Arabic Tweets Emotion Recognition

Linah Hossam Toka Ossama Nerimane ElRefai Team 19

May 26, 2023

1 Motivation

NLP, or Natural Language Processing, is a field of artificial intelligence that focuses on the interaction between human language and computers. some NLP applications involve language translation, chatbots and virtual assistants, Information Retrieval, text summarization, and sentiment analysis which is the topic of our project. It involves analyzing text to determine whether it expresses a positive or negative sentiment. Understanding the underlying attitudes, feelings, and views stated in a text is the motivation behind sentiment analysis. The importance of sentiment analysis has developed as a tool for businesses and organizations to analyze consumer feedback, social media sentiment, and other types of data as the amount of data available on the internet keeps growing.

Sentiment analysis can offer insightful information about how people feel about a particular product, service, brand, or subject. Businesses can spot repeated patterns, sentiment trends, and potential problems or concerns by examining social media posts, customer reviews, and other types of data. Social media is one of the best places to collect useful data for sentiment analysis since the new generation is writing posts, tweets, and comments to express all types of emotions, which is why our proposal focuses on the idea of sentiment analysis of Arabic tweets to address any of the per-mentioned applications. This data is useful because it is realistic, and diverse since it comes from people of different backgrounds, genders, and age categories. As shown in Figure 1, our project aims to classify tweets as a number of positive emotions, negative emotions, and neutral ones which are referred to as none.

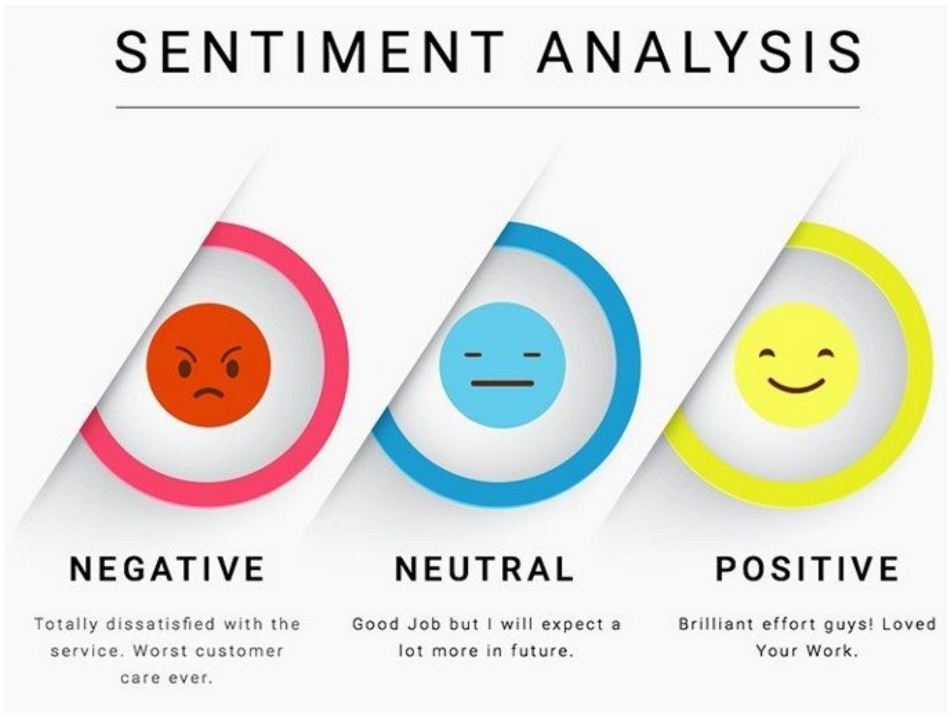

SENTIMENT ANALYSIS
NEGATIVE
Totally dissatisfied with the service. Worst customer care ever.
NEUTRAL
Good Job but I will expect a lot more in future.
POSITIVE
Brilliant effort guys! Loved Your Work.

Figure 1: Basic Sentiment Analysis Classes

2 Challenges

In this project, we face multiple challenges such as preprocessing the data, since we are using a dataset collected from Twitter social media platform as we will elaborate later in section 4, data samples are spontaneous and random sentences with lots of slang words and emojis. This sort of unstructured data will need much preprocessing to be useful. The second challenge is finding a suitable classification base model that we can modify and enhance its accuracy. The third challenge is dealing with Modern Standard Arabic (MSA) and the Egyptian dialect. These are the three main challenges among others.

Our plan of how to solve these challenges is discussed in the following sections where we discuss the dataset used, the analysis of the data, and its preprocessing in section 4. We present an overview of our system architecture and some initial details about it in section 6.

3 Related work

This section contains a discussion of the other approaches that might be used for the sentiment analysis problem which is the same problem we are addressing. According to [1], the general process flow of sentiment analysis includes collecting the dataset of tweets using one of the multiple common ways, following this comes the preprocessing stage which is a very essential step. Preprocessing of the data includes removing the username of the person who wrote the tweet, and the time the tweet was written among others. Another preprocessing that could be done to the data is tagging; that is marking unsentimental content in a tweet that does not have any impact on the tweet sentiment such as; tags, hashtags, and emojis. However, emojis could be useful for sentiment analysis in many cases, therefore, they are tagged with a word representing their meanings, for example, the emoji ':)' could be replaced with happy, and the emoji '<3' could be replaced with heart_emoji.

The next step of the sentiment analysis flow is the Filtering; in this step, we take the output of the preprocessing and filter any misspelled words, remove any repeated letters in a word that is coming from exaggeration or simply misspelling, and remove any stop words that will not help in classifying the sentiment class corresponding to the tweet. The final step before we start classifying the sample is normalization which includes standardization of the form of a certain letter used, removing punctuation, diacritics, elongation, and non-letters among other things.

