Your Name: YongKyung Oh

Your Andrew ID: yongkyuo

Homework 3

1. Training Set Construction (5 pts)

Construct the training set for the amazon review dataset as instructed and report the following statistics.

Statistics	
the total number of unique words in T	21980 out of 327542
the total number of training examples in T	2000
the ratio of positive examples to negative examples in T	1.0
the average length of document in T	163.771
the max length of document in T	3294

In the Positive T: 15067 unique words out of 157565

In the Negative T: 14706 unique words out of 169977

2. Performance of deep neural network for classification (20 pts)

Model hyperparameters:

1. Data processing:
    a. Word embedding dimension: 100
    b. Word Index: keep the most frequent 10k words (max index as 10k)
    Including <unk> and <pad>, the length is equal to 10002

2. Model Train & Evaluation:
    a. Batch size: 64
    b. Learning rate: 0.001
    c. Optimizer: Adam
    d. Objective Loss: Cross entropy
    e. Epoch: 50
3. CNN (modified)
    a. Network: Word embedding lookup layer -> 1D CNN layer * 3-> fully connected layer -> output prediction
    b. Number of filters: 100 * 3
    c. Filter length: [3, 4, 5]
    d. CNN Activation: Relu
    e. Fully connected layer dimension 100, activation: None (i.e. this layer is linear)
4. RNN: (modified)
    a. Network: Word embedding lookup layer -> LSTM layer (Bidirectional) -> fully connected layer(on the hidden state of the last LSTM cell) -> output prediction
    b. Hidden dimension for LSTM cell: 100
    c. Activation for LSTM cell: tanh
    d. Fully connected layer dimension 100, activation: None (i.e. this layer is linear)

	Accuracy	**Training time(in seconds)**
RNN w/o pretrained embedding	75.30	284.52
RNN w/ pretrained embedding	85.80	278.37
RNN w/ pretrained glove 6B	83.20	276.43
CNN w/o pretrained embedding	67.00	49.53
CNN w/ pretrained embedding	85.05	45.33
CNN w/ pretrained glove 6B	82.55	48.88

- ✓ without pretrained embedding: use the Word2Vector model generated in the preprocess
- ✓ with pretrained embedding: use the pretrained embedding data from Amazon review
- ✓ with pretrained glove 6B: use the pretrained embedding data from glove 6B: 100

3. Training behavior (10 pts)

Plot the training/testing objective, training/testing accuracy over time for the 4 model combinations (correspond to 4 rows in the above table). In other word, there should be 2*4=8 graphs in total, each of which contains two curves (training and testing).

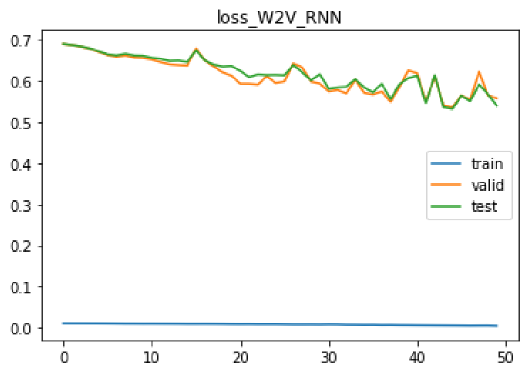
(I included the valid data from train split 8:2 to select the best model)

