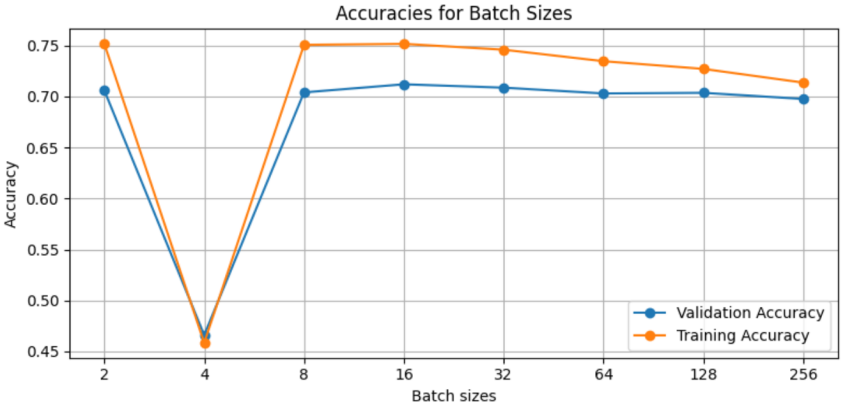


Figure H: Accuracy for different batch sizes

Choosing a too small batch size can lead to overfitting and longer training time due to the increased number of steps during the gradient descent. In our output there was a sudden huge drop in accuracy. It is unclear what exactly caused this huge drop, but we decided to shy away from using low batch sizes (2, 4 and 8) for this reason. For larger batch sizes (16+), the validation and training accuracies start to converge, indicating a reduction in overfitting, but the validation accuracy drops as well. The ladder can be explained by the fact that large batch sizes can lead to underfitting as the model fails to learn from the diversity of the individual examples in each batch. Another consideration is the training times, as displayed below:

Batch size	2	4	8	16	32	64	128	256
Training time (s)	1163	598	382	319	304	283	272	265

We selected a batch size of 16 as it maximises the validation accuracy, while not having a significantly higher training time than batch sizes of 32+. Finally, we tuned the number of epochs using the optimal learning of 0.0002 and a batch size of 16:

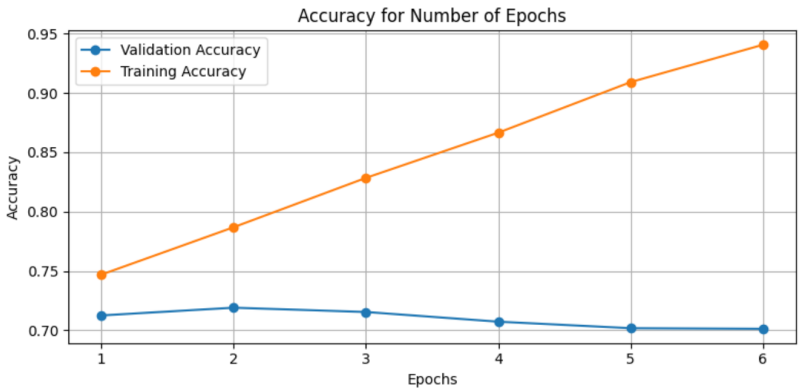


Figure J: Accuracy for number of Epochs

The graph shows a linear increase in training accuracy, while validation accuracy remains flat or is slightly decreasing. This is a clear sign of overfitting. There is an almost linear relationship between number of epochs and training time:

Epochs	1	2	3	4	5	6
Training time (s)	341	627	933	1241	1548	1856

Using a batch size of 1 almost cuts training time in half but it reduces validation accuracy by roughly 0.65% on the validation set. In the end, we decided to use 2 epochs as we had access to a very fast GPU during training and inference. The final hyper parameter settings are summarised below:

Hyper parameter	**Learning rate**	**Batch size**	**Epochs**
Value	0.0002	16	2

6.3 Final results

Model/DF	**VADER**	**Random Forest**	**RoBERTa**
Without emojis (DF2)	52.58%	60.11%	70.52%
Translated emojis (DF1_1)	53.72%	61.38%	70.61%
Visual emojis (DF1)	51.97%	61.80%	71.06%