Now that we have our data ready we can start our classification task. We need to start by annotation these tweets, which is done manually and the samples are classified into negative, positive, or neutral or maybe even further classification into subclasses of positive and negative sentiments. There are two types of classification; supervised and unsupervised. In this project, we mainly address supervised or corpus-based classification. In supervised classification machine learning classifiers are used such as Support Vector Machine (SVM), Naïve Bayes (NB), Decision Tree (D-Tree), K-Nearest Neighbor (KNN). Machine Learning models generally need to be trained using a training dataset that covers all the labels that the models need to learn, the model is then evaluated over a testing dataset. The second type of classification is using unsupervised algorithms which is also called lexicon-based, in these models there is no training as the classifier will classify a dataset directly using a dictionary/lexicon of words. Each word will have a certain polarity and based on that the classification takes place by calculating the overall polarity of a certain sentence.

The work in [2] used the supervised learning methods; naive Bayes and Support Vector Machine (SVM) to perform sentiment analysis on a dataset collected using Twitter API. This classification was performed using a feature vector extracted using both unigram and bigram. As for the authors of [3], they also used Support Vector Machine (SVM) to perform Arabic sentiment analysis but on a dataset of 3015 Arabic tweets from TAGREED corpus. The work in [4] discussed the key issues in conducting Sentiment Analysis on Arabic social media text by comparing the results of 5 different supervised classifiers; Naive Bayes, Support Vector Machine (SVM), Maximum Entropy, Bayes Net, and J48 decision tree. These five algorithms were trained and tested over a dataset of Arabic tweets collected with the help of Twitter filter stream API.

Another example is the work in [5] where the Kuwaiti- Dialect was explored by the authors by building a KDOEST system (Kuwaiti- Dialect Opinion Extraction System from Twitter). This system used two supervised learning algorithms which are SVM and Decision Tree. The authors classified the Kuwaitii terms into multiple classes falling under either positive sentiment or negative sentiment and used these terms to classify the tweets dataset. Another dialect that was explored is the Jordanian dialect in [6], three different classification algorithms were used by the authors; SVM, Naive Bayes, and k-Nearest Neighbors (KNN). They also used RapidMiner for preprocessing and filtering. Annotation of their data samples was done manually by users on CrowdSource website.

The work in [7] tackled the problem of whether emojis in the dataset be used in classification or removed during preprocessing. The classification algorithms used were Decision Tree and Naive Bayes. They managed to increase the accuracy of their model by integrating emojis into their dataset which is close to what we did in our proposed work. Both supervised and unsupervised methods were explored in [8] where the authors build their own lexicon manually in the unsupervised approach. They used the SentiStrength website to build their lexicon and enhanced it by adding synonyms of words. For the supervised examination they used the RapidMiner software on their dataset that included both Modern Standard Arabic and Jordanian dialect tweets.

Another work also used unsupervised learning techniques in [9]. They explored the problems of sentiment analysis in informal Arabic using a small lexicon of size 200 words only. They also used a small dataset of 100 tweets and achieved good results with 73% accuracy. Our final example is the work in [10] where the authors also built their own lexicon by extending a seed lexicon of 380 words. Afterward, they annotated the word in the lexicon using two algorithms. Finally, they used two unsupervised methods to calculate the sentiment of the samples in their tweets dataset that was collected by Twitter search API. The first one is the sum, which sums the polarity of each word in a tweet, and the second is the double polarity method where each word in the tweet has positive and negative weight.

From this Exploration of some of the previous work done in the Arabic Tweets Emotion Recognition, it could be observed that the main two supervised learning models that are commonly used in this task are the Support Vector Machine (SVM) and Naive Bayes.

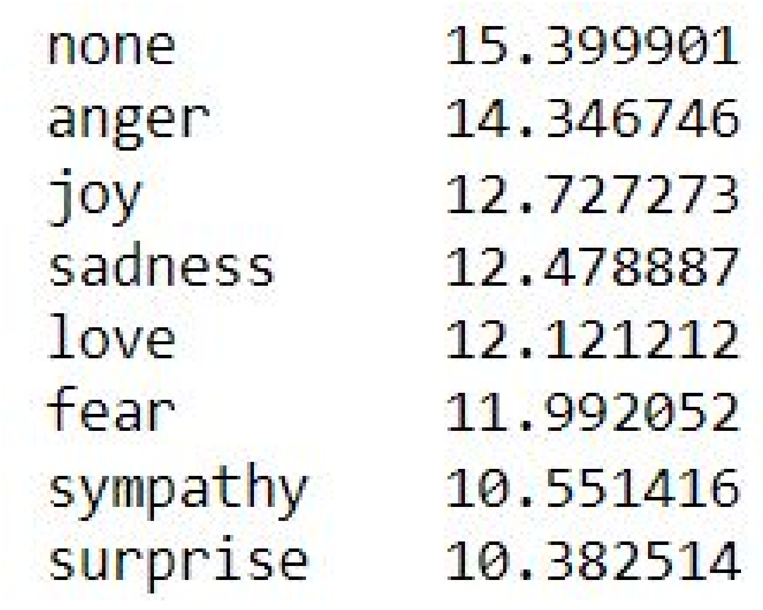
4 Dataset

To find a suitable dataset, we searched for papers with contributions in arabic sentiment analysis. We found a paper titled "Emotional Tone Detection in Arabic Tweets"[11] we started using their dataset in order to use the results as a benchmark to evaluate our model. According to the paper, The dataset has 10,065 tweets and its goal was to cover the most frequently used emotions in Arabic tweets.It was collected from multiple resources : The first source was a corpus made up of 1167 tweets that had previously been gathered and classified as good, negative, or neutral by the Nile University (NU) text mining research group[12, 13].

