Natural Language Understanding, Generation and Machine Translation. Coursework 1.

By

S1885517

S1891076

On

Recurrent Neural Networks

Q2 A:

These are the findings of the required experiment where the parameter settings of learning rate, hidden units and steps were tried in the possible combinations as mentioned in the question.

	Training size	Dev size	Learning Rate	Hidden Units	Steps Back	Batch size	Anneal	Loss Raw
1	1000	1000	0.5	25	0	100	5	5.0170
2	1000	1000	0.1	25	0	100	5	5.2190
3	1000	1000	0.05	25	0	100	5	5.4077
4	1000	1000	0.5	25	2	100	5	5.0351
5	1000	1000	0.1	25	2	100	5	5.2145
6	1000	1000	0.05	25	2	100	5	5.4038
7	1000	1000	0.5	25	5	100	5	5.0180
8	1000	1000	0.1	25	5	100	5	5.2146
9	1000	1000	0.05	25	5	100	5	5.4038
10	1000	1000	0.5	50	0	100	5	**4.9731**
11	1000	1000	0.1	50	0	100	5	5.1329
12	1000	1000	0.05	50	0	100	5	5.3130
13	1000	1000	0.5	50	2	100	5	5.0292
14	1000	1000	0.1	50	2	100	5	5.1366
15	1000	1000	0.05	50	2	100	5	5.3142
16	1000	1000	0.5	50	5	100	5	5.0211
17	1000	1000	0.1	50	5	100	5	5.1368
18	1000	1000	0.05	50	5	100	5	5.3142
1	1000	1000	0.5	50	0	25	5	4.7837
2	1000	1000	0.5	50	5	25	5	4.7826
3	1000	1000	0.5	50	2	25	5	4.7986

Label 10 (Dark pink) represents the lowest loss value with the desired parameters. Value in yellow represents a candidate which should be successful intuitively. The last 3 rows show the same experiment with new batch size values, and showing that eventually the yellow will win, represented by green.

In the table above, we see that learning rate has been the factor that most affected the loss. This is mainly because a lower a learning rate results in a lower change in the matrix requiring a greater number of iterations for the model to eventually converge. Similarly, with the number of hidden units, on average 50 performed better than 25 as seen above. This is because with a greater number of hidden units the model can extract more features to represent the sentence helping make better predictions. However, this probably also results in a relatively higher learning time (more epochs). Interestingly, the step size barely made any difference in loss. This is quite interesting, as you would think that the greater the number of steps the better the model is capturing longer term dependencies. When performing the experiments, we think to ourselves maybe this is because it had a smaller number of gradient updates (higher batch size) and so we decide to lower batch size and perform the experiment again. We do see the parameter with 5 back steps outperform the model with 0 back steps (in terms of loss). However, the difference in loss is quite negligible, which makes us wonder how the back steps really work, or how significant the vanishing gradient problem might be (explored later). Regardless, we pick the parameters highlighted in green as they had the lowest loss.

Q2 B:

Train size	Dev size	Lr	Hidden	Steps	Batch size	Anneal	Loss Raw	Unadjusted Loss	Adjusted loss
25000	1000	0.5	50	5	25	5	4.2106	67.399	88.437

Q3 B:

We hypothesized that the best parameters for this question would be the same as the previous task. The simple reason is that the task to be performed is still quite similar for the model. Also, as the distance of the verb increases from the subject the number of steps the model should look back makes a big difference in what it can predict for the subject-verb agreement. This is because if it does not remember the subject then it cannot predict the verb-agreement accurately. Moreover, there's more emphasize on steps-back this time as the distance of the verb from the subject should make a difference in subject-verb agreement prediction. Also, greater batch size might be a safer way to make the gradient steps, but that was not the case from what we observed. However, we decided to perform the experiments like previous questions again (specially because the train time was super low). Here we see that initially it would seem that lower hidden units perform better, but as we train longer (more epochs, as there is no such constraint) we eventually see that the same parameters as before work well enough!

