Tweet Spectrum: Unravelling the Sentiment Context of Twitter(X)

Abdul Rabbani Syed[1]
Abdul Mannan Mohammed[2]
Mohammed Muqeemuddin[3]
Nasir Ali Hussaini Syed[4]

Abstract

The rapid evolution of social media has required advanced techniques for sentiment analysis, particularly in understanding public opinion and emotions expressed on platforms like Twitter. This paper explores a comparative analysis of traditional machine learning models and large language models in the context of Twitter sentiment analysis. Our goal is to assess the accuracy and efficiency of these models in classifying tweets into positive, negative, or neutral sentiments that emphasizes on twitter hate profanity.

We utilized a dataset comprising a wide range of tweets that were collected and preprocessed for model training and evaluation. The traditional ML models employed in our study include algorithms such as Logistic Regression, Random Forest Classifier, Multinomial Naïve Bayes, and XGBoost Classifier. In the domain of LLMs, we investigated the performance of cutting-edge models like GPT-3 and versions of BERT like DistilBERT and RoBERTa, renowned for their deep learning capabilities.

Our methodology towards this research began with an initial exploratory data analysis to recognize patterns and sentiment distributions, followed by the implementation of each model. We evaluated model performance based on various metrics, including accuracy, precision, recall, and F1-score.

This study highlights the different strengths and limitations inherent to each model category. Traditional ML models, while they are less resource-intensive, faced challenges in processing the sarcastic and informal language of Twitter. In comparison to that, LLMs demonstrated a better ability to interpret contextual and linguistic subtleties but required significantly more computational power. Additionally, we developed a web application prototype to demonstrate the practical application of these models in a user-friendly interface, enabling users to conduct real-time sentiment analysis on Twitter texts.

Our findings revealed that while traditional ML models are beneficial for their simplicity and lower resource demands, LLMs excel in accuracy and contextual understanding. This comparative study contributes to the field by offering valuable insights into the selection of appropriate models for Twitter sentiment analysis, considering various factors such as accuracy, efficiency, and resource availability. Future research will aim to expand the dataset and explore hybrid models that synergize the strengths of both traditional Machine Learning techniques and Large Language Models.

Introduction

Sentiment analysis is the automated process of tagging data according to their sentiment, such as positive, negative and neutral. Sentiment analysis allows companies to analyse data at scale, detect insights and automate processes.

In the past, sentiment analysis used to be limited to researchers, machine learning engineers or data scientists with experience in natural language processing. However, the AI community has built awesome tools to democratize access to machine learning in recent years.

The growth of social media in past two decades especially Twitter, has made it a crucial platform for assessing public sentiment. In the fast-paced digital world, the sentiment behind a tweet can significantly impact an individual or a company's brand. It can either boost a brand by going viral with positive content or harm it with negative suggestions. This concern highlights the importance of accurately capturing sentiment in language, where decisions and reactions evolve in seconds. Understanding which specific words or phrases contribute to the overall sentiment of a tweet is crucial. This not only involves picking out parts of a sentence but analysing the context and intent behind it [6] [1].

Our motivation got us to this research that primarily focuses on categorizing various expressions of communication on Twitter, especially those pertaining to violent communication and hate speech. By identifying patterns in these expressions, we aim to develop systems of rules based on the analysis of each tweet. This effort aligns with the understanding gained from incidents like the Charlie Hebdo attack, where Twitter played a significant role in public discourse (Smith & Granville, 2015). The study employs machine learning approaches to analyse and predict sentiments from extracted Twitter tweets, categorizing them into positive, hateful/negative/abusive/violent, and neutral sentiments.

The objective of this research is multi-layered. Firstly, we seek to establish a classification of communication on Twitter, particularly focusing on violent and hate speech. This sorting is based on a qualitative analysis of a sample of tweets identified for their content related to violence and hate [4]. Secondly, the study aims to validate these categories using quantitative analysis. We also intend to objectify the identification process of different categories of communication by examining patterns and other variables that affect speech.

Addressing these objectives, we utilized a dataset of tweets sourced from the Kaggle data portal. This dataset allowed us to integrate our project into training four distinct machine learning categorization models. The study emphasizes feature engineering after the data analysis to accurately predict vulgarity and sentiment with a lower error rate. The model's performance was evaluated based on accuracy, sensitivity/recall, precision, and the F1 score for each class, identifying the best-performing model for predicting sentiments [5].

In practical application, this research extends to the development of a full-stack web application using Flask API which is a python framework. This application allows users to input text and receive the sentiment of the tweet that demonstrates the real-world applicability of our research. For instance, when a user inputs a seemingly contradictory statement like "My ridiculous dog is amazing," the Sentiment Analyzer discerns the positive intent, showcasing the uncorrupted understanding of the model [20].

This study is not just an academic exercise but a step towards a deeper understanding of sentiment analysis in the world of social media. It aims to contribute significantly to the field of NLP by providing a robust comparative analysis of traditional ML models and LLMs that emphasize their practicality and performance in real-world sentiment analysis on Twitter.

Related work

Microblogging platforms like Twitter have revolutionized the way information is shared and opinions are expressed. The intrinsic nature of these platforms, where users post real-time messages on a variety of topics, makes them a rich source for sentiment analysis. F. Miro-Llinares's 2016 study delves into this realm, exploring models for classifying tweets into positive, negative, and neutral sentiments. This research combines unigrams with features and tree kernels that significantly outperformed the unigram baseline, demonstrating the effectiveness of hybrid models in sentiment analysis [19].

The study introduced a novel tree representation of tweets, integrating various feature categories into a single and efficient model. This approach utilized PT kernels to calculate the similarity between tweet representations shows the work of Haussler who engineered Convolution Kernels for comparing abstract objects. Such methodologies underline the evolution of sentiment analysis techniques, moving beyond traditional feature vectors to more sophisticated models.

The emergence of social media platforms has given rise to two main research directions in sentiment analysis. Alzyout et al. (2021) have identified these as the development of new methods for analysis, such as sentiment label propagation on Twitter follower graphs, and the employment of social relations for user-level sentiment analysis. This shift towards leveraging the social network structure of Twitter for sentiment analysis represents a significant advancement in the field [7].

Concurrently, there's a focus on identifying new sets of features to enhance sentiment analysis models, as noted by Alec Go (2009). His research into microblogging features like hashtags, emoticons, and the presence of intensifiers such as all-caps and character repetitions, and their integration into Naïve Bayes (NB) classifiers using interpolation methods, has been pivotal. This points to the increasing complexity of sentiment analysis tasks, where the mere lexical content of tweets is insufficient to capture the true sentiment [8].

Preprocessing of raw tweets, as discussed by Kandpal, is essential due to the often-noisy nature of Twitter data. This involves standardizing language expressions and increasing the specificity of concepts identified in tweets. Such preprocessing is crucial for accurate sentiment classification and lays the groundwork for more sophisticated analyses [9].

The performance of classifiers like KNN, SVM, and NB in sentiment analysis has been a subject of study, as shown in [10]. Their research, which utilized RapidMiner software, revealed challenges in achieving high accuracy, precision, and recall, underscoring the need for more robust classifiers capable of handling the nuances of Twitter data.

Further complicating the landscape of Twitter sentiment analysis is the prevalence of messages of violence and hate speech, as highlighted in [11]. The study involved analyzing a subset of tweets specifically for violent and hate speech content, providing insights into categorizing such extreme sentiments. This area of research is critical, considering the potential for such messages to incite real-world harm.

The issue of cyberbullying on Twitter that was explored in [12] adds another layer of complexity to sentiment analysis. With Twitter's broad user base and public nature, it has become a platform where aggressive and harmful communication can be disseminated widely. The frequency of cyberbullying on Twitter when compared to other social media platforms shows the urgent need for effective sentiment analysis tools that can identify and mitigate such problems.

XGBoost (eXtreme Gradient Boosting) tands out for its efficiency and performance in classification tasks, including sentiment analysis. As an implementation of gradient boosted decision trees, XGBoost is known for its speed and accuracy, making it particularly effective for processing large and diverse datasets like those found on Twitter[4]. The underlying mechanisms of XGBoost, as explained in Friedman's (2001) work on Greedy Function Approximation and Natekin and Knoll's (2013) tutorial on gradient boosting machines, highlight its proficiency in predictive modeling, a key aspect in sentiment analysis [13][14][15].

