Twitter Sentiment Analysis

Youssef Agiza*
youssefagiza@aucegypt.edu
The American University in Cairo

John El Gallab
john.elgallab@aucegypt.edu
The American University in Cairo

1 Introduction

Social media has profoundly advanced in the past 20 years more than any technological product has ever progressed. A vast majority of humans, nowadays, use social media on a daily basis as their primary source of interaction with other community members, news articles, and even the world. Through these interactions, social media users post their updates, react to other people's updates and interact with what other users share in a way that mimics the human natural face to face daily interactions. So, to understand what a human is trying to express through social media, the sentiment of what they post must be studied. This has only been made possible through machine learning techniques, if one truly aspires to study a large number of human expressions. The manual evaluation of all data generated through social media platforms is not feasible as it requires huge amounts of computations that no number of humans can readily do.

Recently, machine learning approaches have been classifying the text-based human expression on social media into either positive or negative expressions. This automated self-sustaining tool has a multitude of use-cases in the sociology, marketing, and analytics fields.

To define our problem, we aim to use a machine learning-based approach to classify short English texts into either associated with positive sentiment, neutral sentiment, or negative sentiment, Typically these texts are below 140 characters and in form of tweets. We shall proceed in this paper by discussing machine learning-based solutions to tackle the problem of sentiment analysis of human-written English texts from the previous literature.

2 Literature Review

According to Taboada et al. (2011), there exists two main techniques of tackling our defined problem. The first is the Lexicon-based approach in which the sentiment orientation is analysed based on certain words present in the text under investigation. The other approach utilizes machine learning to compare the present text to ones that are already analyzed and labeled according to their sentiment to figure out the sentiment of the newly investigated text.

2.1 Lexicon Approach

The Lexicon approach uses dictionaries of words that are manually identified according to sentiment (positive, negative or neutral) known as seed words that act as indicators of the sentiment of the text [**?**][**?**][**?**][**?**]. These seed words can be later used to expand the dictionary of words used. This approach may be suitable for more general purpose texts but yields low accuracy metrics. A better approach that yields higher accuracy metrics is the machine learning-based approach.

2.2 Machine Learning-Based Approach

According to Aue and Gamon (2005), the machine learning-approach is domain-specific; therefore, will yield much lower accuracy metrics if utilized in a domain other than that used for training the model. This approach uses parts of the data that is present and already classified into either positive, negative or neutral sentiment and then trains classifiers to learn from these examples without relying on any pre-existing lexicon (Dhaoui et al., 2017). Some utilised machine-learning techniques include but are not limited to Naive Bayes, maximum entropy, or support vector machines. Moreover, the sampling of the data set heavily impacts the accuracy of the classification using this approach. According to Babacar et al. (2021), integrating lexical dictionaries into the machine learning-based approach yielded an increase in accuracy metrics of the model when coupled with a linear regression-long short term memory model.

2.3 Machine Learning Approaches Comparison

Different machine learning approaches were used to tackle the problem of sentiment analysis of texts. A recent publication worked on a comparison of different machine learning models performance on sentiment analysis of tweets by Rustam et al. (2021). Multiple different classifiers were used and tested on both short texts and long texts. According to Rustam et al., both Naive Bayes and Logistic Regression model gave a performance of 91 percent and 74 percent accuracy metrics respectively for short texts but didn't behave in a similar manner when it came to longer texts. Natural Language Processing was another approach that is commonly discussed in literature to tackle the problem of sentiment analysis. This classification is done through a Long Short Term Memory Recurrent Neural Network. This technique yields 81 percent accuracy metric but was more successful on longer texts than the Logistic Regression model.

Moreover, Boiy discussed the usage of a Support Vector Machine (SVM) which was found to be robust if many features happen to be in the dataset and if the data was noisy

*Both authors contributed equally to this research.

using a Weka implementation. According to the same reference, the following table shows possible sources of error according to different languages.

Id	Cause	English	Dutch	French	All
1	Features insufficiently known and/or wrong feature connotations	23	21	15	59
2	Ambiguous examples	12	8	8	28
3	Sentiment towards (sub-)entity	3	3	9	15
4	Cases not handled by negation	3	3	4	10
5	Expressions spanning several words	3	5	2	10
6	Understanding of the context or world knowledge is needed	2	2	4	8
7	Domain specific	0	3	3	6
8	Language collocations	2	2	2	6
9	A sentiment feature has multiple meanings	2	1	2	5
10	Language specific	0	2	1	3

Figure 1. Error causes and different reasons for different languages according to Boiy et al.

Another paper by Machova et al., discussed the following process of classification of tweets to detect political bias in the tweets[?].

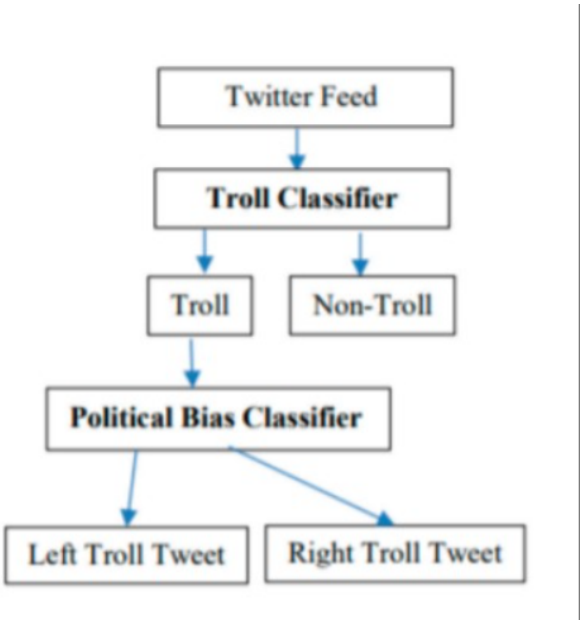

Figure 2. Machova et al. process of classification of tweets

This process used two classifiers as shown, the Troll Classifier and the Political Bias Classifier as described in the paper. The following metrics were obtained. Support Vector Machines obtained 84 percent accuracy, Convolutional Neural Networks achieved 74 percent accuracy, and Bidirectional Encoder Representation for Transformers model (BERT) 99 percent troll/non-troll classification.

