NLU+ Assignment 1 Report

s2041285 s2099657

January 2021

1 Question2

1.1 (a)

	LR=0.5 hdim=25	LR=0.5 hdim=50	LR=0.1 hdim=25	LR=0.1 hdim=50	LR=0.05 hdim=25	LR=0.05 hdim=50
Step=0	5.0170	4.9731	5.2190	5.1329	5.4077	5.3130
Step=2	5.0351	5.0292	5.2145	5.1366	5.4038	5.3142
Step=5	5.0180	5.0211	5.2146	5.1368	5.4038	5.3124

Table 1: Minimum Loss on Development Sets with Different Parameter Settings

Table 1 shows the minimum loss on development dataset with different settings of learning rate, hidden dimension and time steps. Random seed of 2018 is used through all the experiment. From the table, when the learning rate is 0.5, hidden dimension in RNN is 50 and time step is 0, the model could achieve the minimum loss which is 4.9731. With the same hidden dimension and time step settings, RNN with larger learning rate has smaller minimum loss. It is likely that the training epoch number of 10 is too small, so that the models could not be close to optimum weights after 10 epochs. In the case with not enough training epochs, larger learning rate could update more and get closer to the optimum weights. Another observation from the experiment is that RNN models with larger hidden dimension could have better performance. From the experiment results in table 1, in most cases, when the learning rate and time step settings are the same, models with hidden dimensions of 50 have fewer loss than those with hidden dimension of 25. It is likely that hidden layer with larger dimension could keep more information about previous words than hidden layer with smaller dimension. Moreover, because the dimension of the word embeddings in V should be the same as the hidden dimension, when the hidden dimension is larger, more information about the word could be captured. However, with the same learning rate and hidden dimension settings, different time steps have similar minimum loss, which do not seem to affect the result.

1.2 (b)

The best parameter settings found in (a) are {learning rate=0.5, lookback=0, hidden dimension=50}, when the model is trained with this best parameter settings on 25,000 sentences, the mean loss, the unadjusted perplexity and the adjusted perplexity are correspondingly **4.4158**, **82.746**, **111.550**.

2 Question 3

2.1 (b)

In this task, parameters of learning rate, hidden dimension and lookback are taken into consideration. The number of epochs in this task is set to 10, and the training size is set as 25,000. In the implementation of 'acc_deltas_bptt_np', for short input sequences preceding the verb with length smaller than the preset lookback, the lookback of such sequences is set to the length of the sub-sequence preceding the verb.

The final parameters used in this task are {learning rate=0.5, lookback=2, hidden dimension=50}. From table 1 of task 2 (a), it could be observed that when the model is trained in small number of epochs, larger learning rate could lead to quicker convergence and better results with the same hidden dimension and time step. Thus, the learning rate is chosen as 0.5. Another observation from task 2 (a) is that larger hidden dimension could get better results. In this task, hidden dimension is chosen as 50. However, with such learning rate and hidden dimension settings, the step number does not have obvious effect on the loss in task 2 (a). So in this experiment, we compared the performance of step 0, 2, and 5.

	lookback=0	lookback=2	lookback=5
loss	0.4288	0.4095	0.4065
accuracy	0.814	0.827	0.825

Table 2: Loss and Accuracy on Development Dataset with Different Lookback

Table 2 shows the loss and accuracy on development dataset for different lookback settings. It shows that when lookback is 2, the model could have the highest accuracy which is 0.827 compared to the other two time steps. Although when lookback is 5, the model could achieve similar performance as lookback of 2, it requires more computation steps and resources, which is less efficient than time step of 2. Therefore, we finally chose parameter settings of {learning rate=0.5, lookback=2, hidden dimension=50}.

3 Question 4

The parameters **U**, **V**, **W** used in this experiment are from the trained model in 2(a) with the hyper-parameters {learning rate=0.5, lookback=0, hidden dimension=50}. The number prediction accuracy on development set is **0.675**, and the number prediction accuracy on test set is **0.651**. The accuracies are much lower than the ones using the model in question 3. In this task, we try to use this language model to predict the number agreement of the verb. The original goal of the language model RNN is to predict the next word given word sequences. However, for prediction of next word task, it considers not only the morphology but also the semantic aspect to predict the next word, which may make the prediction of the number agreement less accurate. In addition, the test accuracy is lower than that of development set. The size of test set is larger than that of development set. Thus, the number of unseen words contained in test set could be larger than that of the development set. For unseen words, they are converted to 'UNK' and their real meanings are lost, which could lead to less accurate number agreement prediction.

4 Question 5

The **hypothesis** we made initially was using part of speech tags alone could make number prediction more accurate compared to the model in question 3. For the model in question 3, we use the sub-sequence preceding the verb to predict the morphology of the verb. In real life, for number prediction of a verb, the part of speech of preceding words could also be helpful. So in this question, we intend to use part of speech of the words for number prediction task.

In **experiment**, we need to transform the words in the sentences of the given training and development data files to corresponding part of speech. We firstly tokenized each sentence into tokens and then transformed each token into tag of part of speech using 'word_tokenize' and 'pos_tag' functions in nltk package. Then we built a vocabulary containing 37 kinds of tags of part of speech.

An example to convert a sentence into a sequence of part of speech tags is as Table 3 shows:

Sentence	**NNS**	are	responsible	for	colour	perception	.
Part of Speech	**NNS**	VBP	JJ	IN	JJ	NN	.

Table 3: Example of converting a sequence into part of speech tags

In sentences of the datasets, **NNP**, **NNS** and **NN** represent singular proper noun, plural noun and singular noun tags separately. In NLTK package, it treats the three words as **NNP**. If the three words are used as the subjectives, they will lose the noun number information and could affect the number prediction of the verb. Thus, for these words including **NNP**, **NNS**, **NN**, we made them unchanged as the corresponding tags of part of speech directly.

We take the generated part of speech tag sequences as the training dataset and train the model with the same parameter settings {learning rate=0.5, lookback=2, hidden dimension=50, epoch=10} as the model in Question 3, in particular, the model takes the part of speech tags preceding the verb as input and predicts whether the verb is VBP or VBZ. We used development set to test the model and made comparison with the model stated in Question 3. The best accuracy of prediction with part of speech tags is 0.949 compared with 0.827 for model in question 3, which is much higher and validate our previous hypothesis.

From our **analysis**, one possible reason is that using part of speech tags as input has less noise than using word as input. For original RNN models, the vocabulary size is limited. It is possible to encounter unseen

words. The original model takes unseen words as word 'UNK', which loses important information of the unseen words. For one sentence, if the subjective word is unseen from the vocabulary, then it would be transformed into 'UNK', which makes the model difficult to do number prediction without knowing number information about the subject. However, part of speech tags are complete and of small size. By using part of speech tags as input, we could tag each word without using 'UNK', which is more accurate. Another possible reason is that with smaller-size tag vocabulary, the size of matrix 'V' is smaller, which could make training of each vector with more times and final vectors could be more representative. For word inputs, because it always requires large vocabulary size, the training for each word vector may be not enough. Through the experiment, training time with smaller size of tag vocabulary is shorter than using large size of vocabulary, which is more efficient. To summarize, although the way to use part of speech as input could lose word information, it does not affect number prediction too much. Moreover, focusing on the part of speech could reduce the noise in training compared to large-size vocabulary and could give better performance.