The introduction of LLMs marked a paradigm shift in Natural Language Processing (NLP). These models, known for their vast scale and deep learning architectures, excel in understanding and generating human-like text. The GPT series of models, as discussed by Radford et al. (2019) in their study on unsupervised multitask learners, showcases the potential of LLMs in understanding context and nuances in language. This ability is crucial in sentiment analysis, especially when dealing with the varied and often informal language used on social media platforms [16].

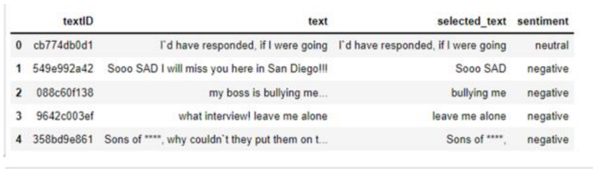
BERT (Bidirectional Encoder Representations from Transformers), developed by Devlin (2018), represents a significant advancement in NLP. Unlike previous models, BERT processes text in both directions, providing a deeper understanding of context. This bidirectional nature is pivotal in sentiment analysis, as it allows for a more nuanced interpretation of text [17]. Sun et al. (2019) in their exploration of BERT's capabilities demonstrate how it surpasses previous models in tasks requiring a deep understanding of context, which is often the case with social media texts [18].

Data

Data Collection:

We obtained our Twitter data from a Kaggle data source to train a machine learning model to predict sentiment in the data gathering process. The data came from a Kaggle competition that was held online. Sentiment Analysis: Emotion in Text Tweets with Existing Sentiment Labels is the name of the dataset. There are 27481 tweets row occurrences in all. The sentiment label is divided into three categories: positive, negative, and neutral. Our objective is to use the retrieved tweets to predict the underlying prospective sentiment.

At this point, the loaded data frame looks like this:



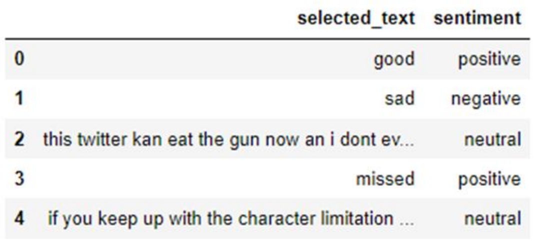
	textID	text	selected_text	sentiment
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative
2	088c60f138	my boss is bullying me...	bullying me	negative
3	9642c003ef	what interview! leave me alone	leave me alone	negative
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative

Figure 1: Data Preview

Data Pre-processing:

To prepare our dataset for machine learning model training, we engaged in several data processing steps, including formatting, examining, and cleaning the tweets data. This process involved removing special characters, extraneous columns, punctuation, and stop words. We utilized Pandas, a Python library for data manipulation, to load and process the data. Columns irrelevant to our machine learning algorithm training were excluded from the dataset. Special characters were also removed by applying a text cleaning function across the entire dataset. This ensured that the data was free from erroneous and unwanted values, aligning it with our objectives.

Following the cleaning process, the data took on a specific format, as depicted in the accompanying figure.



	selected_text	sentiment
0	good	positive
1	sad	negative
2	this twitter kan eat the gun now an i dont ev...	neutral
3	missed	positive
4	if you keep up with the character limitation ...	neutral

Figure 2: Formatted Dataset

Data Analysis:

During the data analysis stage, we expolred into the dataset, seeing patterns and identifying trends in features through statistical visualization. For this purpose, we employed Python packages, specifically matplotlib and seaborn libraries, to create meaningful visual representations of the data. Among the various visualizations we produced are:

- Distribution of Sentiment Classes
- Positive Tweet Text length distribution
- Negative Tweet Text length distribution
- Neutral Tweet Text length distribution

- Pie Chart of Sentiment vs Tweet Count
- Most common words for positive sentiment group
- Most common words for negative sentiment group
- Most common words for neutral sentiment group
- WordCloud generation of positive sentiment group
- WordCloud generation of negative sentiment group
- WordCloud generation of neutral sentiment group

Distribution of Sentiment Classes:

We showcased the spread of different sentiment classes across the entire dataset, illustrating the distribution of sentiments. This involved categorizing and counting the number of positive, negative, and neutral sentiments.

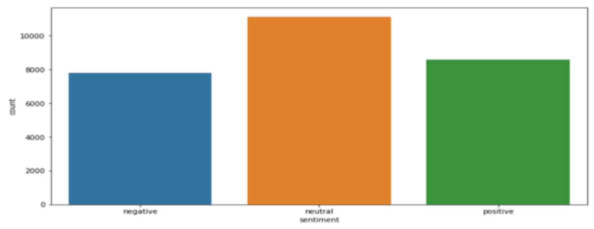


Figure 3: Distribution of Sentiment Classes

Positive Tweet Text length distribution:

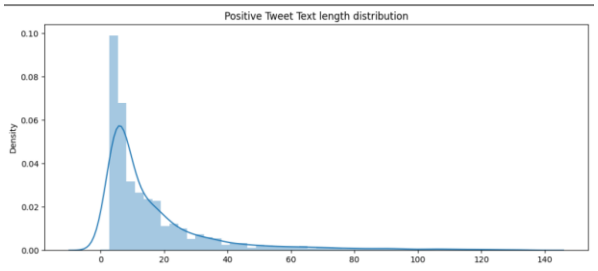


Figure 4: Positive Tweet Text Length Distribution

Negative Tweet Text length distribution:

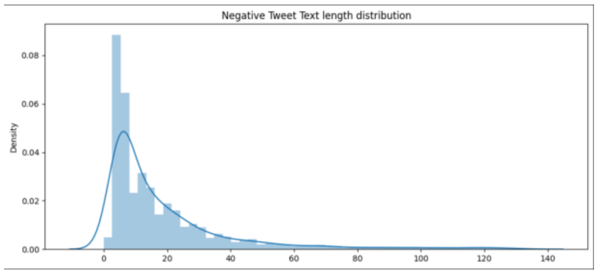


Figure 5: Negative Tweet Text Length Distribution

Neutral Tweet Text length distribution:

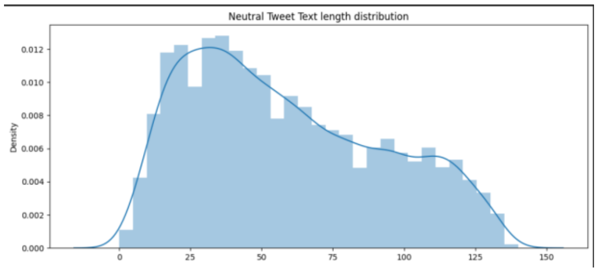


Figure 6: Neutral Tweet Text Length Distribution

Pie Chart of Sentiment vs Tweet Count:

This graph shows the proportion of each sentiment in the whole data for all three classes (positive, negative, neutral).

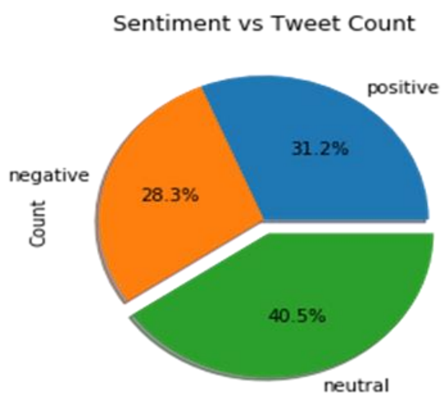


Figure 7: Sentiment vs Tweet Count

Most common words for positive sentiment group:

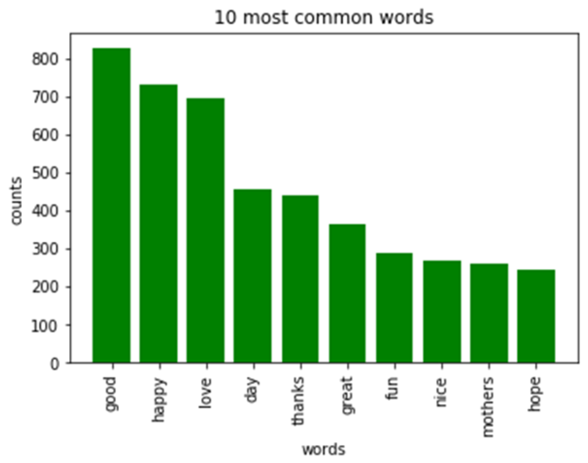


Figure 8: 10 most common words (positive)

