FINAL REPORT TASK 8: LSTM, BI-LSTM AND ATTENTION MODEL

Qing Wang, Tianqi Wu
{qwang55,twu38}@illinois.edu
University of Illinois at Urbana-Champaign
Champaign, IL 61820

December 15, 2019

1 Introduction

Sentiment analysis is a popular topic in Natural Language Processing (NLP). It measures the attitude of customers towards products or services. Tremendous comments and reviews are generated every day and it is impossible to analyze them manually. Automated sentiment analysis with high accuracy help companies with decision making the product improvement. In this work, we will compare different approaches of sentiment analysis on Yelp Review Polarity [1] data set. Methods compared include logistic regression (TF-IDF + LR), Bag of words (BOW), Recurrent Neural Network (LSTM, bi-LSTM), Attention model (LSTM + attention) and Transformer (BERT).

2 Related Work

2.1 Traditional Machine Learning

Sentiment analysis is also known as text classification and it can be done with traditional machine learning methods. Support vector machines (SVM), Random Forest and Logistic Regression are some of them which are still commonly used [2]. They are easy to implement with the help of handy tools like scikit-learn [3]. One could get relatively good result in a short time. When dealing with text, we need to somehow get numeric representations of the words so that calculations can be done. To use those machine learning methods, we could transform the documents to matrices using Term Frequency Inverse Document Frequency (TF-IDF) [4].

There are other basic models for document classification like Bag-of-Words (BOW) [5] which utilizes all of the tokens in the sequence but completely ignores the temporal information. It is able to deal with inputs with varying length but not able to capture the sequence information. Therefore, those methods are not as effective as ones mentioned below.

2.2 Recurrent Neural Network

Recurrent neural networks (RNN) [6] are relatively recent methods which could effectively solve various NLP problems. Texts are represented by word embeddings, which are just another vector representation of fixed length. RNN uses actual text sequences as inputs and it is able to maintain the short-term temporal information. However, it suffers from vanishing and exploding problems, which are typical in deep learning.

To better capture the long-term temporal information, Long Short Term Memory (LSTM) [7] is introduced by adding special units to control the flow of information. Unidirectional LSTM preserve only the information from past since its inputs only contain past information. Bidirectional LSTM (Bi-LSTM) [8], however, takes both past and future information as inputs. Thus, it is even more effective considering its ability to learn information from both past and future.

2.3 Attention Mechanism

There are other variants of RNN like Seq2Seq [9], which uses encoder-decoder architecture. However, those variants of RNN use fixed-length context vectors and it is hard to remember longer sequences. Attention mechanism[10] solves this problem and it is first introduced by Bahdanau to solve the task of machine translation. It measures the relevance of source sentence to target word. Neutral networks with attention mechanism are able to disregard noise and focus on the most relevant part. Despite the effectiveness of RNN with attention mechanism, it adds much more weights to the model and it may take significantly longer training time. Hence, Transformer [11] is introduced and it is built entirely on self-attention mechanism without using time-consuming recurrent architecture. It is the most recent approach for various tasks of NLP and it performs especially well for sentiment analysis.

3 Dataset

The data set used for experiments is Yelp Review Polarity [1]. It consists reviews from Yelp with positive and negative polarity with label of 2 and 1 respectively. For each polarity, there is 280,000 training samples and 19,000 testing samples. In total, there are 560,000 training samples and 38,000 testing samples.

During data preprocessing, we remove all emojis, numbers and special characters. Essentially, only texts are kept in this case. Other common techniques like tokenization and lower case transformation are used. To prepare data for RNN, we need better understanding of data to determine the length of sequence used in models. The mean length of reviews is 134 with standard deviation of 124. Further analysis indicates that 96.35% of the data set is contained within the most common 8000 words. Over 88% (345k) of the unique tokens occur between 1 and 10 times while only 1% occur more than 100 times each. Finally, we convert the tokens to ids for later use. Index 0 is reserved for unknown token.

4 Model Architecture and Objective Function

There are 6 models implemented in total in this paper. Details are provided below:

4.1 TF-IDF+LR

To prepare data for logistic regression, TF-IDF is used to transform the texts to vector representations. Then, logistic regression could be done to classifier the sentiments.

4.2 BERT

Bidirectional Encoder Representations from Transformers (BERT) [12] is based on Transformer model [11]. We fine-tune the released pretrained model `bert-base-uncased` so that it works on our text classification case.

