**NLU coursework 1**

S2025012 s2045321

**General Settings :**

In our entire coursework, all the following parameters use the default values given by the coursework code:

Batch\_size = 100

Mini\_change(early stop) = 0.0001

Annealing size = 5

**Question 2: Language Modeling**

1. Best parameters search

In this section, we train the RNN model to predict the probability of next word of the current input. To find the best hyper-parameters of the RNN model on the wiki data set with 2000 vocabulary size, we use several hyper-parameter combinations including: , number of , . After training, we evaluate our model on development set with size = 1000 and report their loss. We have the following result: The best hyperparameter combination is highlighted (yellow) in the table.

图形用户界面, 应用程序, 表格, Excel

描述已自动生成

Table 1: RNN language model training result with respect to different hyperparameter combinations

In a suitable range, increasing the learning rate can find the minimum point faster in the process of gradient descent. Therefore, given the limited epochs number (10), the higher learning rate can lead to a lower loss. When it comes to the hidden units size, as you can see, sometimes higher units have lower losses, and sometimes it's the other way around. At first, we thought there maybe had some experimental error which made the trend inconspicuous, but in the next three replicates, we all observed the result as we saw at the first time. Therefore, we can't say a higher hidden units size can lead to a lower loss. In regard of back steps, we expected to have a better performance with the increase of lookback steps in backpropagation through time (BPTT) since we use lookback steps in RNN to capture long-distance dependency for language. We suspect the reason for this fact is that our training set is not big enough, we will discuss this in question5.

(B) Training on larger dataset

We used the parameters combination got from A) to train the 25000 sentences with 10 epochs. We plot the log base loss with respect to epoch in figure1:

图表, 折线图

描述已自动生成

Figure 1: model's performance during training.

After training, we evaluate our language model on the development set and test set. For development set, the unadjusted loss is 82.735 and the adjusted for missing vocab in basic mode is 111.533. For test set, the Unadjusted loss is 102.160 and adjusted loss for missing vocab is 141.603.

Question 3: Predicting Subject-Verb Agreement

1. Best parameters search

In this section, we search whether RNN architecture can handle subjective-verb agreement problem. We train our RNN model to do the binary classification (VBZ? or VBP?) given the sequence of words preceding the verb. First similar to the Question 2(A), we need to find the best parameter combination which has lowest loss. we use lr = [0.5, 0.1, 0.05], number of hidden units = [25, 50,100], lookback steps = [0, 2, 5] to train our new model and report their loss, accuracy on development set . The results are shown in table 2.

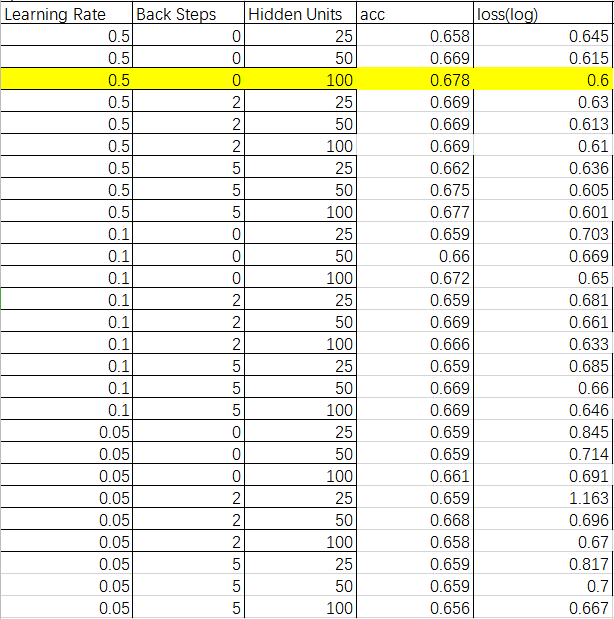


Table 2: For the number-predict function RNN, the test results of several hyper-parameters combinations after 10 epoch training.

The best parameter combination is highlighted in table 2 which are lr = 0.5, backstep = 0, hidden units = 100 resulting in highest accuracy(67.8%) and lowest log base loss(0.6).

From the table, we can clearly see that the higher learning rate, the lower loss, which is same with the 2(A). However, unlike the result in 2(A) where we couldn't find a clear trend over hidden unit size, here we can conclude that more hidden units can improve the performance of RNN model. The most interesting thing is that in this case, the value of the back steps which lead to the best result is also 0. we will also discuss this back steps value in section 5.

1. Training on larger dataset

Using the best parameter combination got from 3(A), we trained our number prediction RNN model with 25000 training size. After training, we evaluate our model over the development size and test size. For development set, the accuracy is 82.6% and the log-base loss is 0.39994. For test set, the accuracy is 82.7% and the log-base loss is 0.39992.

**Question 4: Number Prediction with an RNNLM**

In this section, we use RNN language model got from Question 2 to do the number prediction task. Compared with the method in question 3, here we use language model to predict the probability of the singular verb and plural word directly given the sequence of words preceding the verb. We choose the term with higher probability as our prediction result. Now. we use the best language model which is trained in Q2 (B). After we implement the compare\_number\_prediction function, we evaluate our language model on the development set and test set. The prediction accuracy for development set is 67.5% and the prediction accuracy for test set is 65.125%.