Most common words for negative sentiment group:

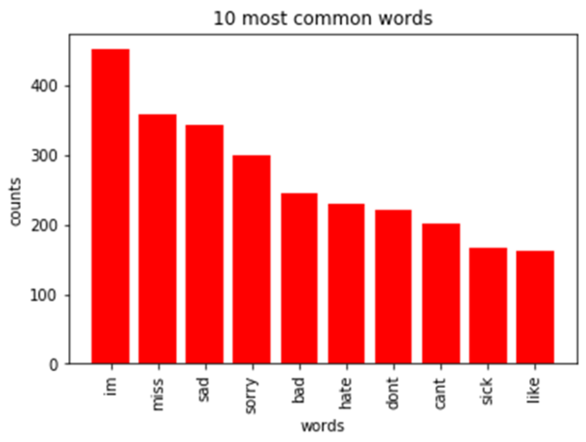


Figure 9: 10 most common words (Negative)

Most common words for neutral sentiment group:

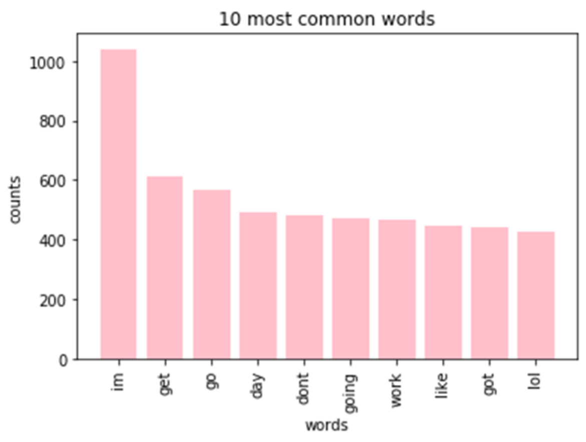


Figure 10: 10 most common words (neutral)

Word Cloud generation of positive sentiment group:

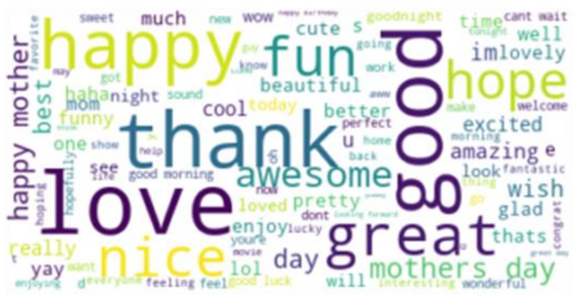


Figure 11: Positive WordCloud

Word Cloud generation of negative sentiment group:

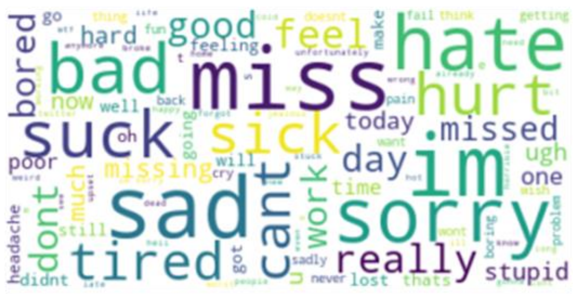


Figure 12: Negative Word Cloud

Word Cloud generation of neutral sentiment group:

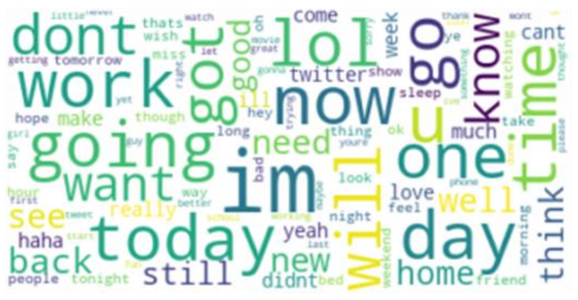


Figure 13: Neutral WordCloud

Now that we've made sense of the information and displayed the key features and data distribution, it's time to train the machine learning models. However, before we can use ml modeling, we must divide the dataset into two parts: training and testing.

Data Train/Test Split:

During the data split procedure, we decided which parts of the data should go into the training dataset and which should go into the test dataset. We randomly divided our dataset into 70/30 percent. This indicates that 70% of the dataset is chosen for training the machine learning model, while 30% of the dataset is utilized for testing or unknown data to the model. Testing data will be used to train the model once it has been constructed, in order to evaluate the model's performance on unseen data.

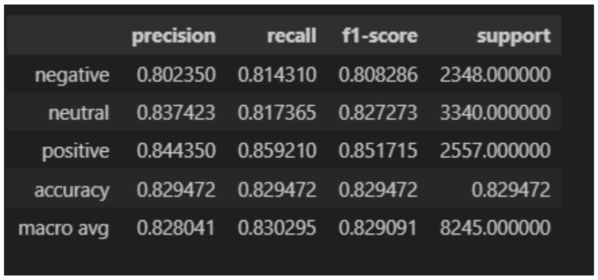
Methodologies

Logistic Regression

Logistic Regression is a predictive classification method that operates by estimating the probability of a dependent or target variable based on independent variables [18]. In our Twitter Sentiment Analysis project, we've utilized this model to predict the sentiment of tweets by designating the extracted features from the tweets as independent variables and the sentiment labels as the dependent or target variable. Initially, the model was trained on a portion of our dataset also called the training data, where it learns to associate specific features with corresponding sentiment labels. This training involved adjusting the model's parameters to minimize prediction errors, effectively making it adept at recognizing patterns indicative of different sentiments.

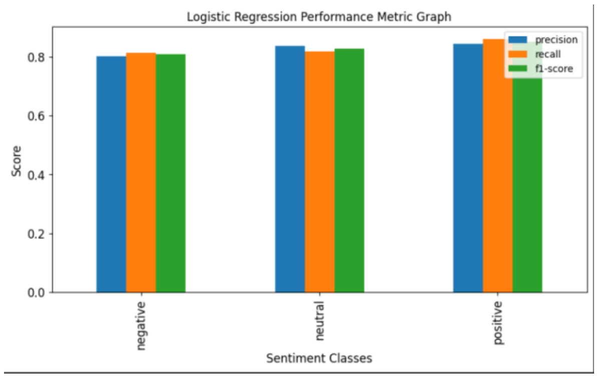
After the training phase, the performance of the Logistic Regression model was evaluated using a separate set of data, known as the test data. This evaluation was crucial to assess how well the model generalizes to new, unseen data. To provide a comprehensive understanding of the model's effectiveness, we generated a classification report for both the training and testing datasets. These reports include key metrics such as accuracy, precision, recall, and the F1-score, offering a detailed view of the model's performance across different sentiment classes. Additionally, we created performance metric graphs for both the training and testing datasets.

Classification report on testing data:



	precision	recall	f1-score	support
negative	0.802350	0.814310	0.808286	2348.000000
neutral	0.837423	0.817365	0.827273	3340.000000
positive	0.844350	0.859210	0.851715	2557.000000
accuracy	0.829472	0.829472	0.829472	0.829472
macro avg	0.828041	0.830295	0.829091	8245.000000

Performance metric graph on testing data:



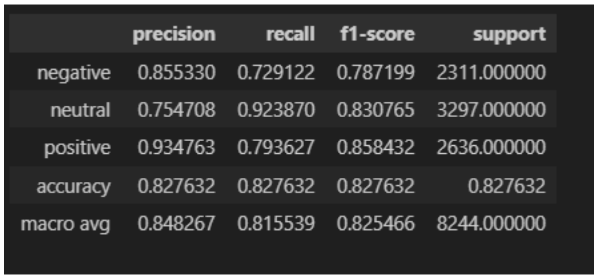
Random Forest

In our Twitter Sentiment Analysis project, the Random Forest algorithm is employed which is a machine learning technique for classifying tweets into different sentiment categories. Random Forest operates by constructing multiple decision trees during the training phase and outputting the class that is the mode of the classes predicted by individual trees. This ensemble approach enhances the predictive accuracy and controls over-fitting which is a common issue in decision tree models. [21]

In the context of our project, features were extracted from the tweets such as word frequencies and text properties. They serve as input variables for the Random Forest model. The sentiment labels of the tweets are the target variables. The model's training involves using these features to build a group of decision trees. Each tree in the forest makes an independent prediction, and the final sentiment classification is decided based on the majority vote from all trees. This method effectively captures the suggestion and complexity of natural language, making it particularly suitable for sentiment analysis.