Figure K: Summary of accuracy measures from all models on the Test set

When considering the final test results, the VADER model has an accuracy of 52.58%, which increases slightly to 53.72% with translated emojis but drops to its lowest point with visual emojis. For Random Forest the accuracy for the dataset with emojis is 61,8% and slightly lower where the emojis are translated (61.38%). As in the initial results the model performs the worst on the data that excludes emojis with the accuracy of 60.11% RoBERTa performs least effectively without emojis with an accuracy of 70.52%, slightly improves with translated emojis, and achieves its highest performance with visual emojis with accuracy of 71.06%. From examining the confusion matrix, it can be seen that the model correctly identified 2104 tweets as positive and 1755 as negative. On the other hand, there were some errors too: 213 tweets were mistakenly predicted as negative when they were actually positive, and vice versa for 169 tweets (Appendix B.3.6)

7. Discussion

7.1 Comparison and evaluation of results

The VADER sentiment analysis model shows the best performance when emojis were converted into text, while it performed worst when emojis were shown as visual images, with scores of 53.72% and 51.97%, respectively. Considering that guessing all tweets to be neutral would still get about 50% right, it's clear that VADER's overall accuracy is not particularly good. Consequently, this makes it difficult to answer the research question regarding the influence of emojis on the performance of the model as the results are quite mixed and somewhat controversial, making it even more difficult to draw profound conclusions.

Random Forest model shows its best performance when dealing with the visual-emoji dataset and worst when emojis are excluded. When lemmatization was tested with the Random Forest model, the outcome was nearly identical to that of VADER. As a result, it has been decided to not include lemmatization, which led to a notable boost in accuracy – to around 60% for all three datasets. Both, the initial and final results, demonstrate that Random Forest works best with visual emojis. Therefore, it can be concluded that the additional context provided by emojis, whether as text or visual form, proves to be useful and beneficial for the Random Forest model to classify the sentiment of tweets. The fine tuning of Random Forest parameters is almost the same as the default parameters, which indicates that the initial results are extremely close to the best possible performance of the model.The results of Random Forest after performing the grid search for the best parameters are marginally lower than in the initial run which can be most likely attributed to the randomness in the training process and the very close tuning of the default parameters. The tuning on the dataset without any sort of emojis has accuracy increased the most out of the three datasets. This can lead to a conclusion that the fine tuning mainly improves the textual context in the sentiment analysis, which is also in line with the improvement of the translated emojis.

The RoBERTa model also showed improvement from training to final results across all three datasets. For example, when analysing the data frame containing visual emojis, the initial score achieved by RoBERTa was 70.29%, while the score increased to 71.06% after fine-tuning. These improvements underline the pivotal role of choosing the optimal hyperparameters, in this case, a learning rate of 0.0002 and a batch size of 32 as they clearly improved the model’s performance.
The translated emoji dataset achieved the second highest accuracy score of 70.61% after the visual emoji dataset, which is consistent with the fact that RoBERTa models tend to perform the best on texts that have not been processed much.
Among all models, RoBERTa achieved the highest accuracy score of 71.06%, which means it correctly classified the sentiment of more than two-thirds of the tweets. This is most likely due to the fact that RoBERTa is the only model, compared to VADER and Random Forest, that takes the context and, thus, much larger scale of the pattern of the tweets into account. Especially in text classification, this factor upholds enormous importance as the sentiment of text, in this case tweets, cannot solely be

classified based on a single word but oftentimes depend on the whole sentence. Moreover, the results initialise RoBERTa's robustness and ability of generalising well to unseen data, meaning the model is prepared to be used beyond the chosen Twitter dataset.

7.2 Limitations and Risks

There are several points to address in terms of risks and limitations connected with this study. Despite being true to nature, our dataset presents a class imbalance towards a larger set of neutral tweets compared to the others. This in turn may affect the accuracy and generalisation of the results. This issue is seen in Random Forest models, as they tend to favour majority classes in the datasets. Moreover, by being labelled by humans, the dataset is prone to limitations arising from human subjectivity in this process. Furthermore, both the Random Forest and VADER models fail to capture the contextual nuances of the data, meaning these models may miss the sentiment a tweet might convey in its particular context. Another notable limitation of Random Forest is the lack of the ability to comprehend the misspelt words and other 'out-of-vocabulary' phrases that are not commonly used. Lastly, the ongoing introduction of new emojis poses a difficulty in sentiment analysis due to the fact that their meaning is not universally established. Emojis can be interpreted in a notably varied manner across different individuals, cultures, and time-periods as highlighted by Barbieri et al. (2016), complicating the task for models to accurately classify sentiments.

