Deep Learning - Practical 2

Ard Snijders
University of Amsterdam
12854913

1 Recurrent Neural Networks

1.1 Vanilla RNNs

Question 1.1

(a)

$$\frac{\delta \mathcal{L}^{(t)}}{\delta W_{ph}} = \frac{\delta \mathcal{L}^{(t)}}{\delta p^{(t)}} \frac{\delta p^{(t)}}{\delta W_{ph}}$$

(b)

$$\frac{\delta \mathcal{L}^{(t)}}{\delta W_{hh}} = \sum_{i=0}^{t} \frac{\delta \mathcal{L}^{(t)}}{\delta p^{(t)}} \frac{\delta p^{(t)}}{\delta h^{(t)}} \frac{\delta h^{(t)}}{\delta h^{(i)}} \frac{\delta h^{(i)}}{\delta W_{hh}} = \sum_{i=0}^{t} \frac{\delta \mathcal{L}^{(t)}}{\delta p^{(t)}} \frac{\delta p^{(t)}}{\delta h^{(t)}} \left(\prod_{j=i+1}^{t} \frac{\delta h^{(j)}}{\delta h^{(j-1)}} \right) \frac{\delta h^{(i)}}{\delta W_{hh}}$$

(c) The gradient $\frac{\delta \mathcal{L}^{(t)}}{\delta W_{hh}}$ is temporally dependent on all previous timesteps that have occured up til t, since we want to compute the gradient w.r.t W_{hh} for all of the previous hidden states, whereas the gradient $\frac{\delta \mathcal{L}^{(t)}}{\delta W_{ph}}$ does not have such a temporal dependency; it is only dependent on the current hidden state.

The problems that might occur for $\frac{\delta \mathcal{L}^{(t)}}{\delta W_{hh}}$ is that for long sequences, the full gradient is comprised of a long series of gradient multiplications. We are using the hyperbolic tangent non-linearity as the activation function for each hidden state, and this function often has gradients which are < 1; there is thus a risk of vanishing gradients when performing BPTT on long sequences, which would cause the network to learn much slower, if at all.

1.2 Long Short-Term Memory (LSTM) network

Question 1.2

(a) The forget gate $\mathbf{f}^{(t)}$ takes as its input the input of the current timestep $\mathbf{x}^{(t)}$ and the previous hidden state $\mathbf{h}^{(t-1)}$, and uses this information to decide which features in the current cell state $\mathbf{c}^{(t)}$ should be 'forgotten'. The sigmoid non-linearity is appropriate here because it outputs a value between 0 and 1 per feature, thereby serving as a weighting mechanism when applied element-wise to the features in the previous cell state $\mathbf{c}^{(t-1)}$.

The input modulation gate $\mathbf{g}^{(t)}$ takes the same input as $\mathbf{f}^{(t)}$ (albeit with a different set of parameters; $\mathbf{W}_{fx}$ and $\mathbf{W}_{fh}$) and passes it through a hyperbolic tangent non-linearity, thereby creating a new vector of so-called 'candidate values'. These values can be interpreted as *potentially* interesting features that *could* be added to the current cell state. The tanh activation is favourable here as its activations lie in the range $[-1, 1]$, which can serve as a suitable 'priming mechanism' before the values are passed through the sigmoid layer in the input gate.

Preprint. Work in progress.

The input gate $\mathbf{i}^{(t)}$ takes the same input as $\mathbf{g}^{(t)}$ (again, with different parameters; $\mathbf{W}_{ix}$ and $\mathbf{W}_{ih}$) but instead passes it through a sigmoid non-linearity. This gate could be understood as a 'selection mechanism', insofar that it solely serves to assign 'importance' to the features of the current timestep, by squashing each feature to a range of $[0, 1]$, where values close to 1 and 0 indicate more salient, and less salient features, respectively. By then combining the candidate vector from $\mathbf{g}^{(t)}$ and the vector from $\mathbf{i}^{(t)}$ via element-wise multiplication, the candidate values which are deemed important will be multiplied with a value closer to 1 (thereby 'let through' mostly intact), whereas the values which are deemed less important are multiplied with a value closer to 0, and thus those features will contribute less to the current cell state.

