Deep Learning practical 2

Jochem Soons
MSc Artificial Intelligence
University of Amsterdam
UvA-id: 11327030
Date: 30-11-2020
jochem.soons@student.uva.nl

1 Recurrent Neural Networks

1.1 Vanilla RNNs

Question 1.1

a)

$$\frac{\delta \mathcal{L}^{(T)}}{\delta \boldsymbol{W}_{ph}} = \frac{\delta \mathcal{L}^{(T)}}{\delta \hat{\boldsymbol{y}}^{(T)}} \frac{\delta \hat{\boldsymbol{y}}^{(T)}}{\delta \boldsymbol{p}^{(T)}} \frac{\delta \boldsymbol{p}^{(T)}}{\delta \boldsymbol{W}_{ph}}$$

b)

$$\begin{aligned}\frac{\delta \mathcal{L}^{(T)}}{\delta \boldsymbol{W}_{hh}} &= \frac{\delta \mathcal{L}^{(T)}}{\delta \hat{\boldsymbol{y}}^{(T)}} \frac{\delta \hat{\boldsymbol{y}}^{(T)}}{\delta \boldsymbol{p}^{(T)}} \frac{\delta \boldsymbol{p}^{(T)}}{\boldsymbol{h}^{(T)}} \frac{\boldsymbol{h}^{(T)}}{\delta \boldsymbol{W}_{hh}} \\ &= \sum_{i=0}^{T} \frac{\delta \mathcal{L}^{(T)}}{\delta \hat{\boldsymbol{y}}^{(T)}} \frac{\delta \hat{\boldsymbol{y}}^{(T)}}{\delta \boldsymbol{p}^{(T)}} \frac{\delta \boldsymbol{p}^{(T)}}{\boldsymbol{h}^{(T)}} \frac{\boldsymbol{h}^{(T)}}{\boldsymbol{h}^{(i)}} \frac{\boldsymbol{h}^{(i)}}{\delta \boldsymbol{W}_{hh}} \\ &= \sum_{i=0}^{T} \frac{\delta \mathcal{L}^{(T)}}{\delta \hat{\boldsymbol{y}}^{(T)}} \frac{\delta \hat{\boldsymbol{y}}^{(T)}}{\delta \boldsymbol{p}^{(T)}} \frac{\delta \boldsymbol{p}^{(T)}}{\boldsymbol{h}^{(T)}} \Big(\prod_{j=i+1}^{T} \frac{\boldsymbol{h}^{(j)}}{\boldsymbol{h}^{(j-1)}} \Big) \frac{\boldsymbol{h}^{(i)}}{\delta \boldsymbol{W}_{hh}}\end{aligned}$$

c) $\frac{\delta \mathcal{L}^{(T)}}{\delta \boldsymbol{W}_{ph}}$ is only dependent on the current state ($h^{(T)}$, which is part of the expression for $\boldsymbol{p}^{(T)}$), while $\frac{\delta \mathcal{L}^{(T)}}{\delta \boldsymbol{W}_{hh}}$ is dependent on all previous recurrent states ($\sum_{i=0}^{T} \boldsymbol{h}^{(i)}$). So the latter has temporal dependence on all previous states while the former is only dependent on the current state. So if we train the model for a large number of timesteps and we want to compute the latter gradient $\frac{\delta \mathcal{L}^{(T)}}{\delta \boldsymbol{W}_{hh}}$, we would then have to compute a very large amount of gradients of all the previous states w.r.t. each other (the term $\prod_{j=i+1}^{T} \frac{\boldsymbol{h}^{(j)}}{\boldsymbol{h}^{(j-1)}}$). Two problems might occur here: 1) it becomes computationally infeasible to calculate all the gradients and 2) we might encounter exploding or vanishing gradients because we multiply many gradients with each other.

1.2 Long Short-Term Memory (LSTM) network

Question 1.2

a) 1. Input modulation gate $\mathbf{g}^{(t)}$:
The purpose of this gate is to create a vector of new candidate values that could be added to the cell state, to update our cell state values. It uses tanh as a non-linear activation function. This is a good

choice, because the output from tanh can be positive or negative (between -1 and 1), allowing for increases and decreases in the values of the cell state. Besides, the tanh function is relatively easy to differentiate.

2. Input gate $\mathbf{i}^{(t)}$:
The purpose of this gate is deciding which values of our cell state we should update. This is done by multiplying the output of this gate with the output of the input modulation gate before adding the input modulation values to the cell state. It uses the non-linear sigmoid activation function, which outputs a number between 0 and 1. This is a good choice because of this output range: here a 1 represents a strong update of a cell state value while a 0 represents that we do not update this value.

3. Forget gate $\mathbf{f}^{(t)}$:
The purpose of this gate is to decide what information we are going to throw away or "forget" from the cell state. This is done by multiplying the output of this gate with the cell state. It also uses the non-linear sigmoid activation function, that outputs values between 0 and 1. This is a good choice because here a 0 means that we get rid of all information from the cell state value, while a 1 indicates that we keep the cell state value.