8. Conclusion

This project has investigated the research question of how the presence of emojis affects the performance of machine learning models such as VADER, Random Forest, and RoBERTa in assigning sentiments to tweets. Our investigation revealed that the three different models deployed in this project were affected in varying ways by emojis. The rule-based model, Vader, benefited from the inclusion of emojis that are translated into text, whereas the inclusion of actual emojis appeared to harm performance. Random Forest as well as RoBERTa benefited from both translated emojis and actual emojis.

Further research is necessary to explore optimal methods for emoji processing and tokenization. However, the findings of this project reveal that the incorporation of emojis into NLP tasks can be very valuable. Consequently, it is highly recommended that researchers and practitioners prioritise the integration of emojis into NLP tasks as emojis offer valuable insights in sentiment analysis. The removal of emojis should only be considered as a last resort when no other viable option exists.

References

Barbieri, F., Kruszewski, G., Ronzano, F., & Saggion, H. (2016). How Cosmopolitan Are Emojis?: Exploring Emojis Usage and Meaning over Different Languages with Distributional Semantics. Proceedings of the 24th ACM International Conference on Multimedia, 531–535. https://doi.org/10.1145/2964284.2967278

Breiman, L. (2001). Random Forests. Machine Learning, 45(5–32). https://doi.org/10.1023/A:1010933404324

Dixon, S. (2021). Share of tweets containing emojis from July 2016 to July 2021 [Graph]. In Statista. Emojipedia. Retrieved May 26 2023. https://www.statista.com/statistics/1367443/share-of-tweets-containing-emojis/https://www.statista.com/statistics/1367443/share-of-tweets-containing-emojis/

Eisner, B., & Rocktäschel, T. (2016). Emoji2vec. GitHub. Retrieved May 05 2023 https://github.com/uclnlp/emoji2vec#readme

Eisner, B., Rocktäschel, T., Augenstein, I., Bošnjak, M., & Riedel, S. (2016). emoji2vec: Learning Emoji Representations from their Description (arXiv:1609.08359). arXiv. http://arxiv.org/abs/1609.08359

Erle, T. M., Schmid, K., Goslar, S. H., & Martin, J. D. (2022). Emojis as social information in digital communication. Emotion, 22(7), 1529–1543. https://doi.org/10.1037/emo0000992

Farías, D. I. H., Patti, V., & Rosso, P. (2016). Irony Detection in Twitter: The Role of Affective Content. ACM Transactions on Internet Technology, 16(3), 1–24. https://doi.org/10.1145/2930663

Gnana Priya, B. (2019). Emoji based sentiment analysis using KNN. International Journal of Scientific Research and Review, 7(2279-543X).

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. 'O'Reilly Media, Inc.'

Hutto, C., & Gilbert, E. (2014). VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. Proceedings of the International AAAI Conference on Web and Social Media, 8(1), 216–225. https://doi.org/10.1609/icwsm.v8i1.14550

Jurafsky, D., Martin, J.H., Kehler, A., Linden, K.V. and Ward, N. (2008). Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition. 3rd ed. Delhi: Pearson Education.

Kralj Novak, P., Smailović, J., Sluban, B., & Mozetič, I. (2015). Sentiment of Emojis. PLOS ONE, 10(12), e0144296. https://doi.org/10.1371/journal.pone.0144296

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach (arXiv:1907.11692). arXiv. http://arxiv.org/abs/1907.11692

Mathur, A., & Foody, G. M. (2008). Multiclass and Binary SVM Classification: Implications for Training and Classification Users. IEEE Geoscience and Remote Sensing Letters, 5(2),

241–245. https://doi.org/10.1109/LGRS.2008.915597

Mbaye, M. (2020). Up-to-date list of Slangs for Text Preprocessing. Kaggle. Retrieved May 26 2023. https://www.kaggle.com/code/nmaguette/up-to-date-list-of-slangs-for-text-preprocessing/notebook