The following metric measures were obtained with their respective models when using the SVMs for online discussions (non-tweets).

Measures	Linear SVM	SVM + GKRBF	SVM + SKF
Accuracy	0.738	0.974	0.727
F1 score	0.762	0.978	0.769
Precision	0.697	0.973	0.780
Recall	0.839	0.983	0.758
Specificity	0.839	0.960	0.663
Matthews Correlation Coefficient	0.485	0.945	0.429

Figure 3. Comparison of SVMs performance using Weka Implementation

3 Datasets Review

This section includes some of the datasets we reviewed while researching for this project.

3.1 Yelp Review Sentiment Dataset

Yelp is well-known website and mobile application that posts crowd-sourced reviews on various businesses. The Yelp Review Sentiment Dataset is a polarity dataset that is publicly available through Kaggle website. The set is constructed from the Yelp reviews dataset which contains reviews from Yelp. It was used as a text classification benchmark in a paper published by Xiang Zhang, Junbo Zhao, Yann LeCun. It is built by considering 2 or less stars as negative and 3 or more stars as positive. This set is one of the large dataset that we found in sentiment analyses, achieving a total of 560,000 training samples and 38,000 testing samples. The data is divided into two classes: positive polarity and negative polarity which are represented by 2 and 1 respectively.
Dataset link

3.2 Amazon Reviews Dataset

This dataset is the largest sentiment analysis dataset found in our research, containing a few million sample points. It contains customer reviews from Amazon.com, the well-known online shopping website, in the form of text and labeled using the star ratings. The dataset is meant to be used to apply fastText, a library that uses Natural Language Processing(NLP) for efficient learning of word representations and sentence classification.
Dataset link

3.3 Sentiment140 dataset

Unlike many other datasets, this dataset was constructed automatically using Twitter Search API without manual annotation. The authors used the API to search for tweets using keywords and classify them. In the classification, they assumed that tweets with positive emoticon are positive and one with negative emoticons are negative. Sentiment104 is a huge dataset that has 1,600,000, providing the following features: *target, ids, date, flag, user, text. Target* is the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive). *Ids* is the id of the tweet and *date* is the date of the tweet. *Flag* is the query used to find the tweet using Twitter API. *User* is

the user who wrote the tweet.
Dataset link

3.4 Twitter Data set for Arabic Sentiment Analysis

This dataset is one of the unique datasets we found as it focuses on Arabic tweets. By using a tweet crawler, the researcher collected 2000 labelled tweets (1000 positive tweets and 1000 negative ones) on various topics such as: politics and arts. These tweets include opinions written in both Modern Standard Arabic (MSA) and the Jordanian dialect. The model is supposed to be used to determine the feelings of the user who wrote the tweet where the feelings are classified into positive or negative. However, this dataset is one of the smallest datasets we came across which disqualifies it from our selection.
Dataset link

4 Project Overview and Solution

As previously mentioned, social media is an essential means of communication that millions are using around the world. Being such an important communication method, it is critical to be able to determine the sentiment communicated through a message, a post, or a tweet. Accordingly, we are aiming in this project to utilize the power of machine learning to achieve such purpose. After extensive research, we settled on using the Sentiment104 data sets for many profound reasons. First and foremost, it offers one of the largest data samples found in this area of sentiment analysis which shall be used to train the model rigorously and effectively. Furthermore, the data was collected and labeled using an automated process utilizing the Twitter Search API. We expect this approach to reduce the margin of human error since no annotation or other forms of intervention were needed. Another reasons for choosing this set is that it focuses on Twitter which is widely-used platform in the current era on which people express their ideas and emotions. In turn, we hope that this project will have the potential to be extended and used in real-life by users and not for the sake of researching only.

5 Experimental Analysis

In this phase we performed a pilot study on different machine learning models to decide the most suitable one for our use case. The chosen models are: Decision Tree, Neural Networks, Support Vector Machine, Naive Bayes, and Logistic Regression. Each of those models is briefly described in the next section along with some of its relevant pros and cons.

5.1 Data Representations

We represented the data representation in three different ways and gave each representation to all of the model to compare their performance. The representations are:

1. **TF-IDF**

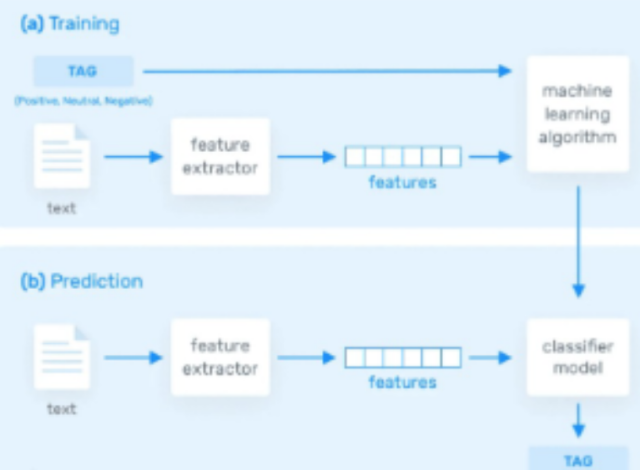

Figure 4. Training a prediction model that will be used for sentiment analysis[**?**].