4. Output gate $\mathbf{o}^{(t)}$:
The purpose of this gate is to decide which parts of the (updated) cell state we are going to output. Once again this layer uses the sigmoid activation function and it is also here a good choice because of the output range of the function: here a 1 indicates that we will fully output the cell state value while a 0 represents that we do not output (update) this cell state value.

b) I assume $d = \text{N\_input}$:

$\{W_{gx}, W_{ix}, W_{fx}, W_{ox}\} \in R^{\text{N\_input} \times \text{N\_hidden}}$
$\{W_{gh}, W_{ih}, W_{fh}, W_{oh}\} \in R^{\text{N\_hidden} \times \text{N\_hidden}}$
$\{b_g, b_i, b_f, b_o\} \in R^{\text{N\_hidden}}$
$W_{ph} \in R^{\text{N\_hidden} \times \text{N\_output}}$
$b_p \in R^{\text{N\_output}}$

So our total number of trainable parameters is:

$$4\Big((\text{N\_input} \times \text{N\_hidden}) + \text{N\_hidden}^2 + \text{N\_hidden}\Big) + (\text{N\_hidden} \times \text{N\_output}) + \text{N\_output}$$

1.3 LSTMs in PyTorch

Question 1.3

The performance results of the three runs with different seeds (handpicked, so that I could use the same seeds for Question 1.4) on the two sequence lengths (10, 20) on the binary palindrome dataset, in terms of steps needed until convergence and the final test accuracies are displayed in table 1. Here the final test accuracies were obtained by generating and testing on 5000 new data samples. The corresponding plots of the different training runs are shown in figure 1.

From the table and the plots it becomes clear that the LSTM model achieved perfect accuracy on all runs. However, the model did have a lot more difficulties in learning the "trick" of recognizing the palindromes for T=20. What happened during these runs is that the model would plateau at some accuracy/loss value but then after a while it suddenly learned the dependency of the palindromes and it would convergence to perfect performance. One remark I would like to make here is that this perfect performance for the T=20 sequence length could only be achieved when I implemented my model with a relatively large embedding dimension: I used 2 * seq_length. If the model had a lower embedding dimension, the sudden step to perfect performance could often not be made. This illustrates the importance of the embedding dimension of our input: a larger embedding dimension increases the expressive power of our LSTM model (which seems logical since we increase the number of parameters in our weight matrices and the embeddings themselves

are also learned parameters).

Another remark I would like to make is that I performed an early stopping method (you can see this in the seperate plots in figure 1a and 1b), which stopped training when the loss during evaluation became lower than $0 + \epsilon$ with $\epsilon = 0.005$ (so training stopped when the model converged to perfect performance). Normally this would not be a good criteria because this method assumes perfect convergence on training data, but in this specific case it was a better choice than some other criteria that would stop training when loss plateaus, because then the model could not have converged to perfect accuracy on T=20 (because of the behaviour pointed out before).

Seed	Sequence length (T)	Steps until convergence	Test acc.	Std. dev.
0	10	300	1.00	-
1	10	300	1.00	-
2	10	360	1.00	-
Averaged	10	320	1.00	0
0	20	1260	1.00	-
1	20	1080	1.00	-
2	20	2280	1.00	-
Averaged	20	1540	1.00	0

Table 1: Overview of performance of the **regular LSTM model** for sequence lengths T=10 and T=20 run over random seeds 0,1 and 2. Testing was performed over N=5000 samples. Respective averaged scores are shown below the separate runs.

1.4 Peephole LSTM

Question 1.4

The performance results of the three runs with different seeds (the same as in Question 1.3) on the two sequence lengths (10, 20) on the binary palindrome dataset for the Peephole LST model are displayed in table 2. Again, the final test accuracies were obtained by testing on 5000 newly generated data samples. The corresponding plots of the different training runs are shown in figure 2. Moreover, I used the same early stopping criteria and the same embedding dimension as in Question 1.3 for the Peephole model, which is 2 * seq_length.

Similar to the regular LSTM model in 1.3, the Peephole LSTM achieves perfect test accuracy on all the training runs. However, one important difference in results compared to the regular LSTM model is that the Peephole model has much less difficulties in learning the binary palindrome pattern for T=20. While the normal LSTM model needed an average of 1540 steps for convergence (table 1, the Peephole LSTM only needs an average of 540 steps to reach perfect accuracy. This shows that directly looking at the cell state instead of the hidden states can bring benefits in learning dependencies along longer sequences.