The output gate $\mathbf{o}^{(t)}$ takes the same input as the other gates (*again*, with different parameters; $\mathbf{W}_{ox}$ and $\mathbf{W}_{oh}$), and its purpose is to determine what to output from the cell state, based on the information from the current timestep $\mathbf{x}^{(t)}$ and the past hidden states $\mathbf{h}^{(t-1)}$. Similarly to $\mathbf{i}^{(t)}$ and $\mathbf{f}^{(t)}$, the sigmoid is an appropriate non-linearity as its output range enables us to weight each feature of the cell state via element-wise multiplication.

(b) Since the LSTM employs shared parameters over time, the number of trainable parameters does not depend on the sequence length T. Instead, we have 9 weight matrices $\mathbf{W}$ and 5 biases $\mathbf{b}$. Total number of parameters:

$$4*(N_{input}*N_{hidden})+4*(N_{hidden}*N_{hidden})+(N_{hidden}*N_{output})+4*(N_{hidden})+1*(N_{output})$$

1.3 LSTMs in PyTorch

For accuracy and loss curves, see Figure 1. For accuracy results, see Table (cite). Observing the loss curves, we can note a number of things. All models converge in the 600-1000 batches range, reaching a perfect accuracy of 1.00. Furthermore, the LSTM seems to converge similarly across sequence lengths, with the exception of the LSTM trained on sequences of length 5; here, the model only appears to converge after 800 batches, whereas for the other experiments it is close to convergence after 600 batches.

These results slightly contradict the behaviour typically exhibited by LSTMs; given the known problem of vanishing gradients (as discussed in 1.1 c), we might expect to see the LSTM converge slower and/or perform poorer as we increase the sequence length - however, aside from very moderate differences in the initial loss values between experiments (which might serve as a crude proxy for the inherent difficulty of each problem), there does not seem to be a noticeable relationship between sequence length and convergence and/or loss behaviour over time. Simultaneously, this is perhaps to be expected due to the simplistic nature of the BSS problem, and the fact that only a small range of different sequence lengths was considered. In that respect, the problems of vanishing gradients and inability to capture long-distance-dependencies might become more apparent as we dramatically increase the sequence length.

1.4 Gated Recurrent Unit (GRU) in PyTorch

For accuracy and loss curves, see Figure **??**. For accuracy results, see Table (cite). Observing the loss curves, we can note several things. For all sequence lengths, the GRU converges after approximately 600 batches (with only minute differences between sequence lengths). The GRU achieves a perfect accuracy of 1.00, for all three sequence lengths. Between sequence lengths, there are slight differences among the loss values in the 0-200 batches range, with the lowest loss values occurring for the sequences of length 4, and the highest loss for sequences of length 6 (here, the losses are in the range of 0.4x and 0.6x, respectively). Although these initial loss values are perhaps a crude measurement, they might reflect how longer sequences translate to more difficult tasks, thereby explaining their higher loss values.

Still, these results align with the hypothesis that LSTM-based architectures (to which the GRU belongs) tend to perform poorer as sequence lengths increase, due to the vanishing gradient problem. Possibly, we should expect to see the observed positive relationship between sequence length and initial loss values, and the number of batches required until convergence, to sustain, were we to experiment with sequences of even greater lengths.