	Trainsize	Devsize	Lr	Hidden	steps	batchsize	anneal	Accuracy	
0	1000	1000	0.5	50	0	25	5	0.709	
0	1000	1000	0.5	50	2	25	5	0.711	
0	1000	1000	0.5	50	5	25	5	0.71	
0	1000	1000	0.1	50	0	25	5	0.667	
0	1000	1000	0.1	50	2	25	5	0.669	
0	1000	1000	0.1	50	5	25	5	0.669	
0	1000	1000	0.05	50	0	25	5	0.666	
0	1000	1000	0.05	50	2	25	5	0.669	
0	1000	1000	0.05	50	5	25	5	0.669	
0	1000	1000	0.5	25	0	25	5	0.669	
0	1000	1000	0.5	25	2	25	5	0.713	
0	1000	1000	0.5	25	5	25	5	0.713	
0	1000	1000	0.1	25	0	25	5	0.658	
0	1000	1000	0.1	25	2	25	5	0.666	
0	1000	1000	0.1	25	5	25	5	0.666	
0	1000	1000	0.05	25	0	25	5	0.659	
0	1000	1000	0.05	25	2	25	5	0.658	
0	1000	1000	0.05	25	5	25	5	0.658	
0	1000	1000	0.5	50	2	25	5	0.732	Epochs 15
0	1000	1000	0.5	50	5	25	5	0.733	Epochs 15
0	1000	1000	0.5	25	2	25	5	0.727	Epochs 15
0	1000	1000	0.5	25	5	25	5	0.726	Epochs 15

After training it on the parameters highlighted in green (Lr=0.5, Hidden units=50, steps= 5, batch size=25, anneal=5). The accuracy on the full development set after training on the complete training set was:

Trainsize	Devsize	Lr	Hidden	steps	batchsize	anneal	Accuracy
25000	1000	0.5	50	5	25	5	0.868

Q4:

Number prediction with RNNLM	
Accuracy on dev set	0.723
Accuracy on test set	0.707

It is interesting to note that even without direct supervision the model can perform decently well. However, we must note that the accuracy value is still much lower than model trained in Q3, which has had direct supervision. In Q2 we train the model to predict the next word (a lot of choices) vs in Q3 where we just predict the subject-agreement of the verb (binary). So, when we apply the weights of Q2 we are hoping that the model learned to capture the subject-verb-agreement while learning to predict next words. This being an implicit task results in lower accuracy.

Also, note this model only compares given two verbs if it picks the one with correct agreement, and doesn't imply that it is predicting the correct agreement in general.

I.e. for

1) The plates are on the table.

2) *The plates is on the table.

3) The plates go on the table.

4) *The plates goes on the table.[1]

We don't know if it also predicts $\mathrm{P}(\mathrm{go} \mid \mathrm{x}) > \mathrm{P}(\mathrm{goes} \mid \mathrm{x})$, if the actual word in the sentence was "are" instead of "goes". I.e. maybe the model learnt to predict the word "are" and not the Subject-Verb Agreements.

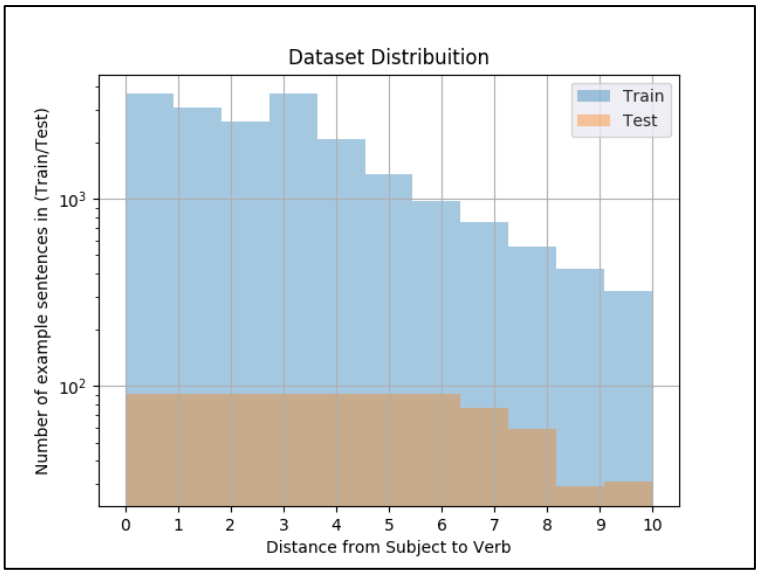
[1] * Represents grammatically incorrect sentences. I.e. sentence 2 and 4 are considered incorrect.