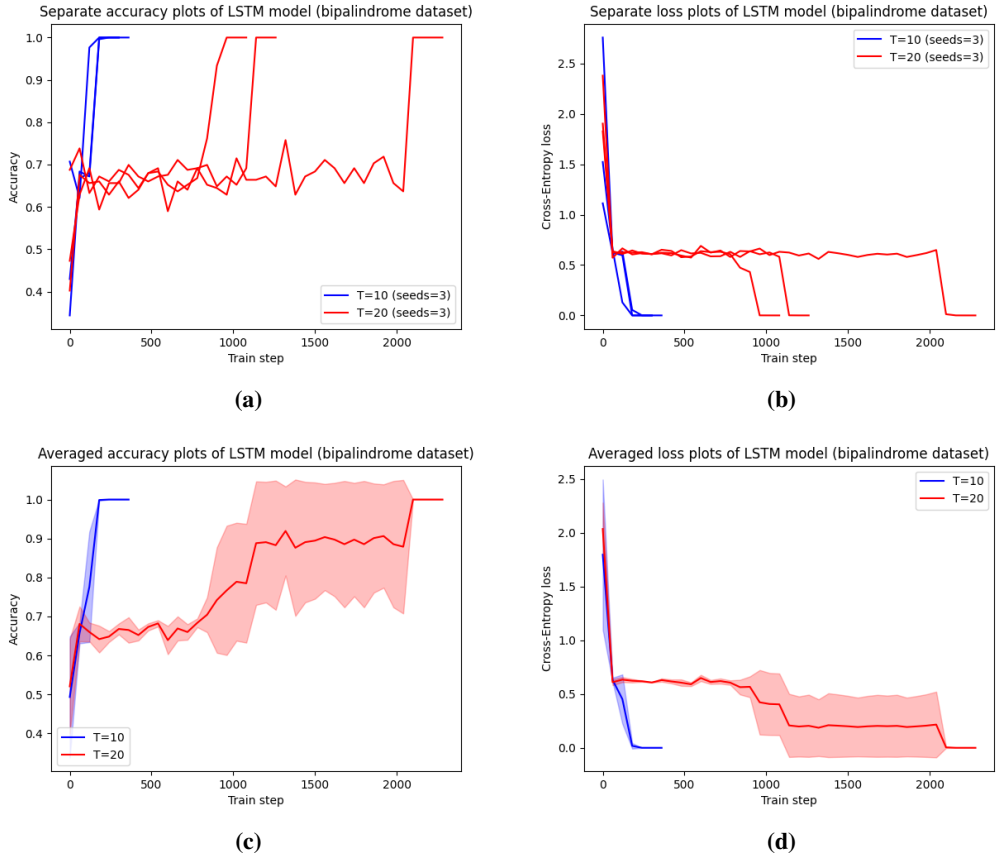


Figure 1: Training accuracy and loss plots throughout the training process of the **regular LSTM model** run with three different random seeds for sequence lengths T=10 and T=20. **(a)** and **(b)** display the separate accuracy and loss curves of the three runs, while **(c)** and **(d)** display the corresponding averaged curves plotted with the standard deviation over the runs.

Seed	Sequence length (T)	Steps until convergence	Test acc.	Std. dev.
0	10	480	1.00	-
1	10	420	1.00	-
2	10	420	1.00	-
Averaged	10	440	1.00	0
0	20	480	1.00	-
1	20	720	1.00	-
2	20	420	1.00	-
Averaged	20	540	1.00	0

Table 2: Overview of performance of the **Peephole LSTM model** for sequence lengths T=10 and T=20 run over random seeds 0,1 and 2. Testing was performed over N=5000 samples. Respective averaged scores are shown below the separate runs.

