NLU Coursework:
Recurrent Neural Networks

s1765864 **s1780875**

General experimental settings

Unless otherwise stated, all experiments in this report use the following default parameters defined in the given coursework code:

- Batch size = 100,
- Learning rate annealing = 5,
- Minimum change (early stopping) = 0.0001.

Moreover, code for question 4 (b) requires Keras 2.1.x.

Question 2: Language Modeling

(a) Hyperparameter search

First we performed a grid search with the suggested hyperparameters: `lr` $\in \{0.5, 0.1, 0.05\}$, `hdim` $\in \{25, 50\}$, `lookback` $\in \{0, 2, 5\}$, and then expanded our search according to our findings. The initial exploration showed that higher learning rates lead to smaller adjusted loss, so we also trained models with a learning rate of 1.0 for 10 epochs. Furthermore, we were interested in whether a model with more parameters would result in lower error, so we measured the error with 100 hidden units as well. Table 1 summarises the adjusted losses with a learning rate of 1.0.

Hidden units	Lookback	Adjusted loss
25	0	5.285
25	2	5.260
25	**5**	**5.248**
50	0	5.377
50	2	5.360
50	5	5.412
100	0	5.604
100	2	5.447
100	5	5.313
15	2	5.277
15	5	5.288

Table 1: Adjusted loss of the model after 10 epochs with a learning rate of 1, with different hyperparameters. Best result is shown in bold.

A lower number of hidden units tended to result in smaller error (adjusted loss), so we performed tests with 15 hidden units as well.

We expected the performance to increase with the amount of lookback in backpropagation through time (BPTT), because the gradient calculation is correct only if we backpropagate the error gradients along the

length of the whole sentence. This was indeed the case with 100 hidden units (5% decrease in error with lookback of 5 vs no BPTT). However, for 50 hidden units the error was 0.5% *higher* with a lookback of 5 than without BPTT. We suspect the reason for this behaviour is the small number of training examples. Backpropagating through 5 time steps instead of only 2 resulted in only a minute decrease in adjusted loss (5.315 vs. 5.336 when averaged over different number of hidden units, with learning rate 1.0).

Overall, the model with 25 hidden units, lookback of 5 and initial learning rate of 1.0 resulted in the lowest adjusted loss with the reduced training set and training time, so we chose these hyperparameters for our model in the next question. According to Mikolov et al. (2010), a higher number of training examples warrants a higher number of hidden units, so maybe the model with 50 or even 100 units would have performed better than ours on the training set of 25,000 examples.

(b) Model performance using 25,000 training examples

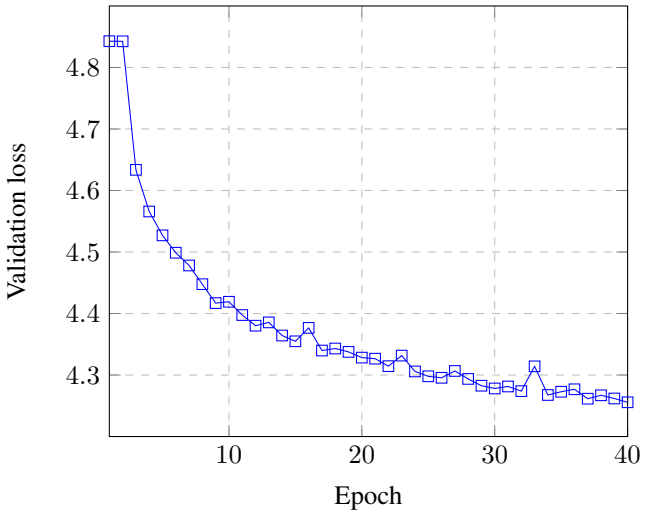

Figure 1: Learning curve with the larger training set.

We trained a model with 25 hidden units and a BPTT lookback of 5, with an initial learning rate of 1.0 for 40 epochs (Figure 1). The validation error was still decreasing at the end of the training, which is in line with the findings of Mikolov et al. (2010), who also found that RNNs are hard to overfit. The best observed validation loss was 4.256 at epoch 40, meaning an unadjusted perplexity of 70.50, which was 91.45 after adjusting for the missing vocabulary. On the test set, this means a loss of 4.265, meaning unadjusted/adjusted perplexities of 71.14 and 92.38.

Question 3: Predicting Subject–Verb Agreement

(b1) Hyperparameter tuning

We applied a similar approach to Question 2 (a), by performing a hyperparameter grid search over the following ranges: `lr` $\in \{1.0, 0.5, 0.1, 0.05\}$, `hdim` $\in \{25, 50, 100\}$, `lookback` $\in \{0, 2, 5\}$. Again, higher learning rates resulted in lower loss on validation set. Table 2 summarises the losses with a learning rate of 1.0.