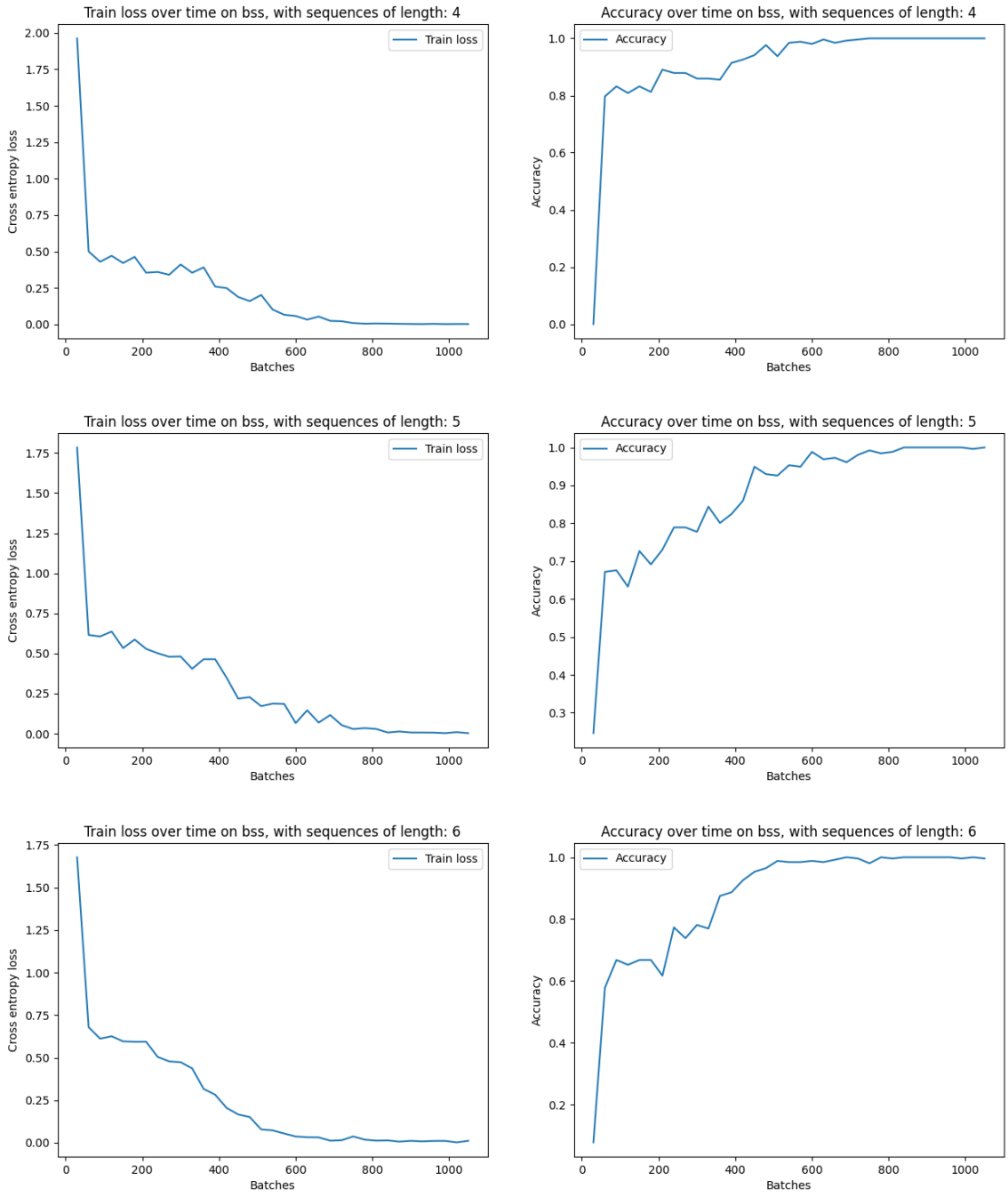

Figure 1: Loss and accuracy curves for the LSTM on the Baum Sweet Sequence toy problem for sequences of length 4, 5, 6.