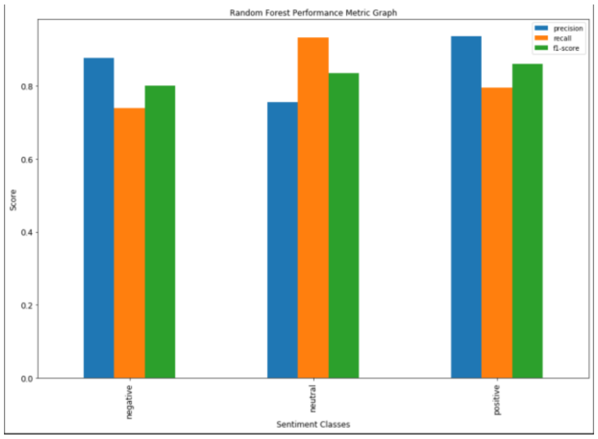
After training, the Random Forest model's performance is rigorously evaluated using the testing data, which consists of tweets not seen by the model during training. This evaluation is crucial to gauge the model's ability to generalize its predictions to new data. We utilize various performance metrics such as accuracy, precision, recall, and the F1-score, and present them in a comprehensive classification report for both the training and testing datasets. These metrics provide a multi-dimensional view of the model's effectiveness, highlighting its strengths and areas for improvement.

Classification report on testing data:



	precision	recall	f1-score	support
negative	0.855330	0.729122	0.787199	2311.000000
neutral	0.754708	0.923870	0.830765	3297.000000
positive	0.934763	0.793627	0.858432	2636.000000
accuracy	0.827632	0.827632	0.827632	0.827632
macro avg	0.848267	0.815539	0.825466	8244.000000

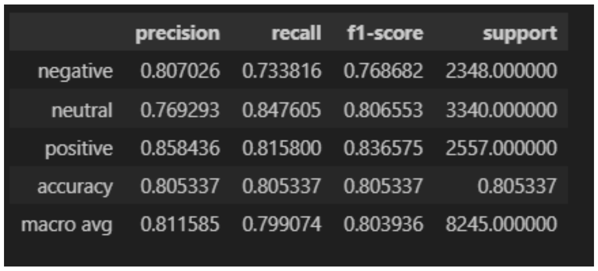
Performance metric graph on testing data:



MultiNomial Naïve Bayes

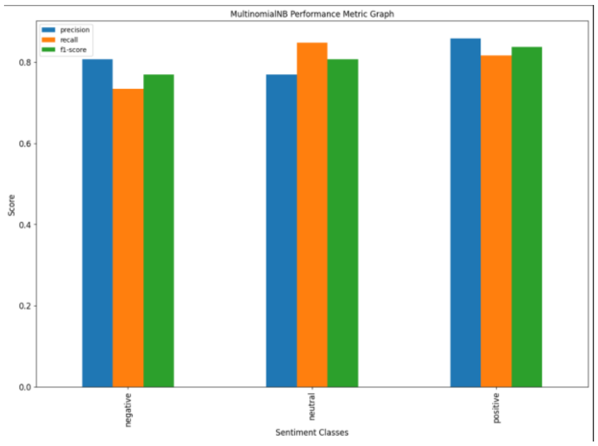
The Multinomial Naive Bayes algorithm is based on Bayes Theorem. It plays a crucial role in classifying tweets into different sentiment categories. This probabilistic learning method is particularly suited for Natural Language Processing (NLP) and is better at handling the high-dimensional text data in our project. We utilized this algorithm by first training it on a dataset where the features are the frequency counts of words in tweets, and the target variable is the tweet's sentiment. After training, the model predicts the sentiment of a tweet based on the probabilities it has learned and it chooses the class with the highest likelihood. We evaluated the model's performance on both training and testing datasets by creating detailed classification reports that include key metrics like accuracy, precision, recall, and the F1-score. We generated performance metric graphs for these datasets, providing a visual representation of the model's ability to accurately classify sentiments.

Classification report on testing data:



	precision	recall	f1-score	support
negative	0.807026	0.733816	0.768682	2348.000000
neutral	0.769293	0.847605	0.806553	3340.000000
positive	0.858436	0.815800	0.836575	2557.000000
accuracy	0.805337	0.805337	0.805337	0.805337
macro avg	0.811585	0.799074	0.803936	8245.000000

Performance metric graph on testing data:



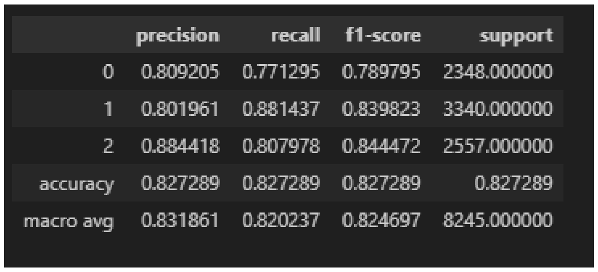
XGboost

The XGBoost algorithm plays a significant role in accurately classifying the sentiments of tweets. XGBoost stands for Extreme Gradient Boosting. It is an advanced implementation of gradient boosting that is known for its speed and performance. It works by building an ensemble of decision trees in a sequential manner, where each new tree attempts to correct the errors made by the previous ones [15]. This approach makes XGBoost particularly effective in handling complex and nuanced datasets like the one used in our project.

In the implementation of XGBoost in our project, we first transformed our text data into a numerical format using vectorization techniques, which is a crucial step for any machine learning model to process text data. The transformed data comprising the features extracted from tweets, were fed into the XGBoost classifier. The model was trained on a portion of our dataset (70%), known as the training data, where it learned to identify patterns and associations between the features and the sentiment labels.

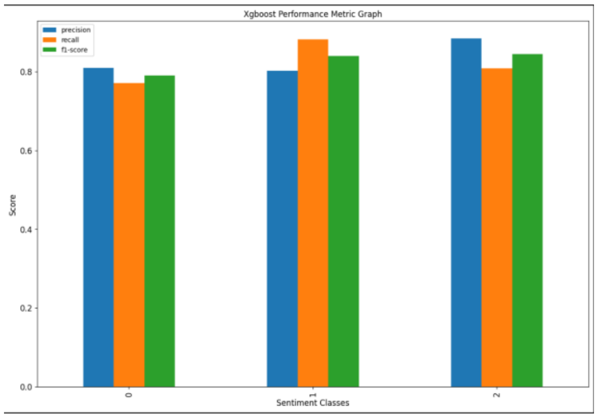
After the training phase, the model's performance was evaluated using the testing data. This evaluation is essential to assess how well the model can generalize its predictions to new and unseen data. We utilized various performance metrics such as accuracy, precision, recall, and the F1-score to evaluate the model. Additionally, we generated a classification report and performance metric graphs for both the training and testing datasets. These tools provided valuable insights into the model's accuracy and its ability to correctly classify the sentiments of tweets.

Classification report on testing data:



	precision	recall	f1-score	support
0	0.809205	0.771295	0.789795	2348.000000
1	0.801961	0.881437	0.839823	3340.000000
2	0.884418	0.807978	0.844472	2557.000000
accuracy	0.827289	0.827289	0.827289	0.827289
macro avg	0.831861	0.820237	0.824697	8245.000000

Performance metric graph on testing data:



BERT (Bidirectional Encoder Representations from Transformers)

BERT, short for Bidirectional Encoder Representations from Transformers, is an extraordinary milestone in the landscape of Natural Language Processing (NLP). Developed by Google AI Language researchers in 2018, BERT serves as a ground-breaking solution to the age-old challenge of enabling computers to comprehend human language.

