Project - Sentiment Analysis on Apple from Twitter Tweets based on Apple Event 2022

Authored by:

Bian Hengwei (CSC) U1923732B

Fu YongDing (CSC) U1921155E

Gavin Neo Jun Hui (CE) U1921265L

Pong Zhi Jun Jeremy (DSAI) U1922951J

Vedula Kartikeya (CE) U1923891G

Luo XiaoYang (CE) U2022803B

Preface

The purpose of the report is to find out the sentiments of Apple from Twitter tweets from the Apple Event 2022. Held recently on 7 September 2022, the event introduced several novel Apple products, namely iPhone 14, iPhone 14 Pro, iPhone 14 Pro Max, Apple Watch Ultra, Apple Watch Series 8, new Apple Watch SE, Airpods 2. This report will delve into the generic review on the latest iPhone and Apple Watch, as well as the overall Apple Event 2022 itself.

We would like to express our sincere gratitude to Google LLC for providing a free collaborative Google Colab workspace which allows our group to brainstorm and code as a team using its free hardware resources.

This report on Sentiment Analysis of Twitter Tweets is divided into 7 parts.

1. Introduction and Background
2. Crawling
3. Classification
4. Innovation
5. Frontend UI
6. Discussion
7. Conclusion

YouTube Link:
https://youtu.be/6S8HrA5dNO8

Google Drive Link: (Dataset + Source Code)
https://drive.google.com/drive/folders/1AEykZZu6Tfl60A9Q-blObAnB_TcIpU0u?usp=share_link

Content

1 Introduction & Background

Powerful events are often a critical success factor for any business. According to a study on event-marketing by Bizzabo, it has shown that 31 percent of marketers believe events are more effective than digital marketing, email marketing and even content marketing. (Rafalson, 2017)

1.1 Why Apple?

Apple is a perfect example of an organization which utilizes event-marketing to its advantage. Annually, Apple hosts the Apple Event which has always been a highly anticipated occasion for the 900 million iPhone users all around the world. (Jay, 2020)

Apple Event is consistently praised for its clear message and simple marketing elements. (May, 2018) As the Apple Event 2022 approaches, we are curious to understand consumer sentiments of Apple from this event: Why is it so popular? Have the products released live up to consumer's expectations?

1.2 Why Twitter?

For any brand, social media traction is important for discovering brand perception, growing its influence and improving through customer service. (Benson, 2021) Twitter is therefore a useful channel to gather feedback through trending topics, hashtags and mention features.

Tweets can be scraped using the Twitter API that will directly mention a brand or hashtag topic. Twitter API is a simple-to-use developer platform that allows up to a generous 500 thousand tweets request per month on a personal access basis. (*Getting Access to the Twitter API*, 2022) Our group decided to utilize this tool to conduct our sentiment analysis given its convenience.

2 Crawling

In this section, we will address how we crawled and massaged our data to produce our final dataset.

2.1 Answers to Question 1

1) **How you crawled the corpus (e.g., source, keywords, API, library) and stored it**

   Using Twitter API, we generated tweets with hashtags related to the Apple Event 2022 and the products that were released during the event. The hashtags are #AppleEvent, #AppleEvent2022, #iPhone14, #iPhone14Pro, #iPhone14ProMax, #AirPodspro, #AirPodspro2, #AppleWatch, #AppleWatchUltra, #AppleWatchSeries8 and #AppleWatchSE.

   The library we used in our code to set up Twitter API is tweepy.

2) **What kind of patterns or insights can be derived from your crawled corpus (e.g., what kind of subtopics or aspects emerge)**

   Twitter tweets can often be descriptive or informal. Hence data pre-processing (specifically mentioned in step 5: Run code) is very crucial in order to extract only the relevant information for sentiment analysis. We further inspect our dataset in csv format to ensure its robustness and readiness for training and testing.

   Observing the tweets crawled, some subtopics that we see are thoughts about the products unveiled at the Apple event, such as the new iPhone 14 series, Apple Watch Series 8, Ultra and Airpods2.

   In addition to the aforementioned tweets returned, irrelevant tweets of presumably spam, ads or wrongly tagged were also found in our crawled corpus. Hence we had to conscientiously skip them during our manual

labeling of data.

3) **The numbers of records, words, and types (i.e., unique words) in the corpus**

   From Twitter API, we are able to extract 15358 unique tweets in 275503 words for a dataset before pre-processing. From the processed tweets, there are 19824 unique words.

2.2 Setting up Twitter API

In using Twitter API, we have to follow a sequence of steps in order: import Twitter packages, retrieve consumer and access token from Twitter Developer, authenticate our access, initialize number of tweets to request and hashtags for the topics of tweets we want to collect. We have passed our desired language in Twitter API parameters to obtain only English tweets.

2.3 Data Pre-processing (Regex)

For every tweet received, we will do pre-processing of data in a series of data manipulations using regular expressions.

Merge Tweets into Single Line

Tweets can come in numerous formats such as paragraphs, bulleted lists, numbered lists. We want to concatenate tweets into a single sentence of string for standard data formatting and ease of data utility. This will be the initial step after obtaining the tweet.

Remove any hyperlink

Tweets can be accompanied by a link to a websource, image, gif, etc. While

links could potentially encapsulate certain elements of sentiments, we will only be focusing on text language processing in our experiments.

Remove any retweets (eg. RT @User1)

Tweets can come in the form of retweets, similar to quoting another person's tweets to demonstrate opinions. While retweets showcase the relationship between the original tweet creator and retweet owner that might influence retweet owners' sentiments, we will neglect that aspect and only be focusing on text language processing in our experiments.

Remove user directs (eg. @User2)

Tweets can come in the form of user directs, similar to dedicating an opinion from the tweet owner to another Twitter user. While the user directs showcase the relationship between the tweet owner and the tweet's intended recipient that might influence tweet owners' sentiments, we will neglect that aspect and only be focusing on text language processing in our experiments.

Remove punctuations

Tweets often have punctuation to form complete sentences in the contemporary English language. While certain punctuation combinations may be used to symbolize sentiments, we will neglect that aspect and eliminate all punctuations.

Remove emojis

Tweets may be accompanied with emojis as a form expression by tweet creators. Emojis are primarily encoded in Unicode on their general appearance and certain emojis may showcase sentiments. However, we will neglect emojis in our experiments but an analysis with unicodes might be an interesting aspect.

Remove hashtag in front of words

Tweets retrieved using the Twitter API according to hashtags often carry along the hashtags. In this case we will be removing the hashtag and retain the remaining word after the hashtag, ie. keeping iPhone in #iPhone.

Remove numbers

Tweets can contain numbers in it. While some number combinations may represent certain sentiments, in this case we will be removing all numbers for text language processing in our experiments.

Remove unwanted spacing between words in a sentence

Tweets might have unnecessary spacing due to the tweet creator's intention. We will remove the unwanted spaces for ease of tokenization (to be explained in later sections).

Lowercase sentence

Case folding by lowercasing all words in a tweet sentence might introduce ambiguity of language, for example 'Apple' as a brand name versus 'apple' as a fruit. However, we will neglect that aspect as we consider that Twitter users have the liberty to tweet in whatever capitalization, and hence ambiguity might be too complex to resolve.

2.4 Checking for Duplicates

Before storing our initial pre-processed tweets, we will check if the same tweet already exists in our storage. If it exists, we will neglect the particular tweet and call the Twitter API again. This ensures minimal duplication and maintains data integrity in terms of the quality and amount of data we work with.

3 Classification

In this section, we will discuss the methods we used to classify the tweets with sentiments.

3.1 Answers to Question 2

Pretext) Covering subjectivity detection and polarity detection subtasks on the crawled data

Our approach to these 2 subtasks is to consider the objective to be a multiclass classification problem. In our labeled dataset, we first eyeballed it to be neutral vs opinionated and then classify the resulting opinionated data as positive or negative.

For both machine learning and deep learning methods, these two subtasks are achieved as we train the models on our manually labeled dataset to classify test tweets into sentiments of “NEGATIVE”, “NEUTRAL” and “POSITIVE”.

In our LSTM method, besides training a multiclass model, we also tried training two binary classification models, one subjectivity and one polarity, and used a pipeline to perform the multiclass classification. We found out that the overall accuracy is similar for a single multiclass classification model and two binary classification models.

1) Motivate the choice of your classification approach in relation with the state of the art

For our project, one major problem we faced is to gather enough labeled data for training. Although the apple event takes place every year, tweets regarding past apple events may not be helpful in determining the sentiment of the company from this year’s event. We also found that auto labellers such as VADER were not able to give us very accurate labeling of the sentiments of the tweets that we crawled. The apple event is also a niche topic, so using labeled datasets available online such as the popular Sentiment140 for sentiment analysis may not give us desirable results. Therefore we had to resort to manual labeling, where we managed to label 4,800 records in total.

With this constraint in mind, our classification approach for this project is to make use of traditional machine learning methods such as Naive Bayes and KNN. Since such models are less complex, they are able to generalize better with a smaller training dataset. These models are also able to train more efficiently, as compared to complex models such as deep neural networks. Since the dataset contains tweets with a neutral sentiment, we chose models that are able to support multiclass classification. We have decided to explore using different machine learning models to see which models work well. Detailed explanations of each machine learning algorithm can be found in the description of each model below, coupled with its advantages and disadvantages.

In addition to machine learning models, we also explored using deep learning methods. Since we have a small dataset, for Long Short-Term Memory we first pre-trained our deep learning models on datasets we found online, before training and testing on the dataset we gathered. For BERT, we use the same dataset as the one used for traditional machine learning, even though it is small, which we will address the drawbacks in its section.

