Arabic Tweets Emotion Recognition

Mohanad Osman and Shahd Koura, Team 6

May 26, 2023

Abstract

Sentiment analysis aims to identify emotions that are being expressed in a given text. It is becoming more relevant than ever since most people tend to turn to their social media platforms to voice their opinions and express their thoughts. The majority of recent works, however, mainly focus on the English language. Research in sentiment analysis in languages other than English falls far behind. Emotion detection in Arabic text is a relatively new field of research that focuses on developing methods for identifying and analyzing emotions expressed in Arabic texts. This field makes use of techniques from natural language processing, machine learning, and psychology to classify emotions into predefined categories such as anger, sadness, joy, fear, or surprise. In this project, we adopt multiple techniques to pre-process and clean an existing labeled dataset, as well as fine-tune a classifier in order to classify emotion in a given tweet into one of the basic emotion categories.

1 Introduction

Social media has prevailed in the last decade and has become a prime way of expressing emotions. This can provide valuable insights into public opinion and sentiment. Emotion recognition, also known as sentiment analysis, is a subfield of natural language processing (NLP) that focuses on identifying and understanding emotions expressed in text. It involves analyzing and classifying textual data to determine the underlying emotional states, attitudes, or sentiments of individuals.

1.1 Motivation

Sentiment analysis in the Arab media has become more important in the past few years, due to the increasing number of users of social media. Sentiment analysis allows researchers and analysts to monitor and extract valuable insights from social media data. It can help track public sentiment on specific topics, identify trends, detect emerging issues, and gauge public reactions to events, products, or policies. However, research on Arabic tweets emotion recognition is relatively limited. In this project we try to bridge this gap, while addressing the specific challenges and nuances of analyzing emotions in Arabic text.

1.2 Challenges

Recognizing emotions in Arabic tweets presents several challenges due to the unique characteristics of the Arabic language and the nature of social media data. The following are some of the challenges we faced when studying our chosen dataset(mostly in Egyptian Dialect):

- Dialect - Arabic has numerous dialects, each with its own vocabulary, grammar, and expressions. This variation makes it difficult to develop or use a single standardized model that can accurately detect emotions across different dialects.
- Noise Generation – most users of the social media do not care about the spelling or grammatical mistakes; this leads to the generation of a number of incorrect words and phrases. In addition, since most tweets are in the Egyptian dialect, an additional challenge was imposed as slang and informal language were tricky to detect. We tried using two libraries: pyspellchecker and spellchecker. These are python libraries that provide spelling correction functionality. They are able to detect and correct spelling errors in text by suggesting alternative words based on the

provided input. However, the results of both libraries were not very promising as they failed to detect the Egyptian 'slang'.

- Emoji Usage - Emojis can be considered a "visual representation" of text. Our dataset contains a wide usage of emojis to convey emotions. These emojis might not always align with the textual content, leading to discrepancies between the perceived emotion and the actual text.

2 Related Work

Osman et al. [3] studied increasing the size of Arabic posts through social media, which increases the importance of sentiment analysis and subsequently increases the interest of researchers, especially since the number of researches of the Arabic language is very few. The difficulty and complexity of the Arabic language in addition to the lack of tools available to extract Arabic sentiments from the text is an obstacle to researchers so the uses of natural language processing facilitate dealing with the Arabic text.

Shoukry et al. used a dataset of 600 tweets to built two lists, one for positive and the other for negative sentiment words[9]. They then used lexical-based classifier to classify Arabic tweets expressed in Egyptian dialect.

Samhaa R. El-Beltagy et. al(whose contributions and dataset are the starting point for us) discussed the importance of sentiment analysis in the Arab media during the past few years, because of the increasing number of users of social media. They found it difficult to deal with texts written on social media, often the slang language, in addition to the great difficulty in dealing with various Arabic dialects. They added a set of features that were combined with machine learning based on sentiment analysis, where they chose an application Complement Naïve Bayes.

Some studies have been conducted in the field of sentiment analysis of Arabic content, especially when it comes to comparing the performance of different classifiers for the Arabic language datasets. One study compared the performance of the Sequential Minimal Optimization (SMO) and Naïve Bayesian (NB) in the classification task[2]. These algorithms are the suggested classifiers based on excel in their performance; SMO comes first, and then NB.