Test data 1600: Valid data 400: Test data 2000

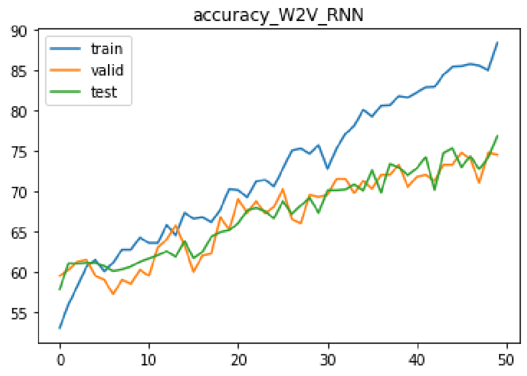
RNN w/o pretrained embedding

Test Loss: 0.53 | Test Accuracy: 75.30

- training/testing objective over time



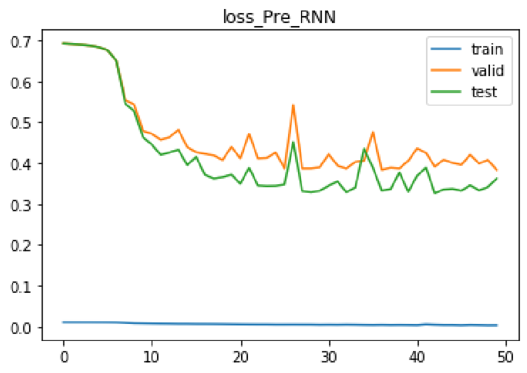
- training/testing accuracy over time



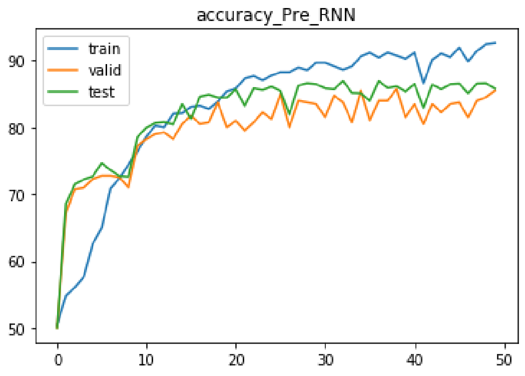
RNN w/ pretrained embedding

Test Loss: 0.36 | Test Accuracy: 85.80

- training/testing objective over time



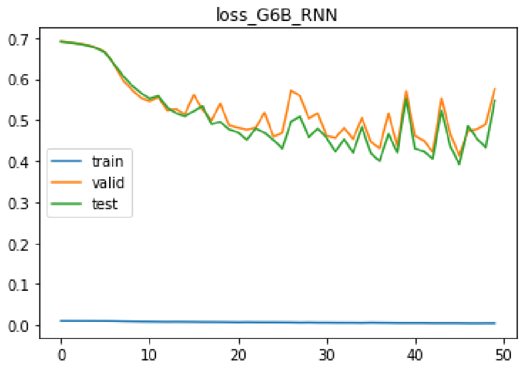
- training/testing accuracy over time



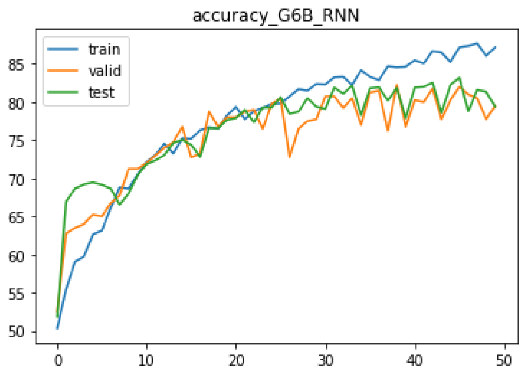
RNN w/ pretrained glove 6B

Test Loss: 0.39 | Test Accuracy: 83.20

- training/testing objective over time



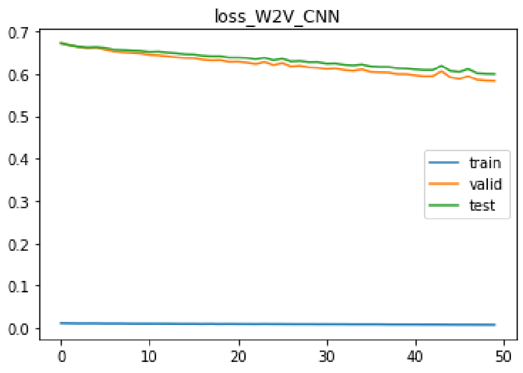
- training/testing accuracy over time



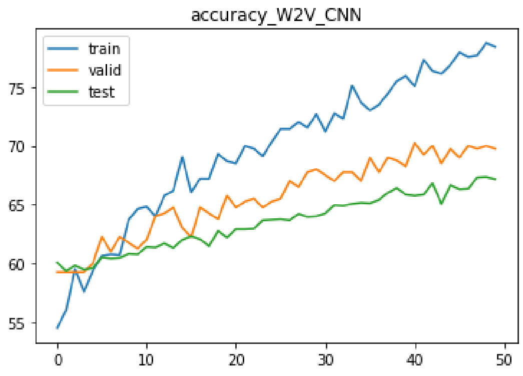
CNN w/o pretrained embedding

Test Loss: 0.60 | Test Accuracy: 67.00

- training/testing objective over time



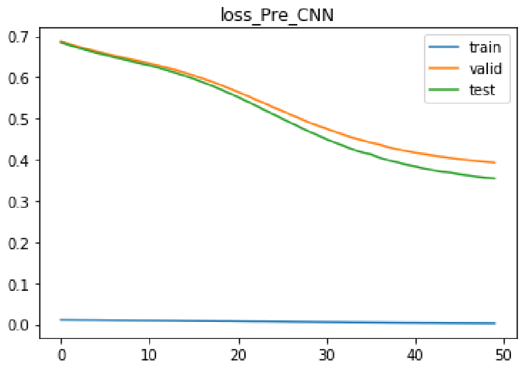
- training/testing accuracy over time



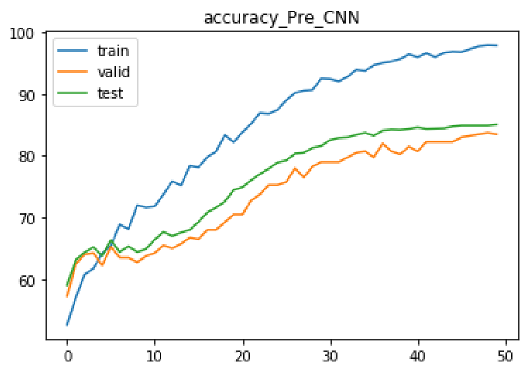
CNN w/ pretrained embedding

Test Loss: 0.35 | Test Accuracy: 85.05

- training/testing objective over time



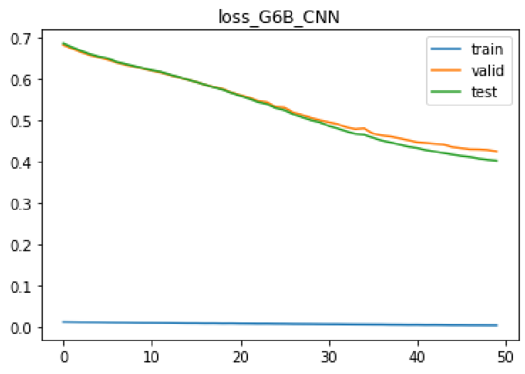
- training/testing accuracy over time



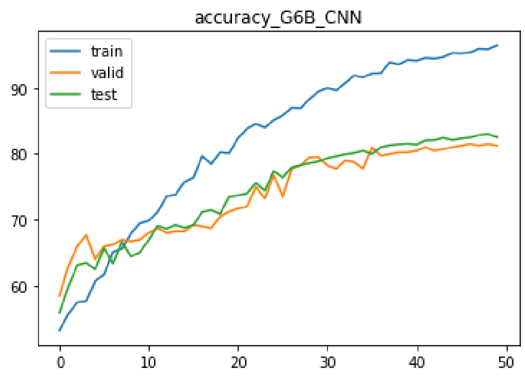
CNN w/ pretrained glove 6B

Test Loss: 0.40 | Test Accuracy: 82.55

- training/testing objective over time



- training/testing accuracy over time



4. Analysis of results (10 pts)

