Sentiment Analysis of Movie Reviews: Positive or Negative

Sibei Zhou
Honghao YU
Chenkun Zhang

1 Introduction

In this project, our NLP task performs sentiment analysis on movie reviews, which is a binary classification task on the text with variable length. It can input an English sentence (movie review) and output the sentiment of that review: positive or negative.

The research value of this project is that humans can easily distinguish whether the feelings of a text are positive or negative, but machines cannot because it is difficult for machines to understand the meaning of sentences. But when faced with a large number of comments, it is impossible to use human resources to classify emotions sentence by sentence, so computers must be used to analyze the emotions of a large number of texts.

The dataset we will use is the Large Movie Review Dataset, which contains 50,000 reviews (negative 25,000 and positive 25,000) for training. For the test, we also have 50,000 reviews to predict. Depending on the number of stars shown on the file's name for each review, we can know the index of the review and its sentiment score from 1 - 10, and the review is negative with 1-4 stars and positive with 7-10 stars.

2 Related Work

In the paper "Large Movie Review Dataset" by Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts(Maas et al., 2011), created a dataset for binary sentiment classification which contains 25,000 highly polar movie reviews for training, and 25,000 for testing. And our project is an implementation of a paper that uses the database they provided to implement an analysis machine, in which we train with 25000 reviews with correct sentiment labels. After training, we input 25000 reviews text without labels and checked the output results to calculate the accuracy.

The baseline approach refers to the article "Sentiment Analysis with Python (Part 1)" posted by Aaron Kub (Kub, 2018), which provides a clear definition and a method that is easy to understand and implement so that we can quickly get the basic structure of the program. Based on the baseline approach, we improved the accuracy of sentiment analysis in the test set by changing its internal components, including word embedding, data training model, and classifier type used.

3 Approach

3.1 Data Preprocess

Read all the TXT files in the positive and negative dictionaries containing the reviews, and combine the 50k reviews into a single CSV file in order. The first 1 - 25000 reviews are the training dataset, and 25001 - 50000 are the test dataset. And the first half training and test dataset are negative, and the rest half is positive. Before inputting it into the program, we need to do some preprocessing on the reviews' text:

- First, remove all line breakers in the text;
- Second, remove all punctuation like ",.?![]+-=";
- Finally, Change the characters to lowercase.

So that we get reviews without noise.Z

3.2 Logistic Regression

The article "Sentiment classification on big data using Naïve bayes and logistic regression", introduces the Logistic Regression method. (Prabhat and Khullar, 2017)We choose to use the method described in the article "Sentiment Analysis with Python" (Kub, 2018) as our baseline (part of code

hyper-parameter C	Accuracy
0.05	0.88432
0.25	0.87996
0.5	0.87684
1	0.87404

Table 1: Different hyper-parameter C vs. accuracy in Logistic Regression model

and idea), which use the model of logistic regression and it consists of the following steps:

Create vectorization for each review. In baseline, it uses the simplest form that creates a very long vector with each dimension for a unique word in the corpus and transforms each review into a vector containing 0's and 1's, where 1 means that the word in the corpus corresponding to that vector appears in that review;

Build the Logistic Regression model by using sklearn module. We only need to care about the hyper-parameter C, which will affect the regularization. In our test case, we tested various C values to get different accuracy and finally chose 0.05 because it got the highest accuracy. Table 1 shows the different accuracy obtained from variable hyper-parameter C values.

Finally, train a model using the entire training set with 25,000 reviews and evaluate the accuracy of the 25,000 test reviews.

The highest accuracy achieved finally is 0.88432 with C = 0.05.

3.3 Simple RNN

In the article ”Sentiment Analysis using Recurrent Neural Network”(Kurniasari and Setyanto, 2020)They used deep learning and neural networks to measure the accuracy of the sentiment analysis classification model, using RNN and Word2vec methods. RNN means recurrent neural network, the simplest structure of RNN contains one input layer, one hidden layer, and one output layer (Figure 3). Each timestep takes the input layer and the hidden layer from the last timestep to calculate the output layer so that it allows the word in the input sequence to affect the words behind it in the sentence. However, when the weights accumulate, it might lead to the problem of gradient vanishing and gradient explosion and make the RNN lose long-term memory.

