Urdu Text Sentiment Analysis

1st Abdullah Tahir (385714-B)
SEECS
NUST H-12
Islamabad, Pakistan
abtahir.bscs21seecs@seecs.edu.pk

***Abstract*—Analyzing Urdu sentiment can provide valuable insights into social and cultural trends. It can be used to understand public opinion on current events, gauge reactions to cultural products like movies or music, and even track the spread of misinformation or hate speech online. This information can be crucial for researchers, journalists, and policymakers in understanding the social fabric of Urdu-speaking communities.**

***Index Terms*—Natural Language Processing, Sentiment Analysis, Urdu Language**

I. Introduction

Sentiment analysis, an important application of NLP, uncovers the underlying emotions behind text. While widely explored for languages like English, Urdu, spoken by hundreds of millions, presents a significant research gap. Analyzing Urdu sentiment offers a window into public opinion on social media, news, and reviews – crucial for understanding social trends, gauging reactions, and even tracking misinformation. However, Urdu sentiment analysis faces challenges due to language complexities and limited research. This paper delves into this domain to bridge the gap and advance Urdu sentiment analysis.

II. Dataset Acquisition

The dataset I chose was the Urdu text sentiment corpus [1] [2] formed from tweet data. This dataset contains tweets written entirely in Urdu classified in three different classes; P for positive, N for negative and O for offensive respectively. The dataset contains 999 entries in total; 499 for the N class, 480 for the P class and 20 for the O class respectively.

III. Pre-processing

Cleaning of the dataset is done to ensure smooth operation of the model. The dataset only contains 20 examples of the O class so that has been removed entirely so that it doesn't influence the model. It also had a null value which had to be removed. Some stop-words from the Urdu language are removed from the text, then the text is tokenized using the Tokenizer() class from Keras. The text is then converted into sequences and the sequences are padded. The training labels are converted into numeric values (1 for P, 0 for N) and the data is split into train and test data.

IV. Model architecture and hyper-parameters

For the model, primarily LSTMs were used in the architecture.

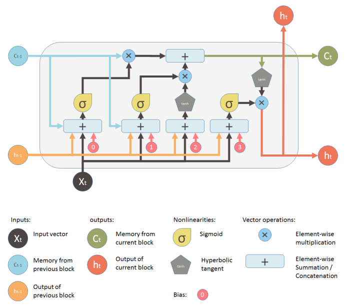

Fig. 1. LSTM

LSTMs (Long Short-Term Memory) are a type of RNN that tackles vanishing gradients. They use memory cells with gates to control information flow. Forget gates decide what to forget, input gates control new information, and output gates regulate what's passed on.

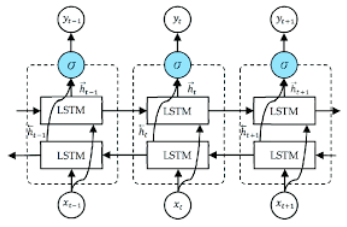

Fig. 2. Bi-directional LSTM

BiLSTMs (Bidirectional LSTMs) build on this by running two LSTMs; one analyzing the sequence forward, the other backward, capturing dependencies in both directions for even richer understanding.
The entire length of the word index from the tokenizer is used for the vocabulary size as it is not a large dataset. An embedding size of 64 was observed to be optimal for training the model. The model architecture starts with an Embedding layer which has an input dimension equal to the vocabulary size, output dimension equal to the embedding size and the input length equal to the max sequence length.
Next, a LSTM is used with 32 units, with a Dropout layer with a rate of 0.2 after that. Then, a BiLSTM is used with 32 units, after which a Dense layer with 64 units and ReLU is used. The output layer is a sigmoid layer.

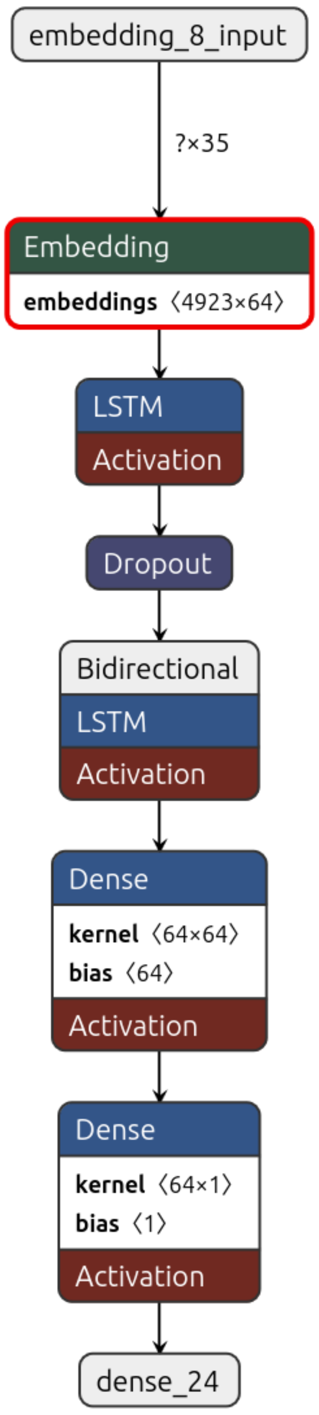

Fig. 3. Model architecture (visualization by Netron)

V. Experimentation

After careful experimentation, the model as described above was chosen. The Adam optimizer was used with an initial learning rate of 7.5×10^{-5}. A learning rate scheduler with a patience of 5 epochs and a reducing factor of 0.75 was used as it was observed to be the best performing. The training data was split into 80% training data and 20% validation data for testing. The model was then evaluated on the testing data from earlier.
After training many times, 40 epochs were chosen to be the right hyper-parameter for this model.

VI. Results

I've chosen the accuracy parameter to present for my results section. The accuracy of the final model after careful experimentation and hyper-parameter tuning was around 64%, as described in the code file.
We may notice from the results in the code file that the model tends to over-fit tremendously on the training data. I've tried to mitigate that factor by reducing the number of units in the LSTM layers, adding a Dropout layer, and also by introducing a learning rate scheduler. That has reduced the over-fitting by a small margin. This might be due to the fact that there is not a lot of data present. Pre-processing was done to the best of my ability to mitigate any faulty text from the corpus. Further work on this can be done by curating a much varied and larger dataset.

References

[1] M. Y. Khan, S. M. Emaduddin, and K. N. Junejo, "Harnessing english sentiment lexicons for polarity detection in urdu tweets: A baseline approach," in *2017 IEEE 11th International Conference on Semantic Computing (ICSC)*. IEEE, 2017, pp. 242–249.
[2] M. Y. Khan and M. S. Nizami, "Urdu sentiment corpus (v1.0): Linguistic exploration and visualization of labeled datasetfor urdu sentiment analysis." in *2020 IEEE 2nd International Conference On Information Science Communication Technology (ICISCT)*. IEEE, 2020.

Appendix

Link: https://github.com/Abdull0100/DL-Assignment3