McShane, L., Pancer, E., Poole, M., & Deng, Q. (2021). Emoji, Playfulness, and Brand Engagement on Twitter. Journal of Interactive Marketing, 53, 96–110. https://doi.org/10.1016/j.intmar.2020.06.002

Mei, Q. (2019). Decoding the New World Language: Analyzing the Popularity, Roles, and Utility of Emojis. Companion Proceedings of The 2019 World Wide Web Conference, 417–418. https://doi.org/10.1145/3308560.3316541

RoBERTa: An optimized method for pretraining self-supervised NLP systems. (2019). Facebook AI. Retrieved May 28 2023. https://ai.facebook.com/blog/roberta-an-optimized-method-for-pretraining-self-supervised-nlp-systems/

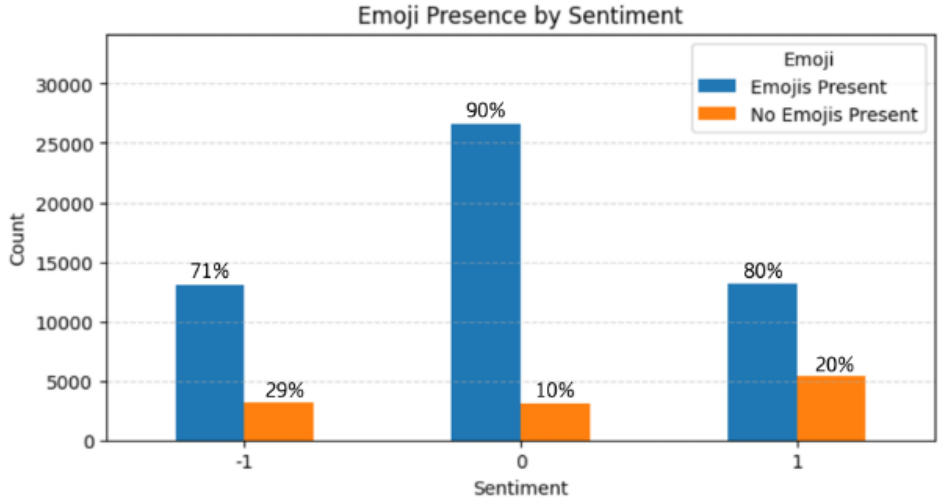
Thorn, J. (2021, December 16). Random Forest: Hyperparameters and how to fine-tune them. *Medium*. https://towardsdatascience.com/random-forest-hyperparameters-and-how-to-fine-tune-them-17aee785ee0d

Wang, L., Niu, J., & Yu, S. (2020). SentiDiff: Combining Textual Information and Sentiment Diffusion Patterns for Twitter Sentiment Analysis. IEEE Transactions on Knowledge and Data Engineering, 32(10), 2026–2039. https://doi.org/10.1109/TKDE.2019.2913641

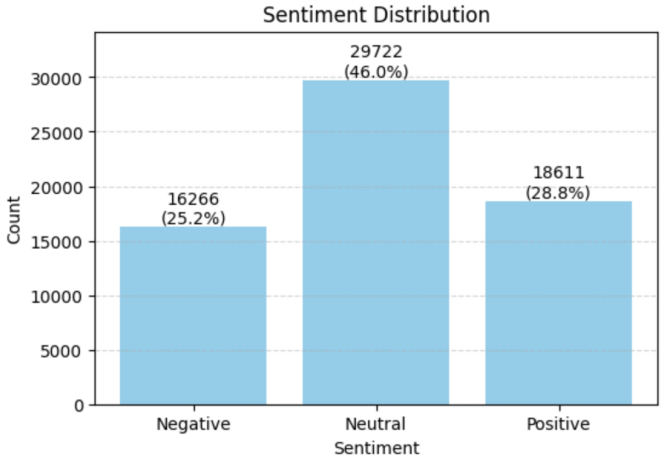
Appendix

Appendix A: Exploratory Data Analysis

Appendix A.1: Visualisation of the emoji presence by sentiment after main preprocessing (df)



