SENTIMENT ANALYSIS

COMP 4332 Project 1

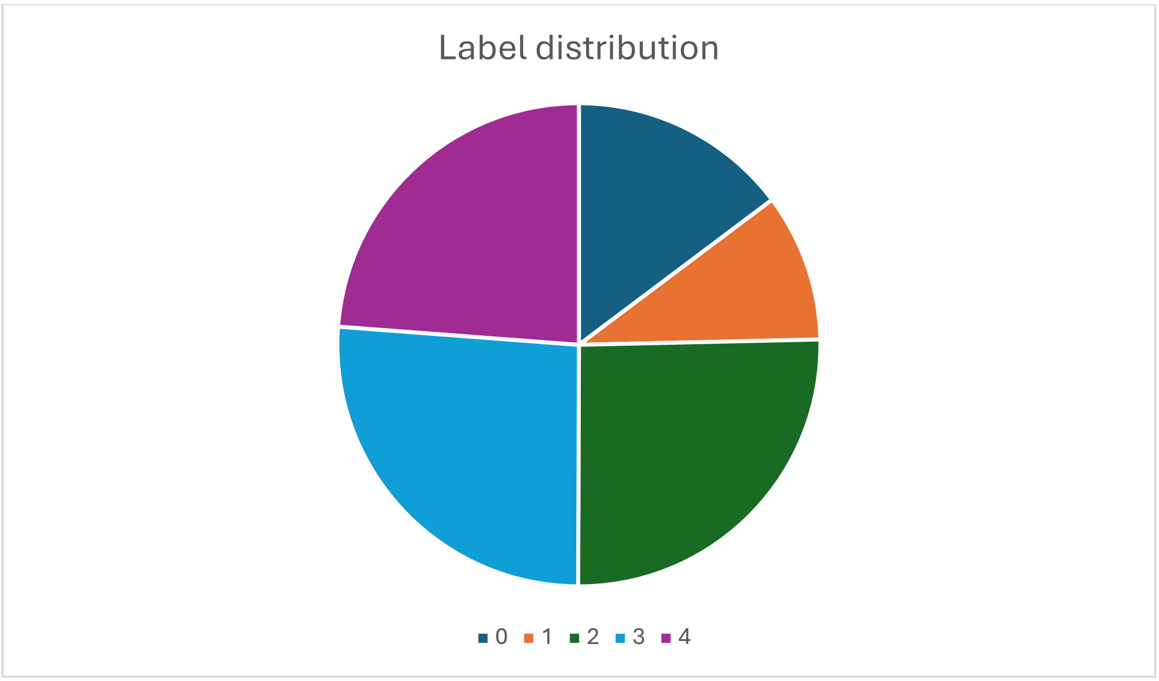
YUEN Man Him, Tsang Hing Ki

Group 31 | 2024 Spring

Before building the model:

We examine the dataset first before building our model as the Professor suggested. We found that the dataset is pretty unbalanced. Here is the distribution of training data:

0	1	2	3	4
295	198	508	523	476



We can see 2, 3, and 4 are three-quarters of the whole data but the total distribution of 0 and 1 is just a quarter. Therefore this may end up in the situation that model cannot predict correctly for label 0 and 1.

Algorithm:

We implement many algorithms like cnn, svm. Here we will introduce the three algorithm we used that achieve the best performance: Logistic regression, LSTM model, and BERT model.

Logistic regression:

This is one of the baseline algorithms but we improved it. For the data preprocessing, we remove stopword and make all characters lowercase before performing stemming so all words can be stemmed using the Porter algorithm. After that, we pass this preprocessed data to CountVectorizer which counts the frequency of words inside the sentence. Finally, this dataset is passed to the logistic regression model.

For the hyperparameters tuning, the min_df is set to 0.0 as low df words are usually keywords inside particular documents. We don’t want to miss it. For the max_df, we found that 0.2 out-perform other values like 0.1, 0.3, 0.5, 0.7 so we choose 0.2.

Second, the ngram_range is set to (1, 3) so both single words relationships and multiple words with order information are both extracted to the model. The model can learn both information for a better understanding of the sentence but not overfitting it using too large n value for n_gram.

LSTM model:

We figured out from some usual examples that RNN models are good for sentiment analysis so LSTM is one of our targets. For the data preprocessing, it is pretty much like Logistic regression, we perform stopword removal, change all characters to lowercase, stemming using Porter algorithm. Without passing to CountVectorizer, the preprocessed dataset is passed to our model. First, it is embedded and then passed into dropout layer first. Then it is passed into LSTM. Then it is passed to a ReLu and a fully connected linear layer for output.

For the hyperparameters, the embedded size so as the input size of LSTM is set to 64, as the size of words in one sentence is 50 after stopword removal and stemming. 64 is the nearest power 2 number.

The hidden_size is set to 128. We tested the LSTM for hidden_size as 32, 64, and 128. Eventually, we found out that 128 performed the best out of the three sizes. In our point of view, we think 128 is more than enough to store all the information needed for the long term. And due to the unbalanced distribution of data, eventually, long-term memory may cause incorrect predictions for labels 0 and 1. This affects the precision of labels 0 and 1, and the recall of labels 2, 3, 4.

BERT model:

We leveraged the BERT (Bidirectional Encoder Representations from Transformers) model for sequence classification due to its ability to understand contextual relationships within text. BERT's pre-training on extensive text corpora allows it to capture complex language nuances, making it ideal for tasks requiring deep language understanding.

The training process involved preparing and tokenizing the dataset, configuring the BERT model for sequence classification with five labels, and fine-tuning it on our dataset. We used the AdamW optimizer for training over ten epochs, monitoring accuracy to gauge the model's learning progress.

For the number of epochs, the original one is 10, but then a drop of valid accuracy is discovered. We then modified it to 3, attempting to avoid overfitting.

Summary of accuracy:

Model	Valid accuracy
CountVectorizer + Logistics regression	52.3%
LSTM	42.0%
BERT	56.8%

Our final model:

We choose BERT as our final model, as it has the highest accuracy.