Hypothesis: The performance should increase (loss decrease) with the number of back steps for a distance between the subject and the verb.

Experiment Design: We will train an RNN-NP model to predict the subject-verb agreement. It will be trained on the best-known parameters while varying the back step-size from (0-10) and observing their loss on the development set. For this, we would need a properly represented training set and test set. We will then analyse the loss found for each distance category. We will also analyse the average gradient values after each time step and find some relation between the average gradient values obtained and the experiment results.

Data:

To test this hypothesis, we had to categorize the sentences based on the distance of the Subject to the verb. Using "wiki-train.txt", we extracted enough sentences to make the model learn while trying the best to make sure the distance measure was evenly split. For development set we had to combine the test set and move some part of train set to make sure that the dev set had enough evenly distributed sentences.



Example sentence, with their category: subject verb

the NNP NNP plant is the first of its kind in Russia. -> Distance = position(is)-position(plant)-1 = 0

the mark of a moderate man is freedom from his own Ideas. -> Distance = p(is)-p(mark)-1 = 4

Experiment:

We tried lots of different parameters like varying the vocab size, varying the anneal value, running for 20 epochs, or changing the hidden units to values like 100 or 150. After all those trials we decided to stick to the Q2 parameter values as they provided a better and faster comparison across the previous question models. So, the parameters were:

Trainsize	Devsize	Lr	Hidden	steps	batchsize	anneal
25000	1000	0.5	50	5	25	5

As described above they were trained on a somewhat evenly split training data total of 20,000 sentences, and 1500 development sentences. The results we obtained are shown in *Table 2*, and *Table 1*.

We see in *Table 2* that for Loss on distance 0-5 there is simple trend where greater back-steps results in a lower loss. For Loss on distance 6-10, there is no clear pattern, where back-step = 2 has a lower loss or back-steps = 5 has a lower loss. This implies that the model can't properly learn long term dependencies even with higher back-steps. Moreover, for loss on distance 0-5 we see a gradual decrease in rate of change of loss, specifically we can see in *Table 1* that after both steps 0,1, and 5 the change in loss very significantly dropped. This means the model is learning little after 2, or 5 lookback-steps.

	Distance of Subject from the Verb.										
Look Back	Loss 0	Loss 1	Loss 2	Loss 3	Loss 4	Loss 5	Loss 6	Loss 7	Loss 8	Loss 9	Loss 10
0	0.4901	0.5815	0.5111	0.5141	0.5183	0.6151	0.4577	0.6323	0.5288	0.4593	0.8133
1	0.4717	0.4725	0.4393	0.4495	0.4593	0.5925	0.4370	0.5706	0.4727	0.6153	0.8493
2	0.4632	0.4036	0.3945	0.4190	0.4038	0.5535	0.4152	0.5373	0.4504	0.6564	0.7688
3	0.4546	0.3920	0.3804	0.4137	0.3968	0.5308	0.4263	0.5416	0.4612	0.6461	0.7565
4	0.4470	0.3801	0.3716	0.4100	0.3828	0.5097	0.4242	0.5399	0.4604	0.6430	0.7563
5	0.4372	0.3724	0.3596	0.4003	0.3714	0.4848	0.4157	0.5419	0.4633	0.6233	0.7443
6	0.4340	0.3688	0.3619	0.4004	0.3692	0.4792	0.4161	0.5424	0.4624	0.6307	0.7487
7	0.4335	0.3644	0.3615	0.3994	0.3671	0.4780	0.4177	0.5437	0.4640	0.6447	0.7479
8	0.4325	0.3615	0.3616	0.3993	0.3646	0.4738	0.4196	0.5437	0.4644	0.6442	0.7512
9	0.4313	0.3594	0.3620	0.3970	0.3638	0.4703	0.4199	0.5448	0.4674	0.6409	0.7550
10	0.4314	0.3585	0.3632	0.3961	0.3644	0.4709	0.4218	0.5467	0.4702	0.6441	0.7596