2) Discuss whether you had to preprocess data (e.g., microtext normalization) and why

Detailed explanation can be found in Chapter 3.2.2 Further Processing and Chapter 3.3 Text Normalization.

3) Build an evaluation dataset by manually labeling 10% of the collected data (at least 1,000 records) with an inter-annotator agreement of at least 80% (it is recommended to have 3 annotators, but 2 is also OK)

Requirement satisfied in Chapter 3.2.3 Data Labeling.

4) Provide evaluation metrics such as precision, recall, and F-measure and discuss results

This is the overall accuracy results:

Machine learning model	Accuracy (%)
K-Nearest Neighbour	58.25
Logistic Regression	66.08
Naive Bayes	66.17
Support vector machine (SVM)	66.17
Decision trees	58.17
Extra trees	68.42
Random forest	64.50
XGBoost	67.33
Long Short-Term Memory (LSTM)	74.08
BERT	76.46

Individual evaluation metrics using precision, recall, and F-measure are shown in the corresponding model's section.

In traditional machine learning approaches, Extra Trees, XGBoost, Naive Bayes, Support Vector Machines, Logistic Regression classifiers performed relatively better. Random Forest gave a mediocre result while Decision Trees and KNN classifiers provided a relatively poorer performance.

Deep learning models of LSTM and BERT provided better accuracies peaking at around 75-76%.

5) Discuss performance metrics, e.g., records classified per second, and scalability of the system

The table below shows the train time complexity of the machine learning models we have attempted to use. n refers to the number of records in the dataset. Based on the table, most of the model is linear in terms of the number of records in the dataset, except for SVM. Therefore SVM does not scale well with regards to the number of records in the dataset, while the other models scale relatively better.

k → number of neighbours/trees
d → number of features

Machine learning model	**Train time complexity**	**Wall time**
K-Nearest Neighbour	O(knd)	159 ms
Logistic Regression	O(nd)	89 ms
Naive Bayes	O(nd)	6.99 ms
Support vector machine (SVM)	O(n^2)	323 ms
Decision trees	O(n*log(n)*d)	105 ms
Extra trees	O(n*log(n)*d*k)	2.44 s
Random forest	O(n*log(n)*d*k)	137 ms
XGBoost	O(tdxlogn)	15.4 s

If we are looking to expand into real world systems, time and accuracy are 2 major focal areas for a good working application. A system should be able to predict with high accuracy in a short amount of time.

6) A simple UI for visualizing classified data would be a bonus (but not compulsory)

Requirement discussed in Chapter 5 Frontend UI.

3.2 Data Pre-processing (Manual)

3.2.1 Data Storage

We store our scraped data from Twitter API in .csv format for ease of access through data frames found in Python Pandas library later. The .csv file is uploaded on a personal Github account for shared usage using a url link.

3.2.2 Further Processing

Before uploading our dataset on Github, we manually inspect our data derived from the initial data-processing through which we discover certain ambiguities.

1. Foreign tweets that managed to be returned through Twitter API even though we requested only for English. For example, "eeii god why the girl sef wait make dem buy iphone for her before she talk say she no do again vawulencespace violence werey werey tiwa savage sad romance sadromance tiwasavage olamide iphonepro cruise".
2. Tweets that do not make logical sense in terms of the English language. For example, "pharmacology daily quiz fluoroquinolones anitbiotics emergency med twitter health medicine".
3. Tweets unrelated to our hashtags that were mistakenly tagged. For example,"goal on pubg pubg gameplay noob gamer watch now iphone".

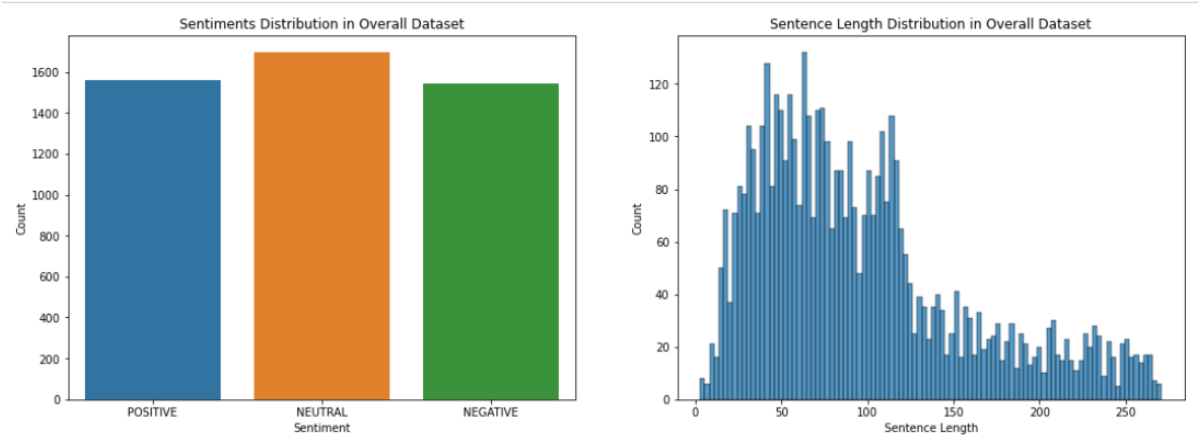
The anomaly tweets are filtered out and removed from our evaluation dataset to ensure purity of data.

3.2.3 Data Labeling

As tweets do not have any metrics such as ratings that are commonly found

in product review datasets, we only can use the texts from the tweets scrapped to evaluate the sentiments. As discussed earlier, we found auto labellers to be quite inaccurate, therefore we had to rely on eyeballing each tweet and label the sentiments individually. We managed to label 4800 records that were used for both training and testing in our models.

This is the distribution of our manually labeled dataset:



3.3 Text Normalization

Defined as the process of transforming text into a canonical (standard) form, text normalization is used to reduce noises generated by a single word in multiple forms. This also helps to increase efficiency in model training.

3.3.1 Stemming

Stemming is a process that stems or removes the last few characters from a word without considering context. For example, from the word "studying", it removes the -ing to form the stem "study". However, as this method is easy and does not require a corpus, it will form words that are not found in the dictionary.

3.3.2 Lemmatization

Lemmatization is a process that considers context and converts word to its meaningful base form. Unlike Stemming, Lemmatization requires a corpus, and thus the base form is still proper English words.

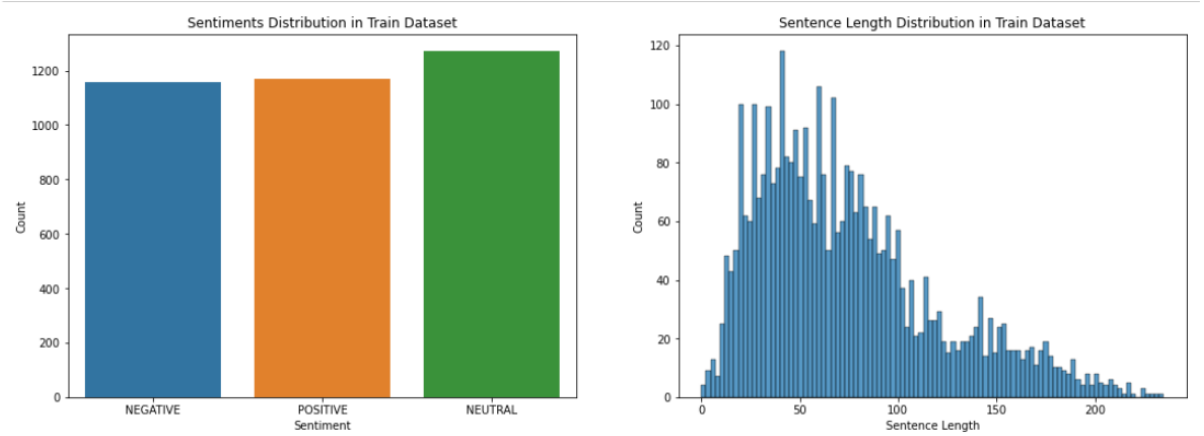
3.3.3 Process

We will consider text normalization in the following sequence: Tokenization → Removal of Stopwords → Stemming → Lemmatization. After the process is done, the model training will be easier as the dataset is now filled with words in the simplest forms.

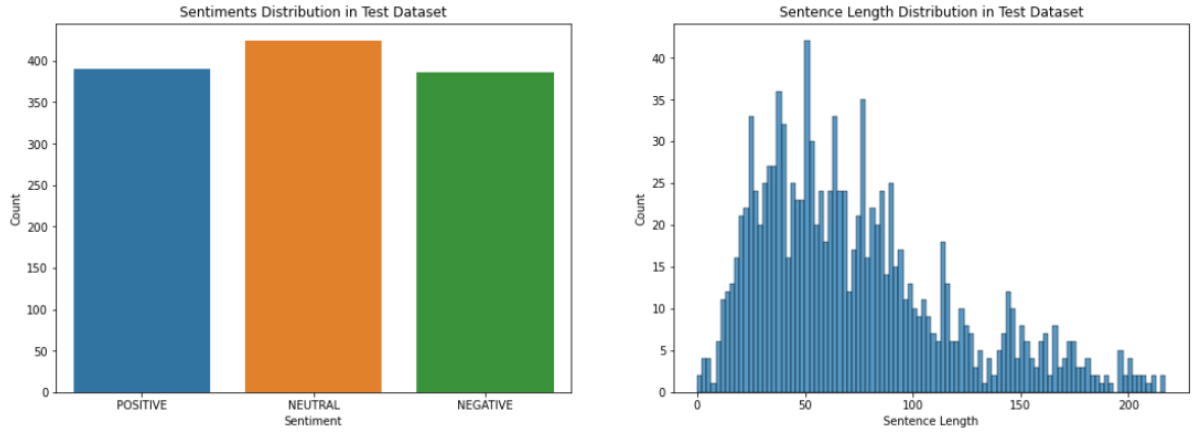
3.4 Split into Train and Test Datasets

In machine learning approaches, we typically split the samples into train, validation and testing data. Training data is the subset of samples used to fit the model on. Validation data is a sample of data used to provide some unbiased evaluation of a model fit while the model is fitting. On the other hand, test data is a sample that is used to provide an unbiased evaluation of the final trained model. For traditional machine learning methods and LSTM, we will split our dataset into 75% training data and 25% testing data for use in our generic classification models. This is because the models we are planning to experiment on do not have an evaluation phase while it is fitting, thus the validation samples are not needed.

