COMP4901I - Opinion Mining Report

Cheng Chi Fung
cfchengac@connect.ust.hk

1 Data

1.1 Data Cleaning

In this assignments, the following data cleaning methods is used. First, turn all the string into lower case and turn all of them into ascii encoding. It is followed by expanding the contradiction. Morverover, we remove all the digits and special characters.

1.2 Data Statistics

The following are the data statistics of the dataset given.

Table 1: Data Statistics

Statistics	–
Number of sentence	1
Number of words	2
Number of vocabs	3
Frequent words	4
Max sentence length	5
Average sentence length	6
Std sentence length	7

2 Implement ConvNet with PyTorch

The following are the data statistics of the dataset given.

Table 2: Development accuracy and Test Set accuracy

Dataset	Accuracy
Test Set	0.01
Development Set	0.01

2.1 Hyperparameters Tuning Results

The following are the hyperparameters tuning results.

Table 3: Char CNN Archiecture we used

Pooling Types	Learning Rate	Kernel Size	Dropout rate	Embedding Dimension	Number of Filters	Results
Average Pooling	0.01	100	0.1	100	2	32

3 Results and Analysis

3.1 Development Set Accuracy and Test set Accuracy

Table 4: Development accuracy and Test Set accuracy

Dataset	Accuracy
Test Set	0.01
Development Set	0.01

3.2 Analysis

For the kernel sizes, we found that for larger kernel size, we need to train more iterations but getting converge.

For the dropout, we found out that, the it requires more time to converge but getting higher best test set accuracy. The reason for that may be due to dropout forces a neural network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.

We found out that for higher learning rate, the higher the convergnce rate. However,

4 Bonus

4.1 Dynamic Padding

The following are the screen shot of the Predictor Network, CNN Encoder Network and loading the pretrained encoder.

4.2 Pretrained Word Embedding

For Pretrained Word Embedding, we have tried to replace the original word embedding layer by the pretrained **word2Vector** with **Google News corpus** (3 billion running words) word vector model. (Google News Corpus: https://github.com/mmihaltz/word2vec-GoogleNews-vectors). And since the dimension of the embedding matrix is enormously big which cause some memory error while training, we have limited to only use ten thousands of vocabs. All the above process can be easily done through by a python libarary named **gensim**.

And the following are the results of using pretrained embedding.

Table 5: Development accuracy and Test Set accuracy

Dataset	Accuracy
Test Set	0.01
Development Set	0.01

4.3 Other CNN Architectures

For other CNN archiectures, we have implmented character CNN by following the paper **Character-level Convolutional Networks for Text Classification**. (https://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.pdf)

Same as the paper, we have defined a list of characters which includes 26 English letters, 10 digits, 34 special characters and one blank characters. (**70 Characters in total**)

In the later part, we transfer those characters as 1-hot encoding and used it to create the sentence vectors for each sentences. For unknown characters, blank characters are used to replace it. The sentence vectors would then be inputed into the CNN with the following archiecture which is quite similiar to the paper.

Table 6: Char CNN Archiecture we used

Layer	Layer types	Kernel Size	Pooling Size / is Dropout	Number of Filters
1	Embedding	100	–	–
2	Conv2d	7	3	256
3	Conv1d	7	3	256
4	Conv1d	3	–	256
5	Conv1d	3	–	256
6	Conv1d	3	–	256
7	Conv1d	3	3	256
8	Linear	1024	Yes	–
9	Linear	1024	Yes	–
10	Linear	3	–	–

And the following are the results of using Char CNN.

Table 7: Development accuracy and Test Set accuracy

Dataset	Accuracy
Test Set	0.01
Development Set	0.01