In contrast to our findings in Question 2 (a), this experiment shows that hidden layers with more units result in better performance. Surprisingly, a BPTT lookback of 2 performed better than a lookback of 5 for a number of hidden units equal to 25 and 100, and resulted in very similar loss for a number of hidden units equal to 50. This is a counter-intuitive result, as we would expect an increase in performance when the model tries to capture longer-range dependencies. However, we can explain this result by observing that the number agreement is mostly determined by just one word (the subject head), and our validation dataset has an average subject–verb distance of ≈ 2.58 words, which indicates the model might be exposed to more noisy information when using a higher BPTT lookback.

Hidden units	Lookback	Loss
25	0	0.637
25	2	0.608
25	5	0.628
50	0	0.609
50	2	0.602
50	5	0.597
100	0	0.587
100	**2**	**0.585**
100	5	0.599

Table 2: Loss of the model after 10 epochs with a learning rate of 1, with different hyperparameters. Best result is shown in bold.

(b2) Training on a larger dataset

Finally, we used the best hyperparameters as shown in Table 2 (`lr` $=$ 1.0, `hdim` $=$ 100, and `lookback` $=$ 2) to train the model on a larger training set consisting of 25,000 sentences, while calculating the loss after each epoch on the same development set of 1000 sentences as before. Figure 2 shows the progression of the loss on development set during training.

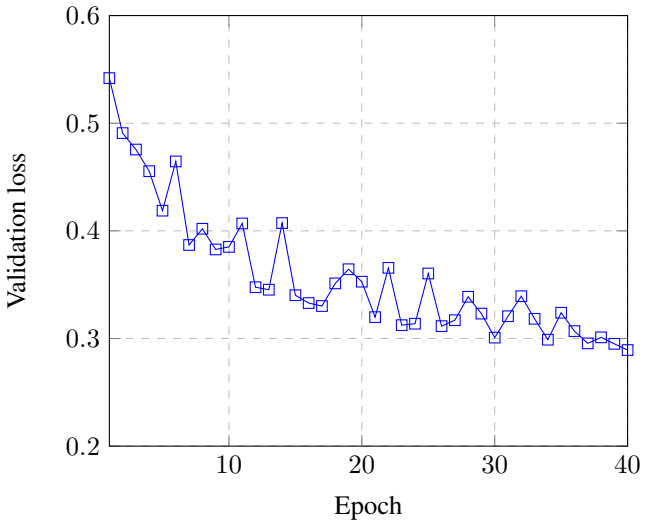

Figure 2: Learning curve with the larger training set.

The final model performance after 40 epochs is summarised in Table 3. The values on development and test sets are very similar and indicate the model has good generalisation capability. We believe the results could be improved by letting the model train for more epochs.

Dataset	Loss	Accuracy (%)
Dev	0.289	87.6
Test	0.285	88.2

Table 3: Model performance for the number prediction task on development and test sets.

Question 4: Number Prediction with an RRNLM

(a) Results with suggested method

For the number prediction without direct supervision, we used the same model as in Question 2 (b). This resulted in an accuracy of 66.9% on the development set and 65.7% on the test set, which is significantly worse than the results reported for the supervised model in Question 3 (b). Moreover, it is worth to note that the validation set contained 659 sentences with singular verbs and 341 sentences with plural verbs, meaning the baseline accuracy on this task is 65.9% on the development set. These results confirm our intuition that a model trained with supervision specifically designed for the task should perform better.

(Report continues on next page.)

(b) Further experiments

Introduction We believe the number prediction task is a good proxy to investigate the power of a recurrent model to learn long-range dependencies. In particular, we set out to answer the following questions: 1) Does the performance of the RNN model degrade as the distance between the verb and subject increases? 2) Does an LSTM perform better than our vanilla RNN on longer verb–subject distances? 3) Can we assess which words in the input sentence the model consider more important for number agreement?

To address the first two questions, we designed a quantitative experiment to measure the model performance with varying verb–subject distances. For the last question, we build an *attentional* model to conduct a qualitative investigation.

Quantitative experiment: long-range dependencies

For this experiment, we implemented a simple RNN and a long short-term memory (LSTM) model (Sundermeyer et al., 2012) with the same number of hidden units we used in Question 3. In theory, the LSTM cell fixes the problem of vanishing gradients, and adds gating units that might help store and retain the most relevant information in the hidden state (i.e. the subject word, or the number of the subject). The validation sentences were binned according to the distance between subject and verb to show statistically more significant results, and the model accuracies were measured on the test set (Table 4).

Distance	Number of examples	Q3 model accuracy	RNN accuracy	LSTM accuracy	Baseline
1	2634	91.5%	92.6%	**93.2%**	66.6%
2–3	454	83.3%	93.0%	**93.4%**	65.6%
4–5	477	84.5%	94.1%	**95.0%**	71.3%
6+	435	77.2%	83.0%	**88.5%**	70.6%

Table 4: Accuracy of RNN and LSTM models as a function of subject–verb distance. "Q3 model accuracy" refers to the RNN model we trained in Question 3 (b), "RNN/LSTM accuracy" refer to the models we trained with Keras, and the baseline is the ratio of singular verb targets in the test set.