Language, a labyrinth of context, historically confounded machines, despite their prowess in collecting, storing, and displaying text. NLP emerged as the beacon in this puzzle, linguistics, statistics, and Machine Learning to impart computers with the ability not just to read but to interpret and derive meaning from text and speech.

Traditionally, NLP relied on a patchwork of specific models, each tailored to tackle singular linguistic tasks. Then, BERT stepped in, reshaping this paradigm. It didn't just solve tasks; it excelled at them, outperforming prior models and earning the mantle of a versatile NLP powerhouse. What makes BERT truly remarkable is its adaptability across a spectrum of language tasks, revolutionizing how we interact with language on a daily basis. For instance, it effortlessly dissects reviews, unveiling the sentiments embedded within—the essence of sentiment analysis.

As we studied numerous papers, we perceive BERT as the Swiss army knife in the NLP arsenal. Its architecture, nurtured on copious amounts of data, navigates linguistic landscapes with finesse, redefining the field and endowing it with unprecedented capabilities. BERT's open-source framework fosters collaboration within the AI community, fostering innovation and the evolution of newer, more refined models. This collaborative ethos propels NLP towards uncharted territories, where language and machines converge seamlessly. In the intricate tapestry of NLP, BERT emerged as a pivotal thread, seamlessly weaving together the intricacies of language and computation. Its impact transcended industries, reshaping how we perceive, engage with, and harness the power of language in the digital era—a testament to the potential of NLP and the transformative capabilities of BERT.

How does BERT Work?

BERT, functions as a remarkable language model owing to its foundation on a vast dataset comprising approximately 3.3 billion words, predominantly sourced from Wikipedia and Google's Books Corpus. This expansive corpus forms the bedrock of BERT's comprehensive comprehension, not solely of English but also of broader contextual knowledge.

The heart of BERT's functionality lies in its training process, made feasible by the innovative Transformer architecture and accelerated by Google's specialized Tensor

Processing Units (TPUs). Training BERT entailed the utilization of 64 TPUs over a span of four days, enabling the model to ingest and distil profound insights from the immense textual dataset. A pivotal technique instrumental in BERT's training is the Masked Language Model (MLM). This approach enforces bidirectional learning by concealing a word within a sentence and compelling BERT to predict the masked word using contextual cues from neighbouring words. The bidirectional prediction mirrors the innate human ability to deduce missing words, leveraging context and pre-existing knowledge. BERT's proficiency in filling these masked segments during training amplifies its understanding of language structures and semantic nuances.

The crux of BERT's accuracy stems from its bidirectional methodology, wherein 15% of tokenized words are masked during training, prompting the model to predict these concealed elements. Consequently, BERT garners a profound understanding of the English language and its intricate usage patterns. At the core of BERT's functionality lies the Transformer architecture, a ground-breaking design enabling highly efficient parallelization during training. Transformers leverage attention mechanisms, initially proposed in the seminal 2017 paper "Attention Is All You Need," to discern complex relationships between words. This breakthrough in NLP models has rapidly propelled Transformers to the forefront of various domains, including natural language processing, speech recognition, and computer vision. They have become indispensable in contemporary deep learning endeavours. The amalgamation of Transformers and the MLM approach is pivotal to BERT's success. The Transformer architecture facilitates the parallelization of machine learning training, enabling BERT to process vast quantities of data efficiently. Meanwhile, the MLM technique compels the model to learn bidirectionally, strengthening its contextual comprehension and linguistic capabilities.

BERT's impact is far-reaching, its abilities extending beyond conventional language tasks. It serves as a testament to the fusion of sophisticated architecture and innovative training techniques. Its proficiency in tasks like sentiment analysis, named entity recognition, and understanding the nuances in text contributes significantly to its widespread usage across diverse applications and industries. In essence, BERT's effectiveness lies in its ability to absorb, process, and comprehend extensive textual data through the intricate interplay of the Transformer architecture, attention mechanisms, and the ingenious MLM approach. This amalgamation forms the bedrock of BERT's prowess, positioning it at the forefront of innovative advancements in natural language processing and machine learning.

Unlike other large learning models like GPT-3, BERT's source code is publicly accessible (view BERT's code on Github) allowing BERT to be more widely used all around the world. It’s important to note that thousands of open-source and free, pre-trained BERT models are currently available for specific use cases if you don’t want to fine-tune BERT.

BERT models pre-trained for specific tasks:

- Twitter sentiment analysis

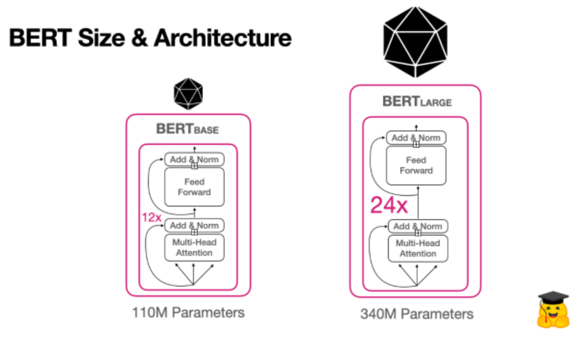


Figure 14: Bert Size and Architecture(Base and Large with respect to parameters)

DistilBERT:

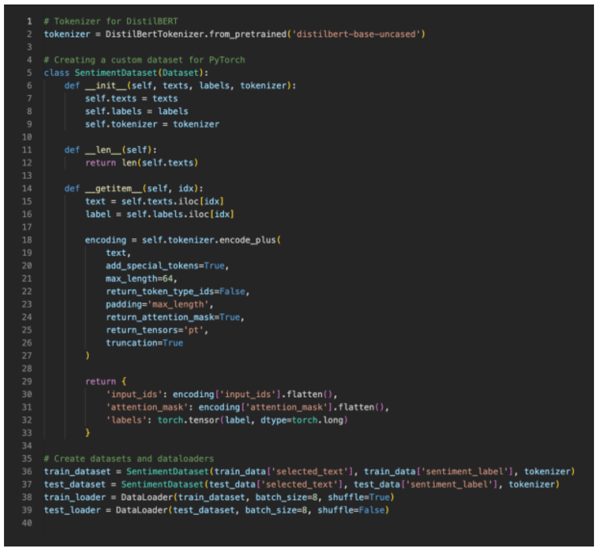
The DistilBERT model was proposed in the blog post Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT, and the paper DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. DistilBERT is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than bert-base-uncased, runs 60% faster while preserving over 95% of BERT’s performances.

The compact, faster, and lighter model significantly reduces pre-training costs, showcasing remarkable potential for on-device computations. In a proof-of-concept experiment and an on-device comparative study, we demonstrate the capabilities of DistilBERT, highlighting its feasibility and efficiency for deploying NLP models within constrained computational environments. This innovative approach not only addresses the challenges of deploying large-scale models but also showcases the practicality and efficacy of DistilBERT in real-world applications requiring on-edge or limited computational resources and better accuracy than our traditional ML models.

DistilBERT simplifies input by eliminating the need for token_type_ids, allowing seamless segmentation using the separation token (tokenizer.sep_token or [SEP]). Unlike BERT, DistilBERT doesn’t support position_ids input selection, a feature that could be incorporated upon request. Functioning as a smaller version of BERT, DistilBERT undergoes training via distillation from the larger BERT model. This training methodology ensures that DistilBERT predicts probabilities akin to its larger counterpart. Its objective includes matching the teacher model's probabilities, accurate prediction of masked tokens (sans next-sentence objective), and optimizing the cosine similarity between the hidden states of the student and teacher models. This distilled approach enables a more compact yet proficient model, retaining the essence of BERT's capabilities in a smaller framework.