This is the distribution of our train dataset:



This is the distribution of our test dataset:



For BERT, we split our dataset into 90% training data and 10% testing data to conduct a 4-Fold cross validation and testing.

In all our dataset splits, we do a stratifying separation to ensure even data distribution of the 3 classes “POSITIVE”, “NEGATIVE”, “NEUTRAL” to avoid the problem of skewed dataset.

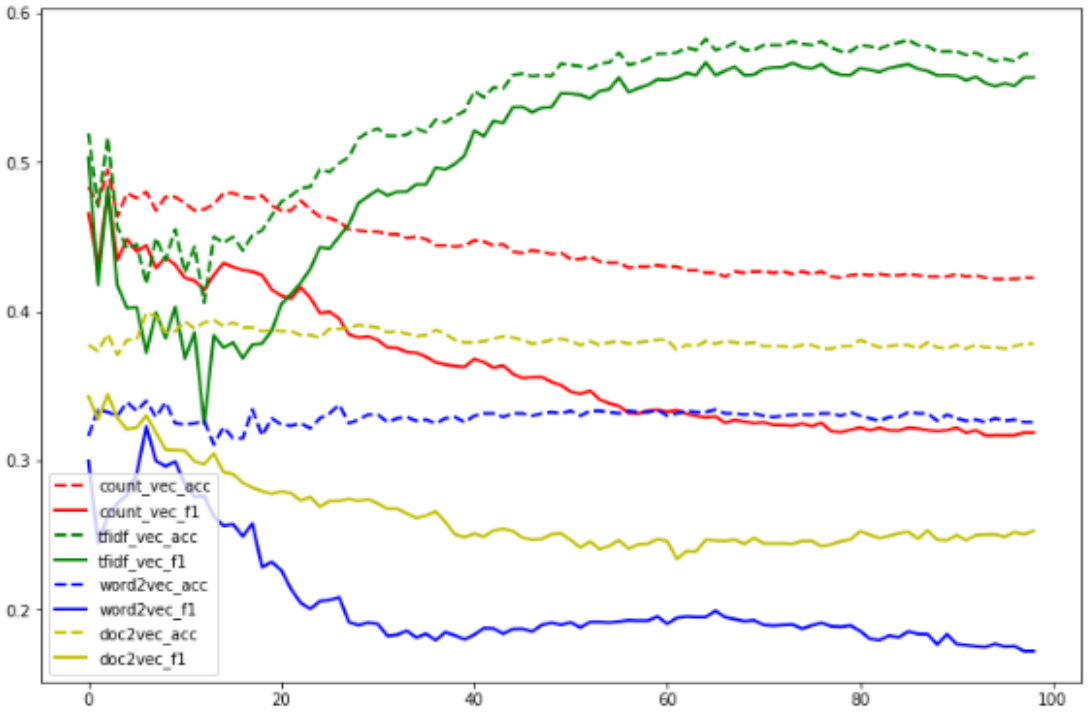
3.4.1 Feature Scaling (Vectorizer)

For machine learning approaches, the models are only able to interpret

numerical data. Therefore, before the models are fitted, we vectorize the samples in both the training and testing set. Vectorization is an approach of converting input data (words, sentences) from its raw form into vectors of real numbers. In this project, we explored 4 different methods of vectorizing for each model:

1. Bag of words (CountVectorizer)
2. TF-IDF (TfidfVectorizer)
3. Word2Vec
4. Doc2Vec

Among these vectorizers, we found that TF-IDF delivered the best results. The picture below shows the results when we tried these 4 vectorizers on a KNN model.



3.5 Traditional Machine Learning

We explored various traditional machine learning methods for classification.

For each method, we tuned the hyperparameters to try and determine a more appropriate parameter for each individual model.

3.5.1 K-Nearest Neighbors (KNN)

3.5.1.1 Overview

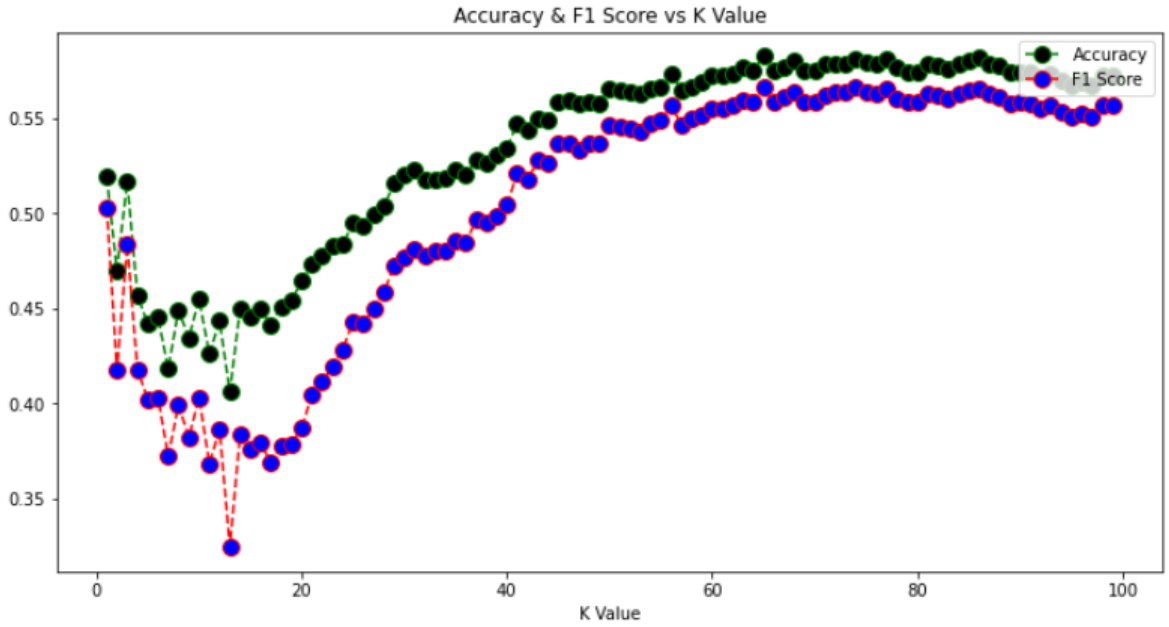
KNN is a supervised learning classification algorithm which predicts values of test data points based on how closely it matches data points in the training set. The KNN algorithm assumes that similar things exist in close proximity. In the KNN algorithm, each unseen or new sample will be given a label based on the majority vote between the “K” closest known samples. Euclidean distance is the traditional method to find the “K” closest match in KNN.

Image: How an unseen sample is classified

In the parameter tuning, we tried a range of values for “K” to determine a better “K” that will be used to fit the model on. This is because the number of neighbors will directly influence the label of the unseen sample as seen from the image above..The optimal K-value will then be used by KNN to classify by finding K nearest neighbors in training data and then using the labels of closest matches to predict.

Optimal K value

The optimal number of neighbors is 65 with the highest weighted F1 score of 0.57.



3.5.1.2 Advantages and Disadvantages

Advantages

- Simple and easy to implement
- Training phase is faster due to lazy learning
- Works better for continuously changing data due to instance based learning

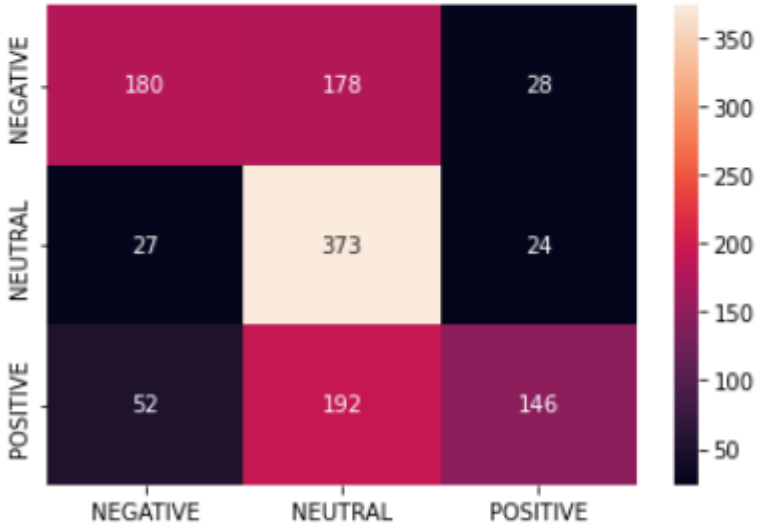
Disadvantages

- Testing phase is slower and costlier in terms of time and memory
- Highly sensitive to outliers
- Tweets used in our experiment are in the form of texts and texts contain a mixture of words. Depending on the size of vocabulary, KNN might not work well with high dimensional data because with a large number of dimensions, it is more difficult to to calculate the distance in

each dimension.

