COMP4901I - Building Interactive Intellengent Agent Assignment 3 Report

Cheng Chi Fung
cfchengac@connect.ust.hk

1 Data

1.1 Data Cleaning

awdawdawd

1.2 Data Statistics

awdawdawd

2 Implement ConvNet with PyTorch

2.1 Embedding Results

awdawdawd

2.2 Hyperparameters Tuning Results

awdawdawd

3 Results and Analysis

awdawdawd

3.1 Development Set Accuracy and Test set Accuracy

3.2 Analysis

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.

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 1: 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.