3.4 GRU and LSTM

To solve the problem of RNNs, we are going to use LSTM and GRU approaches. The GRU (GRU which is referenced in the article ”A GRU Model for Aspect Level Sentiment Analysis” by Xing, Y, and Xiao, C., in this case, (Xing and Xiao, 2019)) and the LSTM which be introduced in ”LSTM with sentence representations for document-level sentiment classification”(Rao et al., 2018)would provide a way to deal with the gradient vanishing and gradient explosion by their special gate structures, which can regulate the flow of information. So both of them can have long-term memory. These gate structures could learn which data in the sequence is important to keep and which is important to delete.

The main difference between them is the GRU is a concise version of LSTM since the LSTM has a very complex structure model. So the GRU has simplified the internal structure. And sometimes the GRU would have a better performance than the LSTM.

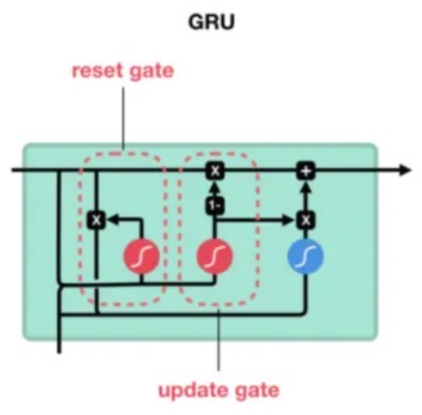

Figure 1: **GRU structure**

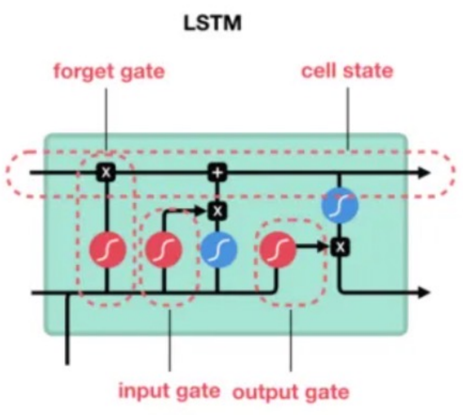

Figure 2: **LSTM structure**

3.5 Bi-LSTM and Bi-GRU

Both LSTM and GRU can only predict the output of the next time based on the timing information of the previous time, but in the problem, the output of the current time is not only related to the previous state but also may be related to the future state because if we want to predict the missing words in a sentence, we need not only to judge according to the preceding text but also to consider the content behind it.

Bi-directional LSTM and GRU consist of two of themselves superposed together, and the output is determined by the states of these two RNNs. The article "Fake News Detection using Bi-directional LSTM-Recurrent Neural Network" (Bahad et al., 2019)mentioned that Bi-LSTM will provide better accuracy than RNN. For each time t, the input will be provided to two RNNs in opposite directions at the same time, and the output is determined by the two unidirectional RNNs as Figure 3 showed. The article "Improving Sentiment Classification of Restaurant Reviews with Attention-Based Bi-GRU Neural Network"(Li et al., 2021) introduced that they try to combine Word2vec, Bi-GRU, and Attention methods to build a sentiment analysis model for online restaurant reviews. This is considered to improve the accuracy of the procedure.

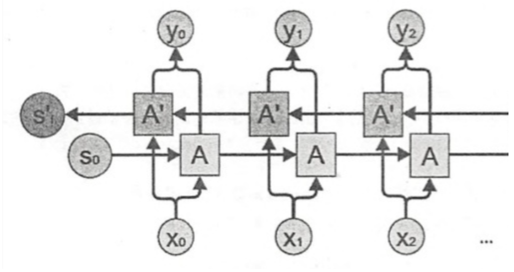

Figure 3: **Bi-directional RNN structure**

4 Experiments

4.1 Experiments setup

Since we have a big dataset, we decided to run our program by GPU instead of CPU. So we download CUDA and CUDNN to prepare an environment for the program. Then we set the torch and PyTorch, letting us use the neural network function. We also set the Matlab online model to make a graph in the ipynb.