2. **Word Embedding**: each word in the sentence is represented by a word embedding vector. All the vectors are then averaged to get one vector representing the whole sentence.
3. **Weighted Word Embedding**: for each word in a sentence, the word embedding vector is multiplied by its frequency in the sentence to get a weighted vector. The vectors of all the words in the sentence are then averaged to give one vector for the whole sentence.

For the word embedding we used the pre-trained embedding provided by the AraVec open source project. Yet, the word embedding provided was not stemmed as the dataset we got from the preprocessing. As a result, we used the raw dataset whenever the word embedding representation was involved in our data. Additionally, we tried normal frequency encoding but TF-IDF gave better results in all the metrics so it was omitted and replaced with TF-IDF. We also attempted doing weighted word embedding using TF-IDF instead of word frequency however, multiplying the TF-IDF vector with the embedding vector didn't work for unknown reasons. One possible reasons may be the difference in Arabic encoding between the two vectors. Another reason may be how sklearn calculates the TF-IDF since we used their library to perform this step.

6 Models Performance

6.1 Decision Tree

The Decision Trees model is suitable for categorical data. This models was not expected to be the best models compared to other models since it is depends on the effect of splitting the data using the features. In our case, the features are the words themselves, which results in high dimensionality of the data. Furthermore, it tries to split the data using the features(i.e. the words) which may result in losing the

value of the context in which the words are present.

TF-IDF Encoding. With TF-IDF encoding, Decision Tree yielded an accuracy of 53% which is the highest compared to the other data representation for this model. This result is expected since in TF-IDF the data is cleaned an prepossessed, so we have smaller number of features resulting in more accurate splits.

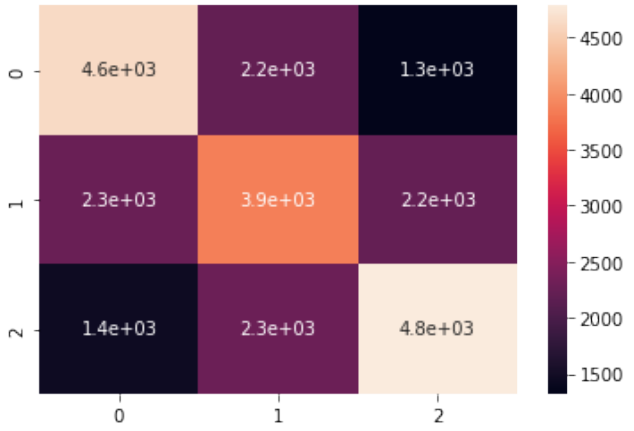

Figure 5. TF-IDF Decision Tree Heat Map

Normal Word Embedding. For the normal word embedeing, the accuracy of this model is 35% which is very low. However, this is not surprising since we are not expecting this model to be suitable.

Weighted Word Embedding. Finally, the accuracy after applying the weights didn't change, giving 35%.

As expected, Decision Tree results are not promising for this use case. One reason for that may be that the number

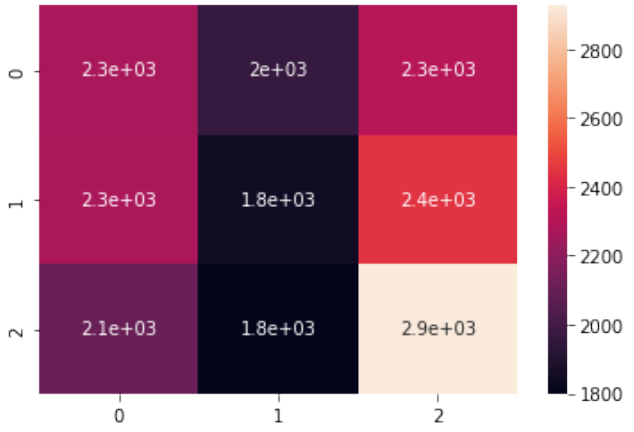

Figure 6. Normal Word Embedding Decision Tree Heat Map

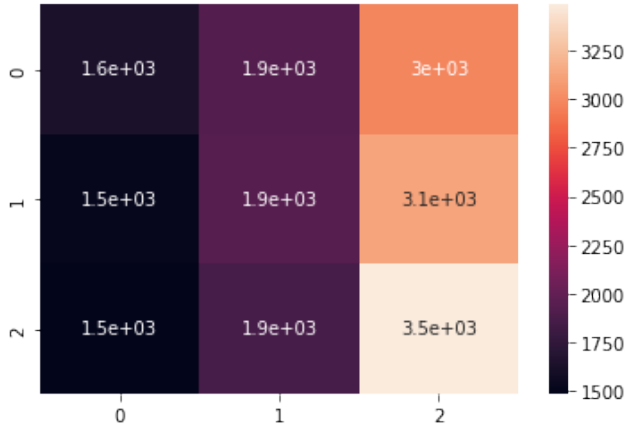

Figure 7. Weighted Word Embedding Decision Tree Heat Map

of features compared to the value gained for each one is not high.