Our code and algorithm:

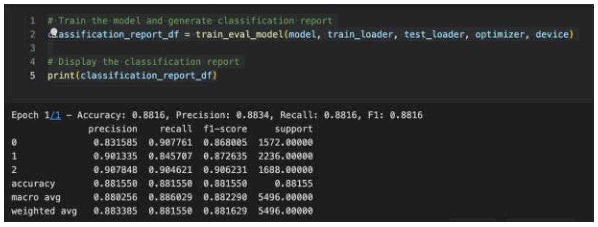
The first step in our research study conducted was to initialise a tokenizer in our case distilbertbase-uncased base which runs 60% faster than BERT. Defining the dataset method called SentimentDataset which functions in encoding the input dataset. Following by creating custom datasets and data loaders which are initialised for our model to get trained upon.



```python
# Tokenizer for DistilBERT
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased')

# Creating a custom dataset for PyTorch
class SentimentDataset(Dataset):
    def __init__(self, texts, labels, tokenizer):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = self.texts.iloc[idx]
        label = self.labels.iloc[idx]

        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=64,
            return_token_type_ids=False,
            padding='max_length',
            return_attention_mask=True,
            return_tensors='pt',
            truncation=True
        )

        return {
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'labels': torch.tensor(label, dtype=torch.long)
        }

# Create datasets and dataloaders
train_dataset = SentimentDataset(train_data['selected_text'], train_data['sentiment_label'], tokenizer)
test_dataset = SentimentDataset(test_data['selected_text'], test_data['sentiment_label'], tokenizer)
train_loader = DataLoader(train_dataset, batch_size=8, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=8, shuffle=False)
```

Figure 15: Initialization of tokenizer and data split

After our model trained on the training data on the number of epochs, we then tested the model on test data and received an outstanding 88.1 % accuracy as depicted in the below figure, the 0 – Negative, 1 -Neutral and 2 – Positive sentiments in the dataset.



```python
# Train the model and generate classification report
classification_report_df = train_eval_model(model, train_loader, test_loader, optimizer, device)

# Display the classification report
print(classification_report_df)
```

```
Epoch 1/1 - Accuracy: 0.8816, Precision: 0.8834, Recall: 0.8816, F1: 0.8816
              precision    recall  f1-score     support
0              0.831585  0.907761  0.868005  1572.00000
1              0.901335  0.845707  0.872635  2236.00000
2              0.907848  0.904621  0.906231  1688.00000
accuracy       0.881550  0.881550  0.881550     0.88155
macro avg      0.880256  0.886029  0.882290  5496.00000
weighted avg   0.883385  0.881550  0.881629  5496.00000
```

Figure 16: Classification report of DistilBERT model

RoBERTa

RoBERTa stands as a transformers model pretrained autonomously on an extensive English text corpus, devoid of human annotations. Its self-supervised learning method utilizes raw texts, tapping into publicly available data for training without human intervention. Specifically, RoBERTa adopts the Masked Language Modelling (MLM) objective. It randomly masks 15% of words in a sentence, tasking the model with predicting these masked words by processing the entire masked sentence. Unlike sequential RNNs or autoregressive models like GPT, which process words linearly or mask future tokens, RoBERTa's bidirectional approach allows it to comprehend sentences in both directions. This bidirectional understanding aids in learning a robust inner representation of English. This learned language representation serves as a foundation to extract valuable features for downstream tasks. By leveraging the features derived from the RoBERTa model, one can train conventional classifiers on labelled sentence datasets, enhancing performance in various applications. RoBERTa's bidirectional mastery of language enables it to generate rich, context-aware representations, fostering superior performance in diverse natural language understanding tasks [2].

It is based on Google's BERT model released in 2018. It builds on BERT and modifies key hyperparameters, removing the next-sentence pretraining objective and training with much larger mini-batches and learning rates.[3] Our study extends beyond BERT, encompassing RoBERTa in a comprehensive evaluation of language model pretraining. Beyond scrutinizing BERT's undertraining and hyperparameter effects, we scrutinize RoBERTa's performance under similar circumstances. Through meticulous exploration of diverse hyperparameters and dataset sizes, we surpass the published benchmarks, exhibiting RoBERTa's exceptional capabilities. Our findings showcase RoBERTa's prowess, demonstrating its ability to rival or exceed the performance of subsequent models, including DistilBERT and traditional ML models we dealt previously. Notably, our best RoBERTa model achieves state-of-the-art results on benchmarks such as accuracy, precision and F1 score mirroring the success observed with DistilBERT. This comprehensive analysis not only accentuates the overlooked nuances in design choices but also underscores RoBERTa's potential for significant advancements in language model. By working our project and code, we provide a foundation for future research, encouraging further exploration and refinement in the domain of language models.

RoBERTa shares the BERT architecture but employs a byte-level Byte-Pair Encoding (BPE) tokenizer akin to GPT-2, employing a distinct pretraining strategy. Unlike BERT, RoBERTa doesn’t necessitate token_type_ids for segment indication. Segments are distinguished by the separation token, tokenizer.sep_token (or </s>), streamlining segmentation without explicit segment labelling.

RoBERTa enhances BERT's pretraining with innovative techniques:

- Dynamic masking, adapting token masking at each epoch rather than a one-time event, optimizing learning diversity.
- Sequentially organizing sentences across multiple documents, maximizing context span within the 512-token limit.
- Training with larger batches for improved efficiency.

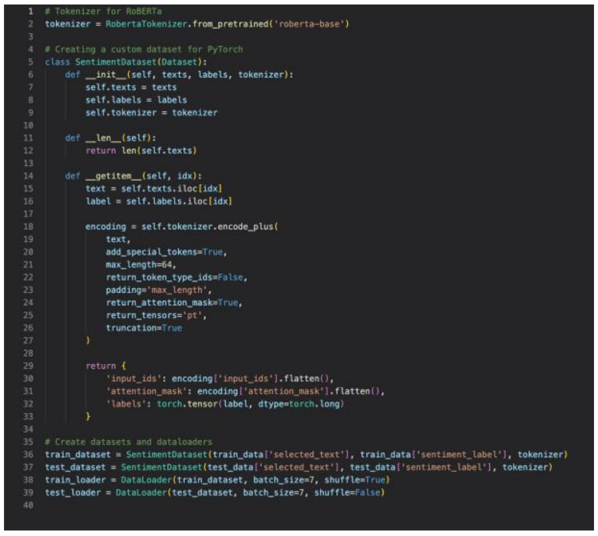
These advancements underscore RoBERTa's evolution, offering superior pretraining methodologies and better accuracy when we test on our datasets after training the model on our environment.

Parameters

- **config** (RobertaConfig) — Model configuration class with all the parameters of the model. Initializing with a config file does not load the weights associated with the model, only the configuration. Check out the from_pretrained() method to load the model weights.
- The RoBERTa Model, a bare transformer, outputs raw hidden-states

devoid of specific head structures. Built upon the PreTrainedModel class, it inherits generic methods for model operations such as downloading, saving, embedding resizing, and head pruning, detailed in the superclass documentation.

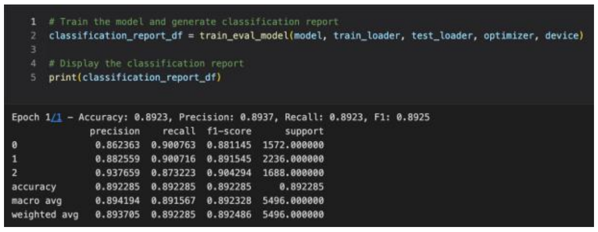
- As a PyTorch torch.nn.Module subclass, it operates like a standard PyTorch Module, following the PyTorch documentation for usage and functionality guidance. This versatile model can function as both an encoder (utilizing self-attention exclusively) and a decoder. In decoder mode, it integrates cross-attention layers between self-attention ones, aligning with the architecture introduced in "Attention is all you need" by Ashish Vaswani and others.



```python
# Tokenizer for RoBERTa
tokenizer = RobertaTokenizer.from_pretrained('roberta-base')

# Creating a custom dataset for PyTorch
class SentimentDataset(Dataset):
    def __init__(self, texts, labels, tokenizer):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer

    def __len__(self):
        return len(self.texts)

    def __getitem__(self, idx):
        text = self.texts.iloc[idx]
        label = self.labels.iloc[idx]

        encoding = self.tokenizer.encode_plus(
            text,
            add_special_tokens=True,
            max_length=64,
            return_token_type_ids=False,
            padding='max_length',
            return_attention_mask=True,
            return_tensors='pt',
            truncation=True
        )

        return {
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'labels': torch.tensor(label, dtype=torch.long)
        }

# Create datasets and dataloaders
train_dataset = SentimentDataset(train_data['selected_text'], train_data['sentiment_label'], tokenizer)
test_dataset = SentimentDataset(test_data['selected_text'], test_data['sentiment_label'], tokenizer)
train_loader = DataLoader(train_dataset, batch_size=7, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=7, shuffle=False)
```