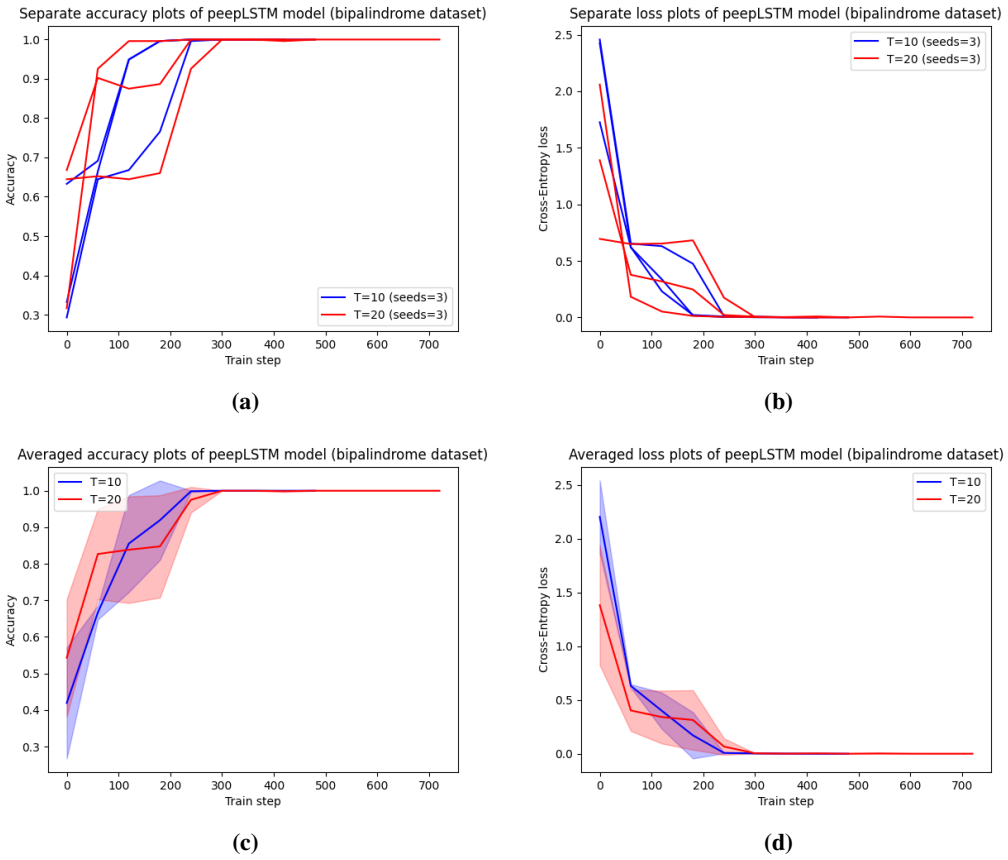


Figure 2: Training accuracy and loss plots throughout the training process of the **Peephole LSTM model** run with three different random seeds for sequence lengths T=10 and T=20. **(a)** and **(b)** display the separate accuracy and loss curves of the three runs, while **(c)** and **(d)** display the corresponding averaged curves plotted with the standard deviation over the runs.

2 Recurrent Nets as Generative Model

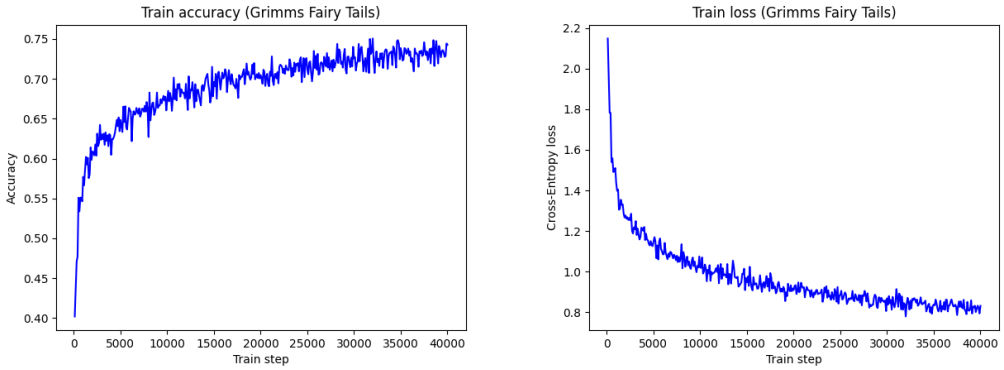
Question 2.1a)

The hyperparameters I used for training the LSTM model for three different datasets are shown in table 3. As can be seen from the table, I did not change many of the default hyperparameters, because training the model with the default configuration already produced decent performance. In fact, the only thing I changed was increasing the number of features in the hidden state to 512, so that the expressive power of the LSTM increased and the model could more easily learn on the training data.

The training accuracy and loss plots of the two-layer LSTM network on the Grimms Fairy Tails book are displayed in figure 3. For simplicity purposes I only included the training plot of this dataset here. Plots for the other two datasets can be found in Appendix A. As can be seen from figure 3, after 40 000 steps the model achieved a training accuracy around 0.70 - 0.75.

Dataset	Learning rate	Batch size	Optim.	Num. hidden	Num. layers	Seq. length	Embed. dim.	Steps
Grimms Fairy Tails	2e-3	64	Adam	512	2	30	60	40 000
Darwins reis om de wereld	2e-3	64	Adam	512	2	30	60	42 000
Democracy in the US	2e-3	64	Adam	512	2	30	60	50 000

Table 3: Overview of hyperparameters used when training the LSTM model on the Grimms Fairy Tails book.



(a) **(b)**

Figure 3: Training accuracy **(a)** and loss **(b)** plots throughout the training process of the LSTM model on the Grimms Fairy Tails book using the hyperparameters specified in table 3.

Question 2.1b)

NOTE: For simplicity purposes (and to limit the number of shown examples) I only show generated sentences from the Grimms Fairy Tails book throughout training for this question. Moreover, instead of generating at 1/3, 2/3 and 3/3 of training I generated sentences at train steps 2000, 10 000 and 40 000 of the training process. I chose these steps because the model converges fairly quickly to reasonable performance (i.e. > 0.6 acc.) and therefore showing generated sentences at 1/3, 2/3 and 3/3 of training would not be that interesting for analyzing differences. The generated sentences for this question can be found below in the section between the solid lines.

