Natural Language Understanding, Generation and Machine Translation
Coursework 1: Recurrent Neural Networks

The University of Edinburgh, School of Informatics
Andreas Neokleous, Dimitra Stasinou

July 15, 2019

Language Modeling (Question 2)

Part 2(a)

The model was trained by considering a combination of different hyper-parameters. A representation of the results for each parameter combination is shown in Table 1. As can be seen in Figure 1, the model performs better when the learning rate is set to 0.5, regardless of the number of hidden dimensions and/or lookbacks. Overall, in terms of the number of hidden dimensions, the model yields better loss results when tuned with 50 hidden dimensions compared to 25. The number of lookbacks does not seem to affect directly the error minimization as it shows a wide variation regarding the results when combined with the other parameters. When training the model with these parameters, the deviation of the minimum compared to the maximum loss achieved is very small 0.43. An additional observation, not shown in Table 1, is that a small learning rate requires more training time, indicating that the model takes more time to converge, and higher dimensions take as well, more time to train.

Learning Rate	**Hidden Dim**	**Lookback**	**Loss**
0.5	50	0	4.973139
0.5	25	0	5.016976
0.5	25	5	5.018009
0.5	50	5	5.021099
0.5	50	2	5.029222
0.5	25	2	5.035141
0.1	50	0	5.132932
0.1	50	2	5.136560
0.1	50	5	5.136824
0.1	25	2	5.214506
0.1	25	5	5.214641
0.1	25	0	5.218973
0.05	50	0	5.312958
0.05	50	2	5.314184
0.05	50	5	5.314224
0.05	25	5	5.403781
0.05	25	2	5.403825
0.05	25	0	5.407678

Table 1: This table shows the results of parameter tuning for Q2. It is sorted in ascending order of loss on the development set and not in descending order of learning rate. It is shown that the smaller the learning rate, the larger is the loss.

We observe that the smaller the learning rate, the bigger is the loss of the model as shown in Figure 1. The learning rate is one of the most important hyper-parameters to tune when training deep neural networks. When the learning rate is very small, the optimization may take a lot more time due to the small steps towards the local minimum. On the other hand, a large learning rate may miss the global minimum. A solution proposed by different researchers [1, 5, 6], is an adaptive learning rate, which is also implemented in our model. We used the default annealing rate of 5. The chosen learning rate parameters do not vary greatly among them and the results indicate a general convergence while training.

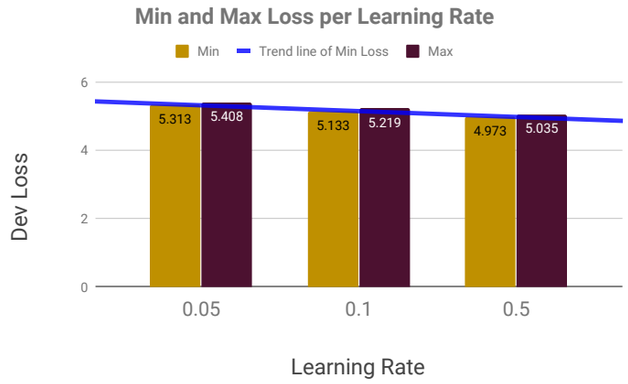

Figure 1: This chart shows the minimum and maximum dev losses that were observed when experimenting with the parameters of Table 1. The losses are grouped by learning rate and the blue line shows that smaller learning rate resulted in a larger loss.

We decided to investigate the performance of the model by taking into account more parameter combinations than the ones recommended in the coursework description. In particular, we trained the model considering a learning rate range from 0.2 to 0.9. For this experiment, we took the optimal parameters of hidden dimensions and lookbacks from Table 1, and combined them with different learning rates. As indicated in Figure 2, the smallest loss was achieved when the learning rate is set to 0.4, with the model having an overall fluctuating performance across the different rates.