Figure 18: RoBERTa tokenizer

The above figure indicates the RoBERTa tokenizer which we used in our study to showcase our model trained and tested on our datasets and how well it performs over DistilBERT and traditional ML models in terms of accuracy. The classification report below shows clearly that the accuracy we achieved after testing our model is 89.2% which is more than DistilBERT model.



```python
# Train the model and generate classification report
classification_report_df = train_eval_model(model, train_loader, test_loader, optimizer, device)

# Display the classification report
print(classification_report_df)
```

```
Epoch 1/1 - Accuracy: 0.8923, Precision: 0.8937, Recall: 0.8923, F1: 0.8925
              precision    recall  f1-score      support
0              0.862363  0.900763  0.881145  1572.000000
1              0.882559  0.900716  0.891545  2236.000000
2              0.937659  0.873223  0.904294  1688.000000
accuracy       0.892285  0.892285  0.892285     0.892285
macro avg      0.894194  0.891567  0.892328  5496.000000
weighted avg   0.893705  0.892285  0.892486  5496.000000
```

Figure 19: Classification report of RoBERTa

GPT-3 (DaVinci Version):

The GPT-3 model developed by Open AI is a large language model designed for text processing and generation. Trained on an extensive dataset of human-generated text, GPT-3 can analyze prompts (input text) and employ statistical techniques to predict sentiment in a text. Among the natural language processing (NLP) models in the GPT-3 family, davinci-003 was introduced in late November 2022. The "text-davinci-003" version represents a specific configuration of this model known for its large scale and impressive language generation capabilities. We aimed to employ GPT-3's natural language understanding to predict sentiment in our given text dataset without the need for explicit training on sentiment-specific data.

Methodology:

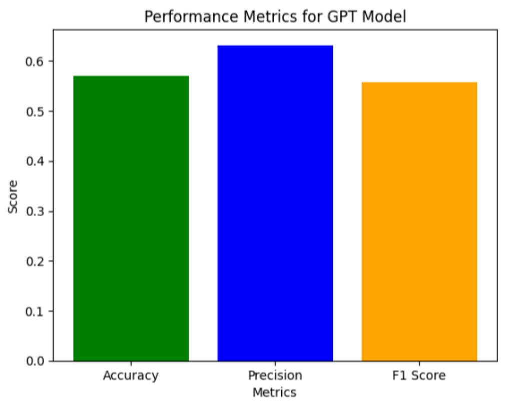
Our approach involved using the Open AI API endpoint to interact with the "text-davinci-003" model. This API allowed us to send text prompts and receive generated responses. By framing our task as a sentiment prediction problem, we input textual data and utilized the model's language generation abilities to extract sentiment-related information.

Implementation:

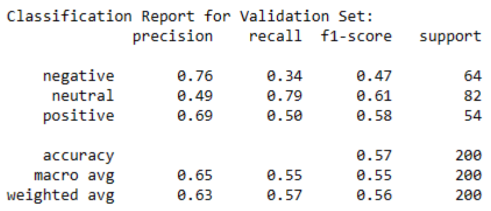
Within our code, we implemented a for loop to iterate through each entry in our dataset. For every iteration, we sent a text prompt, along with prewritten text as a payload, to the text-davinci-003 model. Integration of the Open AI API into our project enabled seamless communication with the GPT-3 model. This process involved obtaining API keys and configuring the necessary parameters. Predicted sentiments from the model are then compared with the actual sentiments to evaluate the model using accuracy, precision, and F1 score metrices. Below, you will find the classification report and performance metric graph identified during the model testing.

In the course of this project, we conducted multiple executions to assess the sentiment prediction capabilities of the GPT-3 model using the "text-davinci-003" variant. The GPT-3 model, taking approximately 12 hours to process our data and generate results, accounted for the time from initiating the API requests to receiving the final responses. In terms of cost analysis, the execution incurred a total cost exceeding $50. This financial investment can be attributed to the utilization of the OpenAI API, with costs linked to the number of tokens processed. To assess variations in the generated responses, we executed the sentiment prediction process twice. These additional executions were aimed at exploring the consistency and reliability of the model's predictions. The time, financial, and execution details provided contribute to the overall evaluation of the GPT-3 model's performance in sentiment analysis, shedding light on both the resource investment and the iterative nature of our exploration. This additional information offers a comprehensive understanding of the practical aspects and resource utilization associated with employing the GPT-3 model for sentiment analysis in our project.

Performance metric graph of GPT 3 daVinci model:



Classification report of GPT daVinci model:



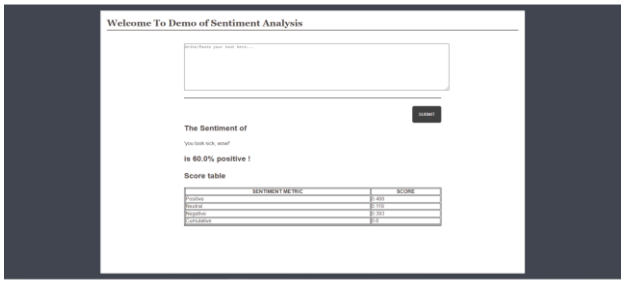
```
Classification Report for Validation Set:
              precision    recall  f1-score   support

    negative       0.76      0.34      0.47        64
     neutral       0.49      0.79      0.61        82
    positive       0.69      0.50      0.58        54

    accuracy                           0.57       200
   macro avg       0.65      0.55      0.55       200
weighted avg       0.63      0.57      0.56       200
```

Models/Evaluation Metrics	**Negative(0)`**				**Neutral(1)**				**Positive(2)**			
	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
GPT	57.86	76	34	47	57.86	49	79	61	57.86	69	50	58

Table 1: Gpt-3 Inference Results

Front-end of Webapp

After following the process of data collection to making machine learning models on the data, we present the demo of how sentiments are analyzed and predicted in our application. Flask, HTML, CSS was used to build this user-interface.
The user writes or pastes their text in the text area to get the predictions on what are the sentiments of the text.


Welcome To Demo of Sentiment Analysis

Write/Paste your text here...

SUBMIT

The Sentiment of

'you look sick, wow!'

is 60.0% positive !

Score table

SENTIMENT METRIC	SCORE
Positive	0.488
Neutral	0.119
Negative	0.393
Cumulative	0.6

Figure 20: Webapp for sentiment analysis

In the above figure, the Sentiment of the sentence “you look sick, wow” is 60% positive. The Score table has Sentiment Metric that gives the spectrum of scores instead of just multilevel classification. This metrices help to understand the context.

Results and Discussion

In our extensive sentiment analysis study on ML vs LLMs, we extensively trained a spectrum of machine learning models, drawing insights from an array of established research papers. Under the guidance of our professor, we delved into large language models (LLMs), aiming to evaluate their performance on our dataset. Traditional models such as Multinomial Naïve Bayes, Random Forest, Logistic Regression, and decision trees proved relatively straightforward due to ample existing research. However, navigating the realm of LLMs presented challenges. These models demanded substantial computational resources, requiring GPU or TPU setups, resulting in prolonged training sessions lasting up to 4 hours per run, necessitating overnight computations. To tackle computational constraints, we adopted DistilBERT and RoBERTa, streamlined variants designed for faster computations based on BERT a LLM. Utilizing open AI APIs, particularly for BERT, GPT, and RoBERTa, incurred initial expenses, notably with larger batch sizes.

To optimize efficiency, we fine-tuned batch sizes, eventually achieving an optimal balance between computational resource utilization and accuracy. Our empirical trials highlighted the superior performance of LLMs, showcasing the exceptional accuracy of both DistilBERT and RoBERTa in sentiment analysis on our tailored dataset. Meanwhile, traditional models maintained anticipated performance levels. This comparative analysis emphasized the efficacy of LLMs, especially DistilBERT and RoBERTa, in handling sentiment analysis tasks compared to conventional machine learning models. It underscored the significance of advanced language models in understanding sentiment nuances, while also highlighting the critical role of judicious model selection and computational optimization in our experiments conducted in our research.