Appendix A.2: Imbalance between the amount of twitter comments based on their sentiment



Appendix B: Results from Models

Appendix B.1: Results from VADER model

Appendix B.1.2: Results from VADER on Validation set

```
df1 (with emojis)
              precision    recall  f1-score   support

    Negative     0.5582    0.4675    0.5089      2419
     Neutral     0.6039    0.4774    0.5333      4455
    Positive     0.4430    0.6516    0.5275      2816

    accuracy                         0.5256      9690
   macro avg     0.5351    0.5322    0.5232      9690
weighted avg     0.5458    0.5256    0.5255      9690

df1_1 (with translated emojis)
              precision    recall  f1-score   support

    Negative     0.5765    0.4869    0.5279      2477
     Neutral     0.6399    0.4571    0.5333      4408
    Positive     0.4495    0.7130    0.5514      2805

    accuracy                         0.5388      9690
   macro avg     0.5553    0.5523    0.5375      9690
weighted avg     0.5686    0.5388    0.5372      9690

df2 (without emojis)
              precision    recall  f1-score   support

    Negative     0.5430    0.4701    0.5039      2391
     Neutral     0.6180    0.4795    0.5400      4561
    Positive     0.4308    0.6421    0.5156      2738

    accuracy                         0.5231      9690
   macro avg     0.5306    0.5306    0.5198      9690
weighted avg     0.5466    0.5231    0.5242      9690
```

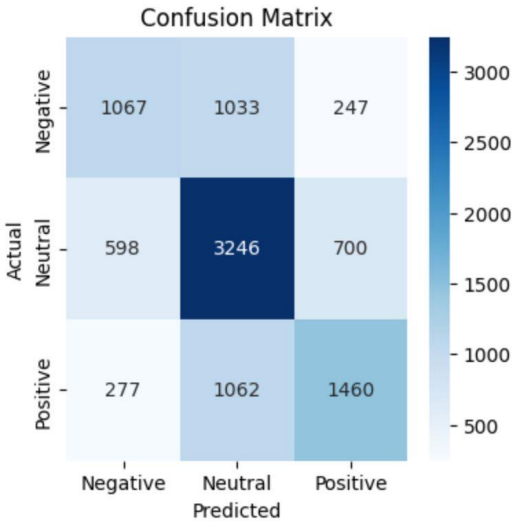
Appendix B.1.2: Results from VADER model on Test set

```
df1 (with emojis)
              precision    recall  f1-score   support

    Negative     0.5586    0.4740    0.5129      2483
     Neutral     0.6052    0.4763    0.5331      4438
    Positive     0.4411    0.6515    0.5260      2769

    accuracy                         0.5258      9690
   macro avg     0.5350    0.5340    0.5240      9690
weighted avg     0.5464    0.5258    0.5259      9690

df1_1 (with translated emojis)
              precision    recall  f1-score   support

    Negative     0.5761    0.4849    0.5266      2458
     Neutral     0.6438    0.4582    0.5354      4476
    Positive     0.4424    0.7119    0.5457      2756

    accuracy                         0.5372      9690
   macro avg     0.5541    0.5517    0.5359      9690
weighted avg     0.5693    0.5372    0.5361      9690

df2 (without emojis)
              precision    recall  f1-score   support

    Negative     0.5501    0.4609    0.5016      2441
     Neutral     0.6051    0.4768    0.5333      4505
    Positive     0.4305    0.6425    0.5156      2744

    accuracy                         0.5197      9690
   macro avg     0.5286    0.5267    0.5168      9690
weighted avg     0.5418    0.5197    0.5203      9690
```

Appendix B.2: Results from Random Forest model

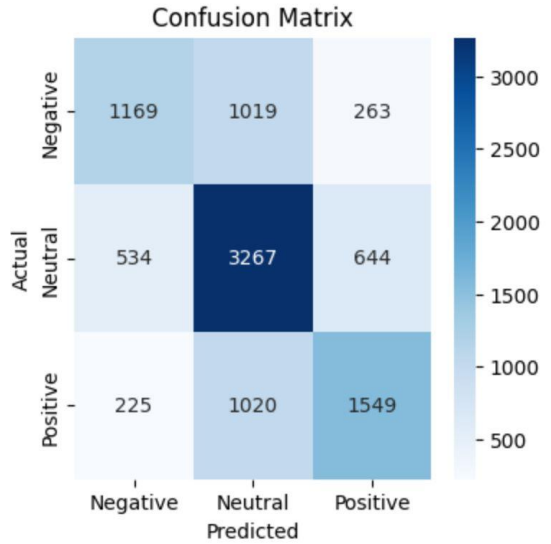
Appendix B.2.1: Results from Random Forest model on Validation set on DF2