6.2 Neural Networks

Neural Networks(NN) models are based on having several layers of nodes connected together with weights to yield an output in the last layer. NN perform feature engineering automatically as the model learns the optimal weights on its own using an error function that needs to be minimized. In this project, NN are expected to be one of the most suitable models since it performs feature engineering to the word and find the right configuration to determine the sentiment of the text. We applied NN using sklearn's Multilayer perceptron classifier (MLPClassifier).

TF-IDF Encoding. As expected, NN yielded an accuracy of 64% which is the highest in this pilot study. The heatmap for the result is shown here.

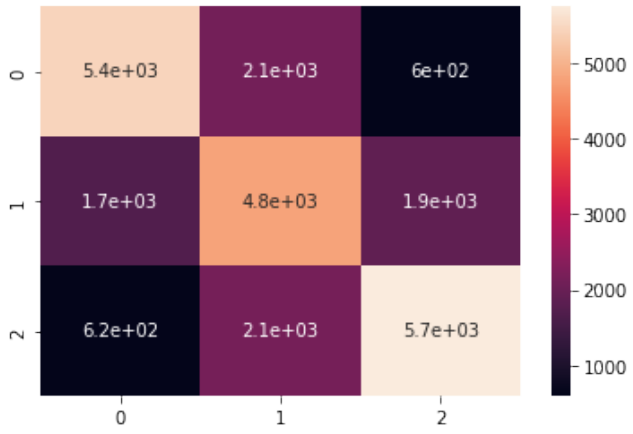

Figure 8. TF-IDF Neural Network Heat Map

Normal Word Embedding. The results here were surprising as the accuracy is 33% which is very low especially considering that we are using Neural Networks. Further more, if we inspect the heat map, we can see that the precision, recall, and f1-score are zero for two of the three classes. The only reason we found for these results are that we are treating all the words equally(i.e. unweighted). Accordingly, we tried the third approach: the weighted word embedding.

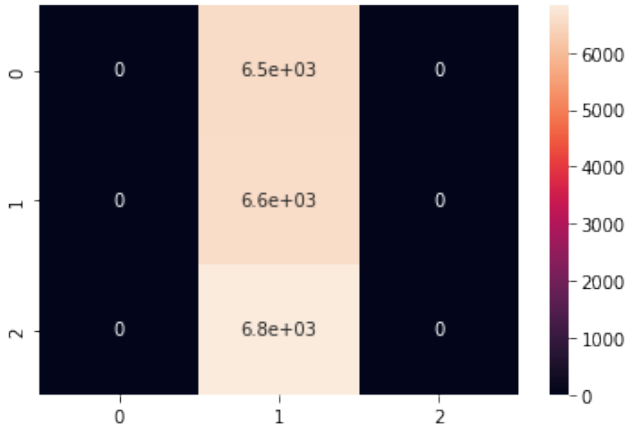

Figure 9. Word Embedding Neural Network Heat Map

Weighted Word Embedding. After factoring the weight of the word in the equation by multiplying its frequency with the embedding vector, the accuracy increased to 36% only which is still significantly low. The only reason we could infer for these results was a problem with the embedding itself or the effect of not stemming the dataset.

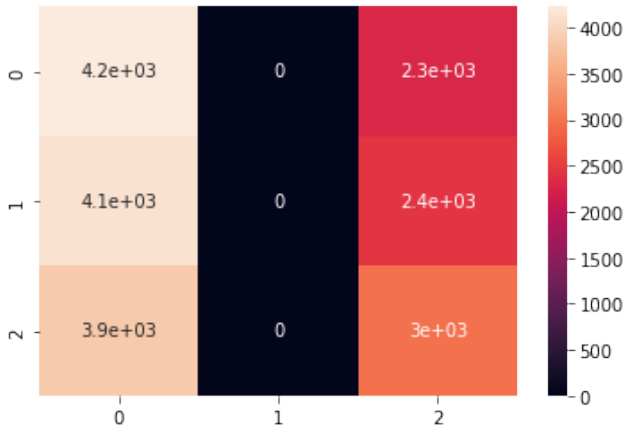

Figure 10. Weighted Word Embedding Neural Network Heat Map

6.3 Support Vector Machine

Support Vector Machine(SVM) is a linear model that tries to separate the data into classes using a line or a hyperplane. It can solve linear and non-linear problems and work well for many practical problems.

TF-IDF. The accuracy of the model is 60% which is acceptable compared to the other models, yet there are better models.

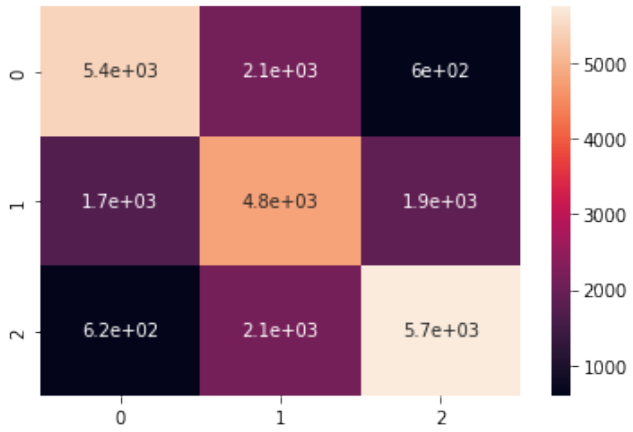

Figure 11. SVM Heat Map

Normal Word Embedding. Similar to the other models, the accuracy dropped to 34% on applying word embedding. The heat ma