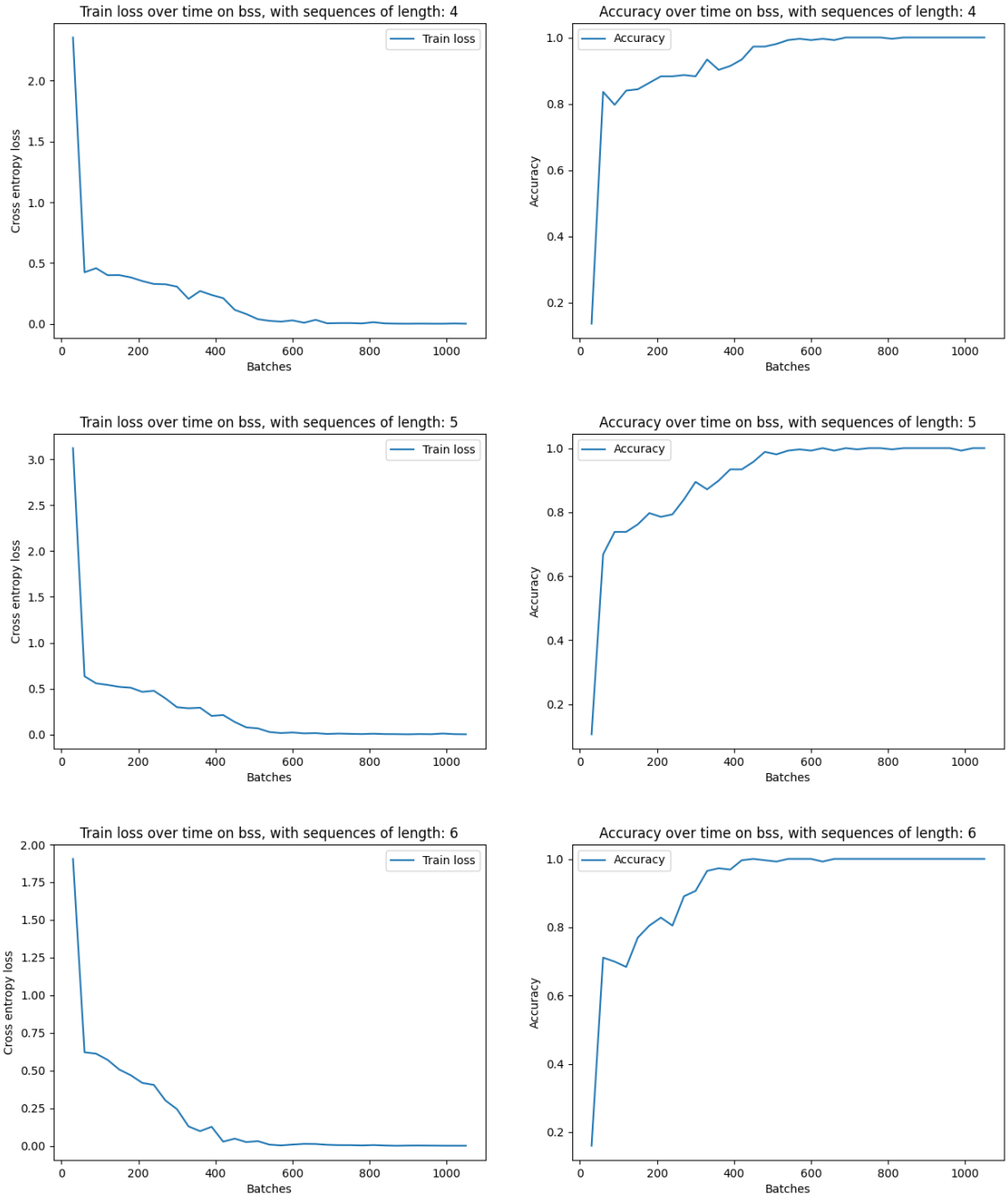

Figure 2: Loss and accuracy curves for the GRU on the Baum Sweet Sequence toy problem for sequences of length 4, 5, 6.