There are several conclusions we can draw from these sentences. Firstly, the results illustrate is that early in the training process (step = 2000) the model has learnt to make simple words but it has not yet learnt to make longer, meaningful phrases or sentences. The model just repeats a small number of words (at most three). When it has seen more sequences, at steps 10 000 and later 40 000,

it does have the ability to generate somewhat more complex text samples.

Secondly, and perhaps more interesting, is that the model exhibits loop-like behaviour in generating new text, even when it is in the later stages of training. Moreover, we can see this at steps 10 000 and 40 000 especially at the longer sentences of T=120 characters and not at the sentences of length T=30. For these longer sentences, the model repeats a phrase like "The son was so sorrowful" or "I will go home and see me free" in almost every generated sentence. Interestingly, these repeated phrases are roughly of length T=30, which is the same sequence length our model was trained on. This behaviour is therefore understandable, as our model cannot learn dependencies stretching longer than 30 characters, which combined with our greedy sampling approach results in these loops during sampling.

GENERATED SENTENCES FOR QUESTION 2.1b)

Sentences generated at train step 2000:

T = 15:
1) -the wind and w
2) ke the wind and
3) quite with the

T = 30:
1) E THE THE THE THE THE THE THE
2) and the wind and when the wind
3) -the wind and when the wind an

T = 120:
1) was all the wind and when the wind and when the wind and when the wind and when the wind and when the wind and when the
2) ’ Then the wind and when the wind and when the wind and when the wind and when the wind and when the wind and when the
3) ing the wind and when the wind and when the wind and when the wind and when the wind and when the wind and when the wind

Sentences generated at train step 10 000:

T = 15:
1) g the star-gaze
2) Cap,’ said th
3) s and said: ‘

T = 30:
1) ing the stars: but the son was
2) Little Red-Cap, however, had n
3) nd the son was so sorrowful th

T = 120:
1) could not see the other, and the son was so sorrowful that he had been a little dwarf said, ‘I will give you a little
2) 1.E.8. If an enchantment so that the old woman was so that he would not be able to find the bed, and the son was so sor
3) Now the seven dwarf said, ‘I will give you a little way of the sea, and the son was so sorrowful that

he had been a li

Sentences generated at train step 40 000 (finished training):

T = 15:
1) s the sisters w
2) was a garden, a
3) VE DIS WARRANTY

T = 30:
1) r the sun shone on his shoulde
2) 590 Fox? He had not the feast
3) would have been a faithful t

T = 120:
1) l the same time again. The king was very angry, and said, 'I will go home and see me free.' She thought to himself,
2) g to him, and the cat climbed up into a baker's oven. And the cook said, 'I will go home and see me free.' She tho
3) s the sisters were gone, she said to her maid, 'Pray get down, and fell on his head, and the cook said, 'I will go h

---

Question 2.1c)

The temperature parameter τ influences the sampling process by either flattening/softening or sharpening/hardening the probability distribution (SoftMax output) over the target classes. A softer distribution means a model is less confident about its predictions (leading to higher randomness) while a harder distribution is more confident in its predictions (leading to less randomness and more greediness). Because we multiply the logits with τ, a higher value of τ will harden our SoftMax output (less randomness) and a lower value of τ will soften our SoftMax output (more randomness).

You can find generated sentences of length T=120 from the three different datasets using different values of τ in the section below between the solid lines (I used T=120 so that I could examine the loop-like behaviour that became evident in Question 2.1b). As we could expect, the model does indeed perform more random sampling for a lower value of τ. Sentences generated with $\tau = 0.5$ are more or less random gibberish, with a lot of incomprehensible words and a lot of grammar/spelling errors. This behaviour does get less when τ is increased. For $\tau = \{1, 2\}$, we notice less obvious grammar or spelling errors, and the model actually seems to benefit from the balanced sampling strategy by being able to escape the loop-like behaviour that the sentences exhibit when they are obtained by greedy sampling. So by using temperature to balance random and greedy sampling strategies, the model becomes more creative in generating new text and it is able to generate sentences (longer than the sequence length it was trained on) without loops.

GENERATED SENTENCES FOR QUESTION 2.1c)

Greedy sampling (τ = None):

Grimms Fairy Tails:
1) Queen looked at the batt of the sun shone on the head out of the window, and said, 'I will go home and see me free.'
2) ch a long time again the stars, and was about to go to the baby in his hand, and said, 'I will go

home and see me free

Democracy in the US:
1) he same principles in the United States are the right of conditions is the case in the United States are the right of co
2) But the present day the same time to time in the midst of the people is at the same time to time in the midst of the peo

Darwins reis om de wereld:
1) quiterschelpen. De stad is de stad is de stroomen verscheidene malen van den morgen van de bewoners van de kust van den
2) f van den morgen van de bewoners van de kust van den morgen van de bewoners van de kust van den morgen van de bewoners v