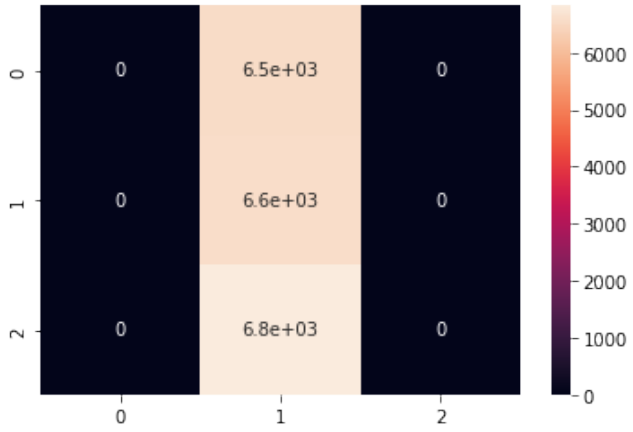

Figure 12. Word Embedding SVM Heat Map

Weighted Word Embedding. The accuracy of the model slightly decreased to 33% percent. Overall, the model accuracy dropped when we used the word embedding.

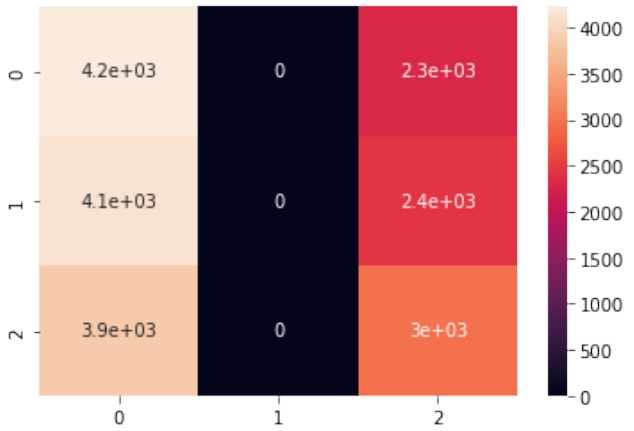

Figure 13. Weighted Word Embedding SVM Heat Map

6.4 Naive Bayes

Naive Bayes is a probabilistic model with an underlying assumption that the features are independent from each other.

TF-IDF. The accuracy of the model is 61% which is among the best we found so far. The heatmap is shown in the figure below.

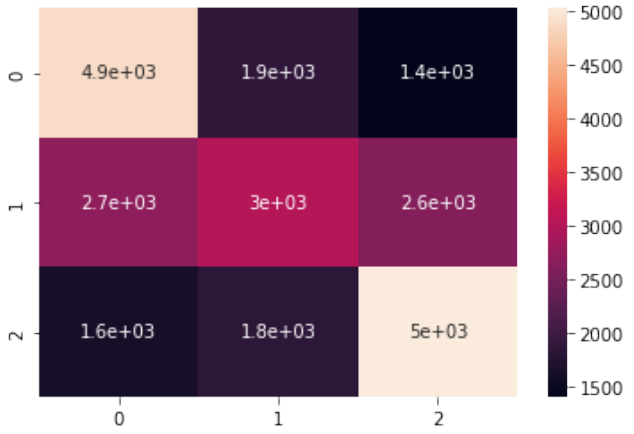

Figure 14. TF-IDF Naive Bayes Heat Map

Word Embedding. The accuracy dropped to 35% similar to the other models. The heat map is given.

Weighted Word Embedding

Again, the accuracy dropped on applying the weighted embedding to 33%. The heat map is given below.

6.5 Logistic Regression

Logistic Regression is a classification algorithm used to predict the probability of a target variable. Yet, Logistic Regression doesn't perform assumption on the data distribution

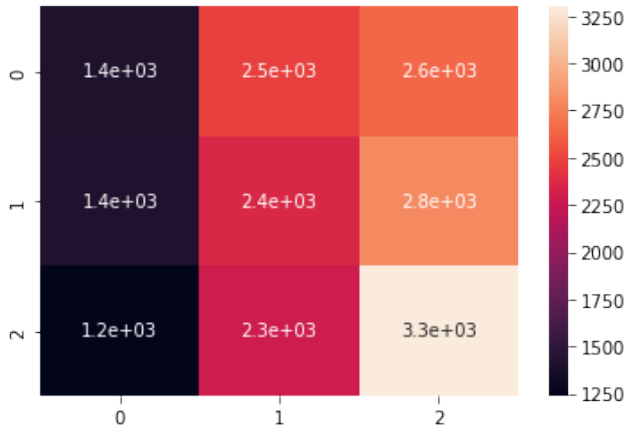

Figure 15. Word Embedding Naive Bayes Heat Map

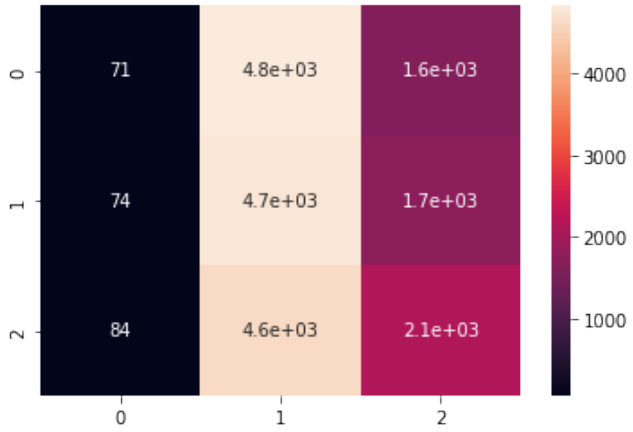

Figure 16. Weighted Word Embedding Naive Bayes Heat Map

unlike Naive Bayes which makes it more generic. Also, One of the main strengths of Logistic Regression is its simplicity. The results of this model were among the best found. One reason for such results may be that the target has a linear correlation with the features which makes since in our case. For example, more positive words are likely to mean positive sentiment.