	precision	recall	f1-score	support
Negative	0.5494	0.4546	0.4976	2347
Neutral	0.6078	0.7143	0.6568	4544
Positive	0.6066	0.5216	0.5609	2799
accuracy			0.5958	9690
macro avg	0.5879	0.5635	0.5717	9690
weighted avg	0.5933	0.5958	0.5905	9690



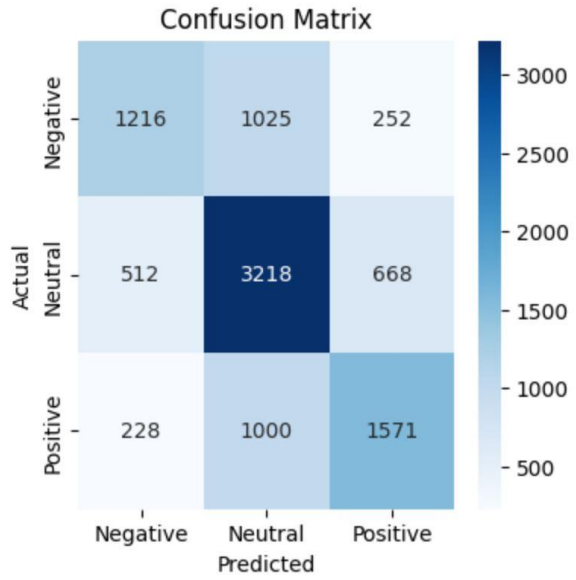
Appendix B.2.2: Results from Random Forest model on DF1_1 on Validation set

	precision	recall	f1-score	support
Negative	0.6063	0.4769	0.5339	2451
Neutral	0.6157	0.7350	0.6701	4445
Positive	0.6307	0.5544	0.5901	2794
accuracy			0.6176	9690
macro avg	0.6176	0.5888	0.5980	9690
weighted avg	0.6177	0.6176	0.6126	9690



Appendix B.2.3: Results from Random Forest model on DF1 on Validation set

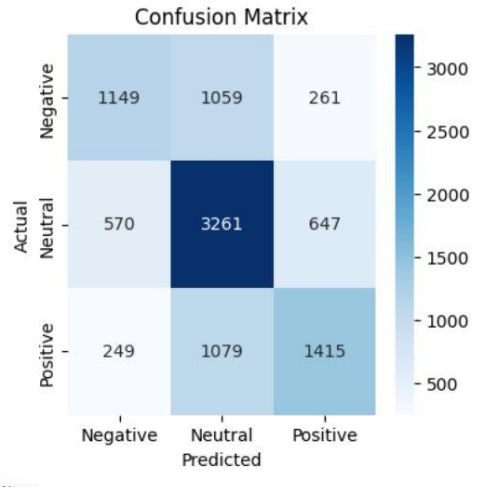
	precision	recall	f1-score	support
Negative	0.6217	0.4878	0.5466	2493
Neutral	0.6138	0.7317	0.6676	4398
Positive	0.6307	0.5613	0.5940	2799
accuracy			0.6197	9690
macro avg	0.6220	0.5936	0.6027	9690
weighted avg	0.6207	0.6197	0.6152	9690



Appendix B.2.4: Results from Random Forest model on DF2 on Test set

```
print(cr2)
print(plot_confusion_matrix(vrf_df2_test_Sentiment, final_predictions_df2))
```