3.5.1.3 Results

	precision	recall	f1-score	support
NEGATIVE	0.69	0.47	0.56	386
NEUTRAL	0.50	0.88	0.64	424
POSITIVE	0.74	0.37	0.50	390
accuracy			0.58	1200
macro avg	0.64	0.57	0.56	1200
weighted avg	0.64	0.58	0.57	1200



3.5.2 Logistic Regression

3.5.2.1 Overview

While logistic regression is usually used for binary classification, it can be extended to multiclass classification by using a multinomial probability distribution. The model will then be fitted to learn and predict a multinomial probability distribution. One difference between a normal binary classification logistic regression and a multinomial logistic regression is the loss function. In order to account for multiple classes, the log loss used in binary classification

has to be changed to a cross-entropy loss.

For logistic regression, we will be tuning the C hyperparameter. This parameter is a form of regularization for the logistic regression model which adjusts the weights given to either the training data or the complexity penalty. This is meant to regulate the model and to prevent overfitting. Unlike the parameter tuning that is done for the KNN model, we will be utilizing another optimization package (BayesianOptimization) which is built upon bayesian inference and a gaussian process that attempts to find the maximum value for a set of values. We have decided to use this instead of a range of integers because unlike “K”, the hyperparameter C can take on decimal real values.

Optimal K value

The optimal C value is 5.16 with the highest weighted F1 score of 0.68.

3.5.2.2 Advantages and Disadvantages

Advantages

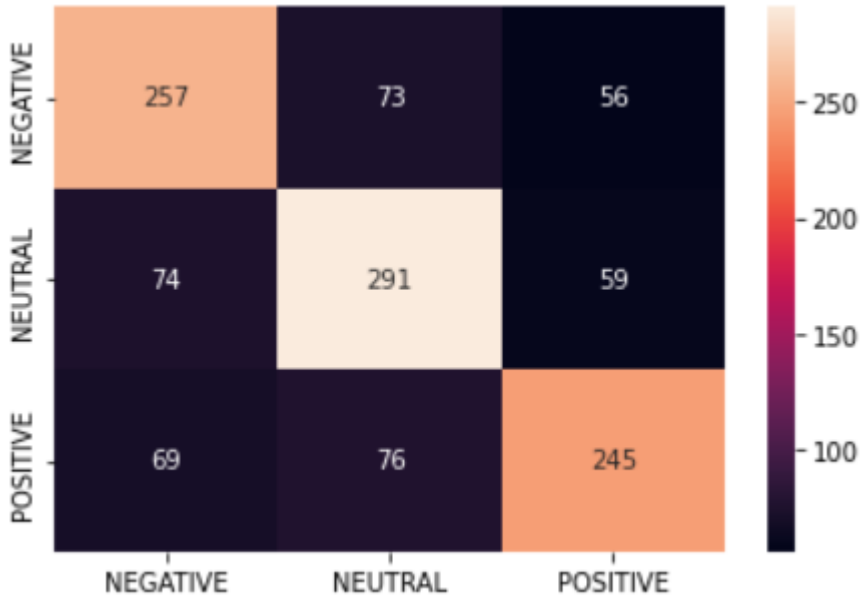
- Easy to implement and trains very efficiently
- Classifies unseen data quickly

Disadvantages

- Construct a linear boundary and may not be suitable for datasets which requires more non-linear boundaries
- Assumption of linearity between dependent and independent variable

3.5.2.3 Results

	precision	recall	f1-score	support
NEGATIVE	0.64	0.67	0.65	386
NEUTRAL	0.66	0.69	0.67	424
POSITIVE	0.68	0.63	0.65	390
accuracy			0.66	1200
macro avg	0.66	0.66	0.66	1200
weighted avg	0.66	0.66	0.66	1200



3.5.3 Naive Bayes

3.5.3.1 Overview

The Naive Bayes Classifier is a probabilistic machine learning model which is

based around bayes theorem, $P(A|B) = P(B|A)P(A) / P(B)$. For text classification, we make use of the multinomial naive bayes model, which take in feature inputs as integers. This model is typically used in various applications such as spam filtering, text classification, sentiment analysis and recommender systems.

For the naive bayes classifier, we decided to tune the smoothing hyperparameter, alpha. This hyperparameter controls the laplace smoothing used by the model. Laplace smoothing typically helps when the model encounters an unseen word in the test set that was not present in the training set. If no laplace smoothing is done, the probability will become 0 which will cause the model to not be able account for truly unseen cases and this will affect how robust the model is.

Optimal Alpha value

The optimal alpha value of the estimator is 1.

3.5.3.2 Advantages and Disadvantages

Advantages

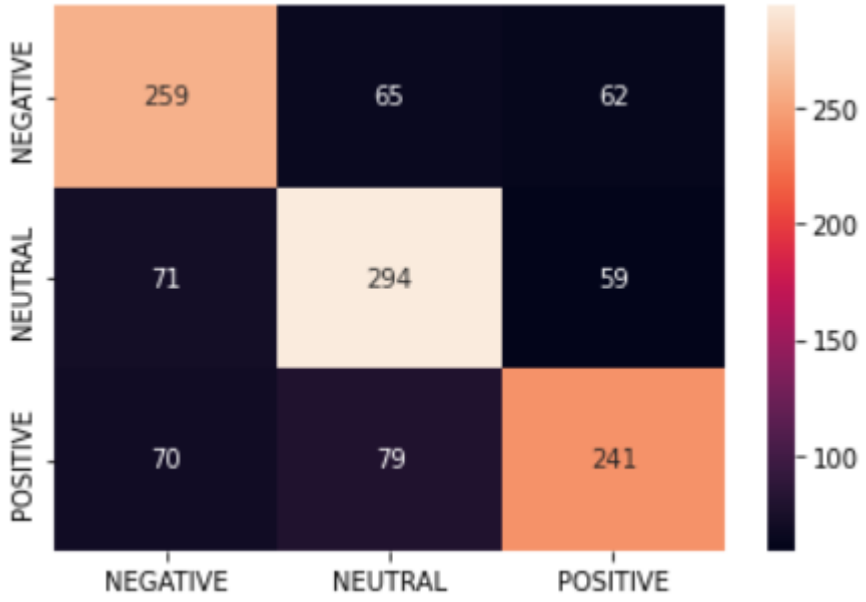
- Simplest and fastest algorithm
- Suitable for large data due to its speed and is highly scalable

Disadvantages

- Assumes the features are mutually independent. This is impossible to achieve in real life as dependency usually assists between attributes.

3.5.3.3 Results

	precision	recall	f1-score	support
NEGATIVE	0.65	0.67	0.66	386
NEUTRAL	0.67	0.69	0.68	424
POSITIVE	0.67	0.62	0.64	390
accuracy			0.66	1200
macro avg	0.66	0.66	0.66	1200
weighted avg	0.66	0.66	0.66	1200



3.5.4 Support Vector Machine (SVM)

3.5.4.1 Overview

SVM is a supervised machine learning algorithm for both classification and

regression problems. SVM performs classification by separating different target classes in a hyperplane in n-dimensional or multidimensional space. SVM draws that hyperplane by transforming the data using mathematical functions called “Kernels”. Since SVM is a binary classification algorithm, it will be performing one-versus-all classification for each of the labels in sentiment analysis, which is negative, neutral and positive. To tune the SVM model for this assignment, there are a few hyperparameters to consider.

There are 3 hyperparameters to be tuned for SVM: Kernels, C and Gamma. Hyperparameter C is the penalty parameter of the error term. It represents how much error is bearable in the model. Gamma defines the distance of influence of a single training point.

To find the best hyperparameters, we used GridsearchCV from scikit-learn library. GridsearchCV iterates through all combinations of the hyperparameters to find out which combination of hyperparameters gives the best results.

Optimal Hyperparameters

The optimal hyperparameters of the estimator are C=1, gamma=1, kernel=’rbf’ (radial basis function kernel) with a score of 0.66.

3.5.4.2 Advantages and Disadvantages

Advantages

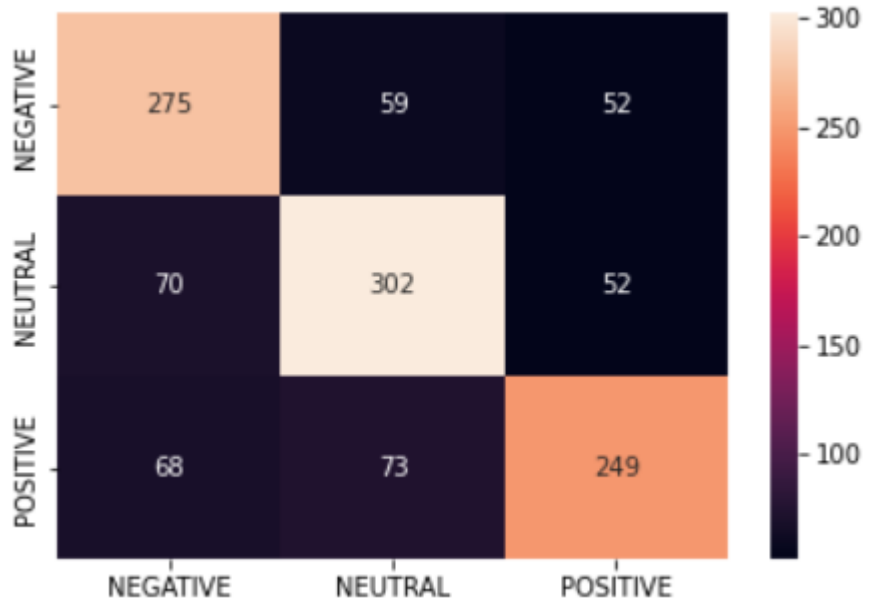
- It works really well with a clear margin of separation
- It is effective in high dimensional spaces.
- It is effective in cases where the number of dimensions is greater than the number of samples.
- It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

Disadvantages

- Large datasets are not a good fit for SVM, and it causes the training time to be very long.
- Performance is not good when the target classes in the data are overlapping.
- SVM will also perform poorly when the number of features for each data point is greater than the number of samples in training data.