TF-IDF. The accuracy of the model is 64% which is the best accuracy found so far. This accuracy was achieved by the Logistic Regression Model and Neural Network Model only.

Normal Word Embedding. The accuracy of this model was 35% which was surprising considering the accuracy of this model with TF-IDF.

Weighted Word Embedding. Unlike most of the other models, Logistic Regression accuracy slightly increased to 36% on applying the weights.

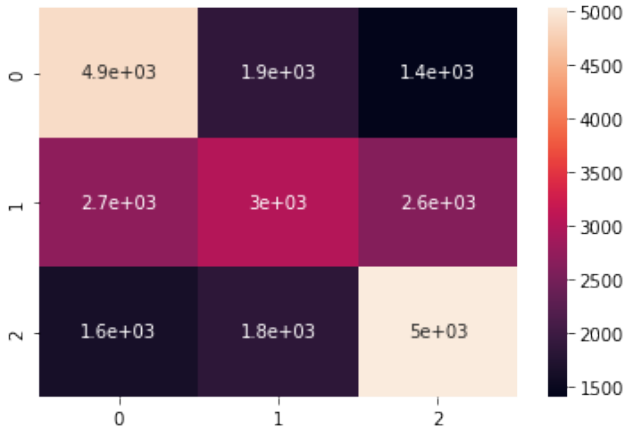

Figure 17. TF-IDF Logistic Regression Heat Map

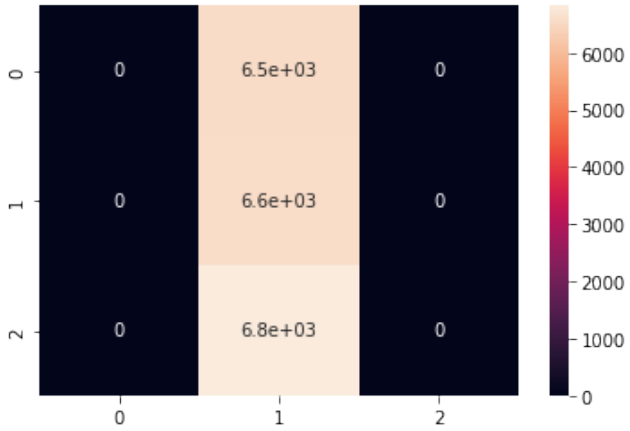

Figure 18. Word Embedding Logistic Regression Heat Map

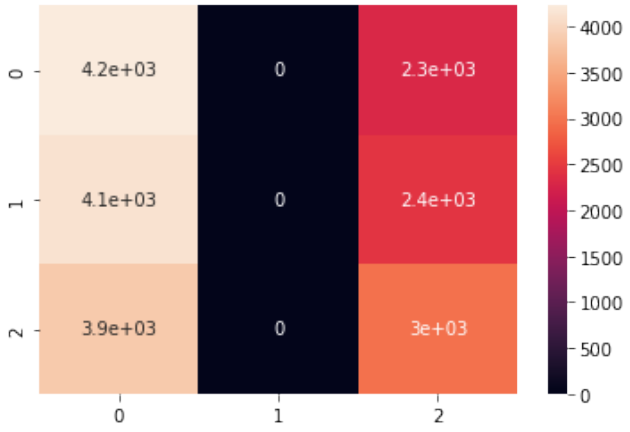

Figure 19. Weighted word embedding Logistic Regression Heat Map

7 Comparison

We can notice a general behavior of the results as the TF-IDF gave the best values in all the metrics. the issue with embedding is not because a normalization for the embedding vectors is needed because we tried normalization and it yielded the same low accuracies. This may either imply that the word embedding approach is not suitable and shall be omitted or that the pre-trained embedding used were not suitable. We assumed that the later reason is more likely to happen since the literature generally shows that word embedding should improve the accuracy.

As already explained, the top three performances are given by the Neural Network, Logistic Regression, and the Naive Bayes.
One advantage that exists in Naive Bayes is being simple and memory efficient. Naive Bayes's assumption of independence between the features makes it computationally fast. However, this assumption may not be valid for our data since different words depend on other words. Similarly, the logistic regression is simple and fast to train. It is also good for generalization since it has no assumptions about the distribution of the class. However, it still assumes a linear correlation between the features and the target which may not always be true. Finally, the Neural Network is the most computationally intensive approach. It requires more memory and time to be trained when compared to the other two models. It may also result in overfitting since the layers may act as memories for the dataset. We avoided this issue by doing early stopping in the training using the 'early_stopping' parameter. Furthermore, we can include word embedding in neural network as part of the training which may solve the issue of the unsuitable embedding mentioned eariler.

8 The Model of Choice

After extensive analysis, the model we decided on choosing is the Neural Network due to several reasons:

1. Neural Networks are flexible as we can configure the layers as we wish to aim for better results.
2. The nature of this model automatically performs features engineering to find the best results. Such approach is useful when trying to predict something as dynamic as sentiment using the text because the natural language usage is not rigorously defined.
3. Word embedding can be integrated with Neural Networks which we expect to improve the results in the following phase.