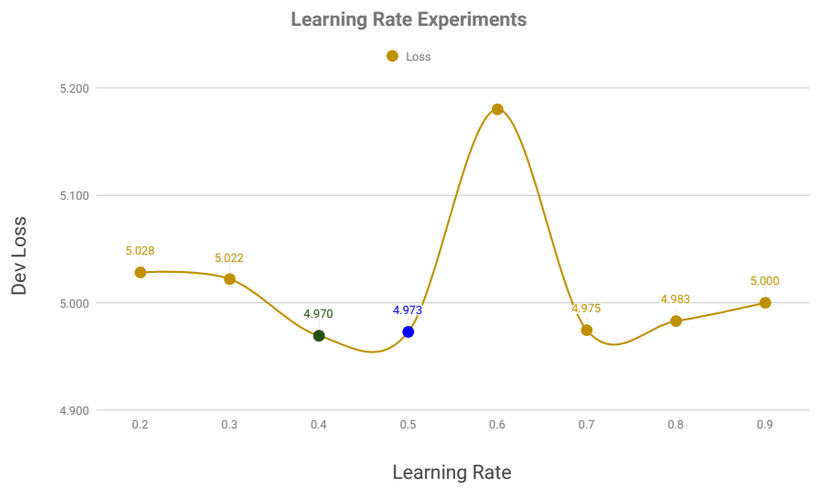

Figure 2: This plot shows the change in loss when experimenting with the parameters of Table 1, 50 hidden dimensions and 0 lookbacks. The blue dot indicates the best, initial loss. The green dot indicates the new, best, initial loss.

We also tested the three, best, initial hyper-parameter combinations with 20 epochs rather than 10. The mean loss decreased to 4.9, with 0.5 learning rate and 50 hidden dimensions, a 0.07 improvement. A further investigation and, possibly, increase of epochs, combined with an appropriate learning rate, could give even better loss results. In terms of the number of lookbacks, we tested the model using 7 lookbacks and observed that there was a slight increase in loss, which may be due to the nature of the dataset.

Part 2(b)

Based on the above results, we trained the model on the bigger training set of 25000 sentences using the following hyper-parameters: **Learning Rate: 0.4, Hidden Dimensions: 50, Lookbacks: 0, Epochs: 10**. Even though 20 epochs had better results, we decided to use just 10 because of time and computation constraints. We evaluated the performance of the model on wiki-test.txt and reported the following **Mean Loss: 4.43567, Adjusted Perplexity: 163.258** and **Unadjusted Perplexity: 114.094** .

Predicting Subject-Verb Agreement (Question 3)

Part 3(b)

For the implementation of this task, we have started investigating the choice of parameters based on those that yielded the best results in Q2. These are the ones mentioned in the Part 2(b) of Q2. Then, we tuned the learning rate, lookbacks and hidden dimensions. This time cross-entropy loss and accuracy as well, were taken into account. We have noticed a decrease in loss but the accuracy was not increasing with some parameter combinations. This could indicate that the network is likely to overfit. Possible solutions for overfitting are dropout [4], L2 regularization (weight decay)[2] or L1 regularization [3].

After experimenting with different parameters we observed that the accuracy is not affected by the lookbacks and it is only affected by the learning rate and hidden dimensions as shown in Table 2. The model's performance was evaluated on the wiki-dev.txt and wiki-test.txt. The lookback parameter affects the cross-entropy loss, however a smaller loss does not always imply a higher accuracy. Since lookbacks do not affect the accuracy, we have concluded that the best parameters are **Learning Rate: 0.8, Hidden Dimensions: 50, Lookbacks: 0**. These yield the following results: **Dev Loss: 3.0315, Dev Accuracy: 0.659, Test Loss: 2.9901, Test Accuracy: 0.675** . The best parameters and their results are highlighted in the Table 2.

Hidden Dim	**Lookback**	**Learning Rate**	**Dev Loss**	**Dev Acc**	**Test Loss**	**Test acc**
50	5	0.8	2.9401	0.659	2.9005	0.675
50	10	0.8	2.9403	0.659	2.9006	0.675
50	2	0.8	2.9431	0.659	2.9034	0.675
50	**0**	**0.8**	**3.0315**	**0.659**	**2.9901**	**0.675**
25	0	0.9	2.6470	0.659	1.6050	0.675
50	5	0.7	3.3978	0.659	3.3599	0.6748
50	0	0.7	3.4967	0.659	3.4577	0.6748
25	0	0.8	4.8906	0.659	4.8672	0.6748
50	0	0.6	4.0220	0.658	3.9861	0.6703
75	0	0.8	2.7240	0.658	2.6580	0.6693
50	0	0.5	4.5817	0.521	4.5495	0.548

Table 2: The hyper-parameter exploration for agreement task using RNN with direct supervision.

Number Prediction with an RRNLM (Question 4)

The Table 3 shows the results of the number agreement prediction task. The accuracy on the development set is higher than the test set's acurracy, as expected. Comparing the results with those of Q3, we can see that the RNN can successfully learn agreement just from the language data. The RNN with direct supervision (from Q3) still performs better on the test set, but with a small difference of 0.0245.

Number prediction accuracy on dev set:	0.666
Number prediction accuracy on test set:	0.6505

Table 3: RNN results on agreement task with just language data.

Exploration (Question 5)

The Recurrent Neural Network Model implemented has the ability of predicting the subject-verb number agreement, as shown in Q4. It scores a significantly good accuracy which shows that it can potentially, with further improvements, achieve higher accuracy results on the agreement task.