Discuss the complete set of experimental results, comparing the algorithms to each other. Discuss your observations about the various algorithms, i.e., differences in how they performed, different parameters, what worked well and didn't, patterns/trends you observed across the set of experiments, etc. Try to explain why certain algorithms or approaches behaved the way they did.

	Accuracy	Training time(in seconds)
RNN w/o pretrained embedding	75.30	284.52
RNN w/ pretrained embedding	85.80	278.37
RNN w/ pretrained glove 6B	83.20	276.43
CNN w/o pretrained embedding	67.00	49.53
CNN w/ pretrained embedding	85.05	45.33
CNN w/ pretrained glove 6B	82.55	48.88

An RNN is trained to recognize patterns across time, while a CNN learns to recognize patterns across space. Which DNN type performs better when dealing with text data depends on how often the comprehension of global/long-range semantics is required.[1]

RNNs perform well and robust in a broad range of tasks except when the task is essentially a keyphrase recognition task as in some sentiment detection and question-answer matching settings. In addition, hidden size and batch size can make DNN performance vary dramatically[2]

In the preprocess time, word2vector model is built. Therefore, if the time is included in the model w/o pretrained embedding, training time will be increased. The performance will be worse if word2vector is not used. (Then embedding parameter will be randomly determined.)

Overall performance is better in the RNN model, which is composed with bi-directional LSTM. CNN with 3 filter (with different sizes) are also well-performed when pretrained data is used. In case of RNN, loss function is not stable compare to CNN case. If the parameters are tuned, performance and training procedure will be improved.

