COMP4901I - BIIS - Assignment 3 Report

Cheng Chi Fung
cfchengac@connect.ust.hk

1 Data

1.1 Data Cleaning

In this assignments, we first convert all the strings into lower case and encode with ASCII. It is followed by expanding the contradiction and 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	10000
Number of words	1195793
Number of vocabs	23602
Number of vocabs with minimum Frequency 3	8798
Frequent words	the, i, to, a, and, it, is, of, not, for
Max sentence length	2186
Average sentence length	119.5793
Std sentence length	137.5261
Class distrubution	0:4000, 1:2000, 2:4000

2 Implement ConvNet with PyTorch

2.1 Using Word Embeddings

The following are the results of using word embeddings.

Table 2: Best Development Accuracy of Using Embedding

Dataset	Best Accuracy
Development Set	0.63

2.2 Hyperparameters Tuning Results

The following are the hyperparameters tuning results.

Best Parameters Obtained learning rate : 0.01, Dropout: 0.1, Number of Filter: 100, Kernel Size: (2,3,4), Embedding Dimension: 100, Average Pooling

Table 3: Hyperparameter tuning results

Pooling Types	Learning Rate	Kernel Size	Dropout rate	Embedding Dimension	Number of Filters	Best Accuracy
Max Pooling	0.1	(3,4,5)	0.3	100	100	0.5984
Max Pooling	0.01	(3,4,5)	0.3	100	100	0.6252
Max Pooling	0.1	(3,4,5)	0	100	100	0.5800
Max Pooling	0.1	(3,4,5)	0.1	100	100	0.5832
Max Pooling	0.1	(3,4,5)	0.3	100	100	0.5804
Max Pooling	0.1	(3,4,5)	0.5	100	100	0.5468
Max Pooling	0.1	(3,4,5)	0.3	100	50	0.5584
Max Pooling	0.1	(3,4,5)	0.3	100	100	0.5808
Max Pooling	0.1	(3,4,5)	0.3	100	150	0.5556
Max Pooling	0.1	(2,3,4)	0.3	100	100	0.5984
Max Pooling	0.1	(3,4,5)	0.3	100	100	0.5652
Max Pooling	0.1	(4,5,6)	0.3	100	100	0.5612
Max Pooling	0.1	(3,4,5)	0.3	50	100	0.5856
Max Pooling	0.1	(3,4,5)	0.3	100	100	0.5828
Max Pooling	0.1	(3,4,5)	0.3	200	100	0.5408
Average Pooling	0.1	(3,4,5)	0.3	100	100	0.5204
Max Pooling	0.1	(3,4,5)	0.3	100	100	0.5788

3 Results and Analysis

3.1 Development Set Accuracy

The following are the results of the final training with the best hyperparameters.

Table 4: Best Development Accuracy in final training

Dataset	Best Accuracy
Development Set	0.6444

3.2 Analysis

For ths size of filters, we found out that the smaller the kernel size, the better the best development accuracy. Since a larger size kernel can overlook at the features and could skip the essential details in the input whereas a smaller size kernel could provide more information leading to more confusion.

For the number of filter, we found out that the number of filter should not be too much and too little. Since the more the number of filters, the more the different convolutions which allow neural network to learn more different features. However, too much filter cause the neural network to difficult to converge. On the other hand, too little filters may cause the neural network to have enough ability to learn different features.

For the dropout probability, we found out that, the higher the dropout probability. it requires more time to converge . The reason for that may be due the neural network require more time learn the robust features by ropout forces. And also we found out that for better best development accuracy, the dropout probability should not be too high and too low. Too high will cause the neural network to difficult to converge. To low will cause the neural network cannot generalize well. According to the results in this assignment, 0.3 is the best.

For the learning rate, we found out that the lower the value, the slower the convergence. On the other hand, the higher the learning rate, the faster the convergence. However, high learning rate also earlier cause early stop. This may because the learning rate too large which cause the gradient descent overshot. Therefore, although low learning rate is slower, It slowly converge to optimal and get better best accuracy. According to the results in this assignment, 0.01 is the best.

For comparison between max pooing and average pooling, we found out that max pooling perform better average pooling. According to the results in this assignment, the Best Accuracy for max pooling is higher than average pooling.

4 Bonus

4.1 Dynamic Padding

For Dynamic Padding, we have defined our custom **collate_fn()** function to process the batch by dynamicially padding the batch with maximum length of the embedding in that batch. Defining our custom **collate_fn()** can be flexibly process the batch. **(This bonus are implemented in the files with postfix _char_dynamic_pad.py)**

Table 5: Best Development Accuracy of Using Dynamic Padding

Dataset	Best Accuracy
Development Set	0.6028

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 during training, we have limited to only use ten thousands of vocabs. All above process can be easily done through by a python libarary named **gensim**. And the following are the results of using pretrained embedding. **(This bonus are implemented in the files with postfix _embedding.py)**

Table 6: Best Development Accuracy of Using Pretained Word Embedding

Dataset	Best Accuracy
Development Set	0.6011

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 use 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. **(This bonus are implemented in the files with postfix _char_cnn.py)**

Table 7: 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 8: Best Development Accuracy of Using Char CNN

Dataset	Best Accuracy
Development Set	0.6312