Natural Language Processing – CS 4420

Assignment 2

Shrey Chhabra

Colab Notebook Link for BERT implementation –

https://colab.research.google.com/drive/12eIjTfEC0obRQ4ITrpGU5ZYrzaE8Jtto?usp=sharing

1. I've chosen the mental health tweets dataset for this assignment. The data exploration yielded a few key insights, which I quantified for analysis by taking into account the average number of words in statements to determine an appropriate sequence length for padding/truncation, identifying common keywords in each class using word clouds to find any patterns in the data, and checking label distribution to see if the dataset is balanced or imbalanced.

2. To prepare the data for training with the LSTM/transformer, I followed a few preprocessing steps, such as text cleaning by converting all text to lowercase and thus avoiding duplicate representations of the same word, removing URLs, hashtags, and special characters (since this is a dataset consisting of tweets), and removal of extra spaces/punctuation for consistency. Next, I used the keras tokenizer to convert every word into number representations, and set a maximum sequence length based on the earlier text length analysis. Finally, I used labelencoder to convert categorical labels into numerical values.

3. For the LSTM, the architecture consists of:
    a. An embedding layer to convert words into dense vector representations
    b. LSTM layer to capture sequential patterns in text
    c. Fully Connected (dense) layer to map LSTM outputs to classification labels
    d. Softmax activation to produce probabilities for each mental health class.

    For training the LSTM, the loss function is crossentropyloss since it's a classification task, adam optimizer (adaptive learning rate), and 10 epochs. I experimented with different learning rates and dropout regularization to fix overfitting.

    For the transformer, I used the BERT model. Unlike LSTMs, it captures bidirectional context, is already pretrained on large corpora, and based on the observations for this task, performs generally better albeit with a higher training ceiling. The architecture consists of 12 transformer layers, each layer capturing different linguistic features, a self attention mechanism to ensure the model focuses on relevant words in a sentence, and a classification head (fully connected layer+softmax) to map BERTs output to class probabilities. The optimizer is AdamW (weight decay regularization for stable

training), trained for 3 epochs on Colab (mainly due to hardware constraints) and warm up steps to gradually increase the learning rate to stabilize training.

4. Insights for the explainability analysis are provided in the notebooks along with visualizations. For the LSTM I have opted to use LIME and Occlusion (wherein a word is hid from the model to evaluate its impact on a sentence). For BERT I have used the same.

5. Observation on class signatures is also provided in the notebook with explainability analysis.

6. Challenges: Besides the time taken to train the actual model, a few things I noticed that were degrading the LSTM output was limited representational power, i.e., for longer sentences, it struggled to capture context, and could not handle out of vocabulary words well. The training was also highly sensitive to batch size, learning rate, sequence length, and required extensive tuning. Improvements for this could include using a BiLSTM or adding an attention mechanism to improve focus on important words. A hybrid model (LSTM+CNN) might also work well for sequence modelling with CNNs for local feature extraction to improve classification.

7. For the BERT model, training and inference was much slower than for the LSTM, and its self attention mechanism also made it harder to understand a few of the predictions it made. Like with the LSTM, I faced overfitting problems (presumably fine tuning on smaller datasets can lead to memorization rather than generalization).

8. Further comments on explainability are made inside the notebooks, with my reasoning for why the class signatures occur the way they do (via use of word clouds as well as the overlap in LIME and occlusion outputs).