3 Methodology

3.1 Data Collection

We used the dataset collected by Samhaa R. El-Beltagy et al.[1], which as far as we know is one of the biggest Arabic datasets collected to date. The dataset consists of more than 10,000 tweets covering the eight basic emotion categories: sadness, anger, joy, surprise, love, sympathy, fear, and none. The dataset came from three different sources: a corpus of tweets collected and labeled for polarity, a search using Twitter's API using "Olympics" as a search keyword, and another search using Twitter's API which focused on revising the emotion categories which resulted in adding 'sympathy' as an emotion category.

3.2 Data Pre-processing

Preprocessing the data helps in preparing it for analysis, improving the quality of the results, and reducing the time and effort required for analysis. Eight pre-processing methods were utilised, including removing redundant white spaces, eliminating punctuation, tokenizing the sentence, shortening words that were written on a long way, removing stopwords, removing diacritization, handling emojis, and stemming the word to its root. But, before that we had to remove null tweets from the dataset to be able to apply these pre-processing techniques.

- **Handling emojis:** We handled emojis in two different ways and compare the result more in the experiment section below;
  - We used library 'emoji' to remove the emojis in all tweets to reduce noise[7].
  - We replaced each emoji with its equivalent meaning. For example, this emoji,"☹" is translated into "وجه محبط" using mtranslate[8].

- **Removing redundant white spaces:** We had to remove redundant white spaces as they made text more readable, reduces noise, and optimises storage space. Also, it helps to ensure consistency and improve the accuracy of tokenization technique.
- **Removing diacritization:** Eliminating diacritical marks will simplify the text by bringing down its complexity, standardise how the text is represented, and reduce noise. We used 'pyarabic.araby' to remove all diacritical marks in the dataset[6].
- **Removing effect of speech:** We removed the effect of speech by shortening elongated words such as "كتيييير" is changed to "كتير". This is done as the wording should be normalised and made more uniform. Shortening words can increase the precision of text analysis and natural language processing activities because they can occasionally be the consequence of typos or improper spelling.
- **Eliminating punctuation:** Punctuation marks like commas, periods, question marks, etc. do not add much meaning to the text and can be safely removed. It also helps with tokenizing the sentences. As we are dealing with Arabic, we had to use some extra punctuation that is not covered in the English language, like '؟' ''؛ ''،. We used 'string' module provided by python to get the all the English punctuation[5]. Since most Egyptians don't write their tweets in official Arabic, there were still some issues. We had to manually add a few Egyptian stop words because they weren't already in the library. For instance, "أنا" is spelled as "انا" by most people on social media platforms.
- **Tokenization:** Tokenization is the process of dividing a stream of text into tokens, which can be words, phrases, symbols, or other meaningful items to be further processed easily. It makes it possible for algorithms to understand the meaning and structure of textual data.
- **Stemming:** Reducing inflected or derived words to their root or basic form is the goal of stemming. This aids in normalizing the words in a text, making it easier to process text data. We used 'nltk' library which stems the word to its root without a root dictionary[4]. We used 'nltk' library as our stemmer[4].
- **Removing stopwords:** Stopwords are words that are often used in a language but do not have a major meaning or value when analysing text data. Stopwords are eliminated from the text data, which concentrates on the most insightful words and makes it simpler to analyse and derive insights. Additionally, by doing this, the data's size can be decreased and processing speed can be increased. We used 'nltk' library to provide us with the most common Arabic stopwords such as "أَنْتَ","إِذَا", etc[4].

3.3 Data Analysis

- **Word Frequency Analysis:** Word frequency analysis can capture the essence of a text corpus by extracting keywords or important terms. Considering words with high frequencies helps in identifying key concepts, topics, or in our case emotions discussed in the text. By considering the frequency of words in a text corpus, feature selection can be easier. On the other hand, words with very low frequencies may be anomalies or errors in the dataset.