4.3 BOW

For Bag of Words model, we first use a for loop iterating through the input, which are batches of sequence of tokens. Then, we feed each of them to an embedding layer and average the results. It will return tensor with shape `sequence length` by `embedding size` Afterwards, we stack the results so that it returns tensor with shape `batch size` by `embedding size`. The output is then passed through a linear layer, a ReLU operation, dropout and a output layer. The objective function used is `nn.BCEWithLogitsLoss()` provided by Pytorch:

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = -w_n \left[y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n)) \right] \tag{1}$$

Where N is batch size, σ is sigmoid operation.

4.4 LSTM, Bi-LSTM and Att-LSTM

For LSTM, input with shape `batch size` by `sequence length` is first passed through embedding layer. The output tensor with shape `batch size` by `sequence length` by `embedding size` is then transposed to shape `sequence length` by `batch size` by `embedding size` and passed through `nn.LSTM()` layer. The output tensor has the same shape as the input tensor for this `nn.LSTM()` layer. Then, we only take the last time step of the output with shape `batch size` by `embedding size` and pass it into the output layer.

Bi-LSTM and Att-LSTM are built based on similar model architecture as LSTM. The LSTM layer of Bi-LSTM is declared with parmeters `nn.LSTM(bidirectional=True)`. Hence, we have double number of hidden units to pass into the output layer.

For Att-LSTM, we add an additional unit after `nn.LSTM()` layer to perform the attention mechanism. The final hidden state with shape `num_layers` by `batch size` by `embedding size` is viewed as shape `batch size` by `embedding size` by `num_layers`. We then perform batch matrix multiplication of transformed final hidden state with output of `nn.LSTM()` layer. Afterwards, we perform softmax operation to the output and batch matrix multiplication with transposed output from `nn.LSTM()` layer to get the context vector with shape `batch size` by `embedding size * num_layers`. Finally, we pass the context vector into the ouput layer.

Note that same objective function `nn.BCEWithLogitsLoss()` in Equation 1 are used for BOW, LSTM, Bi-LSTM and Att-LSTM. Embedding layer and LSTM layer has the same number of `hidden units`.

5 Hyperparameters

The vocabulary size used for TF-IDF+LR is 4000 because larger vocabulary size would cause memory issue. We use default parameters for BERT except batch size of 110 and testing sequence length of 450. The hyperparameters below are the same for different settings of experiments for BOW, LSTM, Bi-LSTM and Att-LSTM.

- Batch size: 200
- Learning rate: 0.001
- Vocabulary size: 8000. It contains 96.35% of the data set.
- Training sequence length: 150. Mean length of reviews is 134 with standard deviation of 124.
- Testing sequence length: 450. Longer testing sequence length yields better performance.

Besides, four settings of hyperparameters are tested. Table 1 shows the settings and best setting is bolded.

Settings	hidden unit	num of layers	dropout rate
1	300	1	0
2	300	2	0.4
3	**500**	**2**	**0.4**
4	800	2	0.4

Table 1: Different Settings of Hyperparameters

6 Training Methods

After getting preprocessed data from Section 3 and defining relevant model architectures, we can start training models. Data is first processed with random permutation and fed into the networks. The output loss is then used to perform back propagation. Adam optimizer is used as optimization algorithm. Learning rate of the optimizer is kept constant at 0.001 with no weight decay. Gradient clipping is used with clip value of 2.0. Also, early stopping is adopted during training. We stop the training and save the model if test accuracy starts dropping by one epoch. It is necessary since the only regularization technique we used is one layer of dropout. It helps preventing overfitting in practice.

7 Result

7.1 Comparison with Baseline

Experiment results are illustrated in following tables. We compare all three models with baseline models BOW and TF-IDF+LR. The test accuracy of baseline models is quite high already, especially for TF-IDF+LR. It is reasonable as we have a lot of training data. It also illustrates that we can achieve acceptable sentiment classification results using only word distribution. However, we definitely need sequential information to better distinguish more complicated expressions such as phrases or metaphors.

All three models improve around 3% over baseline models which illustrates that word order information contributes to sentence sentiment classification. Att-LSTM with 500 hidden unites and 2 layers achieves best test accuracy among three models.

7.2 Comparison of Settings

We also see that test accuracy is generally higher with more number of layer and the number of hidden units actually does not make much improvement for our analysis. A possible reason is that for a binary classification on sentiment, word representations need not to be such complicated to distinguish two classes. If the number of hidden units is too large, it may include more noise that makes classification harder for deep models. This is validated by our experiment as we achieve best classification result with 500 hidden units instead of 800.

