Recurrent Neural Networks for Text Generation

Alessio De Luca 919790
Simone Vaccari 915222

1. Objective:

The objective of this study is to explore the performance of different versions of Recurrent Neural Networks (RNNs) in a character-level language modeling task. Specifically, we aim to investigate the impact of varying the architecture of such networks on the model's performance in generating coherent text sequences.

2. Introduction:

Character-level language modeling is a fundamental task in natural language processing (NLP) where the objective is to predict the next character in a sequence given the previous characters. RNNs are well suited for this type of task as they can effectively capture sequential dependencies inherent in language data. However standard RNNs suffer from two major prbolems: vanishing gradient and lack of context. The first one makes it difficult for them to capture long-range dependencies in sequential data, while the second one doesn't allow the model to correctly generate text based on the previous sentences.

In order to overcome these problems, researchers introduced gated RNNs, a variant of the standard version, in particular: **Long Short-Term Memory (LSTM)** networks and **Gated Recurrent Units (GRU)**. Both models are designed for capturing long-term dependencies more effectively and addressing issues like the vanishing gradient problem.

Unlike traditional RNNs, LSTMs have memory cells with input, forget, and output gates. These gates, controlled by simple neural network layers, help LSTMs retain context over long sequences, making them ideal for tasks like speech recognition and machine translation.

On the other hand, GRUs provide a simpler alternative to LSTMs. They combine forget and input gates into one "update" gate, reducing complexity and the number of parameters. GRUs are able to effectively capture long-term dependencies while remaining efficient, making them suitable for real-time applications and mobile devices.

3. Procedure:

The dataset for the experiment is composed of sentences taken from the book **"Alice's Adventures in Wonderland"** by Lewis Carroll. After loading the whole raw text, the first step consists in manipulating it in order to make it suitable for training a character-based language model. To achieve this, firstly all characters are converted to lowercase to ensure uniformity. Secondly, all the non-alphanumeric characters are removed. Then, an encoding operation is performed by mapping each unique character to a unique integer: the resulting dictionary presents 27 number-character pairs (0 encodes the whitespace, numbers from 1 to 26 the sorted letters of the alphabet).

After these initial pre-processing steps, the text is divided into two subsets: training, representing 70% of the total text, for a total of 108,293 characters, and validation, containing 30,941 characters, i.e. 20% of the whole text. At this point, sequences of 100 consecutive characters are created by running

through the text with a sliding window approach and a stride of 1. At every step the 101th character is saved as the corresponding ground truth. Table 1 shows some examples of training samples with the corresponding ground truth label constructed from the following sentence:

"you know said alice to herself in my going out altogether like a candle i wonder what i should be like then and she tried to fancy what the flame of a candle is like after the candle is blown out"

Sample	GT
you know said alice to herself in my going out altogether like a candle i wonder what i should be li	k
ou know said alice to herself in my going out altogether like a candle i wonder what i should be lik	e
u know said alice to herself in my going out altogether like a candle i wonder what i should be like	' '

Table 1: Training samples of sequences of 100 characters with the corresponding ground truth

As mentioned in the introduction, three main versions of Recurrent Neural Networks were defined:

- standard *RNN*, with tanh non-linearity
- *LSTM*
- *GRU*

Figure 1 shows a schematic representation of the building block of each of these networks.

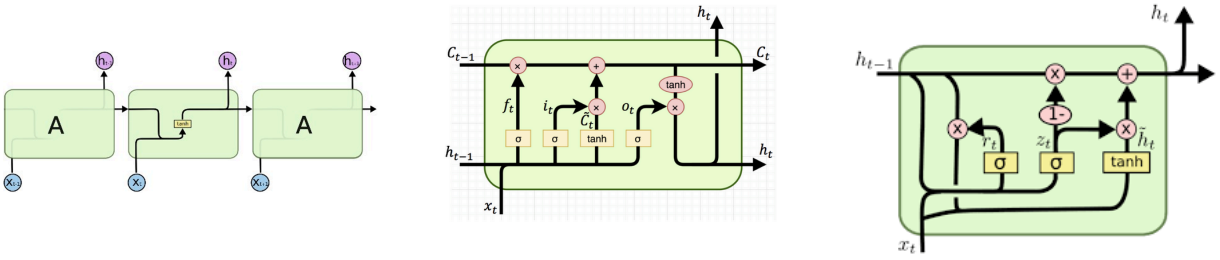

Figure 1: Building block of standard RNN (left), LSTM (center), and GRU (right)

In all three cases, the following parameters will be changed in order to evaluate their effect on the network's performances:

- number of recurrent layers;
- number of features in the hidden state;
- dropout probability;

The performance of the model during training is monitored by computing the cross-entropy loss and the accuracy on the validation set for each epoch. The total number of training epochs is kept set to 30, with a patience of 5, meaning that if after five epochs the loss doesn't decrease the training procedure is arrested.

After being trained, each model will be evaluated by making it generate a sequence of 100 characters starting from a prompt represented by a 100-character sentence taken from the book.

4. Results:

In order to evaluate the effect of the parameters and of the network's architecture on the model's performances we plotted for every configuration the trend of the validation loss and accuracy during the training epochs. Table 2 shows the best loss and the best accuracy achieved with a RNN, and the corresponding parameters. In the same way, Table 3 and Table 4 report the results for the LSTM and the GRU networks, respectively.

Num layers	Hidden size	Dropout	Best validation loss	Best validation accuracy (%)
2	256	0.2	2.31	38.4
2	512	0.2	2.58	33.8
3	256	0.2	2.24	39.4
3	256	None	2.21	40.4
4	256	0.2	2.88	18.9
4	256	0.4	2.87	18.9

Table 2: Training performances of different configurations of RNN

We can observe that in the case of the RNN increasing the number of stacked recurrent layers does not always yield a better performance: when the network becomes too complex and too deep it stops being able to learn.