τ = 0.5:

Grimms Fairy Tails:
1) xace and lanastone
insvered; this troubly reason to cows, by canduinlyo Manankind pair about.' Chanties (fox and villa
2) m wan, he flew awayer off.
And herself up after him, he looked, but polled it as it sumnea money occurred the help and g

Democracy in the US:
1) ?ou.
Rhas reachin. Hewschive regrect
to persuation. It woul-, danctly gere; lived, frcest
egitiet,
like him would demed
2) ; a wanton of ownliarity? hwaccging an election
altor exue
in complaintful
influting, or.–Right of Goid to 31-14,361 tr

Darwins reis om de wereld:
1) Chamispreant)-land, veêrenzed,
ori-fulvizië parnanbaan? Wie juister hem doelde" veecajedsplas, die nu vill
alledat Nwe
2) Rhiep Pedroel. (Wel Wenio Fetike, Vuur: nu bl s,-Pata-ka
hetgehoudt. Nooit uit gewapioné,
tegt de
skel Wiavitel-Bannale

τ = 1.0:

Grimms Fairy Tails:
1) AL, he looked at the door he went to the palace of their apish off, And hath that she called the door, and
how a good
2) y
pain; and at last he said, 'Take me one another,' said the fox. 'Who can go about her; with all my comrades.'

Democracy in the US:
3) American Union Bey, 1830, that tyrannical institutions, his delicious excludes from a convenient,

and it was directed as
4) 3,000 square miles;
for it constitutes as to happiness; but I inhabit them would be said to examine the Federal Governme

Darwins reis om de wereld:
1) 1739, blz. 324. Uit matkomstige huid was eene kreek, die uit
twee zijn sapperen! Zulke zonder la met een opperhoofd; maa
2) midden van Kapitein King vastgroeien. Voorafgebroed deze tafereel kregen, keerde ik nergens terugkeerde worden weinig tr

τ = 2.0:

Grimms Fairy Tails:
1) o the shoe!
The shoe is good for your tail, and was more beautiful that we do not carry to the forest, ate nothing but
2) d the others had been and the gold chain in a good cheese, and the cask quite satisfied, and said, 'Now, she is soon a

Democracy in the US:
1) But the same spot, and the most extreme destiny of the soil, and their efforts in the United States. The Indian was not
2) New World, but it is not the prevailing and his personal interests. In the States of the Union will be to be the cause o

Darwins reis om de wereld:
1) in het eiland James, die er volgens Albemarle (1775-1853) steeg in de veranderingen en zeer vermeld, dat zulk eene hooge
2) Langen zwemmen, en de meeste koraalgeslachten met zijne dagen niet zeer onbezien, doch niet verloren hebben verre eene z

---

Question 2.2 (BONUS)

For this question I manually designed three unfinished sentences per book and I let the trained models finish the sentences to a length of T=80, using temperature balanced sampling with $\tau = 2$. You can find the start en the finished sentences in the section below between the solid lines. The trained models manage to produce some pretty funny sentences given the provided starting setup. Some highlights: in sentence 1), the model seems to take a dark turn pretty early in our generated fairy tail by declaring that sleeping beauty is simply dead. In sentence 5), the model questions the sovereignty of Trump (which Donald himself would probably not agree to) and in sentence 8) the model simply declares that humans are curious creatures, which I guess is a solid statement. It also declares that we humans originate from beaches, which seems like a nice metaphor to our origins from sea creatures? Anyway, these fun little sentences show the capability of LSTM models to understand and produce texts to some level. Naturally, a lot of the generated sentences do not make too much sense and they also contain a lot of errors, but still, given the simplicity of our model, I think it really does a decent job in providing us some fun sentences to read.

GENERATED SENTENCES FOR QUESTION 2.1d)

Grimms Fairy Tails:
1) Sleeping beauty is ... $\longrightarrow$ Sleeping beauty is dead! God will soon bring in the kingdom should do, when one

2) Somewhere far away, ... $\longrightarrow$ Somewhere far away, nothing could be seen, and she agreed to give up the sun sco
3) Once upon a time ... $\longrightarrow$ Once upon a time an old fox went up to him, but they went alone the stairs were

Democracy in the US:
4) Democracy is ... $\longrightarrow$ Democracy is served for the exercise of its decision, and the most prominent ver
5) President Trump is ... $\longrightarrow$ President Trump is not the extent of the sovereignty of the United States the wh
6) I vote for ... $\longrightarrow$ I vote for them to accomplish the same end; and whilst the territory of the Unit

Darwins reis om de wereld:
7) Evolutie is ... $\longrightarrow$ Evolutie is aan het eerst ontstaan leefden, en bereikten wij den vorm van het ei
8) Menschen zijn ... $\longrightarrow$ Menschen zijn zeer merkwaardig, maar veel geleden hebben als de open vlakten van
9) Menschen stammen af van ... $\longrightarrow$ Menschen stammen af van het strand, van deze sterrenzonen een merkwaardige berge

