Emotion Recognition PreProcessing

1 Introduction

In the field of natural language processing, accurately classifying emotions within text is crucial for understanding sentiment and mood. This process involves challenges like handling emojis, diverse writing styles, and linguistic variations. To address these complexities and enable effective emotion classification, thorough data preprocessing and text normalization are essential.

2 Data Cleaning

In this section, we describe the data cleaning process to prepare the text data for further analysis.

2.1 Emojis and Emotions Mapping

We started by mapping emojis to the predefined 8 emotion classes. We dropped emojis with no meaningful emotion representation, leaving us with 80 remaining emojis that were incorporated into the dataset.

2.2 Stopwords Removal

Initially, we employed the NLTK stopwords list, but it proved inadequate in capturing all relevant stopwords. As a solution, we incorporated the Arabic stopwords library. However, we excluded negation and prohibition words from removal.

2.3 Text Normalization

We performed text normalization to enhance the quality of the text data. Steps included:

- Replacing specific Arabic letters with their standardized forms
- Removing numbers and non-Arabic characters

3 Stemming

In this section, we discuss the application of stemming techniques to further refine the text data.

3.1 Light Stemming

We utilized the ISRI stemmer's pre32 and su32 methods for light stemming. This process helped reduce words to their root form while maintaining readability.

3.2 Root Stemming

We applied root stemming using the stem method in the ISRI stemmer, which aimed to achieve a higher level of word normalization.

4 Tokenization and Embedding

In this section, we delve into the critical steps of tokenization and embedding, which form the foundation of our emotion classification model. We explore two different approaches to tokenization and evaluate their impact on the quality of embeddings. Additionally, we leverage the power of BERT tokenizer to further enhance our results.

4.1 Manual Tokenization and MarBERT Embedding

We begin by manually tokenizing our text data using the TensorFlow tokenizer. This step is crucial as it transforms our raw text into a sequence of numerical tokens, which can be effectively processed by our model. The tokenized sequences are then passed through the MarBERT embedding layer to generate embeddings that capture semantic information.

However, despite the initial effort, we observe that the performance of the embeddings with the GRU model falls short of our expectations. This prompts us to explore a more advanced tokenizer for enhanced results.

GRU	precision	recall	f1-score	support
0	0.43	0.84	0.57	307
1	0.40	0.28	0.33	276
2	0.24	0.30	0.27	268
3	0.00	0.00	0.00	258
4	0.46	0.25	0.32	250
5	0.25	0.68	0.36	194
6	0.00	0.00	0.00	201
7	0.56	0.47	0.51	259
accuracy			0.36	2013
macro avg	0.29	0.35	0.29	2013
weighted avg	0.31	0.36	0.31	2013

Figure 1: Manual Tokenization and MarBERT Embedding

4.2 BERT Tokenization and Sentence Embedding

To address the limitations of manual tokenization, we turn to the BERT tokenizer. BERT's sophisticated tokenization mechanism not only handles word splitting but also accounts for subword tokens, thereby capturing the finer nuances of the text. We apply the BERT tokenizer to our text data and generate embeddings using an average pooling technique at the sentence level.

The impact of BERT tokenizer on our emotion classification model is striking. The embeddings exhibit improved quality, leading to a significant enhancement in the performance of the GRU model. This result underscores the importance of employing advanced tokenization techniques for text data preprocessing.

GRU	precision	recall	f1-score	support
0	0.64	0.75	0.69	307
1	0.59	0.69	0.64	276
2	0.66	0.36	0.46	268
3	0.43	0.50	0.46	258
4	0.70	0.71	0.71	250
5	0.72	0.69	0.70	194
6	0.43	0.39	0.41	201
7	0.88	0.88	0.88	259
accuracy			0.63	2013
macro avg	0.63	0.62	0.62	2013
weighted avg	0.63	0.63	0.62	2013

Figure 2: BERT Tokenization and Sentence Embedding

4.3 Utilizing Pickle Files

To streamline our workflow and facilitate efficient data storage, we leverage pickle files to save and load our tokenized and embedded data. Pickle files enable us to preserve the preprocessed data, avoiding the need for repeated preprocessing steps during subsequent runs. This approach not only enhances the reproducibility of our work but also accelerates the experimentation process.

5 Summary

In this project, we aim to develop an accurate emotion classification model for Arabic text.