A team of graduate students at NU then re-annotated this corpus to include emotional content. The students employed Ekman's emotional model and its six emotional classes—happiness/joy, sadness, anger, disgust, fear, and surprise to annotate this dataset. The second resource was made up of 2807 tweets that the same students had collected using Twitter's search API and annotated with the same set of feelings. Egypt's geolocation was used to filter the collected tweets between July 31, 2016, and August 20, 2016. Since many were watching the Olympics at the time, the word أولمبيادْ was used as a search keyword to download the tweets. The third resource was a dataset that was compiled following a search of NileULex sentiment lexicon words added to the Twitter API [14]. More than 500,000 tweets were collected as a consequence of the search. A randomly chosen portion of them was used for labelling. For a non-skewed dataset , A category oriented search was conducted for the under-represented categories (sadness, surprise, love, sympathy, and fear). Towards this end, the Twitter API was queried using hashtags that were expected to return tweets with the desired emotions.

4.1 Data Analysis

The Data Analysis process was mainly about knowing more about the data and making sure that the dataset was downloaded correctly and that no tweets were corrupted. We made sure by exploring some of the tweets appearing at the beginning of the dataset and others appearing at the very end using df.head() and df.tail() as shown in Fig[]. Moreover, as a way of being more aware of the data, a count plot was drawn in order to help maintain a visual representation of the number of tweets falling under every class of emotions represented in (sadness, surprise, love, sympathy, and fear). The count plot was drawn using the Seaborn python library and the output was as shown in Fig [11]. A final representation was conducted in order to make it easier to compare the number of tweets in every class which shows the percentage of every class as follows.



```
none         15.399901
anger        14.346746
joy          12.727273
sadness      12.478887
love         12.121212
fear         11.992052
sympathy     10.551416
surprise     10.382514
```

Figure 2: Percentage of every category

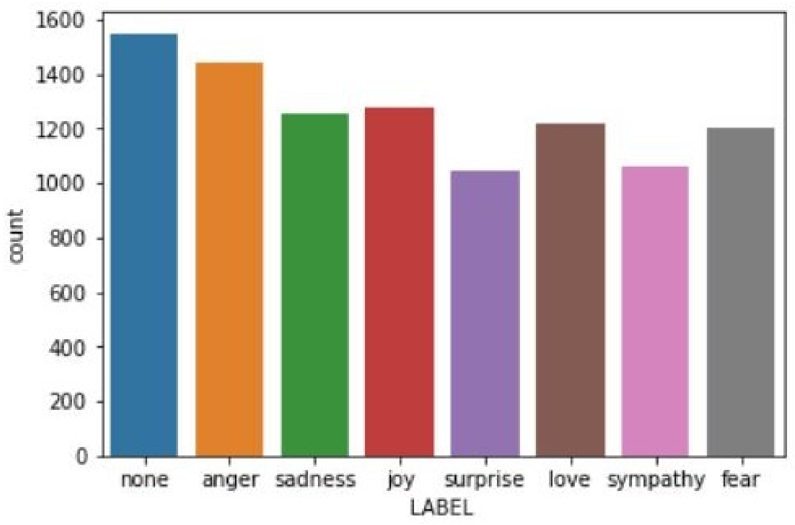
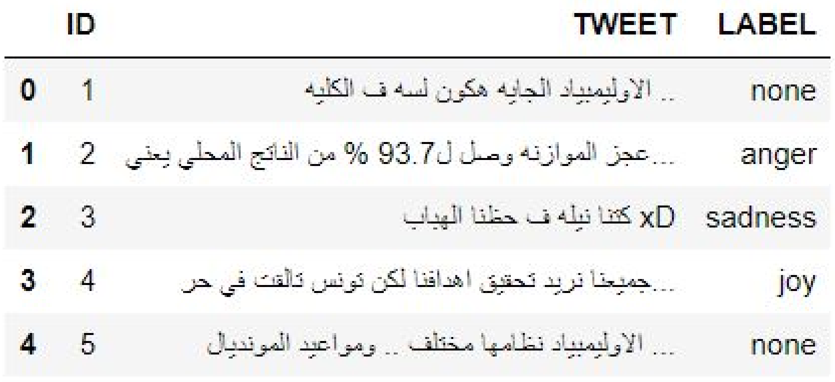


Figure 3: Count plot of categories

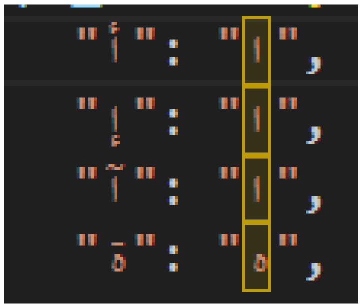


	ID	TWEET	LABEL
0	1	.. الاوليمبياد الجايه هكون لسه ف الكليه	none
1	2	...عجز الموازنه وصل ل93.7 % من الناتج المحلي يعني	anger
2	3	كتنا نيله ف حظنا الهباب xD	sadness
3	4	...جميعنا نريد تحقيق اهدافنا لكن تونس تالقت في حر	joy
4	5	... الاوليمبياد نظامها مختلف .. ومواعيد المونديال	none

Figure 4: showing a very small sample of the data.