---

3 Graph Neural Networks

3.1 GCN Forward Layer

Question 3.1

a) Firstly (and quite evidently) GCN layers exploit structural information of the graph data because they take graph-like data as input (i.e. adjacency matrices A which are a representative description of the graph structure in matrix form and a feature description for every node summarized in the N×D feature matrix H). But more importantly, GCNs are able to exploit this graph structured data by going over all nodes in the graph, where it uses the weight matrix W to transform the input of a particular node, which are the feature vectors of adjacent nodes (and its own feature vector, adding the identity to $\tilde{A}$ takes care of this self connection), to some output feature vector (message). This node in turn provides this output vector or output message to its neighbouring nodes. So by applying this process over all nodes in the graph, a GCN layer exploits the connectivity between nodes and can in fact be seen as a message-passing algorithm over the graph.

b) A problem regarding memory scalability is that the GCN layer uses a adjacency matrix A, which contains the information of connections between all different nodes (while a lot of these connections would be 0). So for large graph models, the memory requirements become extremely large because we need a very large adjacency matrix. A solution to overcome this could be to express the graph as a list of connections instead of the adjacency matrix, or use the list of edges to define some sparse adjacency matrix which requires less memory.

Question 3.2

a) $\tilde{A} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$

b) In each update step every node will receive information only from adjacent nodes. So to forward the information (feature vector) from node C to node E, we need 4 update steps.

3.2 Graph Attention Networks

Question 3.3

Using the attention mechanism means we calculate some weighted average over inputs that describe the "importance" of these inputs for calculating the output. To incorporate attention in the Graph Convolutional Layer, you would have to calculate all attention factors α_{ij} that describe the attention weight between nodes i and j (i.e. the attention we should give to to neighbouring node j in calculating output for node i). Now we only need to extend the formula of computing the output of node i by multiplying the features of all neighbouring nodes j by their respective attention weight α_{ij}:

$$h_i^{l+1} = \sigma\Big(\sum_{j \in \mathcal{N}(i)} W^{(l)} h_j^l\Big) \longrightarrow h_i^{l+1} = \sigma\Big(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} W^{(l)} h_j^l\Big)$$

NOTE: the formula for calculating the attention weights α_{ij} is somewhat more complex and involves a longer equation (and is derived in the Notebook tutorial), so I assume it is not necessary to include this in the answer here.

3.3 Applications of GNNs

Question 3.4

There are many examples, but here are two examples of real-world applications of Graph Neural Networks:

1) Traffic flow predictions [1; 2]: here the graph-based Neural Networks can exploit the spatial dependency of complicated road networks to predict how traffic (cars) flow through a network of roads, which can be represented in graph-like structure.

2) Computational chemistry and more specific: molecular geometry prediction [3; 4; 5]. Here the GNN methods use the graph-like structure of molecules, where ions or atoms are nodes and bonds are the edges between them. GNN techniques can be used to learn more about existing molecular structures but also to generate new molecular structures [6] (e.g. design of new drugs).

Because both examples provided here are both based on data that is inherently structural and can be easily defined in graph-like structures, using GNNs is obviously a good choice. A power of GNNs however, is that they can also be applied to non-structural scenarios where there is no explicit structural relationship, such as text or image classification [7].

3.4 Comparing and Combining GNNs and RNNs

Question 3.5

a) RNNs are well-suited for sequential data (e.g. a time series or a text), were the temporal window can be of infinite length. RNNs do a good job in remembering dependencies through time (long term and short term - using LSTMs). However, RNNs are only applicable when the input data is represented as a directed linear graph (so there is a sequential hierarchy between inputs, i.e. "one comes for the other"), and RNNs can only use the state of "previous" nodes in computing outputs. GNNs on the other hand can deal with far more complex input graphs (with both directed and undirected edges). It manages this by looking at relationships between all nodes and their neighbours (which can take complex structures). Furthermore, in a GNN there is no concept of time and all nodes are updated at each update step using information of neighbouring nodes (the number of update steps is a chosen hyperparameter). GNNs are a very strong and upcoming method, but they have some limitations (according to my understanding): 1) input graphs may be difficult to design due to the complexity of relations between data (and bias of the designer may be incorporated), 2) computational requirements are high when using complex/large graphs, 3) it is hard to retrieve the internal processes of the algorithm (i.e. the black box problem: "why did it get to these results?")