3.5.4.3 Results

	precision	recall	f1-score	support
NEGATIVE	0.67	0.71	0.69	386
NEUTRAL	0.70	0.71	0.70	424
POSITIVE	0.71	0.64	0.67	390
accuracy			0.69	1200
macro avg	0.69	0.69	0.69	1200
weighted avg	0.69	0.69	0.69	1200



3.5.5 Decision Trees

3.5.5.1 Overview

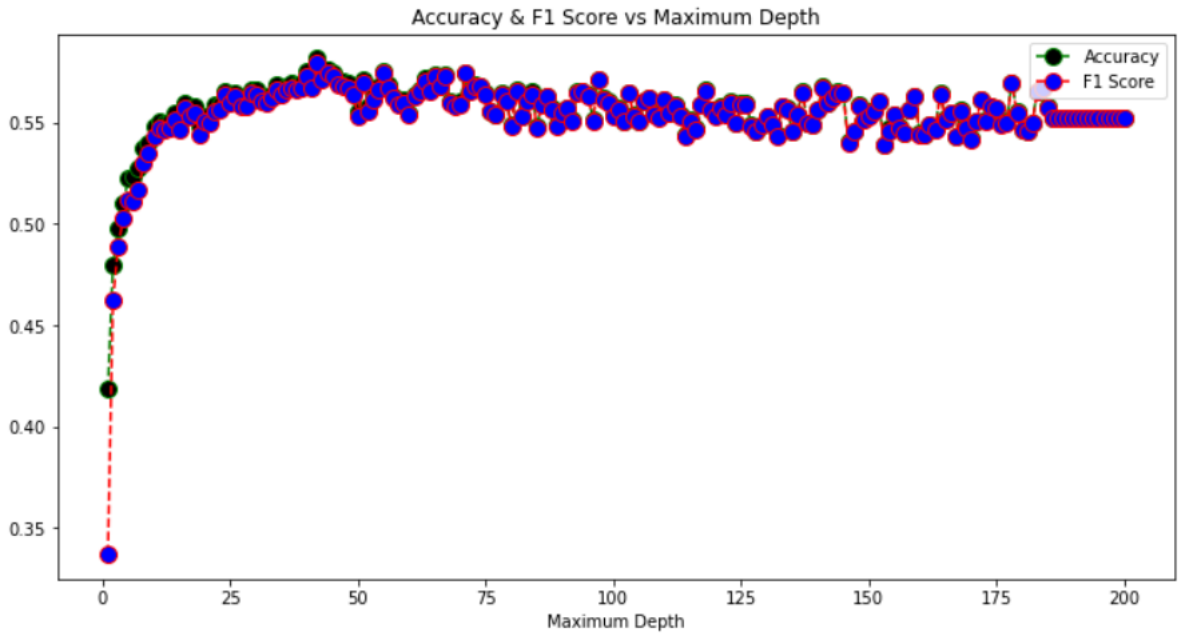
A supervised learning algorithm used for both classification and regression (in this case, we are classifying tweets). The decision tree classifier functions like a rule based classifier. At each node, the samples will be split according to a specific feature. This approach is continuously iterated until all the samples in a single node belong to a single class. As the model continues to narrow down the feature space, we will end up with a pure node, or leaf node, which will contain only samples of the same class. When a decision tree model receives an unseen sample, it follows the branches down the tree based on the features of the unseen sample and when it reaches the leaf node, that will be the predicted label of the unseen sample.

However, one risk of allowing a decision tree to expand all its leaves is that we risk the model overfitting. This is because some samples which may belong to different classes may have very small differences. To reduce overfitting and to make the decision tree model generalize better and more robust, we can tune the maximum depth parameter of the model. Similar to

the KNN model, we try out the decision tree model over a range of integer values to determine a more accurate maximum depth parameter for the model.

Optimal Maximum Depth

The optimal maximum depth is 42 with the highest weighted F1 score of 0.5798.



3.5.5.2 Advantages and Disadvantages

Advantages

- Simple to implement, visualize and explain
- Feature selection is done internally so we do not need to measure feature importance before fitting the model

Disadvantages

- Tendency to overfit

- Time to fit model increases exponentially as data and maximum depth increases

3.5.5.3 Results

	precision	recall	f1-score	support
NEGATIVE	0.58	0.58	0.58	386
NEUTRAL	0.57	0.66	0.61	424
POSITIVE	0.61	0.49	0.55	390
accuracy			0.58	1200
macro avg	0.59	0.58	0.58	1200
weighted avg	0.58	0.58	0.58	1200

3.5.6 Extra Trees Classifier

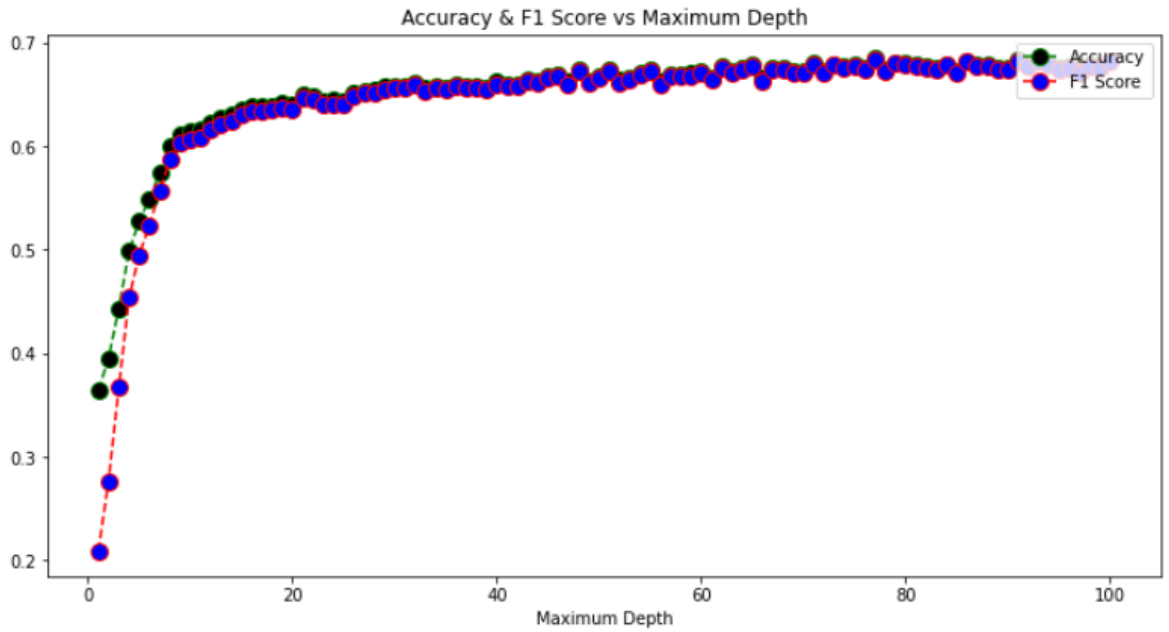
3.5.6.1 Overview

This is a type of ensemble learning, similar to random forest classifiers. ExtraTreesClassifier, like the RandomForest classifier, randomizes certain decisions and subsets of data to minimize over-learning from the data and overfitting. The difference between the 2 ensemble techniques is that the ExtraTrees classifier chooses the split randomly, while the random forest classifiers choose the optimum splits. Furthermore, ExtraTrees classifiers samples without any replacement. Ensemble techniques such as the ExtraTrees classifier aggregates the results of multiple trees to output a classification result.

The ExtraTreesClassifier implementation in scikit-learn has a few parameters we can tune, such as max_depth which controls the maximum depth of each tree and n_estimators which controls the total number of trees which will be used to aggregate the final output. Although there are multiple parameters, we will be tuning only the maximum depth parameter as this parameter works more on a general level to prevent each individual tree from overfitting in the ensemble.

Optimal Maximum Depth

The optimal maximum depth is 77 with the highest weighted F1 score of 0.68.



