CS779 Competition: Sentiment Analysis

Arnav Singla
roll no. 210188
{arnavs21}@iitk.ac.in
Indian Institute of Technology Kanpur (IIT Kanpur)

16 April 20234

Abstract

In this competition we had to predict the sentiment(-1, 0, 1) of a sentence using some model given alot of training data. This problem is basically a classification problem with 3 classes. My approach to this problem was using pre-trained glove embeddings to first convert the sentences into embeddings then passing them through a transformer encoder with multiple attention heads which was then pooled(max, min and mean). Lastly passing the pooled tensors through a linear layer to get the raw logits for each class (-1, 0, 1). Evalutation metrics being F1score : test_set_F1_score(0.646), train_set_F1_score(0.658).

1 Competition Result

Codalab Username: A_210188
Final leaderboard rank on the test set: 14
F1 Score wrt to the best rank: 0.943

2 Problem Description

The problem posed to us is a straightforward classification problem in which, given some text data written by a social media user, we must determine the sentence's sentiment as negative, neutral, or positive. Simply build a model that, given a few sentences, outputs -1, 0 or 1 based on the sentiment of the sentence.

3 Data Analysis

The train dataset given to us had a total of 92228 sentences for which we were given the sentiment, using this data we had to train the model. The average length of the tokenized sentences was around 13, median being 12 and the maximum length is of 258. The following graph shows the distribution of lengths of the pre-processed sentences. The top 5 most commonly occuring words in the train data were 'The', 'I', 'and', 'to' and 'was' which are stop words.

The test dataset had total of 5110 sentences for which we had to predict the sentiment using the model we trained on the training data. The average length of the pre-processed sentences was around 13, median was around 12, max being 141. The following histogram depicts the distribution of the lengths for each sentence in the test-set.

4 Model Description

Given the embeddings(dim=100) of the sentences, they are first passed through a transformer encoder block with 4 encoder layers containing 10 multi-head attention blocks each, yielding encoded tensors for each sentence in the corpus. I then used MaxPooling, MinPooling, and AveragePooling to reduce

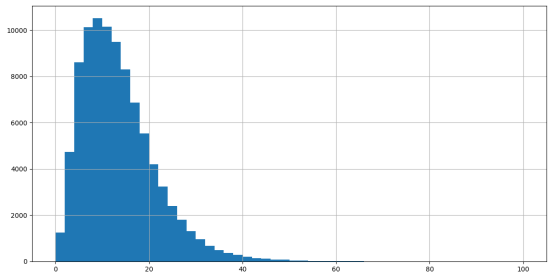

Figure 1: Distribution of lengths of the pre-processed sentences

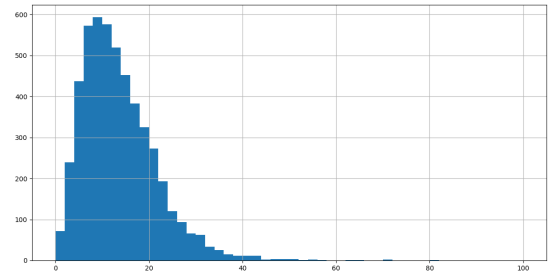

Figure 2: Distribution of lengths of the pre-processed sentences

the dimension from 3D to 2D, concatenating them along the last dimension to obtain a 2D tensor, which was then fed into a feed-forward neural network to obtain the raw scores for each class. On both training and validation data, this model performed exceptionally well. I am using CrossEntropy as my loss function.

5 Experiments

5.1 Data pre-processing

Given a sentence, I first converted it to lowercase and then tokenized the text using spacy tokenizer. I lemmatized each token and used its corresponding part-of-speech tag, both of which were obtained with spacy. Then, I padded or truncated each sentence based on a fixed maximum length (64 in this instance) and created a mask list (consisting of 0s and 1s) to indicate whether a token is a real token or padding token. Using pre-trained GloVe embeddings, the lemmatized tokens were converted to 100-dimensional tensors, and a separate embedding layer was constructed for the POS tags used in the forward pass of the model of the same dimension. I also employed positional encodings for each sentence, which consisted of a tensor with the same dimension as the lemmas embedding. The positional encodings and the lemma embeddings were added and then fed into the model along with the POS tags which were converted to tensor embeddings inside the forward pass.

5.2 Training Procedure

To minimise the loss function during the training phase, I used the Adam optimizer as my optimizer of choice. I trained the model for around 30 iterations, each of which took approximately twenty-five to thirty minutes for each iteration using a batch size of 128. The model was trained of Google Colab. The final learning rate i used was 7e-5 which gave the best results on the training set.

5.3 Hyperparameters

The hyper-parameters i tried to finetune were batch_size, learning_rate, input_dim and max_length. First i tried using max_length of 256, which i found to be redundant because most of the sentences had length of around 13-40, so then i shifted to max_length of 64. Initially i was using an embedding dim of 50 which was working fine but when i tired embedding dim of 100 it gave slightly better results with just a small bump in the training time. Finding the optimal learning rate was the most difficult part, i tried various learning rates ranging from 1e-3 to 1e-8 of which 7e-5 gave the best results. Finding the optimal batch_size was solely based on the computation power, i tried different values of batch sizes like 32, 64, etc but all gave similar results.

6 Results

Table 1: final model

Model	train_phase_score	test_phase_score	Rank_train	Rank_test
Transformer_1	0.658	0.646	8	13

7 Error Analysis

The transformer model was performing good on both the test and train sets, without over fitting to any kind of data, the model did perform poorly when given long sentences.

8 Conclusion

Transformers perform very well on Natural Language processing tasks like Sentiment Analysis, Part of Speech Tagging, etc as can be seen from the results as well. In view of future developments better pre-processing can be done using a self-build vocabulary and removing certain stop words like 'I', 'The', etc