After the performing a pilot study on several models with different prepossessing operations, we proceeded to design a model for this project. However, the models achieved a maximum accuracy of 64% with some models not predicting

full classes as shown in the heat maps. Accordingly, we decided to further investigate the dataset itself in order to find reasons for such result. After some manual investigation, we found that the neutral class(i.e. the class with label '0') had false labels. For example, some text was positive, yet labeled as neutral. As a result we decided to experiment with the dataset after dropping the data labeled as neutral. This decision doesn't affect the problem addresses in our project since our main goal is predicting the positive and negative sentiment of the text, while the neutral sentiment didn't represent an important result.
Surprisingly, the scoring of all the models significantly improved after dropping the neutral labels which encouraged us to completely disregard it and re-perform a pilot study on different models and compare their results. However, for the sake of time and unlike the previous phase, we tested several models only using TF-IDF since it showed highly promising results as explained in the following section.

9 Models Comparison: Updated

In this section we provide a brief comparison between the different models we tried after the change mentioned before.

9.1 Decision Tree & Naive Bayes

Decision Tree scores increased reaching 74% for accuracy, recall, and f1-score of both classes while the precision scored 73% and 75% for the negative class and positive class, respectively. Similarly, Naive Bayes improved reaching approximately 73% for the accuracy, recall, precision, and f1-score for both the positive and negative class. Despite this improvement, these two models have the lowest scores compared to the other models and they are the least suitable for this project. Thus, we mainly consider the next three models.

9.2 Logistic Regression

The values for all the metrics(recall, precision, f1-score, accuracy) of the logistic regression were approximately equal across all the test, so we will be referencing only the accuracy in this section since they all have the same value.
We tested two different solvers, *saga* and *newton-cg*, and they both had equivalent results. Without cross validation, the model scored an accuracy of 84%, while it decreased to 83% on using a 10-fold cross validation.

9.3 Support Vector Machine(SVM)

SVM model improved as well, reaching an accuracy of 81% and almost the same value for f1-score. For the negative class, SVM scored 81% for the precision, and 79% for the recall. For the positive class, it scored 80% and 82% for the precision and recall, respectively. On applying a 10-fold cross validation, all the scores dropped to an average of 80%.

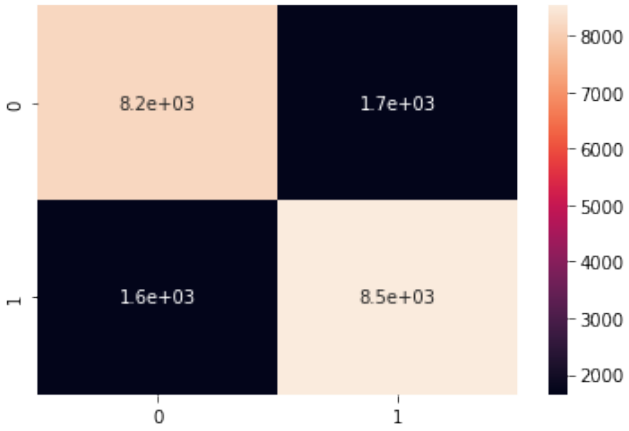

Figure 20. TF-IDF LR Newton solver Heat Map

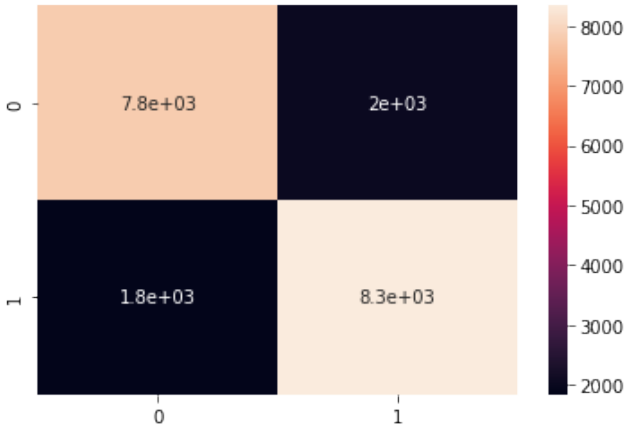

Figure 21. TF-IDF SVM Heat Map

9.4 Neural Network

Finally, the neural network model scores improved reaching an accuracy of 84% without cross validation. The negative class has a recall of 86%, a precision of 82%, and an f1-score of 84%. While the positive class had a precision of 83%, a recall of 87%, and an f1-score of 85%. On applying cross validation, all the scores slightly decreased to approximately 83.7% on average. Thus, this model shows the best values compared to all the other models.

9.5 Chosen Model

According to the metrics and results shown, the neural network has the best values across all the metrics. According to other literature, it is one of the most suitable model regarding the nature of the problem under consideration(i.e. sentiment analysis). It is also suitable for this project due the reasons mentioned in the previous phase, such as having no assumptions regarding the data and auto-generating features. Due to the mentioned reasons, we decided to proceed with this

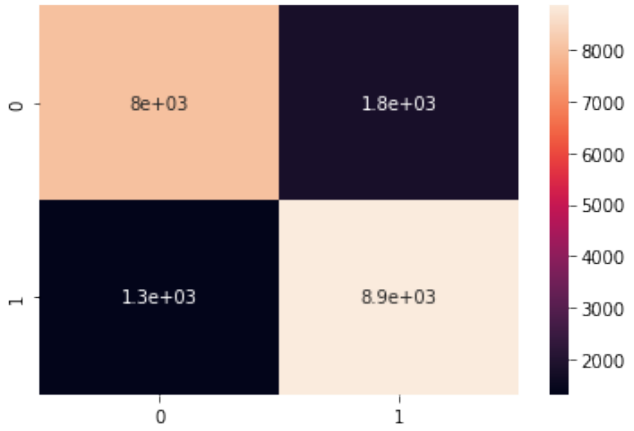

Figure 22. TF-IDF Neural Network Heat Map

model for the upcoming phases; and so, we provide a design for the model that we will implement in the next section.