3.5.6.2 Advantages and Disadvantages

Advantages

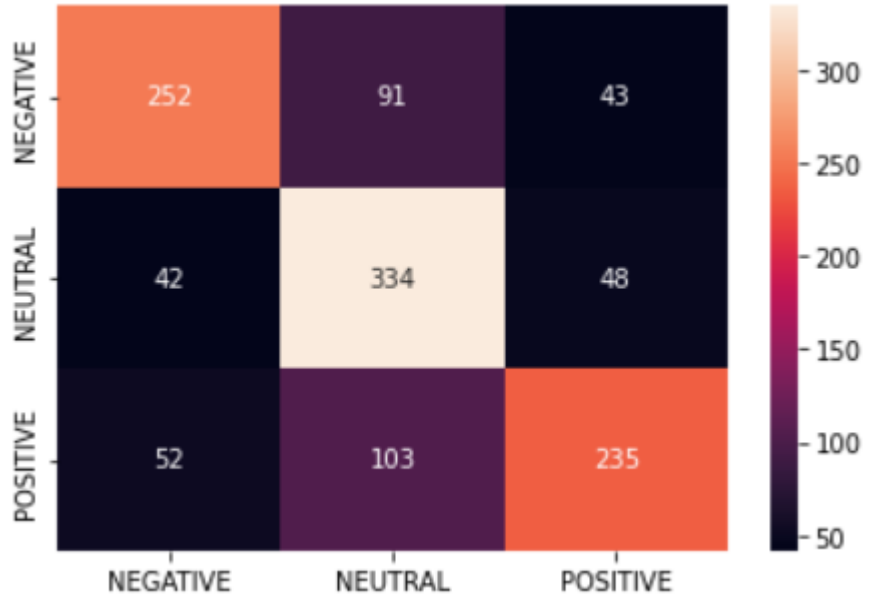
- Computational and execution time is more efficient as compared to random forest classifiers due to the random splits.
- Choosing the split randomly reduces variance and sampling from the entire dataset reduces bias

Disadvantages

- Prone to overfitting

3.5.6.3 Results

	precision	recall	f1-score	support
NEGATIVE	0.73	0.65	0.69	386
NEUTRAL	0.63	0.79	0.70	424
POSITIVE	0.72	0.60	0.66	390
accuracy			0.68	1200
macro avg	0.69	0.68	0.68	1200
weighted avg	0.69	0.68	0.68	1200



3.5.7 Random Forest

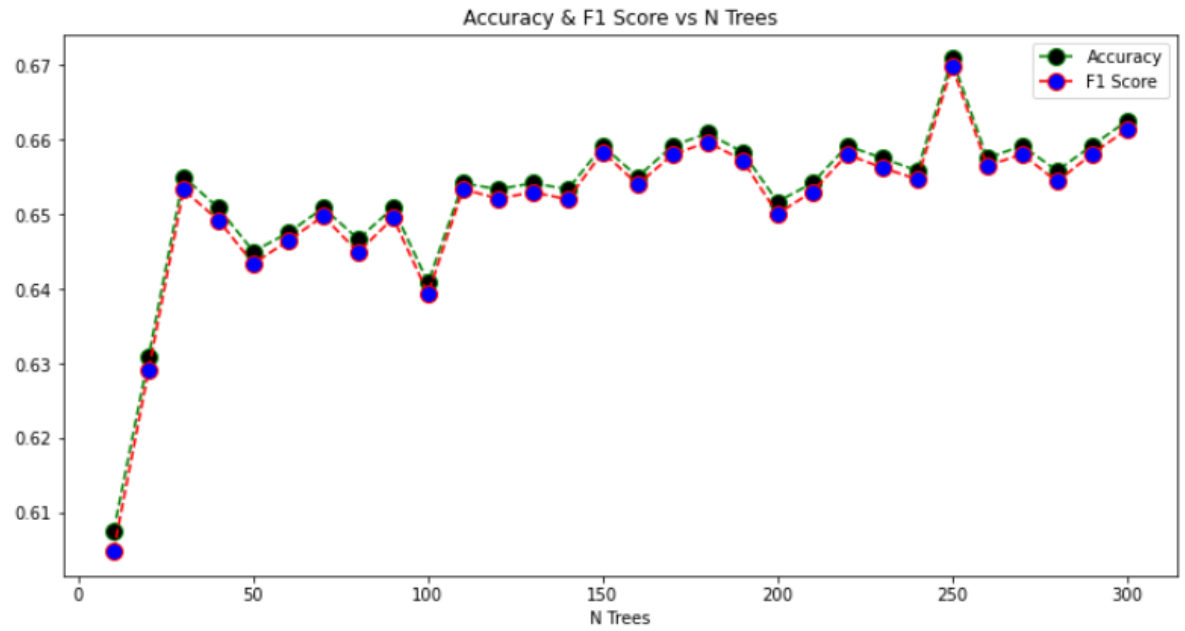
3.5.7.1 Overview

A Random Forest classifier is a type of ensemble machine learning technique which combines a number of decision trees on various sub samples of the dataset and uses the average of the outputs to improve the prediction accuracy and to control overfitting. The random forest algorithm is trained through bagging. Bagging is typically used in machine learning techniques to deal with variance trade-offs and reduces the variance of a model. Generally, as the number of trees increases, the precision of the random forest would also increase as there would be a larger set of sub-samples being trained on. This would allow the random forest to generate more decision trees that would help to make the final prediction.

To find an appropriate number of trees in our random forest, we tuned the n_estimators parameter in the RandomForestClassifier, which controls the number of trees in the scikit-learn implementation of a random forest classifier.

Optimal Maximum Number of Trees

The optimal number of trees is 65 with the highest weighted F1 score of 0.67.



3.5.7.2 Advantages and Disadvantages

Advantages

- More accurate than a single decision tree
- Reduces the overfitting in decision trees
- Robust to outliers due the way random forest classifier generates a final prediction (average of individual decision trees)

Disadvantages

- The training speed is relatively slower as more computational power is needed
- Difficult to interpret and visualize

3.5.7.3 Results

	precision	recall	f1-score	support
NEGATIVE	0.73	0.65	0.69	386
NEUTRAL	0.63	0.79	0.70	424
POSITIVE	0.72	0.60	0.66	390
accuracy			0.68	1200
macro avg	0.69	0.68	0.68	1200
weighted avg	0.69	0.68	0.68	1200

3.5.8 XGBoost

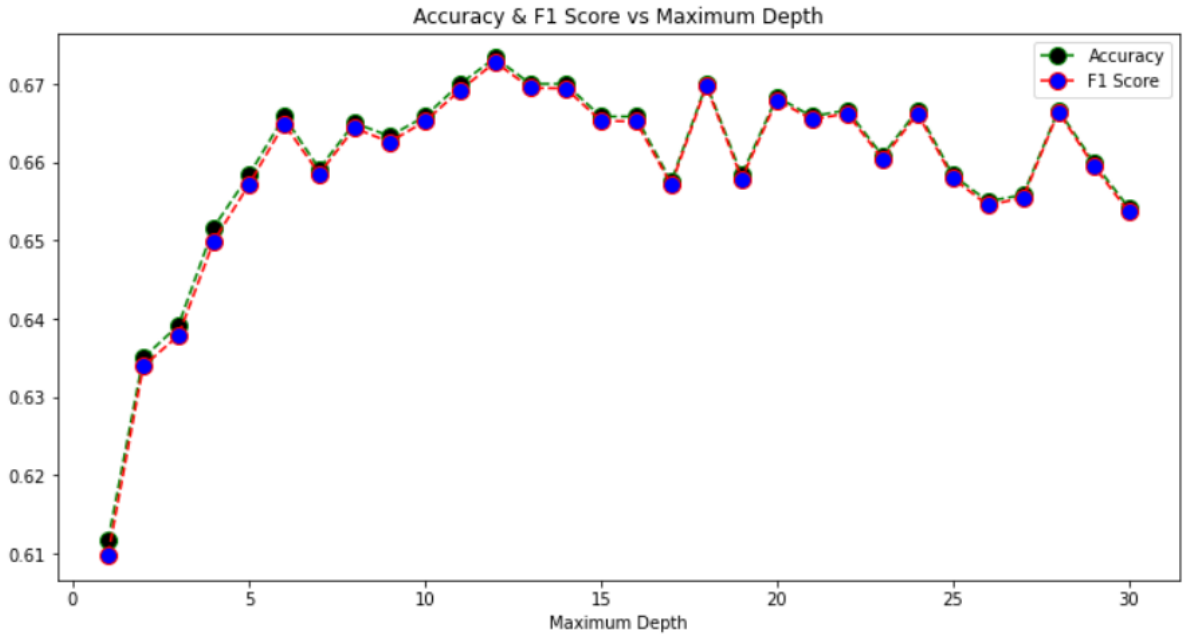
3.5.8.1 Overview

XGBoost is a library which implements distributed gradient-boosted decision trees. A Gradient Boosting Decision Trees (GBDT) is a decision tree ensemble learning algorithm for classification and regression. FIrstly, boosting is a method to create an ensemble. It starts by fitting a model to the training data, followed by building a second model focused on accurately predicting the cases which were incorrectly predicted by the first model. By repeating the process where each subsequent model aims to correct the shortcomings of the previous model, the resulting ensemble will end up robust as it aims to capture and learn the features to correctly predict each data point. Next, gradient boosting is an approach where the subsequent models are created to predict the errors of prior models. They are then added together to make a final prediction. When adding new models, the gradient descent algorithm is then used to minimize the loss.

As the XGBoost classifier can be thought of as an extension and an ensemble of individual decision trees, we chose to tune the max_depth hyperparameter, similar to the decision tree classifier and ExtraTrees classifier.

Optimal Maximum Depth

The optimal maximum depth is 12 with the highest weighted F1 score of 0.673.