7.3 Comparison of Models

7.3.1 LSTM & Bi-LSTM

LSTM and Bi-LSTM achieve very similar results across all settings while Bi-LSTM is slightly better in most settings. A possible reason is that our task is sentiment analysis. Sentence sentiments is mostly encoded in vocabularies and little in word order while the additional information Bi-LSTM encodes is the reverse word order information. It generates slightly better result since it captures sequential order of both directions, yet it does not improve model's prediction for sentence sentiment by a lot. For our setting with 300 hidden units and 1 layer, Bi-LSTM achieves best result yet performances of all models are very similar here.

7.3.2 Att-LSTM

There is larger improvement for Att-LSTM than Bi-LSTM and LSTM as number of layers increases. Since Att-LSTM uses attention layer to generate sentence vector representation from word representations, it encodes information about the sentiment focus in sentence when trained with sentiment loss. Thus, it has a better ability to distinguish useful sentiment information from noise as model becomes deeper and generates better result. Att-LSTM has best performance in settings with 300, 500 and 800 hidden units and 2 layers, which illustrates its advantage in capturing useful information in complicated models.

7.3.3 BERT

Our overall best model is fine-tuned BERT with 500 hidden units. Since BERT is pre-trained on a much larger corpus of various topics, it carries better word representations than our three models. It also has a more complicated attention mechanism and more layers of attention which effectively capture sentiment information of sentences during fine-tuning. We conclude that to achieve a better result for sentiment analysis, we not only need as good word representation as possible, but also an effective mechanism such as attention to capture sentiment focus of sentences.

Models	Train Accuracy	Test Accuracy	Epoch	Time/epoch
LSTM	96.43	95.75	4	10 min
Bi-LSTM	96.14	**95.97**	3	20 min
Att-LSTM	95.78	95.96	3	10 min

Table 2: Performance Comparison of setting 1 (hidden unit:300, num layer:1)

Models	Train Accuracy	Test Accuracy	Epoch	Time/epoch
LSTM	96.59	95.91	4	20 min
Bi-LSTM	96.16	95.91	3	60 min
Att-LSTM	97.10	**96.17**	4	20 min

Table 3: Performance Comparison of setting 2 (hidden unit:300, num layer:2)

Models	Train Accuracy	Test Accuracy	Epoch	Time/epoch
LSTM	97.04	95.99	4	1.1 h
Bi-LSTM	97.41	96.08	4	3.5 h
Att-LSTM	96.63	**96.20**	3	1.1h

Table 4: Performance Comparison of setting 3 (hidden unit:800, num layer:2)

Models	Train Accuracy	Test Accuracy	Epoch	Time/epoch
BOW	89.46	88.43	1	5 min
TF-IDF+LR	92.58	92.63	NA	3 min
LSTM	96.21	95.98	4	0.9 h
Bi-LSTM	95.95	95.99	3	3.3 h
Att-LSTM	95.45	96.22	3	0.9 h
BERT	NA	**97.52**	2	2 h

Table 5: Performance Comparison of setting 4 (hidden unit:500, num layer:2)

8 Computational Cost

In total, we have used approximately 150 Blue Waters GPU hours and 20 Google Colab GPU hours for various testing of hyperparameters. Only interesting results are reported in the paper.

9 Description of Code

BERT is implemented on Google Colab with the help of package `Simple Transformers` [13]. We fine-tune the hyperparameters including `batch_size` and `sequence_length`. A finished copy is named as `BERT.ipynb` and exported as pdf with name `BERT.pdf`.

Codes of other models are written in Blue Waters. Data preprocessing and BOW model are implemented with similar fashion of HW5. LSTM, Bi-LSTM and Att-LSTM models are implemented based on the github repo `nlp-tutorial` [14]. We modify the model architecture so that they can be used on the Yelp Review Polarity [1] dataset. We also try our best to tune the hyperparameters to improve the model performance. Further details can be found on `README.md`.

References

[1] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification, 2015.

[2] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, Donald Brown, Laura Id, and Barnes. Text classification algorithms: A survey. *Information (Switzerland)*, 10, 04 2019.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[4] Juan Ramos. Using tf-idf to determine word relevance in document queries. 01 2003.

[5] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: A statistical framework. *International Journal of Machine Learning and Cybernetics*, 1:43–52, 12 2010.

[6] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[8] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November 1997.

[9] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[13] Thilina Rajapakse. Simple transformers. `https://github.com/ThilinaRajapakse/simpletransformers`, 2019.

[14] Tae Hwan Jung. nlp-tutorial. `https://github.com/leifanus/nlp-tutorial`, 2019.