CSE 582

Home Work # 2

Submitted by: Sadia Anjum Tumpa

ID: 947231364

Sentiment Classification using CNN and LSTM

The task for our HW2 is explained below:

We had to use Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) for sentiment classification. We are provided with the following link as our guideline that performs sentiment classification on Yelp academic review dataset using CNN:

https://towardsdatascience.com/sentiment-classification-using-cnn-in-pytorch-f ba3c6840430

I read through the article making sure that I understood the overall structure of CNN model and how the training is done.

Dataset:

The dataset is provided in the link below:

https://drive.google.com/file/d/1QJFsTPNMQ7D0BaqpJsVgAG7RyuO7Ylnt/view?usp=sharing

I downloaded the data 'yelp_academic_datset_review.json' file. This dataset is a subset of Yelp's businesses, reviews, and user data. It was originally put together for the Yelp Dataset Challenge which is a chance for students to conduct research or analysis on Yelp's data and share their discoveries. In the most recent dataset you'll find information about businesses across 8 metropolitan areas in the USA and Canada. The data provided is not in correct json format readable for python. Each row is dictionary but for it to be a valid json format, a square bracket should be at the start and end of the file with, being added at end of each rowTo accelerate the pipeline, I used the first 1 Million reviews for our experimentation and prepared a csv file that has 1 million rows containing different features.

Data exploration and label mapping:

I used the code from the guideline as a starting point to complete the tasks. I updated the code as required to fit for the given dataset and requirement of our sentiment classification. To start with I explored the data and looked at the number of samples and features in this dataset for further processing. Here ‘star’ is the rating from each user which will be mapped as class labels for our sentiment classification task. Out of 5 ratings, I mapped the review s following:

- If user gives ‘4’ or ‘5’ stars, that is mapped as ‘Positive’ review. [represented as ‘1’]
- If user gives ‘3’ stars, that is mapped as ‘Neutral’ review. [represented as ‘0’]
- If user gives ‘1’ or ‘2’ stars , that is mapped as ‘Negative’ review. [represented as ‘-1’]

The number of rows is not equally distributed across these three sentiments. According to the guideline, the problem of imbalanced classes won’t be dealt with, that is why, simple function to retrieve the top few records for each sentiment is written. In this experiment, we have taken 10000 of each class for further analysis. So, a total of 30000 reviews are taken into account for experimentation purposes.

Data Preprocessing:

Preprocessing the data to use for sentiment analysis is the first step to our task. There are several approaches that can be used to preprocess data. As mentioned in the guideline article, I then performed ‘Tokenization’ and ‘Stemming’ of as a part of data-preprocessing. I have avoided removing ‘stop words’ that includes ‘a’, ‘an’, ‘not’ etc. as removing these words can change the meaning of the review which is not suited for our classification task. ‘Tokenization’ means converting the sentence/text to array of ‘tokens’ or words. And then I performed ‘Stemming’ which refers to reduce the words into its ‘root’ form. Both of the preprocessing is done with ‘gensim’ library.

Then I split the whole dataset at 70-30 with ‘Training’ and ‘Testing’ set with the help of library function. Train data would be used to train the model and test data is the data on which the model would predict the classes and it will be compared with original labels to check the accuracy or other model test metrics.

Preparing Embeddings:

I have used ‘Word2Vec’ with embedding size = 500 to convert the tokens into numerical arrays and that’s how the input tensor is generated using ‘gensim’ library. Then the embedded tensor is saved in working directory to use for further experimentation.

Model Specification:

We have to use CNN and LSTM model for the classification task. First I experimented with CNN model and then I tried LSTM for the task.

The model definitions are mentioned in the relevant part of the code. ‘CNNTextClassifier’ is for CNN based classification and ‘LSTMTextClassifier’ is for LSTM based classification.

I trained for 30 epochs and tracked the accuracy of the training and testing set. Testing accuracy for CNN model reached ~72%.

CNN:

In a typical CNN architecture for sentiment analysis, the input text is first transformed into a sequence of word embeddings. The embedding layer is followed by one or more convolutional layers, each with a set of filters of varying sizes that are applied to the input embeddings. The resulting feature maps are then passed through a max-pooling layer to extract the most important features. The output of the pooling layer is then flattened and fed into one or more fully connected layers, followed by a final softmax layer for classification.

LSTM:

LSTMs, on the other hand, are designed to capture long-term dependencies in sequential data, making them well-suited for text classification tasks. In an LSTM architecture, the input text is first transformed into a sequence of word embeddings. The embeddings are then passed through one or more LSTM layers, where each layer consists of a number of LSTM cells. The output of the final LSTM layer is then fed into one or more fully connected layers, followed by a final softmax layer for classification.

Both CNNs and LSTMs can be effective for sentiment analysis, depending on the specific task and dataset. Examining their performance on different layers can provide insights into how they process information differently and can inform the choice of architecture for a given task. Ultimately, the choice between CNN and LSTM (or another architecture) will depend on a variety of factors, including the size and complexity of the dataset, the computational resources available, and the desired level of accuracy and interpretability.

Parameters for CNN and LSTM layers:

Using model.summary() I have found the number of parameters for CNN and LSTM models.

It is observed that the number of parameters are higher in LSTM compared to CNN since in LSTM there are more complex internal structures than CNNs, such as gates and memory cells, which require more parameters to learn.

CNN	LSTM
13445163	14370995

Time per epoch for CNN and LSTM layers:

LSTM is comparatively slow as it has more number of parameters to train than CNN which is also reflected from the time per epoch for CNN and LSTM model from experiments:

CNN	LSTM

134.2556060552597 seconds	230.19554567337036 seconds

Effect of changing activation function:

In the case of CNN and LSTM models, changing the activation function can affect the performance of the model. For example, using the ReLU function in CNN layers can improve the model's accuracy and speed up the training process. On the other hand, using the tanh function in LSTM layers can help to improve the model's performance by producing outputs with zero mean.