COMP4901I - BIIS - Assignment 4 Report

Cheng Chi Fung
cfchengac@connect.ust.hk

1 Data

1.1 Preprocessing

In the data cleaning 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 preprocessed data are tokenized by the **white space** and are **contructed as sequences** with windows size 30 with adding the start and end of sentence indicator.

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 (Epochs=20, Early Stop=3).

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) with perplexity (201.6539)

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

Table 3: Test Results

Dataset	Best Test Perplexity
Test Set	212.6539

2.2 Discussion of Hyperparameters Tuning Reults

We found out that for **smaller learning rate**, the best validation **perplexity is higher** but converge slower. This may because for higher learning rate which means a larger step size may cause the gradient descent overshot the optimal easier which may trigger early stopping earlier. On the other hand, low learning rate which means a smaller step size may getting less chance to overshot the optimal. Thus, it can achieve better optimal point and getting better performance.

For the number of hidden layer and hidden dimension size. We found out that with **more number of hidden layer and higher hidden dimension size**, the best validation **perplexity is higher**. This may because deeper and wider network can fruitful for difficult and abstract problem such as language generation and getting better performance. On the other hand, little number of hidden layer and hidden dimension size may cause the neural network unable to have enough ability to learn different features.

2.3 Training 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 _word2vector.py)**

We train the model with pretrained word embedding with the best hyperarameters and having same setting as the training without pretrained word embedding. The following are the test results.

Table 4: Test Results with Pretrained word Embedding

Dataset	Best Test Perplexity
Test Set	127.5408

From the test results with pretrained word embedding, we found out that the language model with pretrained word embeddings have better **better performance (lower perplexity)** ($127.5408 < 212.6539$).

2.4 Language Generation

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

Start from "I" i fronting parading twinge wad gleam moliere wasermarrer homestall idolater authoritative convinced engineering unassailable ame enthuse uganda quatrain ramped disturbingly experimentation re finnish noblest orphaned fares', brooked frankness emposo monthly harsh deputy

Start from "What" what imbruted arroyo specter vet unmentionable behest reptiles nick accumulation combings spasmodically mucksy august

Start from "Anyway" anyway neuchatel requirements shlin10 caul inexplicable elysees kindergartens joshua

3 Bonus

3.1 Char Level RNN

For the bonus part, we have implemented Char Level RNN with the reference of the Char Level CNN. Samiliar to 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 characters in the sequence into index and input into the RNN same as the word level RNN performing. **(This part 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