Comparing to the LSTM, the GRU appears to perform slightly better; particularly for sequences of lengths 4 and 5, the GRU achieves close to 1.00 accuracy after 600 batches, whereas at the same point in time, the LSTM's accuracy is in the 0.90 range. Still, these differences are subtle, and further experiments are required to declare a clear winner between the two models. However, it should also be noted that the GRU has less parameters compared to the LSTM, making the GRU a more efficient model.

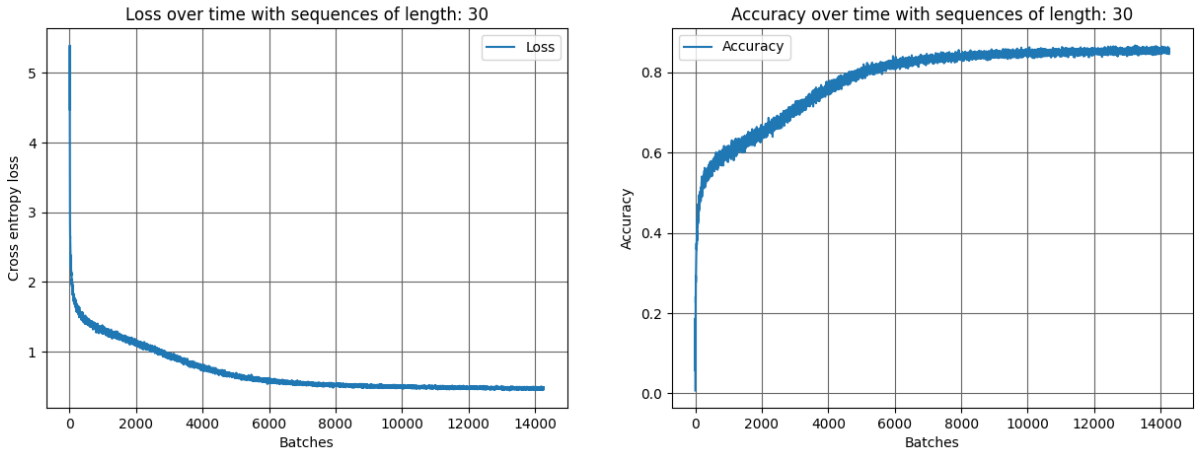

Figure 3: Loss and accuracy curves when predicting the next character using a two-layer LSTM trained on *On The Road*.

2 Recurrent Nets as Generative Model

2.1 a)

Given that the sole goal here is text generation, and not generalising on unseen data (by e.g. having the model predict the next character on held-out chapters of the same book, or even another book), we can choose to engineer the model such that it over-fits heavily, thereby forcing it to 'memorize' the corpus as good as possible. This is primarily achieved by using a large amount of hidden-units to greatly enhance the model's expressive power.

The 2-layer LSTM has a hidden state comprised of 4096 features; characters are embedded in a 2048-dimensional space (generally, embeddings of 1/2 to 1/4 the size of the hidden state tend to work well). We choose not to apply Dropout - this is generally done to regularize the model to benefit *generalisation*, which is not our objective. We optimize with Adam using a learning rate of 0.001, with a batch size of 128. These values appeared to yield the best improvements after a brief hyperparameter search. For loss and accuracy curves, see Figure 2. The model achieves its highest performance after approximately 10.000 batches, which translates to roughly 2.5 epochs, reaching an accuracy in the range of 0.85 (as can be observed, the model performance is somewhat variable throughout training).

2.1 b)

We can generate sentences by feeding a random character to the LSTM, and then using the subsequently predicted character, along with the changed hidden and cell states, to predict additional characters conditioned on previously seen characters. See Table 1 for generated sentences. Note: I choose to sample not at intervals of 1/3, since most of the learning takes place in the 0-7500 batch range, therefore we ought to see more interesting differences between sentences there.