[1] Text Classification — RNN's or CNN's? https://towardsdatascience.com/text-classification-rnns-or-cnn-s-98c86a0dd361

[2] Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*. https://arxiv.org/pdf/1702.01923.pdf

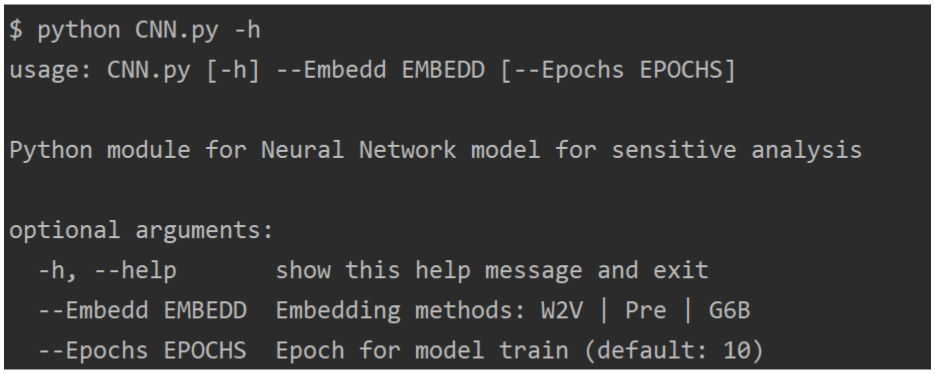
5. The software implementation (5 pts)

Add detailed descriptions about software implementation & data preprocessing, including:
1. A description of what you did to preprocess the dataset to make your implementations easier or more efficient.

Original data is cleaned up. I conducted data preprocessing and tokenization using NLTK. Also, I change the data into torchtext format to utilize the features. Using vector, I can easily load the pretrained embedding data in substitute as weight. Also, torchtext module help me to build batch and iteration bucket to design train and evaluate.

2. A description of major data structures (if any); any programming tools or libraries that you used;

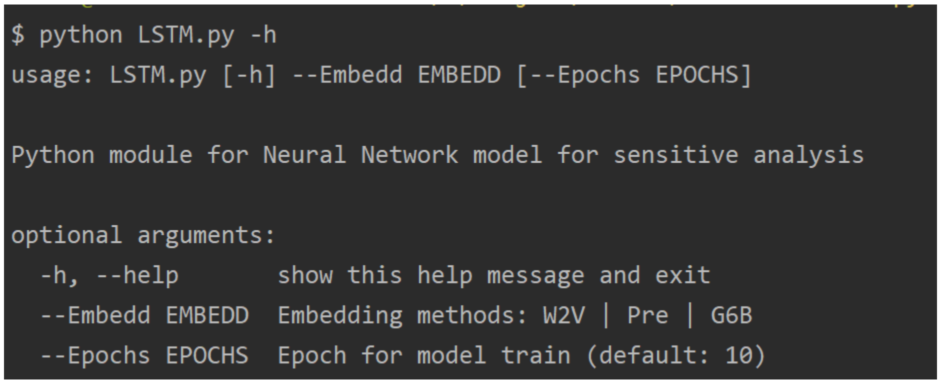
requirement.txt is attached. Most of the data is deal with ‘torchtext’ format. DNN models are based on the pytorch pipeline.



```
$ python CNN.py -h
usage: CNN.py [-h] --Embedd EMBEDD [--Epochs EPOCHS]

Python module for Neural Network model for sensitive analysis

optional arguments:
  -h, --help       show this help message and exit
  --Embedd EMBEDD  Embedding methods: W2V | Pre | G6B
  --Epochs EPOCHS  Epoch for model train (default: 10)
```



```
$ python LSTM.py -h
usage: LSTM.py [-h] --Embedd EMBEDD [--Epochs EPOCHS]

Python module for Neural Network model for sensitive analysis

optional arguments:
  -h, --help       show this help message and exit
  --Embedd EMBEDD  Embedding methods: W2V | Pre | G6B
  --Epochs EPOCHS  Epoch for model train (default: 10)
```

3. Strengths and weaknesses of your design, and any problems that your system encountered;

Most of all, performance is highly depending on the parameters. At this moment, most of the parameters are manually selected. So, if I can develop module to find the best parameters for the model, the performance will be improved.

Reference

[1] Text Classification — RNN's or CNN's?

https://towardsdatascience.com/text-classification-rnns-or-cnn-s-98c86a0dd361

[2] Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative study of cnn and rnn for natural language processing. arXiv preprint arXiv:1702.01923.
https://arxiv.org/pdf/1702.01923.pdf

[3] https://github.com/bentrevett/pytorch-sentiment-analysis

I use this tutorial for custom dataset for torchtext and model design. I implemented the suggested model structure and custom train/evaluate function for my scripts.