And there are some requirements needed so that we can use a virtual environment to run the code: numpy, notebook, jupyter-contrib-nbextensions, jupyter-nbextensions-configurator, tqdm, torch, pandas, matplotlib, sklearn, scikit-learn, and nltk.

4.2 Data set

There are 2 data sets (train set and test set). Both the train set and test set have 12500 positive sentences in the dictionary named "pos, " and 12500 negative sentences in the dictionary named "neg," and each sentence is written as the file content in a single TXT file, with the file name in form 'indexOfReview + starsScore.txt,' so that we can know the star score for each review by reading their file's name. The number of star score is an important indicator for analyzing and evaluating the sentiment of the movie review, a review with many stars between 1-4 is a negative review, and a review with several stars between 7-10 is a positive review, and reviews with 5 or 6 stars were left out.

4.3 Implementation

We write the code our self except the baseline part is copied by the article introduces the method of using Logistic Regression, but the main part of the implementation of the RNNs we simply call the function in torch.nn module to create the models, and the part of code about creating the models of RNNs we refer to the code of Programming Homework 3, mainly the class LSTMTaggerModel and class LSTMTagger.

4.4 Evaluation

We use the accuracy to evaluate the different approaches, which is calculated by dividing the number of the reviews which are classified correctly by the total number of the reviews input.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

4.5 Method compare and Results

We compare all the approaches above Logistic Regression (baseline), simple RNN, LSTM, GRU, Bi-LSTM, and Bi-GRU.

Figure 4 shows the highest accuracy and fastest running time spent on each epoch we got in methods with different RNN models in all tests. Compared to the baseline with the Logistic Regression model which got an accuracy of 0.88432, it achieves improvement at methods with the GRU model and Bi-directional GRU.
We find that the Bi-GRU would have the best performance with the highest accuracy at 88.7540. Although the method with the Logistic Regression

model is the baseline, it is not the worst method with great accuracy and very high time efficiency. And we also focus on the accuracy-increasing rates in RNN models including Simple RNN, LSTM, Bi-LSTM, GRU, and Bi-GRU. Figure 4 shows the relation between the accuracy of those different models and a variable number of the epoch they have tokens between 1-30.

It can be clearly seen from Figure 5 that: the accuracy of simple RNN is stable around 50 and did not show any increase before the 30 epoch, and the accuracy of simple LSTM and Bi-LSTM did not rise rapidly with the increase of epoch but hovered around 50 in the early stage when the epoch reached an uncertain value (usually greater than 10), it suddenly began to surge but tended to balance around 85. GRU and Bi-GRU showed an obvious upward trend at the beginning, then reached the peak in a few iterations, then began to decline slightly, and then tended to balance around 87-88.

