Deep Learning 2020 **(Due: 2 dec 2020)**

Submission Assignment #4

Instructor: Jakub Tomczak *Name:* Mick IJzer*, Netid:* 2552611

The goal of this assignment was to implement a recurrent neural network (RNN) in PyTorch. Two options were given; sentiment classification, and autoregressive modeling. The autoregressive modeling was chosen, which entails training a model to predict the next token in a sequence given the preceding tokens.

1 Problem Statement

Recurrent Neural Networks (RNNs) are a type of neural networks with a temporal nature. RNNs are mainly used for data with a temporal structure, such as text and sound. The standard objectives of classification and regression can also be achieved by using RNNs, but RNNs can also be used for autogenerative modeling. Autogenerative modeling comes down to predicting the next token in a sequence given the preceding tokens. Tokens can for example be the characters in a word, or even words themselves. The nice thing about autogenerative modeling is that no target has to be provided, since the target is just the input shifted by one time step. Since the objective is to predict the next token in the sequence it makes sense that the objective function that will be used for evaluation during training is the cross entropy loss. However, for validation/test evaluation sampling will be used. A sample will be generated and its correctness will be checked manually.

Natural language has a complex structure and dependencies between the tokens. Therefore, several datasets will be used. Namely, the non-deterministic finite automaton (ndfa) dataset, the bracket-dataset, and the toy-dataset. The ndfa-dataset contains sequences like ”sabc!abc!s” and ”suvw!s”. The bracket-dataset has sequences with matching brackets (e.g. ”((()()))”). Lastly the toy-dataset has simple sentences (e.g. ”the woman walks”).

2 Methodology

All three datasets are sorted by length of the sequences, which breaks one of the fundamental assumption of gradient descent. So the first step is to shuffle the data. Next, the dataset is split into batches. To maximize memory utilization, a maximum number of tokens per batch is chosen instead of a maximum number of sequences. Since the created batches contain sequences of varying lengths, the sequences are padded. However, before padding a '.start' token is prepended and a '.end' token is appended to each sequence. This should help the model to identify the start and end of a sequence.

During the process of creating the batches several other aspects are prepared. For example, the length of each sequence in a batch is saved to apply packing. Packing is used to optimize memory utilization. Furthermore, the target values are also created by shifting each sequence a single timestep and adding a column of zeros at the end (to keep the dimensions of the input and targets equal).

The inputs and targets are now prepared. For each dataset the (hyper)parameters of the RNN are tuned manually. The chosen parameter values are described in section . While some parameter values will differ between the RNNs, there are also some aspects that will be equal. For instance, the cross entropy loss function and ADAM optimizer are used for all RNNs. Furthermore, the basic structure of the RNN is also the same; the input is given to an embedding layer, after which it is packed. Next, it is passed through a single or multiple LSTM layers. Lastly, the output of the LSTM layer is unpacked and passed to a fully connected layer to compute the final output of the model.

3 Experiments

As described in section 2, the (hyper)parameters will be tuned manually. For each (hyper)parameter several values were tested, and one value was chosen. The resulting parameter values are shown in table 1. The Toy-model has more hidden layers, and a larger hidden layer size, since the Toy-dataset is more complex than the other two. For each dataset the model was trained a number of times (i.e. repetitions) to combat random effects from for example weight initialization and batch order. After each epoch, the loss of each batch is computed and saved. The loss is saved to be able to inspect the loss-curves. Finally, as described in section 1 there is no validation or test set. To evaluate the performance of the models 10 samples will be generated per epoch. For the NDFA and Bracket model a check is performed whether the generated samples are correct according to the sequence requirements. For the Toy data a check is performed on the spelling of the words, and the grammatical correctness. Note that, due to the repetitions, there are $repetitions * 10$ samples to be evaluated per model. The performance of a model is seen as good if the majority of the generated samples is correct.

	NDFA	**Brackets**	**Toy**
Training Sequences	15.000	15.000	20.000
Max Tokens	500	400	400
Embedding Dimension	32	32	32
Hidden Layer Size	16	16	32
Number of Hidden Layers	1	1	2
Epochs	15	15	15
Repetitions	5	5	3
Learning Rate	0.0005	0.0005	0.00025

Table 1: Hyperparameter Settings.

