清華大學

Tsinghua University

Department of Computer Science and Technology

Natural Language Processing

Assignment 3

Sahand Sabour

2020280401

1 Task Definition

Opinion mining (sometimes known as sentiment analysis or emotion AI) refers to the use of natural language processing, text analysis, computational linguistics, and bio-metrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to the voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine. Industrial circles utilize opinion mining techniques to detect people's preference for further recommendation, such as movie reviews and restaurant reviews. In this assignment, we need to establish a sentiment classification model for the given sentence.

2 Basic Version Implementation

For this assignment, Pytorch was chosen as the deep learning framework for implementing this model. This implementation consists of the following steps:

2.1 Data Processing

2.1.1 Reading the SST dataset

We are provided with an the Stanford Sentiment Treebank datasets. These datasets have already been split into three sets: training, development (validation), and testing. Each line in these datasets is a sentence that is written in a tree form. For instance, given the sentence "Effective but too-tepid biopic", this sentence is provided as (2 (3 (3 Effective) (2 but)) (1 (1 too-tepid) (2 biopic))), which shows that the sentiment of each word (the number next to the word) as well as the sentiment for the whole sentence (the first number from the left). Initially, each line of the dataset was read, the first digit from the left was recorded as the label. Then, all the digits were removed using regular expressions and the remaining text was tokenized using the nltk library. Lastly, all the tokenized were put together to create the input text. Since the given files are provided in the text format and torchtext supports csv files rather than txt, the obtained input and labels were saved in csv files using the pandas library.

2.1.2 Creating Vocabulary Dictionary

The torchtext library was utilized to create the necessary vocabulary. By feeding in the created csv files, torchtext automatically create the requried vocabulary for both the input and the labels. Additionally, this library could also be used to automatically create the batch iterators for training, validation, and testing.

2.2 Model

For this assignment, a Bi-LSTM model was implemented. Firstly, let us explore the LSTM architecture. LSTM is an improved Recurrent Neural Network (RNN) model in a sense that it resolves the vanishing gradient issue that exists with normal RNNs. These type of models are useful where data is presented in sequences (either on a time-scale or in the case of this assignment, sentences). The feature that makes LSTMs superior to a normal RNN is the addition of a forget gate that enables the model to discard really old inputs. In the unidirectional version of these architectures, which were initially proposed and widely used, at each step, the model produces an output based on the previous outputs. However, in cases that information from the future (outputs from the steps after this step) are useful, the model would have no way to utilize this information. For instance, when analyzing a word in sentence in order to classify the sentence's sentiment, we need information from the words before and after each word. Hence, the bidirectional structure, which provides information from both the past and the future could be utilized. The Bi-LSTM model that is implemented in this assignment consists of two LSTM layers, three dropout layers, and a linear (FC) layer. When a sentence is given as input, each word of the sentence is transformed into its corresponding word vector based on the pre-trained embeddings. Accordingly, this input is fed to both the normal lstm layer and the reverse lstm layer. The output of these two layers is passed through two respective dropout layers and then fed to the second lstm layer. Accordingly, the corresponding outputs of these layers (outputs for the same steps) are concatenated. The final output would be taken from the last steps output. This output would then be passed through the third and final dropout layer and then fed into a linear layer to make the classification.

2.3 Training

The created train and validation batch iterators, with batch size of 64, are used for training. The models are trained for maximum of 10 epochs, with learning rate of 0.001. An early stoppage condition, which states that the model has overfitted if the validation results don't improve for 2 epochs, is also implemented.

2.4 Evaluation

The models' accuracy on the sentiment classification is calculated based on the test batch iterator. The obtained results for each of the required parameters are provided below.

# Hidden Layers	Hidden Dimension	Embedding	Dropout Rate	Accuracy (%)
1	256	None	0	35.83
1	256	None	0.2	36.88
3	256	None	0.2	37.28
3	512	None	0.2	37.73
3	512	GloVE	0.2	40.41

Table 1: Experimental Results of the Sentiment Classification Task

In the above table, None for the embedding section means that the no pre-trained word embeddings were used and the embedding weights for the vocabulary were initialized randomly. Based on the obtained results, it can be observed that with the addition of dropout and number of hidden layers, increase in the hidden dimension, and using pre-trained embeddings, the model's performance increases; the effect of using GloVE in the performance is the most noticeable.

3 Improved Version

The model proposed by [1] is implemented as the improved version for this assignment. They propose a Tree LSTM model that is able to outperform the simple LSTM model on the sentiment classification task by utilizing the word and phrase information of sentiment trees.

3.1 Methodology

Similar to LSTM units, the unit j in this architecture has an input gate i_j, output gate o_j, cell state c_j, and hidden state h_j. However, the gating vectors and memory cells are updated based on the values of their children. Accordingly, each unit j has a separate forget gate f_{jk} for each children k of its children. The authors claim that this allows the model to utilize information from the children, which is beneficial for word-level or phrase-level sentiment classification. The calculations for this architecture, given an input x_j to a unit (node) j with C(j) children, are provided respectively below:

$$
\begin{aligned}
h_j &= \sum_{k \in C(j)} h_k \\
i_j &= \sigma(W^i x_j + U^i h_j + b^i) \\
o_j &= \sigma(W^o x_j + U^o h_j + b^o) \\
f_{jk} &= \sigma(W^f x_j + U^f h_k + b^f) \\
u_j &= tanh(W^u x_j + U^u h_j + b^u) \\
c_j &= i_j \cdot u_j + \sum_{k \in C(j)} f_{jk} \cdot c_k \\
\hat{h}_j &= o_j \cdot tanh(c_j)
\end{aligned}
\tag{1}
$$

The above naming conventions are followed when writing the code for this implementation to ensure easier understanding and material consistency.

3.2 Training

First, the nltk library was utilized to parse the trees in the SST dataset and create the training, validation, and testing sets. However, instead of batch of inputs, one tree at a time was fed to the model. In addition, since the data had a tree structure, the labels were extracted during the training, where child nodes are explored and therefore, the labels are not directly available after parsing. The models are trained for maximum of 10 epochs, with learning rate of 0.001. An early stoppage condition, which states that the model has overfitted if the validation results don't improve for 2 epochs, is also implemented. The model's hidden dimension is set to 300 with a dropout ratio of 0.2. GloVE word vectors are used as they were shown to boost the performance.

3.3 Evaluation

The models' accuracy on the sentiment classification is calculated based on the test set of trees. The model was able to achieve 76.75% accuracy, which may be due to an unknown issue in the way accuracy is calculated. However, since at the time of writing this report, no issues were found, this value is reported as the obtained testing accuracy for this implementation, which is a considerable improvement in comparison to the base version.

References

[1] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075, 2015.