Sentence analysis

After 1300 batches, the model achieves approximately 0.65 accuracy when predicting the next character. The generated sentences here are coherent, insofar that words are comprised of contiguous characters, and sentences are comprised of sequences of words with spaces. A noticeable problem is the model getting 'stuck' in a loop, indefinitely generating the same phrases. Moreover, while the sentences are somewhat coherent grammatically, they describe semantically abstract situations, which could possibly be ascribed to the LSTM being unable to model more distant dependencies. Interestingly, all the 'narration' sentences here lack punctuation, while 'quoted' sentences have commas and periods.

At an accuracy of 0.75, the model no longer 'loops'. Furthermore, the sentences are more punctuated, and exhibit fewer grammatical errors. Semantically, the sentences are still somewhat 'poetic', albeit slightly more sensible than previously.

After the model surpasses the 0.80 accuracy mark, the over-fitting becomes more obvious. Here, many of the sentences are concatenations of phrases lifted directly from the source text. For instance, the phrases *'Johnny fell asleep on my arm. We drove back to Sabinal. On the way we pulled up', 'across the state of Michigan'* and *'and Dean went mad again with sweats and insanity'* occur literally in the source text. Similarly, the second piece of text is built up from phrases *'Paris, all those places; we'll sit at sidewalk'*, *'sidewalks crowded with Hongkong-like humanity. "Yow!" yelled Dean.'*, *'yelled Dean. "Si! Mariana!" Music was coming from all sides, and all kinds of'*, and so-forth. We can interpret this as the model having effectively memorized the training set, generating sentences by stitching together existing phrases.

Table 1: Generated sentences for LSTM trained on *On The Road* by Jack Kerouac.

Batches	**Acc.**	**Generated sentences**
1300	0.65	*I woke up with the bar when he was a driver when he was a driver when he was a decision that he was a driver when he was a decision* *"No, you see, man, you got to see the time I saw a strange shirt and said to me. "Sal," said Dean, "I don't know what to see you again."*
3000	0.75	*Victor talked to the streets of Mexican and Negro homes; soft voices were the beginning of the road and went out the party. "Howd'y'do. My name was Eddie. He repeated it, following Dean.* *In the morning I wanted to get out of this manner, and the spectators had to wait for the first time, in some lost bliss that was over a thousand miles, and finally I went to the back and*
7500	0.84	*Johnny fell asleep on my arm. We drove back to Sabinal. On the way we pulled up and backed up across the state of Michigan and moaned in his face, and Dean went mad again with sweats and insanity* *Paris, all those places; we'll sit at sidewalks crowded with Hongkong-like humanity. "Yow!" yelled Dean. "Si! Mariana!" Music was coming from all sides, and all kinds of wonderful technicolor visions*

2.1 c)

The temperature parameter τ can be interpreted as a scaling operation applied element-wise prior to feeding output logits into the SoftMax module. Scaling with values of $\tau > 1$ will cause already large logits to explode, thereby causing those logits to dominate the resulting distribution over characters. Sampling from such 'sharpened' distributions results in selecting the most probable character more often, and for very high values of τ, we effectively end up taking the argmax. Conversely, for values of $\tau < 1$, we can introduce more 'doubt' by allocating more probability to logits with lower values, 'flattening' the distribution over characters. Similarly, for a very small value for τ, the prediction becomes a random guess, irrespective of the model's accuracy. Thus, we can control for the amount of randomness during sampling by adjusting the τ parameter.

For results, see Table 2. As could be reasonably expected, a temperature $\tau = 0.5$ introduces more randomness, with the model generating mostly non-sensical words. Interestingly, the model still 'understands' that sentences are made up of words with spaces, and that they start with a capitalized letter; in this sense, the results give the impression of some strange language. Conversely, a higher value of $\tau = 2.0$ leads to sentences with 'real' words. Naturally, sentences with $\tau = 1.0$ are somewhere in-between, consisting of both real and fake words. These trends persist even for higher accuracies, though there are differences also; at 0.84 accuracy, even for a $\tau = 1.0$, few of the generated sentences have 'fake' words. Possibly, given how much the model is over-fitting on the

source text, it has been able to overcome the 'doubt' of sampling from the distribution of words, thereby

Comparing to non-'sampled' sentences in 2.1 b), the model does not exhibit the repetitive behaviour we observed with greedy sampling. Possibly, by virtue of it sampling, as opposed to selecting the most probable character, it has a probability of 'escaping the loop', explaining the absence of repeated phrases. Otherwise, we can also note parallels between the results in 2.1 b). For instance, at 0.65 accuracy, the models still tend to generate sentences that go on endlessly.