Results as a table:

Models/Evaluation Metrics	**Negative(0)`**				**Neutral(1)**				**Positive(2)**			
	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score	Accuracy	Precision	Recall	F1 Score
Random Forest	82.7632	85.533	72.91	78.719	82.7632	75.4708	92.39	83.0765	82.7632	93.4763	79.36	85.8432
XGBoost	83.212	82.3077	77.11	79.626	83.212	79.2793	88.86	83.7974	83.212	90.1307	81.66	85.6856
Logistic Regression	82.9472	80.235	81.43	80.828	82.9472	83.7423	81.74	82.7273	82.9472	84.435	85.92	85.1715
Multinomial Naïve Bayes	80.5336	80.7026	73.38	76.868	80.5336	76.9293	84.76	80.6553	80.5336	85.8436	81.58	83.6575
DistilBERT	88.16	83.1585	90.78	86.800	88.16	90.1335	84.57	87.2635	88.16	90.7848	90.46	90.6231
RoBERTa	89.2285	86.2363	90.08	88.114	89.2285	88.2259	90.07	89.1545	89.2285	93.7659	87.32	90.4294

Table 2: ML vs LLM Results

Future work

Future work could involve the full-scale implementation of the BERT (Bidirectional Encoder Representations from Transformers) model. This would focus on opting for greater computational efficiency while maintaining or improving accuracy in sentiment analysis. The optimization could include exploring various BERT architectures and fine-tuning the model parameters to better suit our specific dataset and analysis requirements.

Another area of development is the expansion of our current dataset to include a broader range of tweets, possibly from different regions or languages, to increase the model's generalizability. Alongside this, the implementation of user-specific sentiment analysis could provide more personalized insights. This approach would analyze the context of specific users' tweeting history which in turn captures more nuanced sentiment expressions.

The project could also explore the training of a Generative Pre-trained Transformer (GPT) model on our preprocessed datasets. Fine-tuning a GPT model, possibly GPT-3.5, would involve adjusting it to better understand the specific linguistic nuances and sentiment expressions found in our Twitter dataset. Analyzing the accuracy of the GPT model post-training would provide insights into its efficacy in sentiment analysis compared to other models used in the project.

Future research could include the implementation of emerging models like Llama 2 and Bard. These advanced models, which might offer improvements or different approaches compared to existing LLMs, could be explored to assess their effectiveness in sentiment analysis. This would involve training these models on our dataset and comparing their performance with that of currently used models in terms of accuracy, computational efficiency, and their ability to handle nuances in sentiment expression.

Conclusion

Our research aimed to unravel the efficacy of traditional machine learning models Vs cutting-edge large language models (LLMs) like DistilBERT and RoBERTa in sentiment analysis. The comparative study, conducted on our dataset collected from Kaggle competition, illuminated distinctive characteristics and performance benchmarks of each model category. Traditional models, encompassing Decision trees, Random Forest, Logistic Regression, and Multinomial Naive Bayes, demonstrated robustness and familiarity in sentiment analysis tasks. These models showcased reliable performance levels. Their interpretability and ease of implementation stand notable, particularly in simpler sentiment analysis scenarios.

Conversely, our exploration into LLMs revealed a paradigm shift in sentiment analysis. DistilBERT and RoBERTa showcased exceptional accuracy and sophistication in comprehending intricate sentiment nuances. The bidirectional context aggregation and advanced learning mechanisms inherent in these models enabled a profound grasp of nuanced sentiment expressions within text data. Their ability to capture subtle contextual emotions and semantics was remarkable. This comparative analysis delineates a dual narrative. Traditional models present reliable baselines, offering interpretable solutions suitable for less complex sentiment tasks. In contrast, LLMs redefine sentiment analysis capabilities, elevating accuracy to unprecedented levels, especially in complex and nuanced contexts.

Our findings underline the critical importance of model selection aligned with specific task requisites. While traditional models suffice for simpler sentiment tasks, the unparalleled accuracy and nuanced understanding exhibited by LLMs offer an enticing prospect for sophisticated sentiment analysis needs. Ultimately, our study underscores the transformative potential of advanced LLMs like DistilBERT and RoBERTa in unravelling intricate sentiment nuances. These models pave the way for heightened accuracy and deeper understanding in sentiment analysis, heralding a new era in natural language processing applications.

References

1. TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification by Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, Leonardo Neves
2. https://huggingface.co/docs/transformers/model_doc/roberta
3. RoBERTa: A Robustly Optimized BERT Pretraining Approach Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer

Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov

4. Chen, J., & Zhang, J. (2016). Design Collaborative Formative Assessment for Sustained Knowledge Building Using Idea Thread Mapper. In: Proceedings of the International Conference of the Learning Sciences (ICLS 2016). Singapore: International Society of the Learning Sciences.
5. Impulsivity in the self-harm and suicidal behavior of young people: A systematic review and meta-analysis Catherine M McHugh 1, Rico Sze Chun Lee 2, Daniel F Hermens 3, Amy Corderoy 4, Matthew Large 5, Ian B Hickie
6. Sentiment Analysis on Twitter Akshi Kumar, Teeja Mary Sebastian
7. Alzyout, Bashabsheh, Najadat, & Alaiad (2021). Alzyout, M., Bashabsheh, E. A., Najadat, H., & Alaiad, A. (2021). Sentiment Analysis of Arabic Tweets about Violence Against Women using Machine Learning. 2021 12th International Conference on Information and Communication Systems (ICICS). IEEE. doi:10.1109/ICICS52457.2021.9464600
8. Alec Go (2009). Alec Go, L. H. (2009). Twitter Sentiment Analysis. Stanford: Stanford Univesity NLP.
9. Kandpal, P., Wadkar, Y., Attri, H., & Bhorge, S. (2020). Comparison of Sentiment Analysis on Auto-Summarized Text & Original Text using various Summarization Techniques. 2020 IEEE Pune Section International Conference (PuneCon). Pune: IEEE. doi:10.1109/PuneCon50868.2020.9362395
10. Jabbar, J., Urooj, I., JunSheng, W., & Azeem, N. (2019). Real-time Sentiment Analysis On E-Commerce Application. 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC). IEEE Xplore. doi:10.1109/ICNSC.2019.8743331
11. B. Aakash, A. S. (2020). MAGE: An Efficient Deployment of Python Flask Web Application to App Engine Flexible Using Google Cloud Platform
12. Irawaty, I., Andreswari, R., & Pramesti, D. (2020). Development of Youtube Sentiment Analysis Application using K-Nearest Neighbors (Nokia Case Study). 2020 3rd International Conference on Information and Communications Technology (ICOIACT). IEEE. doi:10.1109/ICOIACT50329.2020.9332151
13. Friedman, J. H. (2001). "Greedy Function Approximation: A Gradient Boosting Machine." Annals of Statistics.
14. Natekin, A., & Knoll, A. (2013). "Gradient boosting machines, a tutorial." Frontiers in Neurorobotics.

15. Chen, T., & Guestrin, C. (2016). "XGBoost: A Scalable Tree Boosting System." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
16. Radford, A., et al. (2019). "Language Models are Unsupervised Multitask Learners." OpenAI Blog.
17. Devlin, J., et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805.
18. “An Introduction to Logistic Regression Analysis and Reporting” Joanne Peng, Kuk Lida Lee The Journal of Educational Research 96(1):3-14 DOI:10.1080/00220670209598786

19. F. MIRO-LLINARES, J. R.-S. (2016). Cyber hate speech on twitter: Analyzing disruptive events from social media to build a violent communication and hate speech taxonomy. International Journal of Design & Nature and Ecodynamics 11(3):406-415. doi:10.2495/DNE-V11-N3-406-415
20. Suryavanshi, C. M. (2021). Sentiment Analysis of Product Reviews: Opinion Extraction and Display. CALIFORNIA STATE UNIVERSITY, NORTHRIDGE. From https://scholarworks.csun.edu/bitstream/handle/10211.3/219520/Suryavanshi-Chinmay%20Milind-thesis-2021.pdf?sequence=1
21. Random Forest Classifiers: A Survey and Future Research Directions Vrushali Y Kulkarni, Pradeep K. International Journal of Advanced Computing, ISSN:2051-0845, Vol.36, Issue.1