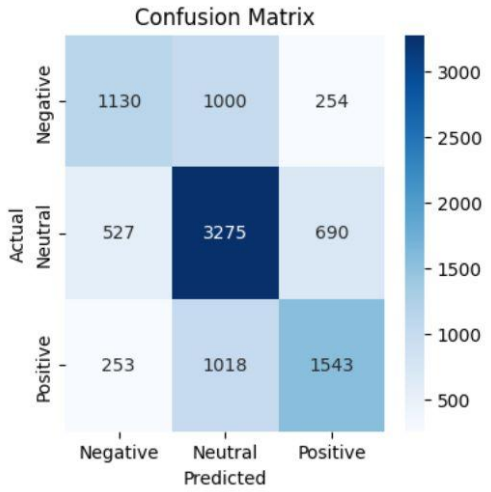
```
              precision    recall  f1-score   support

    Negative     0.5838    0.4654    0.5179      2469
     Neutral     0.6040    0.7282    0.6603      4478
    Positive     0.6091    0.5159    0.5586      2743

    accuracy                         0.6011      9690
   macro avg     0.5990    0.5698    0.5790      9690
weighted avg     0.6003    0.6011    0.5952      9690
```



Appendix B.2.5: Results from Random Forest model on DF1_1 on Test set

```
print(cr1_1)
print(plot_confusion_matrix(vrf_df1_1_test_Sentiment, final_predictions_df1_1))
```

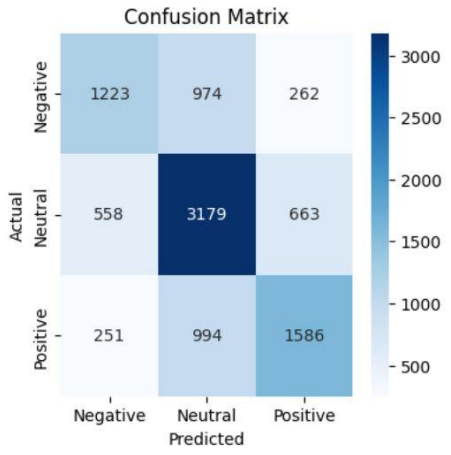
```
              precision    recall  f1-score   support

    Negative     0.5916    0.4740    0.5263      2384
     Neutral     0.6187    0.7291    0.6694      4492
    Positive     0.6204    0.5483    0.5822      2814

    accuracy                         0.6138      9690
   macro avg     0.6103    0.5838    0.5926      9690
weighted avg     0.6126    0.6138    0.6089      9690
```



Appendix B.2.6: Results from Random Forest model on DF1 on Test set

```
[47] print(cr_1)
     print(plot_confusion_matrix(vrf_df1_test_Sentiment, final_predictions_df1))
```

```
              precision    recall  f1-score   support

    Negative     0.6019    0.4974    0.5446      2459
     Neutral     0.6176    0.7225    0.6660      4400
    Positive     0.6316    0.5602    0.5938      2831

    accuracy                         0.6180      9690
   macro avg     0.6170    0.5934    0.6015      9690
weighted avg     0.6177    0.6180    0.6141      9690
```



None

Appendix B.3: Results from RoBERTa model

Appendix B.3.1: Results from RoBERTa on Validation set on DF1, DF1_1, and DF2

With emojis (df1)

	precision	recall	f1-score	support
Negative	0.6605	0.7017	0.6805	2387
Neutral	0.7444	0.6895	0.7159	4516
Positive	0.6769	0.7216	0.6985	2787
accuracy			0.7018	9690
macro avg	0.6939	0.7043	0.6983	9690
weighted avg	0.7043	0.7018	0.7022	9690

With translated emojis (df1_1)

	precision	recall	f1-score	support
Negative	0.6803	0.7138	0.6966	2421
Neutral	0.7453	0.6795	0.7109	4505
Positive	0.6697	0.7373	0.7019	2764
accuracy			0.7045	9690
macro avg	0.6985	0.7102	0.7031	9690
weighted avg	0.7075	0.7045	0.7048	9690

Without emojis (df2)

	precision	recall	f1-score	support
Negative	0.6681	0.7156	0.6911	2521
Neutral	0.7176	0.6722	0.6941	4411
Positive	0.6728	0.6972	0.6848	2758
accuracy			0.6906	9690
macro avg	0.6862	0.6950	0.6900	9690
weighted avg	0.6920	0.6906	0.6907	9690