4.2 Data Preprocessing

We started preprocessing the data by tokenizing the data set using the NLTK library and wordpunct tokenizer. We then downloaded the arabic stop words from NLTK corpus and filtered the dataset. The third step was to represent similar representations of the same letter only in one way as in Fig [5] to improve the model's accuracy. We then removed the diacritics(اَلتَّشْكِيلُ) and all the elongated characters, for example : (اَلْقااااهرةْ) using the regular expression (re) library. We also removed any punctuation such as "?:!.,;". At the end, we lemmatized the data using qalsadi lemmatizer.



```
"أ" : "ا",
"إ" : "ا",
"آ" : "ا",
"ة" : "ه",
```

Figure 5: Example of the replaced letters

5 System Architecture

The main system architecture components to deal with the aforementioned challenges, we first start by preprocessing the data which is done in a number of steps: First of all, we tokenize and segment the data using the NLTK library and wordpunct tokenizer, then we remove stop words, unify letter representations, and remove diacritics. Afterwards, we remove elongated characters and punctuation. The final data preprocessing stage is when we perform lemmatization to return all words to their original root. After the data is preprocessed, we start with the feature extraction phase using for example bag of words or tf-idf. Following the feature extraction is the data splitting into training and testing portions to prevent data leakage. Finally, we select a classification model, train it and test it against the testing data. We then evaluate it by calculating the confusion matrix among other evaluation metrics. The whole system architecture is shown below in figure 6.

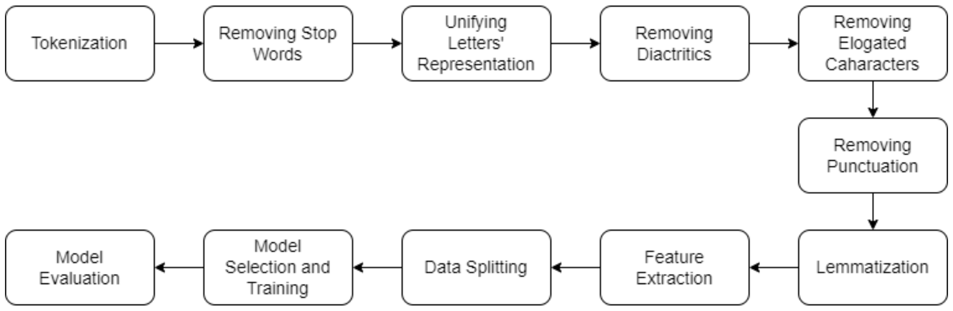


Figure 6: Flow Chart of System Architecture

These system components are subject to change according to their effect on the accuracy of the classification model. We will investigate the effect of each preprocessing step on the model performance and choose the best combination of components that result in the best results. We also might consider removing emojis or replacing them with a standard emoji for each class/emotion and testing the effect of this step on the model evaluation results.

6 Methodology

6.1 Text Classification

Our first task was to classify the data into our 8 labels : none, anger, sadness, joy, surprise, sympathy, fear and love. We used two types of classification to compare the results between them. Our first choice was a simple Logistic regression model.Logistic regression is a a common statistical technique for binary classification [15].It aims to predict a binary result such as (yes/no, true/false, or 0/1) based on a set of input features. For multi-class classification, scikit-learn's logistic regression internally uses the "one-vs-rest" (OvR) strategy. This strategy depends on training multiple binary logistic regression models, where each model differentiates between one class and the rest of the classes. During prediction, the class with the highest predicted probability across all models is selected as the final predicted class. Using sklearn libraries, we split the data into training and testing sets.we allocate 20 % of the data for testing.We create a pipeline that consists of two steps: TfidfVectorizer() which is a text feature extraction method that converts text data into numerical features using the TF-IDF (Term Frequency-Inverse Document Frequency) representation , while the second step is the LogisticRegression(). As for choosing the best hyper-parameter, Grid search is performed using cross-validation. We create and fit the model where we feed it the training sets.We then give the model the testing set of the tweets, make the predictions and evaluate the model.The accuracy score is calculated by comparing the predicted labels with the true labels using the accuracy score function. The classification report function is used to generate a report

Our second choice was a multinomial Naive Bayes classifier. It is a probability algorithm usually used for text classification tasks. It is based on the Naive Bayes principle and assumes that the features (TF-IDF values) follow a multinomial distribution .This classifier is based on Bayes' theorem. the theorem describes the relationship between the conditional probability of an event given previous knowledge and the previous probability of the event. The naive assumption is that features are conditionally independent, given the class, which simplifies the computation and makes it efficient [16].As before, we create a pipeline using the the TfidfVectorizer(), but instead of logistic regression , we use the MultinomialNB(). the 'TfidfVectorizer()' is fitted on the training data to learn the vocabulary and compute the TF-IDF features, and then the 'MultinomialNB()' classifier is trained on the transformed features and the corresponding training labels.

6.2 AraBERT

In this section of the project, we wanted to demonstrate how much of a specific emotion a tweet represents. For that, we used the architecture depicted in fig[7] using a version of BERT that supports Arabic called AraBERT. This architecture is widely used in sentiment analysis and it combines BERT (Bidirectional Encoder Representations from Transformers) with PyTorch Lightning and TPU (Tensor Processing Unit). The architecture is made up of a number of layers which are : 1. For the input layer: The tokenized input text is received by the BERT input layer. BERT requires that the input be tokenized into subword units, which is typically accomplished using a tokenizer. Each subword token is given its own ID. These token IDs are fed into the input layer.