Table 2: Generated sentences for two-layer LSTM trained on *On The Road* by Jack Kerouac, with varying temperature values.

Batch	**Acc.**	**Sentences with $\tau = 0.5$**	**Sentences with $\tau = 1.0$**	**Sentences with $\tau = 2.0$**
1300	0.66	*Welcome foy's galt foriTical. Amû'dallin cat?" The fog kepto word, "Tagm T! Vars' thoo tand, Vandlook'r"*	*Quarty. He made us in. If it was the Holten wound botherous and we didn't know it was making ever find him that time then Dean used to sleep on the car as I am* (...)	*Jane was a lovely piece in the night-street and the plain on February twenty-five days and said, "I have a bath!"*
2800	0.75	*Victor past, reappentiness, eve{yven T-zur, Porring M"da Ekyou,, RoR6No, I Ameche. You JosHu kil? < I wouldn't nel ibvul; her ligits> ohd, everymorow.*	*Bettys and Amy, and announcerged again and I go crazy, I can go to Canada." Dean and Dean just walked on to the stolen unlonely room.*	*Victor was proud of us. We went out. Lights were burning in the back seat and she was making love to two girls to watch the food of the newer, and gave it up front.*
7600	0.84	*Knsoy voices warred around over tangerine porestmax ocrwit Stan for Buick Keeing voice and was sp/itchily daWhed in girls in rhidrhildhia dreams of night.*	*Good's going! We both knew how good I feel coming out. Terry got her clothes and no catches. As I labored at this absurd task, great Kleig lights of a Hollywood premiere stabbed in the rain.*	*Wall's ranch in the middle of the road that unwound, and it was one of the best chapters in the bushes. "Someday we'll meet, and you'll dry all my terrified look as I don't know what to say to you."*

2.2

For auto-completed sentences, see Table 3. Again, at lower accuracies of 0.64, the model does a poor job at finishing the sentence, with finished sentences exhibiting grammatical errors and describing semantically abstract situations. Note again the 'loop' behaviour. For higher accuracies in the 0.75 range, the generated text becomes more plausible and the connection between input and generated text is more coherent (for instance, *'I first met Dean and'* and *'I had to make him'* are co-referential with *'him'* and *'Dean'* having the same referent). Similarly, for accuracies in the 0.84 ranges, the sentences continue to be coherent, though at times it appears as if the words *'I first met'* are disregarded quickly; *'I first met Dean called Ernest Burke'* is non-sensical, whereas *'Dean called Ernest Burke who lived with his father in a hotelroom on Third Street'* is a coherent sentence.

References

Table 3: Auto-completed sentences for two-layer LSTM trained on *On The Road* by Jack Kerouac. Note that all characters were selected via argmax, and not sampled from a distribution. Prompt: *'I first met Dean'*.

Batches	**Acc.**	**Generated sentences**
1100	0.64	*"I first met Dean was a strange red light and was a strange red light and was a strange red light"*
2800	0.75	*I first met Dean and I had to make him comfortably resolved into it. They had the beatest suitcases of his mind that we were a strange ways that we know what they were* *I first met Dean who was wakened for the farewell, gathered in the backyard and said to sleep in a moment.*
7600	0.84	*I first met Dean and I plotted to make Tim Gray come with us, but Tim was stuck to his Denver life. He had pinned up there three weeks ago.* *I first met Dean called Ernest Burke who lived with his father in a hotel room on Third Street. Originally he'd been on good terms with them,*