3.5.8.2 Advantages and Disadvantages

Advantages

- Able to capture complex patterns in data
- Generally more accurate than a single decision tree

Disadvantages

- Prone to overfitting
- May be computationally expensive and take a long time to train
- Hard to interpret and visualize the ensemble of trees

3.5.8.3 Results

	precision	recall	f1-score	support
NEGATIVE	0.65	0.67	0.66	386
NEUTRAL	0.68	0.72	0.70	424
POSITIVE	0.69	0.62	0.66	390
accuracy			0.67	1200
macro avg	0.67	0.67	0.67	1200
weighted avg	0.67	0.67	0.67	1200

3.6 Deep Learning Methods

3.6.1 Long Short-term Memory (LSTM)

3.6.1.1 Overview

Long short-term memory (LSTM) is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. Such a recurrent neural network (RNN) can process not only single data points (such as images), but also entire sequences of data (such as speech or video). For example, LSTM is applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition, machine translation, robot control, video games, and healthcare. LSTM has become the most cited neural network of the 20th century.

The name of LSTM refers to the analogy that a standard RNN has both "long-term memory" and "short-term memory". The connection weights and biases in the network change once per episode of training, analogous to how physiological

changes in synaptic strengths store long-term memories; the activation patterns in the network change once per time-step, analogous to how the moment-to-moment change in electric firing patterns in the brain store short-term memories. The LSTM architecture aims to provide a short-term memory for RNN that can last thousands of timesteps, thus "long short-term memory".

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.

3.6.1.2 Advantages and Disadvantages

Advantages

- Long Short-Term Memory (LSTM) stores historical information by constructing a memory unit, and each temporal state saves the previous input information, effectively alleviating Recurrent Neural Networks' long-distance dependence (RNN).

Disadvantages

- LSTMs fail to completely remove the vanishing gradients.
- They require a significant amount of time and resources to train and become ready for real-world applications.
- LSTMs are affected by different random weight initializations and thus behave similarly to feed-forward neural networks. Hence, they prefer small weight initialization.

- LSTMs are prone to overfitting, and using the dropout algorithm to mitigate this problem is difficult.

3.6.1.3 Results

```
model1.evaluate(X6, Y6)
```

```
38/38 [==============================] - 1s 14ms/step - loss: 0.7644 - accuracy: 0.7408
[0.7644030451774597, 0.7408333420753479]
```

3.6.2 Bi-Directional Long Short-term Memory (Bi-LSTM)

3.6.2.1 Advantages and Disadvantages

Advantages

- LSTM, at its core, uses the hidden state to preserve information from previously processed inputs. Because the only inputs it has seen are from the past, unidirectional LSTM only preserves information from the past.
- When using bidirectional lstm, the inputs are run in two directions, one from past to future and one from future to past. What distinguishes this approach from unidirectional is that in the LSTM that runs backwards, we get to preserve information from the future, and by combining the two hidden states, we can preserve information from both past and future at any point in time.
- What they are best suited for is a complex question, however, BiLSTMs produce better results because they understand context better. Consider the following situation:
    a. When we try to predict the next word in a sentence, on a high level what a unidirectional LSTM will see is:

        “The boys went to ....”

        And will try to predict the next word only by this context.

b. However, bidirectional LSTM will be able to see information further down the road:
   i. Forward LSTM:
      "The boys went to …"
   ii. Backward LSTM:
      "... and then they got out of the pool."

Hence, we can see that using the information from the future it could be easier for the network to understand what the next word is.

Disadvantages

- BiLSTM is a much slower model and requires more time for training.

3.6.3 BERT Transformer

3.6.3.1 Overview

Bidirectional Encoder Representations from Transformers (BERT) is a deep neural network that is designed to pre-train bidirectional representations of text, considering both left and right context.

A small baseline BERT model is used for our small dataset. Text preprocessing of BERT uses English vocabulary extracted from Wikipedia and BooksCorpus.

3.6.3.2 Advantages and Disadvantages

Advantages

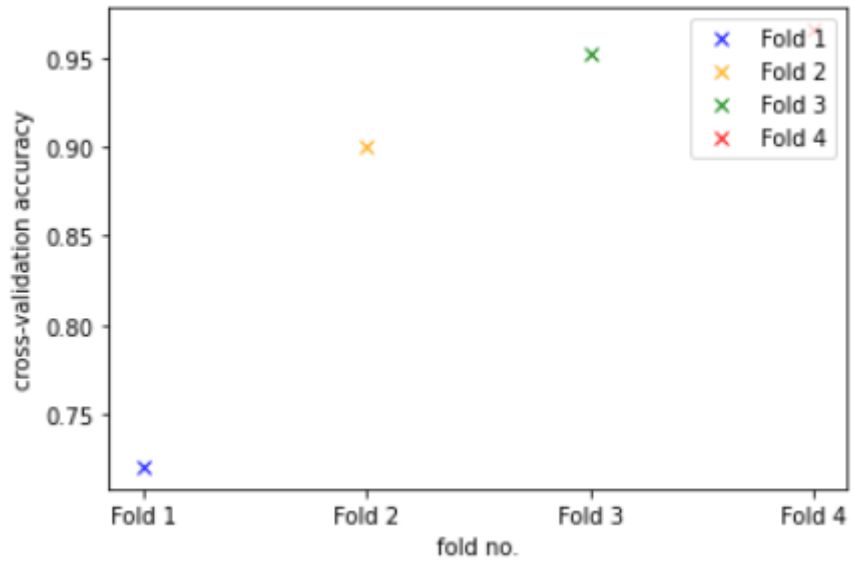
- As opposed to directional models, which reads text input sequentially, usually left to right, BERT transformer encoder reads the entire sequence of words bidirectionally, allowing the model to learn the context of a word in a sentence based on its surroundings, left and right, giving better word interpretation.

Disadvantages:

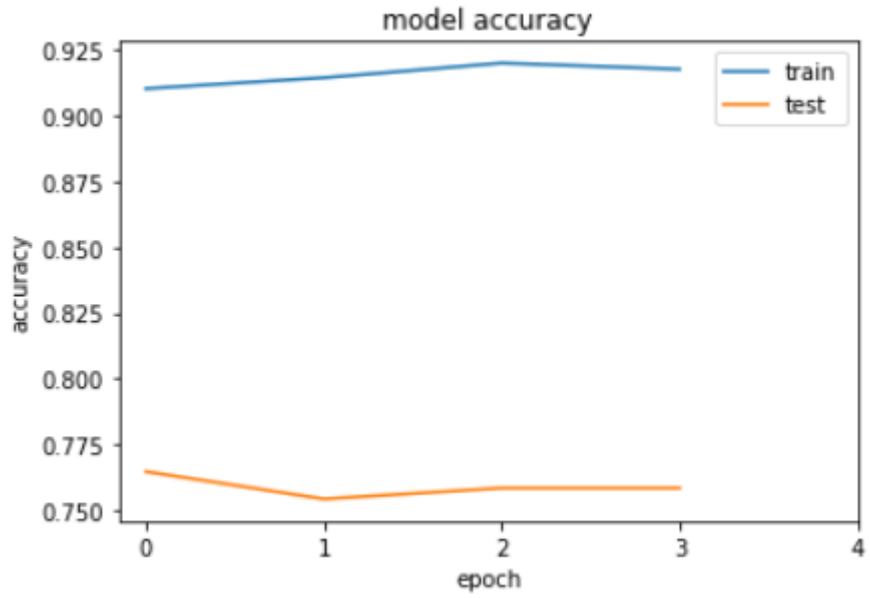
- Since transformers are more complicated with more layers within the network, it thus takes a longer period of time to fully train BERT

3.6.3.2 Results

We conducted a 4-Fold cross validation on our training dataset which is 90% of our overall data. The outcome of above 90% accuracy shows BERT generally works well in Natural Language Processing.



The accuracy on our test split which is unseen.



As we have implemented the EarlyStopping callback, the training was halted after 4 epochs. From the graph above, we observe that the BERT transformer achieved similar results as the LSTM network, even outperforming it slightly. Although the transformer is a model which is capable of learning complex patterns in data, there are drawbacks. Due to our lack of samples, the BERT transformer displayed signs of overfitting after the first epoch as the test accuracy started to decrease.

4 Innovation

In this section, we will discuss the innovation we did to enhance classification.

4.1 Ensemble Classification (stacked classifier)

As seen from the scores above, the classifiers individually were not able to classify the tweets very accurately. A stacked classifier combines the predictions of a few baseline models and a final meta classifier will learn to make a prediction based on the combined output of the baseline models. By combining the output of various models, the ensemble allows the full model to utilize the strengths of each individual model. Unlike bagging (like in random forest classifiers), not all the models need to be the same (i.e. decision trees).

Our stacked classifier will be implemented using the StackingClassifier from the scikit-learn library. The following 4 models will be used as the baseline classifiers: Decision Tree, XGBoost, Naive Bayes, Extra Trees, SVM. These 4 models are chosen because they provided the best results individually. For the final model, we opted for the logistic regression model, which is also implemented in the scikit-learn library. The role of the final classifier is to learn how best to combine the outputs from each of its baseline classifiers.

We observed about a 2% improvement in accuracy and f1-score from implementing this stacked classifier model.

4.1.1 Advantages and Disadvantages

Advantages

- Possible improvement in overall accuracy and performance
- Creates a more robust model since it combines the predictions of baseline models, able to filter out the pros of each classifier to make up for the cons in each classifier

Disadvantages

- Increased training time
- Generating predictions are more computationally expensive, may not be scalable

5 Frontend UI

Our frontend application is specially designed using Streamlit UI. The main functionalities are:

- Ability to input csv to test on our backend pretrained model, output predicted labels and accuracy of our model in prediction
- Interactive portion where users can ask a question or post sentiments to related hashtags on Apple Event 2022, and we will predict his/her sentiments in the query and produce a word cloud of similar sentiments and hashtags found in our test dataset.