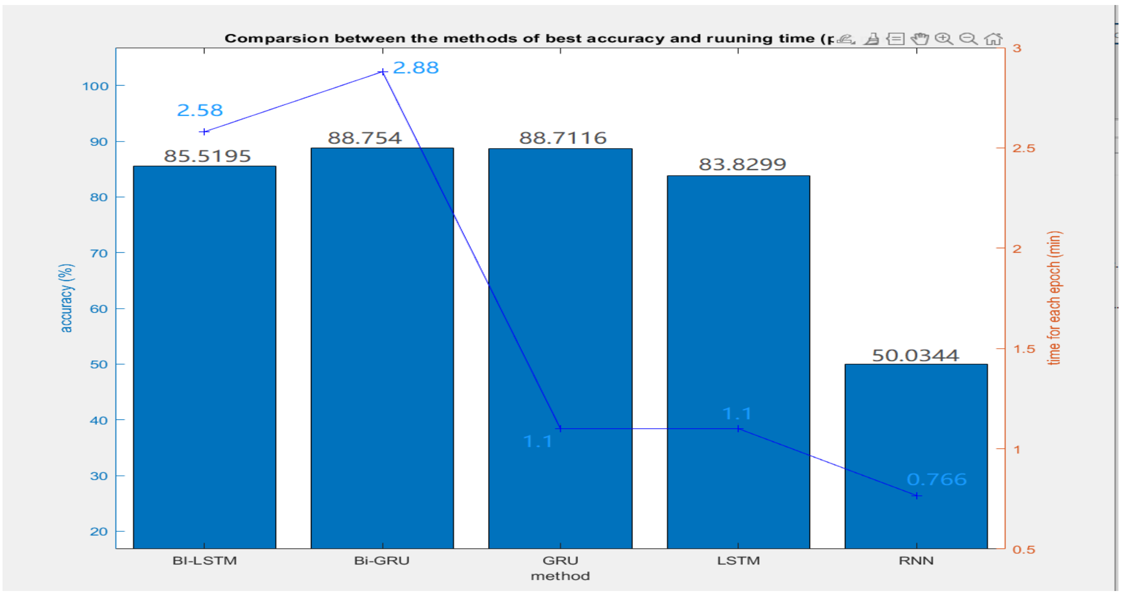

Figure 4: **Comparison between the methods of the best accuracy and running time**

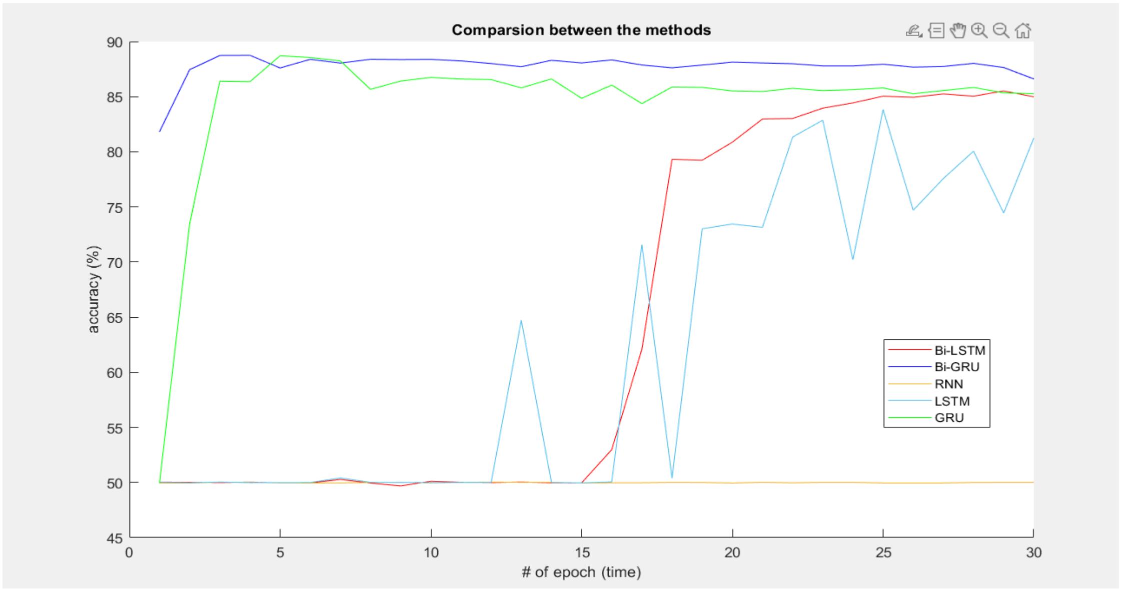

Figure 5: **Comparison between the methods**

Interestingly, the accuracy of LSTM on the test set is almost unchanged while the accuracy on the training set fluctuates in a small range of around 50, while the loss on the training set is always stable at 0.0108.
We have tried to change the learning rate of the optimizer and loss function, increase the number of hidden layers of LSTM, and reduce/expand the batch size, but they have no effect at all. We try to use a smaller train and test dataset but it did not work and even go a worse result, so we finally think that it is the characteristic of the LSTM model used for large databases: sudden increase of accuracy in the uncertain epoch. It is not that they will not learn, but they need a larger epoch.