2. AraBert Transformer: The BERT model is made up of a series of transformer encoder layers stacked on top of one another. The transformer encoder is the central component of BERT that captures the contextual representations of the input tokens [17]. Each transformer encoder layer consists of self-attention mechanisms and feed-forward neural networks. The BERT encoder layers are in charge of learning contextualised representations of the input tokens by observing the surrounding tokens. The Arabert model has the same internal structure as Bert however, unlike BERT which is trained on English corpra it is trained to understand and differentiate between different Arabic dialects.

3. Classification Layer(Fine Tuning): The BERT model is commonly used for sequence classification in the context of sentiment analysis. A classification layer is added on top of the BERT encoder to adapt it for this task. This layer maps the AraBERT output representations (usually the [CLS] token, which summarises the entire input sequence) to the desired number of sentiment classes which are 8 classes in our case. The classification layer used is a linear layer.

4. Embedding to vocab + softmax layer: This is the final layer that gives out the final output. it consists of embedding to vocab step which converts the token embeddings from the BERT model to a predefined vocabulary size. The goal of this layer is to project the token embeddings into a space

where the model can learn to predict the probability distribution over the vocabulary. finally, for the softmax layer, its used to give final probabilities for every class in the label column of the data. which is then used to choose the class it belongs to by choosing the higher probability. However, in our case, the output needed is how much this tweet belongs to all the classes.

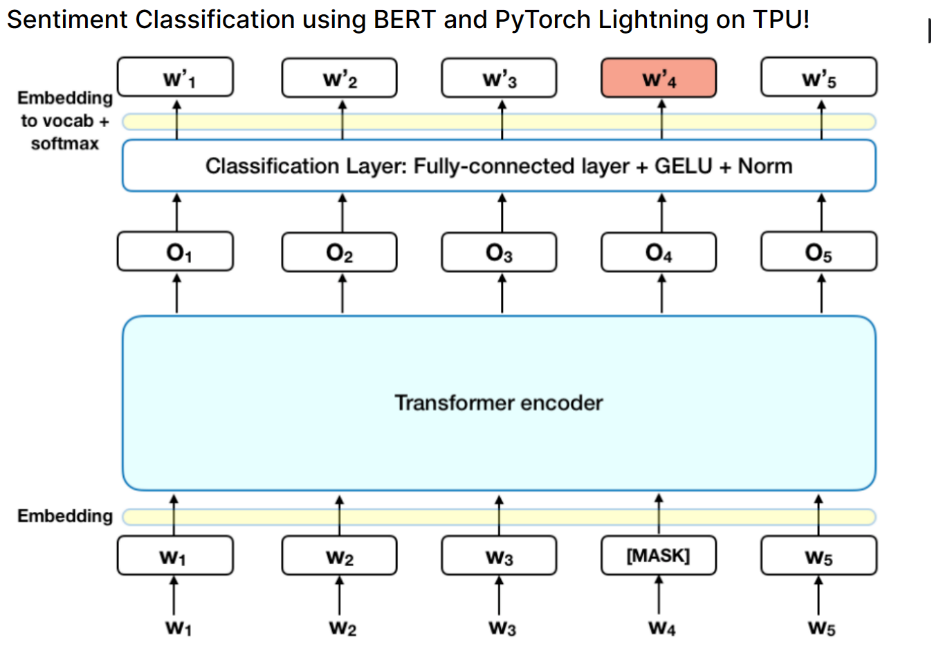

Sentiment Classification using BERT and PyTorch Lightning on TPU!
Embedding to vocab + softmax
w'1
w'2
w'3
w'4
w'5
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

Figure 7: Architecture and how model is used. [18]

6.3 Code structure

Moving on to how the previously explained architecture was used, After loading, preprocessing, and analyzing the data, The needed libraries for the model are imported such as Pytorch lightning, Torch, Transformers and Torch metrics. The architecture was then represented in three classes, every class has a responsibility, the 'SentimentAnalysisModel' class is responsible for initializing the model by loading our pre-trained 'bert-base-arabertv02' model. which is a pre-trained araBERT model fine-tuned for sequence classification tasks. Then the model is configured with appropriate parameters, such as the number of labels (8 in our case), hidden dimensions, and attention heads. then comes the 'SentimentAnalysisDataModule' class, this class is responsible for everything concerning the data and it uses 'LightningDataModule' from pyTorch Lightning to create data modules for training and validation data. These data modules are in charge of data loading, tokenization, batching, and other data-related tasks. The tokenizer is used to convert text data into BERT-compatible tokens. The 'SentimentAnalysisDataModule' is also responsible for the inference method which will be explained in detail. The final class is the one that combines the previous two classes together, which is the 'SentimentAnalysisModule' class which is a child of LightningModule. This class basically contains all of the functionality required to train, validate, and test a sentiment analysis model based on BERT with PyTorch Lightning. The forward pass, training/validation steos, loss computation, optimizer configuration, and logging of relevant metrics during training and evaluation are all defined. To go into detail, the forward pass is represented in a method that is responsible for passing the input ids and attention mask through the BERT model to obtain the model's output logits. The logits represent the predicted sentiment probabilities for each class. The training step for the module is defined by the training-step method. The forward pass is used to obtain logits from a batch of input data (input ids, attention mask, and labels). The cross-entropy loss function (self. loss-fn) is used to calculate the loss between the logits and the ground truth labels. The self.log function is used in the method to record the training loss (train-loss) and accuracy (train-accuracy). It gives back the loss value. For validation step methods, the functionality is exactly the same as the training step. Focusing on the inference method which gives us the needed output, it consists of a number of steps:

1. Using the chosen tokenizer, the input sentence is tokenized and encoded. The sentence is encoded using the encode-plus method, which offers options for padding, truncation, and maximum length. The input IDs and attention mask are included in the encoded sentence.

