COMP4901I - BIIS - Assignment 4 Report

Cheng Chi Fung
cfchengac@connect.ust.hk

1 Data

1.1 Preprocessing

In the data clearning part, we first turn all the unicode string into ASCII and lower case. Then, we expand the contradiction and remove all the characters except (A-Za-z0-9,!?;). Finally, the preprcessed data are tokenized by the white space.

1.2 Data Statistics

The following are the data statistics.

Table 1: Data Statistics

Statistics	–
Number of sentence	240234
Number of words	6966786
Number of vocabs	84624
Number of vocabs (After Limited)	15000
Number of vocabs with minimum Frequency 3	38221
10 Most Frequent words	the, and, of, to, a, i, in, he ,was, that
UNK Token Rate (Limited 15000 Vocabs)	0.00312

2 Results and Analysis

2.1 Hyperparameters Tuning Results

The following are the hyperparameters tuning results.

Table 2: Hyperparameter tuning results

Learning rate	Num of Hidden Layers	Hidden Dimension Size	Validation Perplexity
0.0001	1	128	227.2763346960948
0.0001	1	256	211.13846383756055
0.0001	1	512	206.77620995328408
0.0001	2	512	201.65396298175335
0.001	2	512	535.7280983715222

Best Parameters obtained = (Learning rate: 0.0001, Number Hidden Layer: 2, Hidden Dimension Size: 512)

We use the model with the best Hyperparameters and obtain the following test perplexity.

Table 3: Test Results

Dataset	Best Test Perplexity
Test Set	0.63

2.2 Discuss of Parameters Tuning Reults

2.3 Train with Pretrained Word embeddings

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 part are implemented in the files with postfix _embedding.py)**

We train the model with pretrained word embedding with the best hyperarameters and obtain the following test results.

Table 4: Test Results with Pretrained word Embedding

Dataset	Best Test Perplexity
Test Set	0.63

2.4 Language Generation

The following are the short paragraph generated from our models using the folloing words as starting.

Start from "I"

Start from "What"

Start from "Anyway"

3 Bonus

3.1 Char Level RNN

For the bonus part, we have implmented Char Level RNN with reference of the Char Level CNN. Samiliar as Char Level CNN, we have defined a list of characters which includes 26 English letters, 10 digits, 5 special characters ,a blank character, a start sentence and a end of sentence character. (**44 Characters in total**).

In the later part, we convert the each character in the seqence into index and input into the RNN same as the word level RNN performing. **(This bonus are implemented in the files with postfix _char_rnn.py)**

We train the char level rnn with the best hyperarameters and obtain the following test results.

Table 5: Test Results of Char Level RNN

Dataset	Best Test Perplexity
Test Set	3.762