Word	Frequency
الاوليمبياد	2694
خايفه	677
مصر	618
الهم	314
الحب	306
حاجه	297
خايف	242
الناس	221
حد	216
ربنا	207
يوم	185
يارب	183
اول	175
قلبي	166
حب	156

Table 1: Most Frequent Words in the corpus

- **Word Count** Word counts are a visual way to analyze data. They can help determine the prominence of certain terms or concepts. We created a Word Cloud representing the most common words in the dataset.

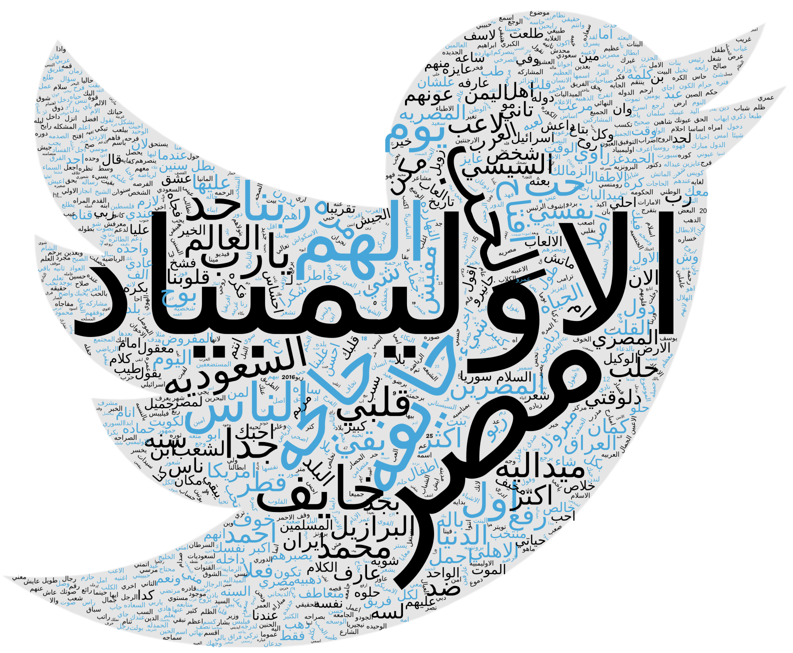

Figure 1: Word Cloud showing the most frequent words

- **Most Frequent Word Per Label** We grouped our dataset by the 'label' column, and calculated the most frequent word for each label. We used the FreqDist library in the nltk package[4]. We removed الاوليمبياد as a keyword from our search since our dataset was heavily related to the Olympics, so it was the most frequent word appearing in the whole corpus as shown in Table 1. We also removed countries' names such as مصر and قطر, which appeared very often in the dataset but will not affect the sentiment analysis.

Label	Word
Sadness	الناس
Anger	ضد
Joy	مبروك
Surprise	بجد
Love	الحب
Sympathy	ربنا
Fear	خايفه
None	ميداليه

Table 2: Most Frequent Word Per Label

3.4 Model

The model we used is built by fine-tuning AraBERT on our dataset. AraBERT is based on Google's Bidirectional Encoder Representations from Transformers (BERT) architecture, which is a powerful deep learning model for language representation. AraBERT was specifically designed and trained on Arabic social media text. In addition, AraBERT has a morphological analyzer that separates Arabic words into their individual morphemes in order to better represent the Arabic language's complex morphology. Moreover, AraBERT is trained on both diacritized and undiacritized texts. It also features a diacritic restoration module that predicts the proper diacritics for undiacritized text during inference. Hence, we opted using AraBERT.

It is a pre-trained transformer-based language model created exclusively for Arabic natural language processing applications, allowing it to learn patterns, syntax, and semantics specific to the Arabic language. The AraBERT model is trained on a large corpus of Arabic texts, which makes it very powerful in handling the complexities and nuances of the Arabic language. The model has a multi-layer bidirectional transformer encoder with 12 encoder blocks, 12 attention heads, 768 hidden dimensions, 512 maximum sequence length, and around 110 million parameters.

Here's a breakdown of how the model works:

- **Forward Pass:** During the forward pass, the model is given the input tensors and runs calculations to produce raw predictions. These calculations involve performing a number of operations and transformations on the input tensors that are specified by their weights.