2. Getting the Input Tensors: The input IDs and attention mask tensors that will be used as input to the model are retrieved from the encoded sentence.

3. Model Prediction: The model is called with the input IDs and attention mask tensors to obtain the logits, representing the predicted sentiment probabilities for each class.

4.Torch.softmax is used to pass the logits through a softmax function in order to calculate the probabilities for each sentiment class. After being compressed to eliminate any singleton dimensions, the probabilities tensor is changed to a Python list.

5. Sentiment class indices and their corresponding labels are mapped using a dictionary called sentiment labels. The predicted labels are obtained by using ”torch.argmax” to choose the class index with the highest probability.

6. Probability Sorting: Based on the probabilities, the sentiment labels and probabilities are combined into tuples and sorted in descending order.

7. Returning Results: Using the predicted label obtained earlier (predicted-labels.item()), the predicted sentiment label is retrieved from sentiment-labels. The function returns the sorted list of sentiment probabilities and labels as well as the predicted sentiment label.As for preprocessing the outputs, we mapped each output label to its designated class : 0: ”none” , 1:”anger”, 2:”sadness”, 3: ”joy”, 4: ”surprise”, 5:”sympathy”, 6:”fear”, 7:”love”

7 Experiments

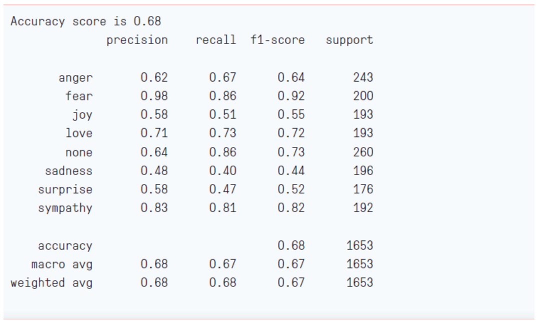
This part explains how different models were used and the results will be shown in the results section. For classification, we used a Logistic regression classifier and multinomial Naive Bayes classifier to classify the tweets into our 8 classes of emotions after applying the same preprocessing steps to the data.

For the Task of calculating the percentage of every emotion, we trained the data using the Arabert model. Once without lemmatization and once after lemmatization since we noticed that the result of the lemmatization changed the some of the meanings of the words.

We also noticed that sometimes the model used to overfit so we set 30 % of the weights to zero using the dropout method.

8 Results

The comparison between the results of logistic regression and Multinomial Naive Bayes Classifiers resulted in the following figures:

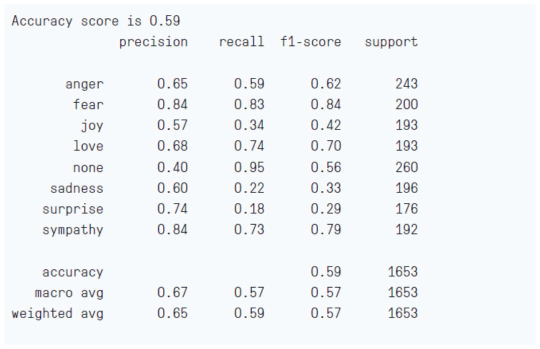


```
Accuracy score is 0.68
              precision    recall  f1-score   support

       anger       0.62      0.67      0.64       243
        fear       0.98      0.86      0.92       200
         joy       0.58      0.51      0.55       193
        love       0.71      0.73      0.72       193
        none       0.64      0.86      0.73       260
     sadness       0.48      0.40      0.44       196
    surprise       0.58      0.47      0.52       176
    sympathy       0.83      0.81      0.82       192

    accuracy                           0.68      1653
   macro avg       0.68      0.67      0.67      1653
weighted avg       0.68      0.68      0.67      1653
```

