Neural Networks Project
Arabic sentiment analysis (NLP)

Table of Contents

Team Information

Team ID: 5

Member name	Department	IDs
عمر احمد فاروق احمد الجبالي	CS	20201701162
عماد محمود عمر الفاروق محمود	CS	20201700517
عمر احمد محمد عبدالمعطي	CS	20201700524
عمر عبدالفتاح عبدالسميع ابراهيم	CS	20201700538
محمد ابراهيم سعيد امام	CS	20201700650
محمد اسامه كمال يوسف	CS	20201700665

Model Analysis

Transformer model:

Parameters

Hyper-Parameters	***Values***
Epochs	2
Embedding dimension	32
vocab_size	65000
max_length	80
num_heads	4
ff_dim	32
droupout_factor	0.2

Metrics

Measures	***Values***
loss	0.4968
Train-Accuracy	0.8741
Test-Accuracy	0.84939

Bi-LSTM:

Parameters

Hyper-Parameters	***Values***
Epochs	2
Embedding dimension	10
vocab_size	18000
max_length	100
num_units	32
droupout_factor	0.5

Metrics

Measures	***Values***
loss	0.4673
Train-Accuracy	0.8402
Test-Accuracy	0.83885

Our work

In the preprocessing phase, we began by removing punctuation from the dataset. Following that, we eliminated stop words and identified each emoji within the dataset to understand its context. We further removed meaningless English words and translated the entire dataset. Additionally, stemming was applied to standardize word forms.

Average review length and vocabulary count were calculated as part of our exploratory analysis. Tokenization on the stemmed dataset was carried out using the Keras tokenizer. This involved splitting sentences into words and creating a dictionary of all unique words, assigning an index corresponding to their location in the dictionary. Subsequently, each sentence was converted into an array of integers, representing all words present in the sentence.

Sequence padding was then applied to ensure that each array represented a sentence of the same length.

Moving on to the LSTM model, an embedding layer was utilized to convert each word in the sentence into a fixed-length vector. A bi-directional LSTM layer was chosen over a unidirectional LSTM layer, as it takes into consideration both the context of the past and the future. This was followed by a dense layer with a 'relu' activation function and another dense layer with 3 units and a softmax activation function. The latter outputs the probability of each label, and the prediction is determined by the label with the highest probability.

Through experimentation with LSTM, it was observed that as the number of epochs increased, the training error decreased while the test error increased, indicating potential overfitting.

In Transformer, Taking our preprocessed data and putting it in a format that matches the desired transformer model input using Keras' built-in tokenizer. We used Keras to construct our transformer model layer by layer, achieving reasonably good accuracy. However, through hyperparameter tuning, we attained the best possible accuracy. After trying multiple different activation functions, we achieved our highest accuracy in this project, which is 0.84939 – a record-setting performance on the Kaggle competition leaderboard to that day.

In conclusion, the preprocessing phase played a crucial role in preparing the dataset for both LSTM and Transformer models. Punctuation, stop words, and meaningless English words were removed, emojis were identified for contextual understanding, and translation and stemming were applied for standardization. Exploratory analysis included calculating average review length and vocabulary count. Tokenization using the Keras tokenizer facilitated the conversion of sentences into integer arrays.

For the LSTM model, an embedding layer transformed words into fixed-length vectors, and a bi-directional LSTM layer captured both past and future context. Overfitting concerns were addressed through experimentation, revealing a trade-off between training and test errors with increasing epochs.

Transitioning to the Transformer model, the preprocessed data was formatted to match the model's input requirements using Keras' tokenizer. Layer-by-layer construction of the Transformer model, coupled with hyperparameter tuning, resulted in substantial accuracy improvements. The utilization of various activation functions, along with careful parameter adjustments, led to a noteworthy achievement of 0.84939 accuracy—a performance record on the Kaggle competition leaderboard up to that point. Overall, the combination of meticulous preprocessing and thoughtful model design and tuning contributed to the project's success in achieving top-tier performance.