We can see that every model attains their peak performance when the subject–verb distance is small, and with high distances the accuracy drops. The accuracy of the Keras models stays approximately the same for distances up to 5. Intuitively, as the distance grows, the 100-dimensional state vector accumulates more and more information about the sentence, and the information about the subject is more likely to be overwritten by an irrelevant word later on.

The LSTM model performed better than the simple RNN in every distance group, but especially on longer subject–verb distances. This empirically confirms our hypothesis that the LSTM cell is better at modelling long-range information.

The RNN we trained with Keras for 10 epochs outperformed the Question 3 implementation trained for 40 epochs by a wide margin. Both had the same number of hidden units and input vocabulary size, and both were trained on the same training set, but in Keras we used different weight initialisers, and the RMSprop optimiser, which is a per-parameter adaptive learning rate method (Tieleman and Hinton, 2012). To prove that these causes account for the differences, we trained an RNN in Keras with the same settings that our implementation used, which resulted in accuracies $\pm 1\%$ from those of our model. This shows that these choices have a huge influence when training a model, and our model was very far from reaching its full potential.

Qualitative experiment: modelling attention

The number prediction task has a very peculiar property: nearly all the information that determines the verb inflection is concentrated in one word, which is the head of the syntactic subject. Inspired by Bahdanau et al. (2014), we designed an *attentional* model that allows us to qualitatively interpret how our recurrent model works, and if it is able to pay more attention to the relevant word in the input sentence.

Model description This model is conceptually very similar to our previous RNN model from Question 3. One important difference concerns the use of a bidirectional recurrent layer to address the problem

of bias towards recent words (Bahdanau et al., 2014). We also add an attention layer that calculates an *expected context vector* based on learned attention weights. Since we're predicting a single binary output (and not sequences), our model does not use information from previously generated tokens, and we calculate the attention weight α_{it} for a word t in sentence i as follows (Yang et al., 2016):

$$u_{it} = \tanh\left(W_w h_{it} + b_w\right) \qquad \alpha_{it} = \frac{\exp u_{it}^\top u_w}{\sum_t \exp u_{it}^\top u_w},$$

where h_{it} is the concatenation of the forward and backward hidden representations given by the bidirectional RNN. W_w, b_w, and u_w are learned parameters. We used the default implementation provided by the Keras framework for bidirectional recurrent layers. For the attention layer, we used a custom Keras layer code published by Baziotis (2018). For more implementation details refer to file `attention.py` accompanying this report.

Experiment Our training procedure is almost the same as for the RNN model in Question 3. We use the same training set of size 25,000 and development set of size 1000. Both the bidirectional encoder and the decoder have 200 hidden units. We train the model for a maximum of 40 epochs and apply early stopping when the model fails to improve the best validation loss for 3 epochs. For training, we use an RMSprop optimiser (Tieleman and Hinton, 2012) with a minibatch of 16 sentences and a dropout rate of 0.5 is applied for regularisation.

Sentence/Attention map	True Label	Prediction
indirect written **sources** 	VBP	VBP
the foundation 's **mission** 	VBZ	VBP
journals about law 	VBP	VBP
the adult critical care **specialist** 	VBZ	VBZ
the **goal** of dynamic priority NN 	VBZ	VBZ

Table 5: Sample attention maps for sentences in test set. The true head of the subject in bold. Below the sentences, a grayscale bar show the attention weights ranging from 0 (black) to 1 (white).

Results and Discussion After only 9 epochs, the training stopped with the model scoring an accuracy of 92.9% on the test set. During inference we can examine the values of the attention weights and check if the model is able to determine the head of the noun phrase and choose the number accordingly. Table 5 illustrates examples of the attention distribution over the input sentences.

In most cases, the highest attention values are over or near the true head of the subject, and in 55.9% of cases the model attends to the correct head. Notably, in the fourth sentence, the model gave the correct label by accident as it wrongly considered the word "adult" as the most relevant. The second sentence shows that the possessive construction *'s* may be a problem for this number prediction task.

In conclusion, we demonstrated how the attentional model can convey some intuition about how the model is operating during inferences. The attention weights can be useful to uncover weaknesses and give insights on how to improve the model architecture. Finally, such qualitative analysis may inform the acquisition of new labelled datasets, to increase the number of training examples addressing the attentional failures.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Baziotis, C. (2018). Keras layer that implements an attention mechanism. `https://gist.github.com/cbaziotis/7ef97ccf71cbc14366835198c09809d2`. Accessed: 2018-03-15.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, 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.

Sundermeyer, M., Schlüter, R., and Ney, H. (2012). LSTM neural networks for language modeling. In *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*, pages 194–197. ISCA.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.