10 Model Design

To decide on the best parameters for our model, we ran grid search CV to compare the results of applying different parameters.

10.1 Hidden layers

Deciding on the number of hidden layers, we tried several values ranging from 2 layers and up to 16 and 32 layers. We found that increasing the number of layers to large numbers didn't result in a significant difference. Yet, a large number of layers is significantly more computationally intensive. Accordingly, we settled on a 3 hidden layers where each of them has 5 nodes. This configuration gave an accuracy of 84% and all the other metrics ranged from 83% to 86%.

10.2 Solvers (Optimizers)

After choosing the number of the hidden layers, we moved on to choose the solver that optimizes the weights. We tested *sgd*, *lbfgs*, and *adam*, each using 10-fold cross validation. *sgd* gave the best results of an average of 83.7% for all the metrics.

10.3 Activation function

We then continued to decided on the activation function. Here, we tried RELU, identity, logistic, and tanh functions. The worst results came from the logistic function which gave approximately 50% for the accuracy and precision, and 30% for the recall. The best results were achieved using the tanh and identity giving an average across all the metrics of 83.9% and 83.7%, respectively. Thus, we decided to proceed with the identity function since it is simpler and the difference is insignificant. For the output layer, we may use sign function to give as -1 and 1 for the output which matches the labels we are using.

10.4 Loss Function

The loss function for sklearn model supports only the Cross-Entropy loss function. Thus, we only experienced with this function at this stage. Thus, we didn't fully eliminate other loss functions. We are planning to research and experiment more with different loss function in the upcoming phase to decide the best loss function.

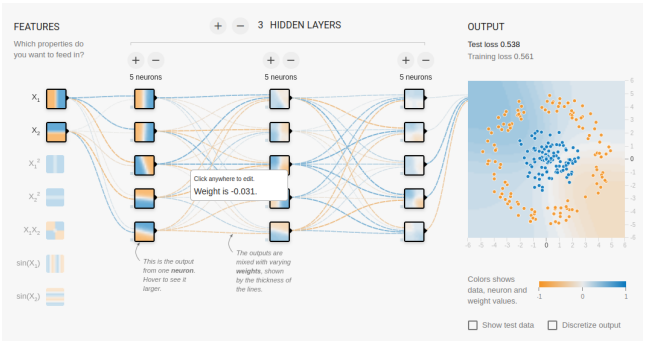

Figure 23. Neural Network diagram

11 Application

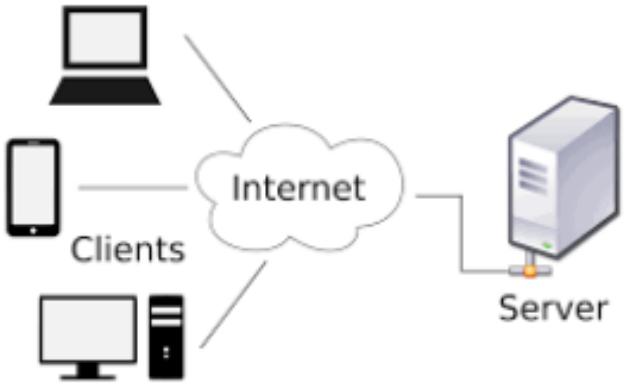

Figure 24. Client Server architecture diagram

The application is supposed to be user-friendly, where the users inputs the features and the application outputs the prediction. In our case, the user shall enter a text through a text field. The text is then sent to the server which will apply the same pre-processing pipeline as that we applied on our data before running the algorithm to give predict the sentiment of the text. The prediction is then sent back to the user through the application. The server will have the neural network weights stored in the database and all the needed data in order to perform the prediction. We can also improve the model by asking the user for their feed-back and use the data to train the model to achieve better

results later. The architecture we are going to implement is a client-server architecture. The application is planned to be a web application, using either Django for the backend development.

12 Final Implementation

We have decided to implement a fully-connected neural network with 5 layer. All layers have relu as their activation function except the output layer which has a sigmoid activation function. An Adam optimizer was also implemented and initialized with parameters $\beta_1 = 0.8$, $\beta_2 = 0.999$ and $\alpha = 0.001$. Our loss function is binary cross entropy as it best suits our problems and yields the best outcomes.

We did not normalize the weights in the training process because of the vanishing gradient problem in deep learning. Our workaround was to increase the learning rate and normalized our value by dividing by the square root of n. A better solution approach would have been to implement a mini batch training but due to limitations it was not implemented.

The preprocessing components remain the same as what was implemented in the initial phases. We start by detecting the language and ensuring that it is Arabic. If the input is not in Arabic we reject it. We also strip the tashkeel, tatweel and harakat from the text. We then remove the stop words according to a corpus we generated from the original dataset. We then apply $TF - IDF$ to the dataframe. The result of the $TF - IDF$ is split into 80 percent for training of model and 20 percent for testing. Training then takes place.

After doing a literature review, we have decided to retrain the model using the following method:
We first append the new input along with the users suggested sentiment to the dataframe through the available *APIs*. We then retrain the whole model using the current dataframe (old original data and the newly appended data) after a certain specificed period of time. This method of retraining was preferred over the ensemble method as the newly inputted data is not large enough to properly train a new neural network. The ensemble method would have constituted of training a new neural network on the newly inputted data and used along with the old neural network to decide the output of the whole ensemble model