The question we raised, after implementing the main task, is whether the same model could have the ability of predicting the inverted task, which means the verb-subject number agreement. Although it may seem to be a quite basic further approach, we wanted to test if it is possible to learn in the same supervised and later, unsupervised way to complete language tasks that usually, humans complete inevitably.

Firstly, we have decided to use the same data set in order to have comparable results in terms of vocabulary tokens and length. We have created copies of the original data, but the vocabulary data set, and adjusted them in the code as wiki-train-new.txt, wiki-dev-new.txt and wiki-test-new.txt. As the vocabulary data set contains all the word tokens and their probabilities, it would make the task more complicated if there were changes in it.

Then, we modified the copied data files in order to represent the relative POS tags and the position of the appropriate words in the correct order. In particular, the subj-idx position was replaced by the verb-idx one, the verb-pos tag with the relevant subj-pos and lastly, the subject-verb dependency positions were inverted. As we needed to include the nouns in the form found in the data set and their inflected one, we implemented the PyPI inflection library which, for this task, would deal with the singularization and pluralization of the English nouns. That way we could transform the nouns in our vocabulary without having to deal with the inflection variations across different cases in the vocabulary. We have also, taken into consideration the cases where nouns where originally replaced by their noun-phrases. Particularly, we have dealt with the correct replacement of NN to NNs and NNs to NN, respectively. During the process of creating the new data sets, we have noticed that the main data did not deal with many cases where a plural verb would have two or more plural subjects or two or more singular subjects, connected by or or nor, would have a singular verb. As these cases were very few, we have decided to emit them from the new data sets in order to check a basic functionality of the model, without handling exceptions. Consequently, the size of the development and test set were reduced to 900 sentences, from the original 1000. The modified data and python files created are included in the zip submission in Q5 folder.

As mentioned before, we have tried to implement a basic functionality of the model, the verb - subject agreement, without taking into consideration special syntactic cases. For that reason, the original implementation was not changed, rather modified in specific parts, as, in general, we are still implementing the number agreement task. We are using the same recurrent neural network that learns to predict first, in a supervised and later, in an unsupervised manner.

Regarding training the model, we have used, after experimentation, parameters similar to the subject-verb agreement prediction task. The parameters that yielded the best loss result of **mean loss: 4.98587** are **learning rate: 0.4, hidden dimensions: 50** and **lookbacks: 0**. Also, the perplexity results are quite satisfying under this parameter tuning; particularly, **unadj loss: 219.098** and **adj loss: 290.565**. Some of these results are shown in Table 4 below.

Learning Rate	**Hidden Dim**	**Lookback**	**Dev Loss**	**Test Loss**	**Unadjusted**	**Adjusted**
0.4	50	0	4.98587	5.0122	219.098	290.565
0.5	50	0	4.99188	5.0164	220.133	291.777
0.7	50	0	4.99785	5.0223	221.607	293.503
0.4	50	5	5.03764	5.0538	229.662	302.909
0.4	50	2	5.04585	5.0611	231.566	305.126
0.2	50	0	5.04703	5.0685	233.52	307.4

Table 4: Q5 - Parameter exploration for language model RNN

In terms of the number agreement task, after experimenting with different parameter combinations and number range, the parameters are similar to the original model regarding backpropagation steps and hidden dimensions. However, the model yields better results when tuned with higher learning rates, specifically greater than 0.8. The results can be seen in the Table 5 below.

Hidden Dim	**Lookback**	**Learning Rate**	**Dev Loss**	**Dev Acc**	**Test Loss**	**Test acc**
50	0	1	4.531853	0.6633	4.318461	0.706638
50	0	0.9	4.97014	0.6633	4.773926	0.706638
50	0	0.8	5.43063	0.6255	5.252466	0.659529

Table 5: Q5 - Parameter exploration for verb-subject number agreement RNN - train-np.

Overall, it seems that the inverted task performed well, based on the results shown above. However, this model was implemented with many limitations so, these could not possibly reflect accurately the reliability of its performance. Further development and investigation may yield better results, especially when all linguistic and computational factors are taken into consideration.

References

[1] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[2] Anders Krogh and John A Hertz. "A simple weight decay can improve generalization". In: *Advances in neural information processing systems*. 1992, pp. 950–957.

[3] Mee Young Park and Trevor Hastie. "L1-regularization path algorithm for generalized linear models". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 69.4 (2007), pp. 659–677.

[4] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[5] Tijmen Tieleman and Geoffrey E. Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural Networks for Machine Learning* (2012).

[6] Matthew D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: *CoRR* abs/1212.5701 (2012).