Table 2. It describes the relation of Look back steps with the Loss at distance d. The Green values indicate high loss and gradients towards Red indicating lower loss.

We'll now refer to the average gradient values and how they change after every epoch to better understand the results.

Average Gradient Values:

We see in Figure 1 that with the average gradient value decrease after every step back. Moreover, in epoch 1 it particularly drops off after back step 0,1,2, 4. For later epochs like 3 and 5 the gradient values seem to be exploding. This is probably because with the decrease in gradient backstep=0 implies that weight U is making better predictions at the final time step but is having difficulty to extract the features from the previous hidden layers, thus we can find the higher gradients in larger time steps at higher epochs.

Look Back	Loss overall
0	0.5449
1	0.5021
2	0.4665
3	0.4604
4	0.4525
5	0.4426
6	0.4417
7	0.4415
8	0.4407
9	0.4400
10	0.4410

Table 1 Average Loss for the look back steps.

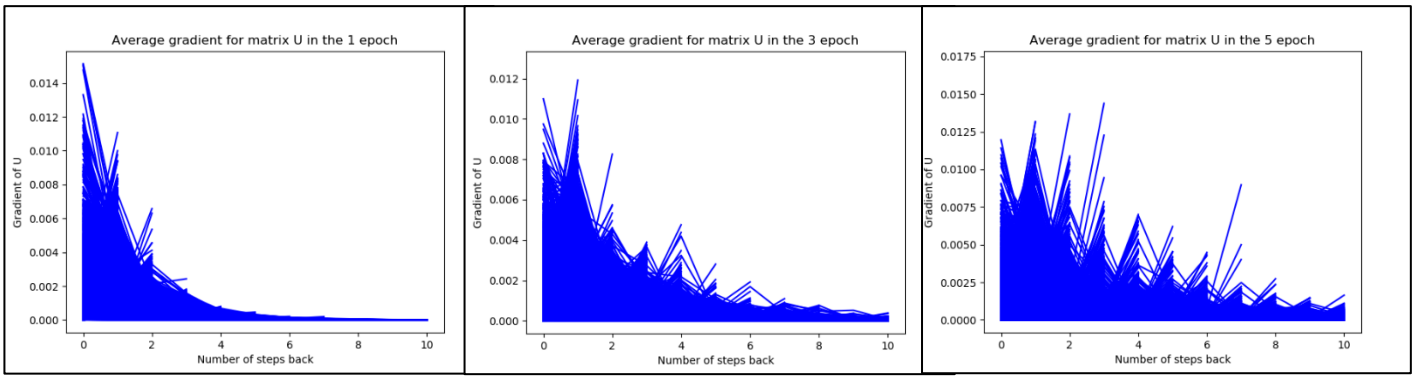


Figure 1 Average gradient values decreasing with respect to the number of steps back in the back propagation for 25,000 sentences.

Conclusion:

We proved that for the short distances between the subject and the verb (0-5) the performance increase with respect to the number of back steps. However, for longer distances such a relation does not hold due to the vanishing gradient problem. We observe in the Average gradient values as they decrease rapidly with the number of steps back. This means that the RNN cannot easily capture long term dependencies.