CS395T: Neural Networks for Sentiment Analysis Project Report

Zeyuan Hu
Computer Science Department
University of Texas at Austin
Austin, Texas
iamzeyuanhu@utexas.edu

Abstract

In this project, I implement a Feedforward Neural Network based on averaged word vectors over the input sentence and a convolutional neural network (CNN) to perform the sentiment analysis on the sentences from Rotten Tomatoes introduced by Pang and Lee (2005).

1 Collaborators

Zhan Shi, Danlu Wang

2 Introduction

Sentiment analysis is an umbrella term for several closely-related tasks. One of them is to classifiy the sentences as either negative or positive. Traditionally, Naive Bayes or SVM bag-of-words models are well-suited for those tasks. However, neural network approaches including convolutional networks (Kim, 2014) and feedforward networks (Iyyer et al., 2015) can substitute those traditional models to achieve similar performance. In this project, I implement a feedford network and a convolutional neural networks for the sentiment analysis. In addition, I explore various architectural and parameter options to tune the systems to gain good classification performance.

3 Implementation Details

3.1 Feedforward Neural Networks

The implementation of the feedforword neural network (FFNN) follows closely to the architecture of the deep averaging network (DAN) described by Iyyer et al. (2015). In the paper, the authors break down the architecture into three simple steps: 1. take the average of word embeddings of words in a given sentence 2. pass the average into a FFNN with one or more hidden layers 3. perform classification on the final output. I follow those three steps closely and my network contains one hidden layer. Essentially, I calculate $P(\boldsymbol{y}|\boldsymbol{x})$ in the network. `one_best` contains the classifiction result of the sentences in the computataion graph `graph`. I implement batching and during each epoch, I shuffle the training examples at the very beginning. For the optimizer, I experiment with both the Adam and the stochastic gradient descent (SGD). The parameter tuning and the impact of the optimizer to the performance will be disccued in details in the experiment section.

3.2 CNN

The implementation of CNN follows Zhang and Wallace (2015)'s empiricial study of one-layer CNN hyperparameter setting on sentence classification. In my implementation, I use one convolution layer with multiple filter widths and feature maps and then I use 1-max pooling suggested in the paper. The code structure is divided into five steps: 1) construct embedding layer by looking up pre-trained word embeddings, 2) create convolution layer and implement 1-max pooling strategy, 3) combine all the pooled features, 4) add dropout, and 5) make final predictions. The rest of work is to focus on the hyperparameter setting, which includes filter region size(s) (i.e., `filter_sizes`), the number of filters to use with each size (i.e., `num_filters`), the activation function(s) (i.e., `activation_func`), and so on.

4 Experiments

The experiment is carried out in a similar way as Zhang and Wallace (2015)'s. For each exploration task, I report the mean, minimum, and maximum average accuracy values observed over three replications.

Table 1: FFNN Configuration

Description	Values
batch size	10
number of hidden layer	1
number of hidden units	150
number of epochs	50
initial learning rate	0.1
decay steps	10
learning rate decay factor	0.99
optimizer	SGD

4.1 Feedforward Neural Networks

For the FFNN, I explore the impact of the architecture and the choice of the optimizer to the system performance. Without special note, the experiments is based on the hyperparameter setting in table 1.

Optimizer: I compare the performance between Adam and SGD, which is controlled by the environment variable `OPT` in the code. One surprising finding is that whether supplying `global_step` argument inside the function call `apply_gradients` can have impact on the performance of the system . I use tensorflow version 1.3.0 and the result is shown in table 1.

Table 2: Optimizer Performance with "global_step"

Optimizer	Arg	Accuracy (training)	Accuracy (dev)
SGD	Yes	76.53 (76.44, 76.59)	74.83 (74.67, 75.05)
ADAM	Yes	77.58 (76.72, 78.23)	**74.98 (74.48, 75.33)**
SGD	No	**81.72 (81.15, 82.70)**	74.42 (74.20, 74.77)
ADAM	No	50.06 (50.06, 50.06)	49.06 (49.06, 49.06)

As shown in the table, `global_step` does not have impact for SGD measured by the accuracy on the dev set. Adam has similar performance as SGD with `global_step`. However, without `global_step`, Adam performs poorly and SGD has significant increase in training set accuracy compared with `global_step` unset case.

Architecture: The architecture of FFNN is determined by the number of hidden layers. In this experiment, I explore the FFNN with zero and one hidden layers. The result is shown in table 3.