- **Loss Calculation:** After the model has produced its predictions, the loss function is computed by comparing the output with the labels from the ground truth. The difference between what was expected and the actual values is measured by the loss function. We use the CrossEntropyLoss function, which is suitable in this case as it encourages the model to assign high probabilities to the correct class while suppressing the probabilities of incorrect classes. This results in one sentiment class being the most dominant for each tweet.

- **Backward Pass (Backpropagation):** The gradients of the loss function with respect to the model's inputs (weights) are calculated. The gradients from the output layer are then transmitted back to the input layer.

- **Weight Update:** The optimizer uses these gradients to update the model's weights after acquiring the gradients. The specific updating rule is determined by the optimizer method (we used Adam Optimizer). The optimizer helps the model in adjusting its parameters to minimize the loss function.

4 Experiment

Our dataset is divided into training and testing subsets (80%, 20%). The training dataset is split into batches in this method, with each batch containing 16 samples. 5 epochs of training are completed by the model in this process. The model can learn from the data, refine its parameters, and boost its ability to predict with each new epoch. The computational power available, the size of the dataset, and the intended model performance are taken into consideration when deciding on the batch size and number of epochs. We tried two epoch sizes: 5 and 10, to determine which epoch size was more suitable. The accuracy obtained after training the model for both 5 and 10 epochs was almost the same. This means that the model has already reached its optimal performance and further training does not lead to significant improvements, which indicates that the model has converged and learned the underlying patterns and relationships in the training data. Therefore we went with an epoch size of 5 to avoid overfitting, which is when the model becomes too specialized to the training data and performs poorly on unseen data.

After the data has been preprocessed, the BERT tokenizer is used to encode the data by adding special tokens, applying padding and truncation, and returning the encoding as a PyTorch tensor. We then extract the individual tensors (input_ids and attention_mask). We create a single tensor containing all the input IDs for the training dataset, as well as a single tensor containing the attention masks of the training dataset.

Label encoding is performed to transform the eight emotion classes into numerical representations, and then convert the encoded labels into a PyTorch tensor.

The input IDs, attention masks, and labels tensors are used to create a PyTorch TensorDataset, which combines the three tensors allowing them to be easily accessed as batches during training.

The same steps are repeated for the testing dataset.

After creating the PyTorch TensorDatasets, we start training our model. The model gradually learns from the training data and optimizes its parameters to improve its predictions by iteratively executing forward passes, computing the loss, computing gradients using backpropagation, and updating the weights.

We also implement an early stopping mechanism, which monitor the progress of the model during training and stops the training process if there is no significant improvement in the loss function. This mechanism involves tracking the best loss achieved so far and setting a predefined patience value(3 in our case), which determines the number of epochs to wait for improvement. For each epoch, the average loss is calculated. If the current average loss is lower than the previous best loss, the best loss is updated, and a counter is reset. However, if the average loss does not improve, the counter is incremented. If the counter reaches the patience value, indicating no improvement for several epochs, the training is stopped early.

This helps in saving computational resources and makes sure that the model is not being unnecessarily trained if it is not showing any improvement.

We carried out two experiments, as we wanted to investigate how emojis affect the overall accuracy of the model. In the first experiment, emojis were completely removed from the dataset, while in the second one emojis were converted to their textual equivalent.

5 Evaluation

During the testing phase, the trained model's performance and generalization capabilities are evaluated on the test dataset. Additionally, it offers information about how well it generalizes outside of the training set. The model is applied to each sample in the test dataset, and its predictions are contrasted with the real labels to assess the model's accuracy as well as other assessment metrics including precision, recall, and F1-score.

When we removed the emojis from our dataset, our model was able to accurately identify and assign the appropriate emotion to a given sentence with an accuracy of **73.17%**. This high level of accuracy demonstrates the model's strength and ability to accurately capture the intricate expressions of emotions in text. The recall rate for the model was **73.26%**. This shows that the model is capable of recovering examples of each emotion, effectively capturing a significant percentage of the true positives. It implies that the model is reasonably successful in identifying and categorizing emotions contained in the dataset. Precision and recall of the model are balanced, as shown by the F1 score of **0.72**. This score gives an overall performance indicator by accounting for both false positives and false negatives.