4 Results and Discussion

The goal of this paper was to implement multiple RNNs for autoregressive modeling using PyTorch. This was done by training RNNs on multiple datasets; ndfa, brackets and toy. The models were trained with the (hyper)parameters as described in table 1. The resulting loss curves are shown in figures ??, ??, ??. Note that the thick blue line reflects the average loss over multiple repetitions and that the light blue area around the line indicates the standard deviation over multiple repetitions. Overall, the learning curves for all three datasets look good.

However, these are the results of just the training process. The actual performance of the models was evaluated by generating samples. Some examples of the generated samples for each model are shown in the appendix. For the 50 samples from the ndfa-models only 1 error was found. For the brackets-model 5 errors were found of which 1 sequence was not finished yet (i.e. reached the maximum number of tokens generated). Respectively this means that 98% and 90% of the generated sequences were correct. For the Toy model no spelling mistakes were found. However, a portion of the generated sequences (±20%) did not make grammatical sense. For example "the bunny went on a nice woman quickly". In general it can be concluded that the use of RNNs with LSTM layers provides remarkable results in autogenerative modeling tasks. Obviously, the chosen evaluation method is very informal, so any conclusions drawn based on this method should not be blindly taken as the truth. A more formal method of evaluating the performance would be to use *perplexity*, a measurement of how well a probability model predicts a sample.

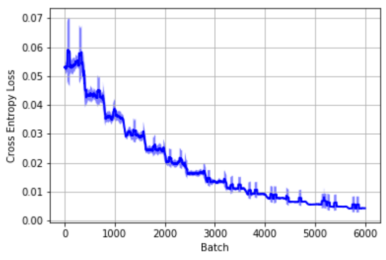

Figure 1: NDFA

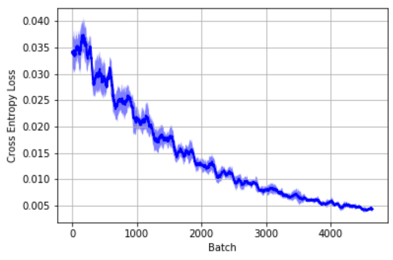

Figure 2: Brackets

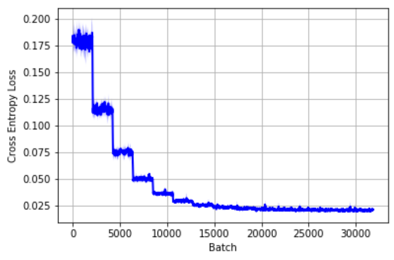

Figure 3: Toy

Appendix

.1 NDFA

Epoch 5:
.startsabc!s.end
.startsuvw!uvw!u.pad.startsabc!klm!klm!klm!s.end
.startsuvw!s.end
.startsuvw!uvw!s.end
.startsabc!abc!ac!ab.padc!abc!s.end
Epoch 15:
.startsabc!abc!abc!s.end
.startsabc!s.end
.startsabc!abc!abc!abc!abc!abc!abc!abc!abc!abc!abc!abc!abc!abc!s.end
.startsuvw!uvw!s.end
.startsklm!s.end

.2 Brackets

Epoch 5:
.start(()((())()(()))).end
.start(())).end
.start((())).pad.pad.end
.start(().unk).end
.start(()()(()())).end
Epoch 15:
.start().end
.start((())).end
.start(()).end
.start((()(()))((())()(())())()).end
.start(()()).end

.3 Toy

Epoch 1:
.startn a muen) qiithieycomay ((ut).end
.starta nu woman walklos.end
.startnt the muny ran walds whilke pegoke.pads di a bnnnwrt uen.end
.startgile a cas ban welkss (woahe cat mann welks w tat a mam ous whkile moia quttl mours goes (o.end
.starta coti a wopren man goes but a dor persou us bunn.pad the qucide person wels bui qthiefl does (walkley.end
Epoch 5:
.starta woman runs to a cat runs while a bunny ran (to a quick dog).end
.starta bunny walked to a gorgeous quick person.end
.startthe gorgeous person walks on a short man.end
.startthe gorgeous dog rus.end
.starta busy bunny walks while a dog goes (ran (on the short bunny)).end
Epoch 10:
.startthe mouse ran to a short bunny).end
.starta nice bunny runs but the man walked with the quick cat.end
.startthe gorgeous mouse ran (but a mouse runs to a person).end
.starta bunny walks (while a dog walks (with a woman)).end
.startthe cat runs quickly.end
Epoch 15:
.startthe woman went (while the cat ran impatiently).end
.starta woman runs impatiently.end
.starta mouse went (with the nice cat).end
.starta quick bunny runs.end
.startthe woman walked with a person.end