Appendix B.3.2: Batch size tuning for RoBERTa

Batch Size	Train Accuracy	Val Accuracy	Train Time
2	0.752007	0.702374	1153.75
4	0.75382	0.708256	600.122
8	0.751476	0.708669	379.629
16	0.746279	0.708359	318.49
32	0.743581	0.711662	303.637
64	0.732435	0.702993	281.796
128	0.72443	0.703302	270.319
256	0.713019	0.698039	265.537

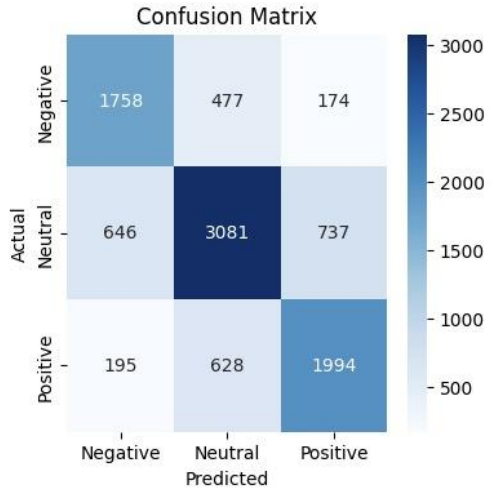
Appendix B.3.3: Learning rate tuning for RoBERTa

Learning Rate	Val Accuracy	Training Time
0.0006	0.455108	302.335
0.0002	0.455108	302.966
6e-05	0.7	302.687
2e-05	0.71032	303.529
6e-06	0.700103	302.643
2e-06	0.686378	303.415
6e-07	0.604954	302.178
2e-07	0.455108	304.499

Appendix B.3.4: Results from RoBERTa on Test set on DF2 after tuning

```
[47]  1 print(cl_2)
      2 plot_confusion_matrix(df2_test["Sentiment"], df2_predictions)
```

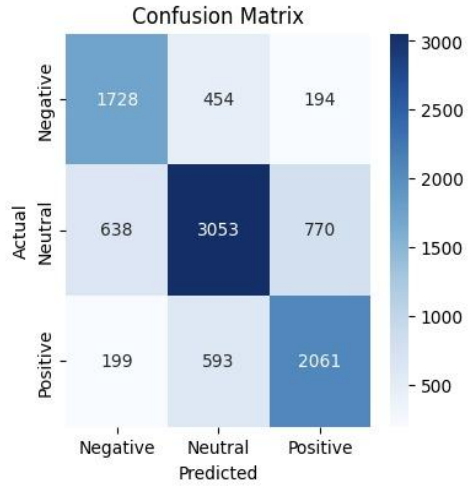
```
                precision    recall  f1-score   support

      Negative     0.6764    0.7298    0.7021      2409
       Neutral     0.7360    0.6902    0.7124      4464
      Positive     0.6864    0.7078    0.6970      2817

      accuracy                         0.7052      9690
     macro avg     0.6996    0.7093    0.7038      9690
  weighted avg     0.7068    0.7052    0.7053      9690
```



Appendix B.3.5: Results from RoBERTa on Test set on DF1_1 after tuning

```
1 print(cl_1_1)
2 plot_confusion_matrix(df1_1_test["Sentiment"], df1_1_predictions)
```

```
              precision    recall  f1-score   support

    Negative     0.6737    0.7273    0.6995      2376
     Neutral     0.7446    0.6844    0.7132      4461
    Positive     0.6813    0.7224    0.7013      2853

    accuracy                         0.7061      9690
   macro avg     0.6999    0.7113    0.7046      9690
weighted avg     0.7086    0.7061    0.7063      9690
```



Appendix B.3.6: Results from RoBERTa on Test set on DF1 after tuning

```
[46]  1 print(cl_1)
      2 plot_confusion_matrix(df1_test["Sentiment"], df1_predictions)
```

```
                precision    recall  f1-score   support

      Negative     0.6781    0.7249    0.7007      2421
       Neutral     0.7342    0.6954    0.7143      4353
      Positive     0.7063    0.7215    0.7138      2916

      accuracy                         0.7106      9690
     macro avg     0.7062    0.7139    0.7096      9690
  weighted avg     0.7118    0.7106    0.7107      9690
```