5 Conclusion

5.1 Learn from experiments

- Be familiar with several common text preprocessing methods before text classification: change all characters to lowercase, remove punctuation marks, remove line breaks, delete stop words (we've tried that before but got worse results for unknown reasons), word form restoration (not actually used yet)

- Be familiar with the use of multiple RNNs for text classification tasks, as well as their respective advantages and disadvantages.

- Learned how to judge the fitting ability of neural networks and the corrective measures in case of underfitting and overfitting.

  Iterating the same data (identical data in each batch), observe the change of loss and accuracy of the model on the test set. If the loss and accuracy remain unchanged for a long time, there is an underfitting. Possible solutions include increasing the number of layers of the neural network, or increasing the number of neural units of the neural network, modifying the weight initialization scheme, and adjusting the learning efficiency.

5.2 Problem fixed

- It could be used at any text binary classify assignment, but the hyper-parameters might need some changes to satisfy different datasets and specific assignments.

- It also can be useful for the text multi-classification task by setting the output size and adding the softmax function after the output layer, so that the classifier can work on multiple classification tasks.

- It can be modified to be used in image classification assignment, which can convert the picture into a pixel sequence, and input the pixel value of each line of the picture into each batch in turn to the RNNs and finally get the classification results of pictures.

5.3 Useful additions

- Word form restoration in data preprocessing.

- Augment the RNNs implementation with the correct attention module might improve the performance, cause the attention module can not only help the RNN model to make better use of the valid information in the input, but also provide a certain ability to explain the behavior of the neural network model.

- Method with BERT model might improve the accuracy.

6 Contribution

- Sibei Zhou:

  Write the code about the RNNs model. Adjust the hyper-parameters of the RNN to optimize the model, try various optimizers and adjust the learning rate. Help to complete the report.

- Honghao Yu:

  Discussing with the group, setting the program environment in a speedway. Writing the report and the slide. Make the graph and try to get the conclusion for the result. Trying to improve the program's accuracy and debug the code.

- Chenkun Zhang:

  Find the essay and reference data for the project. Help improve the Logistic Regression baseline. Discussion and implementing the LSTM GRU approach. Complete the report and the presentation slides.

References

Pritika Bahad, Preeti Saxena, and Raj Kamal. 2019. Fake news detection using bi-directional lstm-recurrent neural network. *Procedia Computer Science*, 165:74–82. 2nd International Conference on Recent Trends in Advanced Computing ICRTAC -DISRUP - TIV INNOVATION , 2019 November 11-12, 2019.

Aaron Kub. 2018. Sentiment analysis with python (part 1). *Towards Data Science*.

Lilis Kurniasari and Arif Setyanto. 2020. Sentiment analysis using recurrent neural network. *Journal of Physics: Conference Series*, 1471(1):012018.

Liangqiang Li, Liang Yang, and Yuyang Zeng. 2021. Improving sentiment classification of restaurant reviews with attention-based bi-gru neural network. *Symmetry*, 13(8).

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Anjuman Prabhat and Vikas Khullar. 2017. Sentiment classification on big data using naïve bayes and logistic regression. In *2017 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–5.

Guozheng Rao, Weihang Huang, Zhiyong Feng, and Qiong Cong. 2018. Lstm with sentence representations for document-level sentiment classification. *Neurocomputing*, 308:49–57.

Yongping Xing and Chuangbai Xiao. 2019. A gru model for aspect level sentiment analysis. *Journal of Physics: Conference Series*, 1302(3):32042.