So, when we have data that has no specific hierarchical temporal ordering or we have input graphs in which nodes are related in a very complex manner (undirected), a GNN would be a much better technique to implement compared to a RNN. On the other side, when there is strong temporal dependency between input nodes and the input is linearly directed (there is a linear ordering), or if we deal with a input sequence of undefined length, or if we want to take a more simple approach, a RNN would probably be a good method to use.

b) Combining RNN and GNN methods has been researched by [8], who introduce the general concept of Graph Recurrent Neural Networks (GRNNs). Such a GRNN framework is created by taking a RNN and retaining the recurrence relation of the RNN nodes but replacing the input-to-state and state-to-state linear transformations within the nodes of the RNN by linear graph filters (so GCN operations). This allows the resulting GRNN model to take both the temporal and the graph structure of input data into account. Examples of tasks for which GRNNs could be useful are therefore also learning tasks on graph-structured and time-varying data, so tasks where we have graph input signals that change over time. Examples of such tasks are predicting traffic on some network of roads, epidemic studies in a community or society (the spread of the coronavirus for example), improving recommendations on a social network application, and many more.

References

[1] C. Chen, K. Li, S. G. Teo, X. Zou, K. Wang, J. Wang, and Z. Zeng, "Gated residual recurrent graph neural networks for traffic prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 485–492, 2019.

[2] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[3] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, pp. 2224–2232, 2015.

[4] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley, "Molecular graph convolutions: moving beyond fingerprints," *Journal of computer-aided molecular design*, vol. 30, no. 8, pp. 595–608, 2016.

[5] E. Mansimov, O. Mahmood, S. Kang, and K. Cho, "Molecular geometry prediction using a deep generative graph neural network," *Scientific reports*, vol. 9, no. 1, pp. 1–13, 2019.

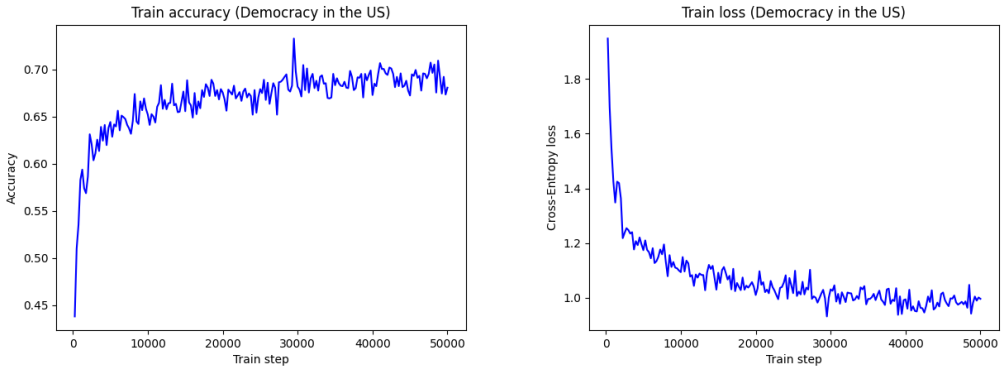
[6] J. You, B. Liu, Z. Ying, V. Pande, and J. Leskovec, "Graph convolutional policy network for goal-directed molecular graph generation," in *Advances in neural information processing systems*, pp. 6410–6421, 2018.

[7] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434*, 2018.

[8] L. Ruiz, F. Gama, and A. Ribeiro, "Gated graph recurrent neural networks," *arXiv preprint arXiv:2002.01038*, 2020.

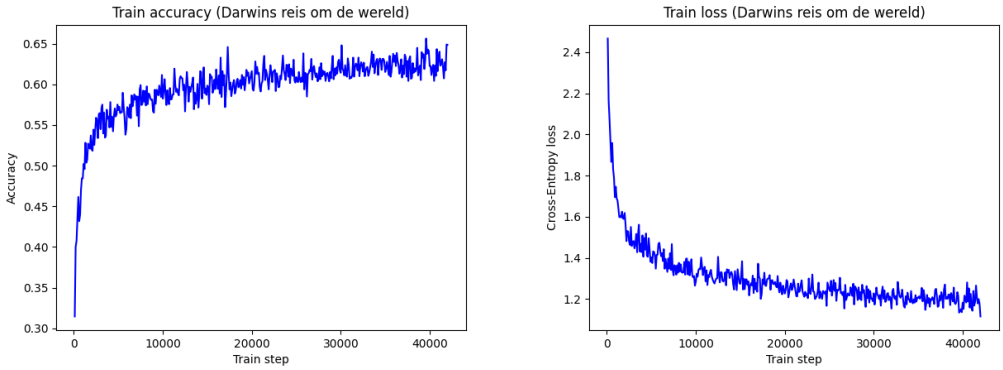
A Training plots

In this appendix the training plots are shown for the LSTM model with hyperparameters configuration described in table 3 on the "Democracy in the US" and "Darwins reis om de wereld" datasets.



(a) **(b)**

Figure 4: Training accuracy **(a)** and loss **(b)** plots throughout the training process of the LSTM model on the "Democracy in the US book" using the hyperparameters specified in table 3.



(a) **(b)**

Figure 5: Training accuracy **(a)** and loss **(b)** plots throughout the training process of the LSTM model on the "Darwins reis om de wereld" book using the hyperparameters specified in table 3.