On the other hand, when we replaced the emojis with their equivalent meanings, the model had an accuracy of **68.5%**, a recall of **69%**, and an F1 score of **0.69**. This shows that replacing the emojis with their equivalent text may not be as successful in accurately classifying the sentences' emotions, as evidenced by the reduced accuracy compared to removing them. This may be due to one of the following reasons:

- **Loss of Cultural Context:** Emojis are commonly used to express emotions and subtlety in communication and frequently have cultural significance. Emojis may lose cultural context and nuanced subtleties that are unique to a single language or location when they are translated into their corresponding meaning. Emojis can be eliminated so that only the text itself is highlighted, allowing for a more language-centric analysis.
- **Variability in Emoji Interpretation:** The same emoji could not always convey the same meaning or mood to various people or cultures because emojis are up to individual interpretation. Depending on their own experiences, cultural background, or context, different people may use the same emoji to symbolise various feelings or concepts. Emojis' comparable meanings are assumed while translating them, although this common interpretation might not reflect the original text's intended meaning.

6 Conclusion

In this project we tried to implement a new approach in sentiment analysis of arabic tweets. The paper from which we obtained our dataset attempted to classify tweets into one of 8 emotion classes using a Complement Naïve Bayes classifier, which resulted in an accuracy of **68.1%**. Our fine-tuned AraBERT model resulted in an accuracy of **73.17%**. Tt can be concluded that the Arabert model is slightly more effective(by almost 5%) in capturing the sentiment of Arabic tweets compared to the Complement Naïve Bayes classifier. The higher accuracy of the Arabert model suggests that it better understands the nuances and intricacies of the Arabic language, enabling it to make more accurate sentiment predictions.

In addition, removing emojis from this specific dataset resulted in a higher accuracy, which suggests that emojis are still very challenging to handle.

7 Future Work

Given the restrictions of the current dataset, which is rather small and concentrated on the Olympics, there are a number of potential directions that can be investigated to further improve the accuracy of the sentiment analysis model.

The size and domain coverage of the present dataset might be constrained. A deeper understanding of sentiment in multiple contexts would result from expanding the dataset to include a wide range of texts from many domains, such as another social media platform other than Twitter, news articles, product reviews, and customer feedback. This would enable the model to generalize better and capture sentiments across different topics.

It would be beneficial to increase the model's ability to handle multilingual text in order to better capture the thoughts expressed in many languages. If the model is trained equally on each language, using multilingual training data and utilizing these models might help the model comprehend and classify sentiment in many languages. This would increase its adaptability to various linguistic contexts.

References

[1] Amr Al-Khatib and Samhaa R El-Beltagy. Emotional tone detection in arabic tweets. In *Computational Linguistics and Intelligent Text Processing: 18th International Conference, CICLing 2017, Budapest, Hungary, April 17–23, 2017, Revised Selected Papers, Part II 18*, pages 105–114. Springer, 2018.

[2] Bassam Al-Shargabi, Waseem Al-Romimah, and Fekry Olayah. A comparative study for arabic text classification algorithms based on stop words elimination. In *Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications*, pages 1–5, 2011.

[3] Ghadah Alwakid, Taha Osman, and Thomas Hughes-Roberts. Challenges in sentiment analysis for arabic social networks. *Procedia Computer Science*, 117:89–100, 2017.

[4] Steven Bird, Edward Loper, and Ewan Klein. *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.

[5] Python Software Foundation. Python 3.9.5 documentation. https://docs.python.org/3/index.html, 2021. Accessed: [Insert date here].

[6] Karim Ghassan. pyarabic: Arabic language toolkit for python. https://github.com/linuxscout/pyarabic, 2011. Accessed: [Insert date here].

[7] Taehoon Kim and Kevin Wurster. emoji: Emoji for python. https://github.com/carpedm20/emoji, 2021. Accessed: [Insert date here].

[8] mouuff. mtranslate. https://github.com/mouuff/mtranslate, 2023. Accessed on 25 May 2023.

[9] Amira Shoukry and Ahmed Rafea. Preprocessing egyptian dialect tweets for sentiment mining. In *Fourth Workshop on Computational Approaches to Arabic-Script-based Languages*, pages 47–56, 2012.