As shown in the table, there is a significant performance gain when we increase our hidden layer from zero to one. However, with the given configuration, two layers architecture has notable performance drop. This suggests that hyperparameter setting and architecture has to be carefully picked to achieve good performance.

Table 3: FFNN architecture

Hidden layers	Accuracy (training)	Accuracy (dev)
0	71.73 (66.66, 77.40)	69.48 (65.76, 73.92)
1	**81.72 (81.15, 82.70)**	**74.42 (74.20, 74.77)**
2	51.22 (49.94, 52.05)	51.88 (50.38, 53.00)

Table 4: CNN Configuration

Description	Values
input word vectors	GloVe-300
filter region size	(3,4,5)
feature maps	128
activation functions	ReLU
pooling	1-max pooling
dropout rate	0.5
$l2$ norm constant	0
optimizer	Adam
number of epochs	20
batch size	64

4.2 CNN

For the CNN, I explore the impact of the hyperparameter setting to the system performance. Without special note, the experiments is based on the configuration in table 4. Figure 1 shows the performance of the CNN versus the number of epochs.

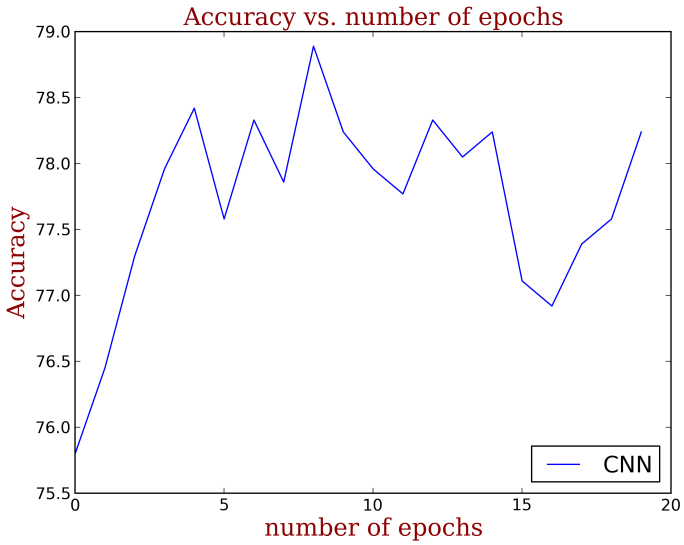

Figure 1: Accuracy vs. number of epochs in dev set

In the following experiments, I train the CNN for 5 epochs. **Filter region size(s):** In this experiment, I examine whether the filter region sizes will have the impact on the CNN performance. The choice of filter region sizes are picked based on

Zhang and Wallace (2015)'s empirical study. The result is shown in table 5.

Table 5: Filter region sizes and CNN performance

Filter region sizes	Accuracy (dev)
(3,4,5)	**77.52 (77.39, 77.77)**
(7,7,7,7)	77.20 (76.92, 77.58)
(7)	76.92 (76.08, 77.39)

I try different region sizes and find out that the impact of the region sizes to the performance is not significant. Even we have one region size 7, there is almost no performance drop.

Feature maps: Zhang and Wallace (2015) suggests that the number of feature maps for each filter region size is ranged between 100 and 600. This is quite large range. I experiment with three feature maps with one value between 100 and 200, one value between 200 and 400, and one value between 400 and 600. The result is shown in table 6.

Table 6: Feature maps and CNN performance

Feature maps	Accuracy (dev)
128	77.52 (77.39, 77.77)
300	**77.86 (77.67, 78.05)**
600	77.61 (76.83, 78.52)

Activation functions: In this experiment, I experiment with two activation functions: `tanh` and `relu`. The result is summarized in the table 7.

Table 7: Activation functions and CNN performance

Activation function	Accuracy (dev)
`tanh`	77.49 (77.30, 77.77)
`relu`	**77.52 (77.39, 77.77)**

As shown in the table 7, the choice of the activation function does not have impact on the CNN performance.

5 Conclusion and Future Work

In this project, I implement a feedforward neural network and a CNN for sentiment analysis. I explore the impact of architecture choice and hyperparameter setting on the performance of the neural networks to the task. In the future, I can carry out similar empirical study towards other NLP tasks with different type of neural networks (i.e., LSTM).

References

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Association for Computational Linguistics*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR* abs/1408.5882. http://arxiv.org/abs/1408.5882.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '05, pages 115–124. https://doi.org/10.3115/1219840.1219855.

Ye Zhang and Byron C. Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR* abs/1510.03820. http://arxiv.org/abs/1510.03820.