Figure 8: Results of Logistic Regression



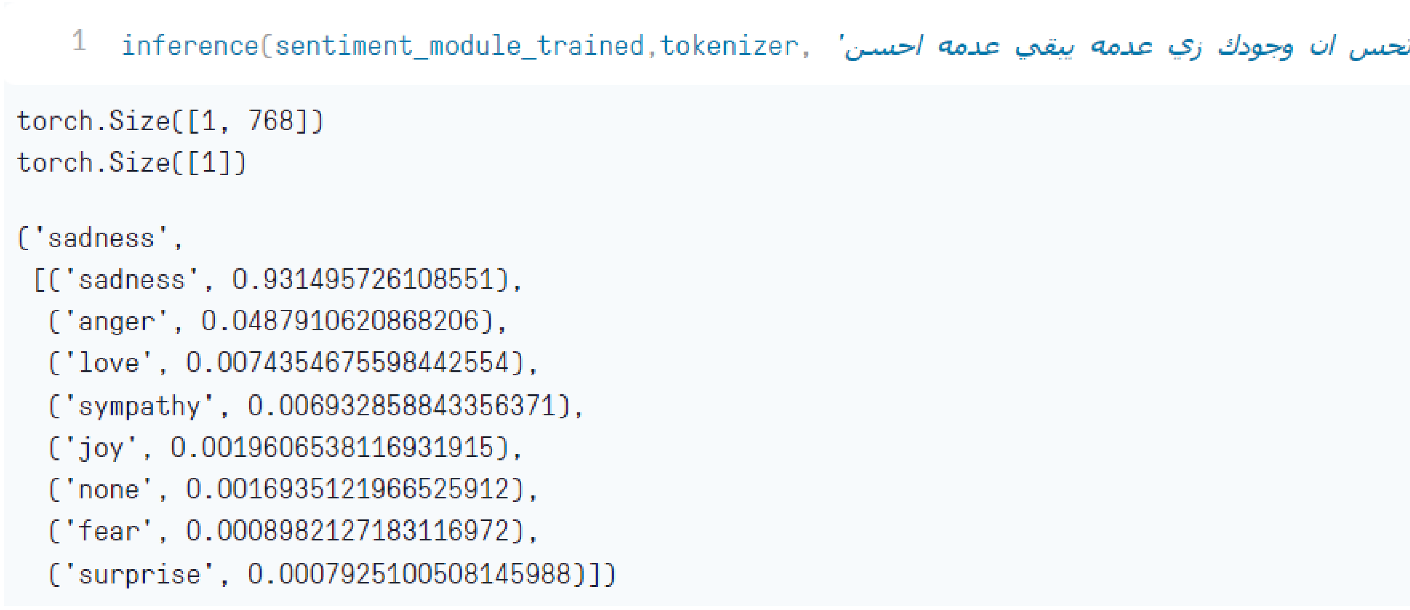
```
Accuracy score is 0.59
              precision    recall  f1-score   support

       anger       0.65      0.59      0.62       243
        fear       0.84      0.83      0.84       200
         joy       0.57      0.34      0.42       193
        love       0.68      0.74      0.70       193
        none       0.40      0.95      0.56       260
     sadness       0.60      0.22      0.33       196
    surprise       0.74      0.18      0.29       176
    sympathy       0.84      0.73      0.79       192

    accuracy                           0.59      1653
   macro avg       0.67      0.57      0.57      1653
weighted avg       0.65      0.59      0.57      1653
```

Figure 9: Results of Multinomial Naive Bayes

As for the results of AraBERT, without any lemmatization, it reached a training accuracy of 95% and validation accuracy of 69% in the 15th epoch. But with lemmatization, it reached a training accuracy of 100% and validation accuracy of 65%

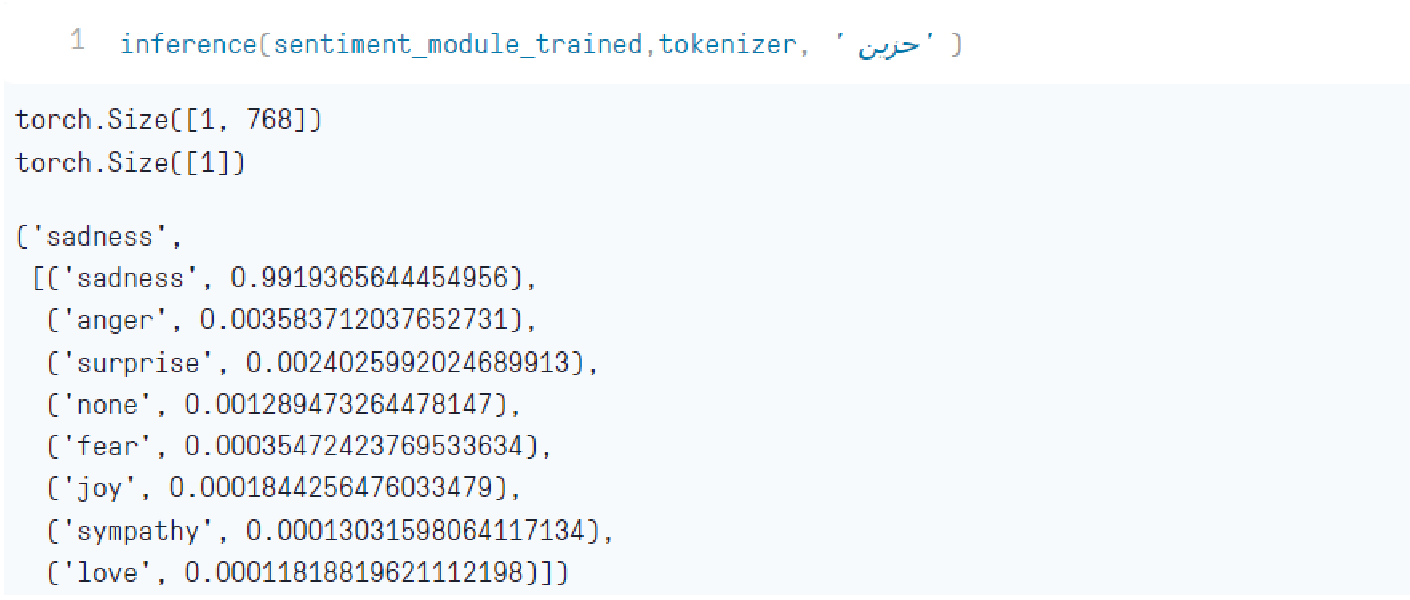


```
inference(sentiment_module_trained,tokenizer, 'تحس ان وجودك زي عدمه يبقي عدمه احسن'
```

```
torch.Size([1, 768])
torch.Size([1])

('sadness',
 [('sadness', 0.931495726108551),
  ('anger', 0.0487910620868206),
  ('love', 0.0074354675598442554),
  ('sympathy', 0.006932858843356371),
  ('joy', 0.0019606538116931915),
  ('none', 0.0016935121966525912),
  ('fear', 0.0008982127183116972),
  ('surprise', 0.0007925100508145988)])
```