6 Discussion

In this section, we will discuss takeaways from our overall experiments.

6.1 Error Analysis

Casefolding

As mentioned in Chapter 2.2 Data Pre-processing using Regex in lowercasing of sentence, the use of casefolding might result in inaccuracies when we determine tweet's sentiments due to ambiguity of language. This results in potential irrelevant tweets being wrongly involved in our test dataset, which will then not value-add to gauging the overall sentiments of our chosen topic.

Non-text elements and stopwords contributing to sentiments

The techniques used in Chapter 2.2 Data Pre-processing using Regex largely involved removal of non-text descriptors. However, some non-text descriptions may contain information that adds to sentiments of a particular tweet. For example, a tweet might be "Wow, I really look forward to receiving my #iPhone14 today!" with an image of an extracted kidney. Clearly, we are not able to detect the sarcasm intended by the tweet creator purely just through analysis of text sentiments. This depicts a loophole in our sentiment analysis approach, since we are only considering texts and not non-text elements in tweets.

As discussed earlier in chapter 3.3 Text normalization, we found out that stop words in our crawled tweets do play a part in the sentiment by providing important contexts. Therefore we have removed the stopword removal step in our data processing process.

Perspective bias in labeling dataset

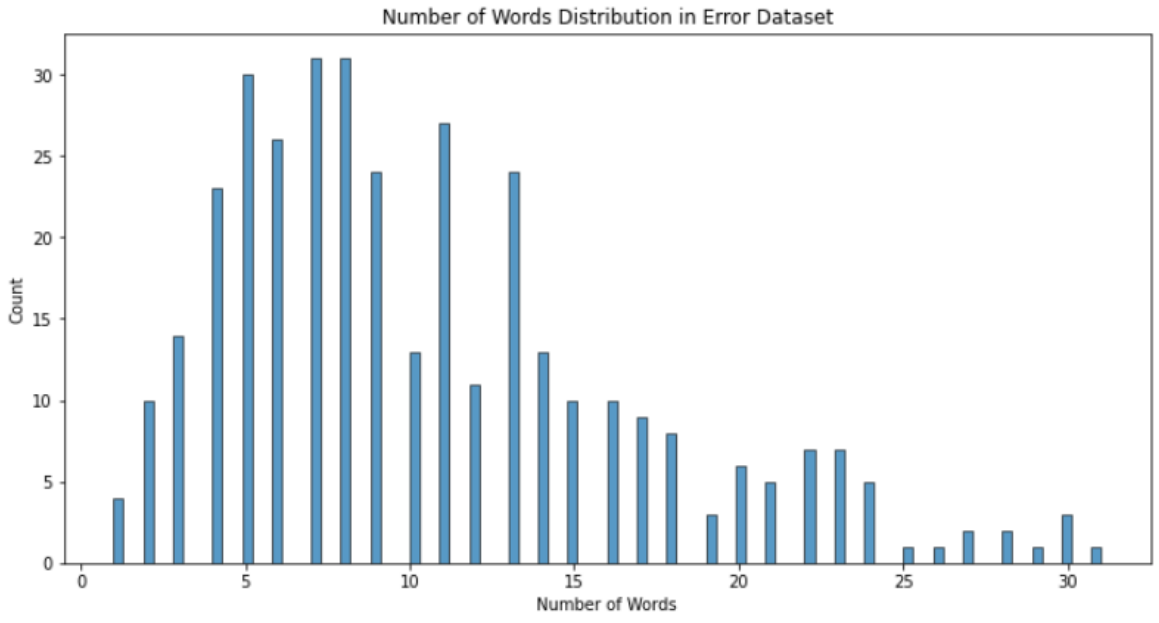
Since the dataset used in this project is manually labeled by ourselves, we realized

that some of the samples could be labeled differently when we conduct error analysis. This could be due to the fact that labeling sentiments of a text manually is tedious and sometimes subjective in interpretation, which resulted in some human errors despite multiple annotators agreement. The table below showed some samples that were recorded differently.

Samples	Labeled sentiment	Sentiment it should be
testing the apple world applewatchseries	Positive	Neutral
apple to make iphone in india	Positive	Neutral
seeing that new iphone commercial really turned my head	Negative	Positive

Error by Removal of Stopwords

After running the models using the datasets with the stopwords removed, we realized that the performance of the models were not that good. The initial experiment shows that tweets with length 8, 13, 16 form the majority of errors.



We then consider the removal of stopwords after the tokenization step. After inspecting the tweets that we have crawled, we realized that the stopwords that were removed actually provide important context and meaning to the tweets. Some examples are:

566	what		NEGATIVE	NEUTRAL	True
615	end of discussion	end discussion	NEGATIVE	NEUTRAL	True
619	the biggest feature of iphonepromax is that it...	biggest feature iphonepromax budget	NEGATIVE	POSITIVE	True
646	iphone se on ios nah	iphone se ios nah	NEGATIVE	POSITIVE	True
675	it wears her out	wears	NEGATIVE	NEUTRAL	True

Removing stopwords in this example changed the sentiment of the text from negative to neutral. Therefore, we experimented to selectively remove stopwords, and totally not removing stopwords in the normalization process in multiple experiments. After which, we observed slight improvements in results across the models.

The results are as follows:

	svm	extree	xgboost	nb	logreg	randforest	dectree	knn
accuracy	68.833333	68.416667	67.333333	66.166667	63.666667	63.250000	58.166667	58.250000
f1 score	68.798921	68.274057	67.278216	66.132103	63.641661	63.185838	57.979595	56.679634

Without Stopwords

	extree	logreg	svm	xgboost	nb	randforest	dectree	knn
accuracy	69.750000	69.000000	68.250000	67.333333	67.250000	61.833333	57.416667	58.00000
f1 score	69.617285	68.935094	68.117936	67.275189	67.104054	61.629309	57.045919	56.47433

Stopword removal for tweets length > 8

	extree	svm	xgboost	logreg	nb	randforest	knn	dectree
accuracy	69.750000	69.166667	67.833333	67.583333	67.500000	63.833333	58.083333	56.333333
f1 score	69.625499	69.035981	67.787145	67.449807	67.353621	63.781453	56.581430	56.261545

Stopword removal for tweets length > 13

	svm	logreg	extree	xgboost	nb	randforest	dectree	knn
accuracy	69.666667	69.166667	68.833333	68.333333	66.583333	62.916667	58.25000	59.166667
f1 score	69.549540	69.048097	68.696350	68.279377	66.542296	62.797846	58.23008	57.803304

Stopword removal for tweets length > 16

	svm	extree	xgboost	nb	logreg	randforest	knn	dectree
accuracy	69.916667	69.000000	68.666667	68.250000	67.083333	62.750000	61.166667	57.166667
f1 score	69.819264	68.822826	68.617442	68.150088	67.047627	62.530615	60.028851	57.144635

With Stopwords

6.2 System Considerations

Use Cases of Different Models for Scalability

Deep learning techniques provide better results than traditional methods for NLP tasks including sentiment analysis, however, they tend to be slower and more expensive to learn and use.

Large dataset provides the foreground for deep learning models to generate better results. On the other hand, traditional machine learning methods might just be enough for small datasets.

6.3 Room for Improvements

Consider emojis from the tweets in sentiment classification

Since emojis are commonly used in twitter, it may provide some features in determining the overall sentiment of a tweet. One possible improvement is to include the emojis by representing the emojis by their unicode into our models.

7 Conclusion

Through this project, we have crawled tweets regarding the Apple Event 2022 and used the text data to conduct sentiment analysis, in order to gain insights on the sentiment people have on Apple as a company. Our group has tried various methods to classify the sentiments, primarily through machine learning and deep learning methods.

Through this project, we have learnt the importance of having enough data that are labeled accurately for the sentiment classification task. Although the current state of the art classification approaches are powerful, they would not deliver good results without enough data to train the models.

Another important takeaway from this project is that each text classification task is different and the most common practices may not necessarily apply to every task. One example from this project would be the removal of stopwords. Although removing stopwords can help models focus more on important information, it can also change the meaning of certain texts by affecting the context, which is especially important for sentiment analysis. Therefore there is a need to consider the context and the data we are working with before deciding on the actions to take for the task, instead of strictly adhering to common practices.

8 Reference

1. Rafalson, B. (2017, September 26). *Event Marketing 2018: Benchmarks and Trends*. Bizzabo. https://www.bizzabo.com/blog/event-marketing-2018-benchmarks-and-trends/
2. Jay, A. (2020, January 29). *How many smartphone users are there in the world? There are over 5.22 billion smartphone users in the world, r*. Financesonline.com; FinancesOnline.com. https://financesonline.com/number-of-smartphone-users-worldwide/#:~:text=Smartphone%20Operating%20System%20Preference%3A%20Android%20vs%20iOS,-Android%20democratized%20smartphone&text=Various%20sources%20put%20the%20total,against%20900%20million%20active%20iPhones
3. May, C. (2018, December 16). *What Makes Events Like Apple's So Effective?* Entrepreneur; Entrepreneur. https://www.entrepreneur.com/growing-a-business/what-makes-events-like-apples-so-effective/324779
4. Benson, H. (2021, May 8). *4 Benefits of Twitter Sentiment Analysis for Your Business | Scraping Robot*. Scraping Robot. https://scrapingrobot.com/blog/twitter-sentiment-analysis/
5. *Getting access to the Twitter API*. (2022). Twitter.com. https://developer.twitter.com/en/docs/twitter-api/getting-started/getting-access-to-the-twitter-api