However, after we processed the data in the development set, we found that the above prediction results were not good. In development set with size 1000, the number of sentences with a singular verb (VBZ) was 341 and the number of sentences with a plural verb (VBP) is 659. This means that we can give a baseline prediction accuracy (65.9%) by just predicting all the verbs are plural in development set. The class number in test set is similar to development set. Therefore, our RNN LM just give a slightly higher result than the baseline. We are going to improve this result by introducing LSTM language model in Q5 and discuss the reason.

**Question 5: Explorations**

**Experiment 1:**

In the process of finishing this coursework, we left a question worth discussing: we expected to have a better number prediction performance with the increase of backsteps in backpropagation through time (BPTT) which can help us to capture the long-distance dependency. But when we evaluate the performance of hyperparameter backsteps in Question2(A) and Question3(A), we surprisingly find that the best backsteps number is 0. In this section, we discuss this question and we will use the number predication task in question3 as our experimental subject.

Firstly, we process the original data to calculate the distance between the subject and verb. We do the statistics to the data according to the distance and plot them in the following histograms. The ordinate represents the proportion of sentences with different distances to the total data set, and the abscissa represents the distance between the subject of the sentence and the verb. In addition, we find that the test set and the development set have similar data distribution as the training set. You can see the Figure 2.

图表, 直方图

描述已自动生成图表, 直方图

描述已自动生成

Figure 2: the distance between verb and subject in training set and development set

The distance 1 means subject and verb in sentence are next to each other. Because the verb's singular or plural form depends on the subject, this is a local dependency. We find that the no matter for training set or development set, the sentence with local dependency takes the majority of data set. So, our first guess is whether introducing too much context information by increasing backsteps will cause the model to confuse the local dependency data and perform poorly. In order to evaluate our conjecture, we need to explore the model's performance in sentences with different distances between subjects and verbs.

What we're using here are two models trained by 1000 training set, lr = 0.5, hidden units = 100, lookback steps = 0 and 5 respectively. After training, we evaluate our two models on the development set. We can see the performance in following Figure 3.

图表, 条形图

描述已自动生成

Figure 3: explore the performance of RNN model with 1000 training size for different distance sentence.

For RNN model with backsteps 5, we didn’t see a worse performance as we expected for local dependency sentence compared with the RNN model with backsteps 0. Also, for long-dependency, we don’t see a better performance compared with the model with backsteps 0. This shows that our initial guess is incorrect.

Then, we wondered if this was because the training set was too small. Therefore, we use the same hyperparameter combination and model as above, the only difference is that we increase the number of training sets to 25,000. You can see the performance in Figure 4.

We can see that this time, the model with backsteps=5 generally performed better than the model with backsteps=0. This verified our conjecture that the experimental results were inconsistent with the theory precisely because the number of training sets was too small.

图表, 条形图

描述已自动生成

Figure 4: explore the performance of RNN model with 1000 training size for different distance sentence.

**Experiment 2:**

In Question4, We find that the RNN language model don’t lead to a much better number prediction performance to baseline which motivate us to implement the LSTM language model. Here, we use pytorch to build a LSTM language model, we do the similar hyperparameter search like what we do in Q2(A) and Q3(A). We set the vocabulary = 2000 and training size = 1000. Then we evaluate hyperparameters with hidden units = [25 50 100] and learning rate [0.5, 0.1, 0.05]. The backsteps in LSTM is set by training the gate. After training, we evaluate each model on development size with size = 1000 and report their loss. We have the following result:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| learning rate | hidden units | loss(log) | Unadjusted loss | Adjusted loss |
| 0.5 | 25 | 4.5 | 94.667 | 129.905 |
| 0.5 | 50 | 4.55 | 94.61 | 129.817 |
| 0.5 | 100 | 4.52 | 91.801 | 125.464 |
| 0.1 | 25 | 4.72 | 112.155 | 157.382 |
| 0.1 | 50 | 4.711 | 111.15 | 155.787 |
| 0.1 | 100 | 4.673 | 109.152 | 152.621 |
| 0.05 | 25 | 4.789 | 120.192 | 170.205 |
| 0.05 | 50 | 4.788 | 120.168 | 170.167 |
| 0.05 | 100 | 4.793 | 120.705 | 171.028 |

Table 3 LSTM language model training result with respect to different hyperparameter combinations

Then, we use the best hyperparameter combination in Table3 to train our number prediction RNN model with 25000 training size. After training, we evaluate our model over the development size and test size.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | baseline | RNN classification model | RNN LM | LSTM LM |
| dev set | 65.90% | 82.60% | 65.13% | 74.85% |
| test set | 67.50% | 82.70% | 67.50% | 76.90% |

For number prediction task , our LSTM LM have made huge strides compared with the RNN LM. The most important reason why we think the LSTM model can perform better is that In RNN language model, in order to avoid gradient vanishing, we set small back steps, such as 5 or 2. In real languages, however, some dependencies are usually much larger than this value. You can see this in Figure 2. For this part of the information, RNN is usually not perceptible. As for LSTM, it determines the input and output of information through the control of the gate, so that the model can perceive the information long ago, which is very beneficial to capture long dependence.

Besides, we observe that for number prediction task, the performance of our LSTM LM still can not catch up with the performance of RNN classification model. We see if we can further improve our LSTM language model by adding an attention mechanism, and this is what we will work on in the future.

Reference：

Mikolov, T., Karafifiat, M., Burget, L., Cernocky, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In Kobayashi, T., Hirose, K., and Nakamura, S., editors, *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048. ISCA.