Figure 10: AraBERT Inference method result with no lemmatization



```
inference(sentiment_module_trained,tokenizer, ' حزين' )
```

```
torch.Size([1, 768])
torch.Size([1])

('sadness',
 [('sadness', 0.9919365644454956),
  ('anger', 0.003583712037652731),
  ('surprise', 0.0024025992024689913),
  ('none', 0.001289473264478147),
  ('fear', 0.00035472423769533634),
  ('joy', 0.0001844256476033479),
  ('sympathy', 0.00013031598064117134),
  ('love', 0.00011818819621112198)])
```

Figure 11: AraBERT Inference method result with no lemmatization

9 Dataset Limitations

In this section, we will discuss the limitations of the dataset; they are briefly listed below

- The Data samples that were labeled as "None".
- Some corrupted data samples
- The emotion faces (emoji)

The first limitation in our dataset was that most of the data samples labeled as "None" were very confusing for the model, upon inspection, many of these samples were even confusing to us which means that even human beings can not classify them into a specific class of sentiment. Another very critical limitation was that during the training of our model, the creation of the embeddings of some of the data samples could not be successfully done to allocate these specific samples we had to train the model on batches of the data with only 1000 samples per batch. Whenever we found any batch that caused an error in the training we kept decreasing the size of the batch to narrow the scale to be able to find the corrupted samples. This has resulted in dropping some data samples; specifically the samples 1000-1500, 4200-4500, and 9000-10000. Upon inspection, we found out that some of these samples were nan. Finally, the emojis in the tweets were a challenge to us; we tried training and testing the dataset with the emojis in and also we tried removing them. Eventually, we performed tagging which is replacing every emoji with a corresponding text that represents its meaning.

10 Conclusion

In conclusion, in our project, we performed sentiment analysis on a dataset of Arabic Tweets. The dataset initially consisted of 10,065 tweets, we performed multiple preprocessing steps on the data samples and even cleaned the dataset and dropped some corrupted samples. Moreover, we classified them into 8 different sentiment classes; anger, joy, sadness, love, fear, sympathy, surprise, and the none class which represents neutral tweets or tweets that don't belong to any of the other classes. We used three main supervised classification algorithms which are Naive Bayes, Logistic Regression, and AraBERT. We achieved promising results from the three models, we compared the results obtained from the three models. Finally, we compared our results to the results obtained by the authors of the dataset.

References

[1] Sarah O Alhumoud, Mawaheb I Altuwaijri, Tarfa M Albuhairi, and Wejdan M Alohaideb. Survey on arabic sentiment analysis in twitter. *International Journal of Computer and Information Engineering*, 9(1):364–368, 2015.

[2] Amira Shoukry and Ahmed Rafea. Sentence-level arabic sentiment analysis. In *2012 International Conference on Collaboration Technologies and Systems (CTS)*, pages 546–550, 2012.

[3] Muhammad Abdul-Mageed, Mona Diab, and Sandra Kübler. Samar: Subjectivity and sentiment analysis for arabic social media. *Computer Speech Language*, 28:20–37, 01 2014.

[4] Soha Ahmed, M. Pasquier, and Ghassan Qadah. Key issues in conducting sentiment analysis on arabic social media text. pages 72–77, 03 2013.

[5] Janan Ben Salamah and Aymen Elkhlifi. Microblogging opinion mining approach for kuwaiti dialect. *international conference of computer technology and information management*, 04 2014.

[6] Rehab Duwairi, Raed Marji, Narmeen Sha'ban, and Sally Rushaidat. Sentiment analysis in arabic tweets. pages 1–6, 04 2014.

[7] Salha Al-Osaimi and Khan Badruddin. Role of emotion icons in sentiment classification of arabic tweets. *MEDES 2014 - 6th International Conference on Management of Emergent Digital EcoSystems, Proceedings*, pages 167–171, 09 2014.

[8] Nawaf A. Abdulla, Nizar A. Ahmed, Mohammed A. Shehab, and Mahmoud Al-Ayyoub. Arabic sentiment analysis: Lexicon-based and corpus-based. In *2013 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, pages 1–6, 2013.

[9] Lamya Albraheem and Hend Al-Khalifa. Exploring the problems of sentiment analysis in informal arabic. pages 415–418, 12 2012.

[10] Ahmed Ali and Samhaa El-Beltagy. Open issues in the sentiment analysis of arabic social media: A case study. 03 2013.

[11] Amr Al-Khatib and Samhaa El-Beltagy. Emotional tone detection in arabic tweets. 04 2017.

[12] Talaat Khalil, Amal Halaby, Muhammad Hammad, and Samhaa El-Beltagy. Which configuration works best? an experimental study on supervised arabic twitter sentiment analysis. 04 2015.

[13] Samhaa El-Beltagy, Talaat Khalil, Amal Halaby, and Muhammad Hammad. *Combining Lexical Features and a Supervised Learning Approach for Arabic Sentiment Analysis*, pages 307–319. 01 2018.

[14] Samhaa El-Beltagy. Nileulex: A phrase and word level sentiment lexicon for egyptian and modern standard arabic. 05 2016.

[15] Joanne Peng, Kuk Lee, and Gary Ingersoll. An introduction to logistic regression analysis and reporting. *Journal of Educational Research - J EDUC RES*, 96:3–14, 09 2002.

[16] Shuo Xu, Yan Li, and Wang Zheng. Bayesian multinomial naïve bayes classifier to text classification. pages 347–352, 05 2017.

[17] Wissam Antoun, Fady Baly, and Hazem M. Hajj. Arabert: Transformer-based model for arabic language understanding. *CoRR*, abs/2003.00104, 2020.

[18] TANAY MEHTA. Bert with pytorch lightning + tpu for sentiments. 2021.