Num layers	Hidden size	Dropout	Best validation loss	Best validation accuracy
2	256	0.2	2.29	39.0
2	128	0.2	2.27	39.5
3	256	0.2	2.19	42.3
5	256	0.2	2.11	42.7
5	128	0.5	2.86	18.9

Table 3: Training performances of different configurations of LSTM

Num layers	Hidden size	Dropout	Best validation loss	Best validation accuracy
2	256	0.2	2.30	39.0
3	256	0.2	2.21	42.0
3	256	0.1	2.20	41.9
4	256	0.2	2.19	41.7
4	128	0.2	2.18	40.6
5	256	0.2	2.18	40.8

Table 4: Training performances of different configurations of GRU

It can be noted that both LSTM and GRU achieve acceptable and very similar results with a number of recurrent layers ranging from 2 to 5. Furthermore, some of the experiments showed that the absence of dropout layers or performing dropout with a probability of 0.1 / 0.2 was preferred to dropout with a higher probability.

In general, the total number of training epochs ranged between 10 and 20; in no case all the 30 epochs were completed.

Figure 2 allows a visualization of the trend of the validation loss and the validation accuracy in the case of the model that yielded the highest accuracy value (LSTM with 5 recurrent layers).

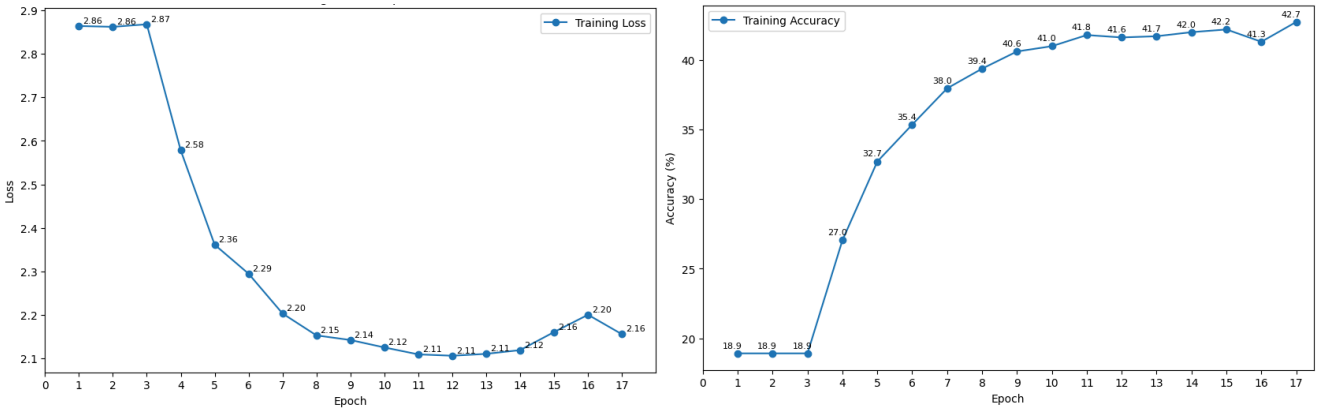

Figure 2: Illustration of validation loss and accuracy through the training epochs for the best model

Once the training was completed, each model was assessed in its ability of text generation, starting from a predefined set of eight 100-character sequences, used to test all the models to facilitate comparative analysis.

An example of generation is shown here:

Prompt: *"y be used on or associated in any way with an electronic work by people who agree to be bound by the"*
Generation: *"was a but it was the rabbit wiile the was a great her and the door and the door and the door and th"*
Real sequence: *"the terms of this agreement there are a few things that you can do with most project gutenberg electroni"*

It is clear that the generated text contains actual words, even though they don't correspond to the true ones of the book, and overall the output seems to not make much sense. Moreover, after a few words the generated characters start repeating. In most cases, this repetition starts after 8-10 words, with the exception of the very badly trained models, for which it can start after the generation of a few characters.

Below is an example of sequence generated by the best model:

Prompt: *"ing this but all he said was why is a raven like a writingdesk come we shall have some fun now thou"*
Generation: *"ght alice ald the mouse was so she was soon and the same the mock turtle seplied to she was soon and"*

Real sequence: *ght alice im glad theyve begun asking riddles i believe i can guess that she added aloud do you mean*

In this case the model was able to generate the first few characters correctly.

During our experiments we noticed that most models end up generating the same set of words (e.g. *the mock turtle*, *the dormouse*) even when the prompt sequences are very different. In some cases, when the prompt contained many whitespaces, some models produced a sequence made of very few initial characters followed by 80-90 whitespaces.

As a final experiment, we changed the length of the training sequences, initially from 100 to 20, and subsequently to 200. In the first case the results were much worse, suggesting that the network lacked enough information and context to yield an acceptable output, while in the second one the generated characters were much more random, leading to reduced word repetition but an increase in the occurrence of meaningless words.

5. Conclusions:

In this study, we explored the capabilities of Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Gated Recurrent Units (GRU) in generating text. These models were trained on 100-character sequences from "Alice's Adventures in Wonderland". The training aimed to predict subsequent characters, effectively learning the book's unique style and vocabulary. Our evaluation focused on both quantitative metrics, such as accuracy and loss on a validation set, and qualitative analysis of the text generated.

We observed that RNNs, while simpler in architecture, struggled with long-term dependencies, often repeating or generating nonsensical sequences. LSTMs showed improved performance, capturing longer phrases and demonstrating better overall coherence in text generation. Finally, GRUs provided a balance between complexity and performance, with faster training times than LSTMs without a significant drop in text quality.