Deep Learning Lab: Report Assignment #3

Marco Latella
marco.latella@usi.ch

December 9, 2021

1.1 Text Data

1.1.1 From the text, I have gathered the following properties:

- Lines: 5033
- Lines length avg: 35.27
- Sentences: 1364
- Words: 38403
- Characters: 177517

1.1.2 For a preprocessing step on a text I would remove all URLs, all irrelevant characters like punctuation and Numbers, all stopwords, convert all characters in lowercase, use Stemming and Lemmatization.

1.2 Batch construction

1.2.1 The if branch checks whether the string is present in the vocabulary before referencing it, to return the related ID. In case the variable "extend_vocab" is True, and the string is not present in the vocabulary, the string will be added and will be returned the id, otherwise will be returned 1.

1.2.2 The Keys of the dictionary **id_to_string** are the IDs and the values are the strings. Instead, the Keys of dictionary **string_to_id** are the strings and the values are the IDs.

1.2.3 It returns the number of characters present in the dictionary.

1.2.4 It returns the number of batches contained in the batches list.

1.2.5 **input_data.data** is a Tensor containing the id of the characters present in the vocabulary. **new_full** returns a Tensor, which has a new size that has to be specified as an argument and is filled with a value also passed as an argument. The two lines mentioned allow to increase the size of the **data** Tensor to a length that can be divided by the value of **batch_size**. To be more precise, the first line returns a new Tensor, containing all zeros and called "padded", which is bigger than the starting one. The second line assigns the starting data Tensor in the first part of the padded Tensor.

1.2.6 The shape of the tensor is [bptt_len, bsz].

1.2.7 The shape of all the tensor except the last one is [bptt_len+1, bsz].
The last tensor has a shape of [x, bsz] where x is between 1 and bptt_len + 1.

1.3 Model and Training

1.3.1 The model has been implemented as required.

1.3.2 The function has been implemented in the Trainer class and its name is *predict()*. It uses a greedy decoding algorithm and takes the character with the highest probability.

1.3.3 An argument has been added to the *predict()* function to decide whether use a greedy or sampling behavior.

1.3.4 The *train()* function has been implemented as required. I monitor the measures every 150 steps. The measures reported are the perplexity and the text generation ability of the model.

1.3.5 I trained the model with the hyper-parameters specified. I used colab pro with a Tesla p100 GPU that was able to train the model in about 2 minutes. To reach a perplexity value below 1.03 it takes about 2175 steps.

Below you can see the evolution of perplexity and the text generation quality:

Step: 0 > Perplexity: 107.1562728881836
Dogs likes best to
————————————

Step: 150 > Perplexity: 6.59898042678833
Dogs likes best to the was a sood the Wolf and the was a sood the Wolf and the was a sood the Wolf and the was a sood t
————————————

Step: 300 > Perplexity: 4.488205909729004
Dogs likes best to the share of the shall was not be a little said, “What is not so the share of the share of the share
————————————

Step: 450 > Perplexity: 3.3146677017211914 Dogs likes best to the terms of the forest.

But the Bat and the Mouse was so took at the same to the trouble to the tr
————————————

Step: 600 > Perplexity: 2.76772141456604
Dogs likes best to the work of the work of the work of the work of the work of the work of the work of the work of the work of the
————————————

Step: 750 > Perplexity: 2.418003797531128
Dogs likes best to be safe for the same time they came to the sheep.

THE WOLF AND THE CROW

A WOLF accused out to
————————————

Skipping some steps

———————————— Step: 1950 > Perplexity: 1.0568287372589111
Dogs likes best to danger in it.

When she looked in, she saw that there was nothing, the Hound cried up and set him sh
————————————

Step: 2100 > Perplexity: 1.073156714439392

Dogs likes best to eat at home, and he went trotting along with the meadows in the water.

“Can you swim?” asked the Ma

———————

Step: 2175 > Perplexity: 1.0273483991622925
Dogs likes best to be an instant, with his heavy coat of mail, he mounted the
Donkey himself. In a little mischief to t

———————

Training the model, the perplexity decrease in a constant way until it reaches a value of 1.02 and without any increase. As you can see with a high value of perplexity the model is not able to create a meaningful sentence but merely repeats a few words. With a low value of perplexity, the model is able to create sentences with some sort of sense, but he still has difficulty creating sentences that follow logic. However, if you consider only the first predicted characters, sometimes they have sense and can be also fun.

1.3.6 As required I trained the model with two others values of Learning Rate and BPTT. I had in total 4 new combinations of hyper-parameters. I used LR: 0.002, 0.00075 and BPTT: 32, 128.
Each combination takes a different time and number of steps to reach a Perplexity value below 1.03:

- LR: 0.002, BPTT: 32 > 12876 steps
- LR: 0.002, BPTT: 128 > 3740 steps
- LR: 0.00075, BPTT: 32 > 5046 steps
- LR: 0.00075, BPTT: 128 > 1144 steps

Looking at the results I can say that thanks to the bigger dimension of the BPPT the model takes less time and fewer steps to complete the training phase reaching 1.03 of Perplexity. However, looking at the text generation quality I didn’t notice very different results between combinations. This can be associated to a small dataset and the complexity of the model not very high.

1.3.7

a)

Input: THE HARES AND THE FROGS

Output:

IN A FOREST, deep, shady, and still, there once lived a companion with him.

THE FOX AND THE GOAT
A FOX who had fallen into a deep well was casting about to find how he should get out again. At length a Goat came to the place, and seeing Reynard, asked whether the water was good, and if there was plenty

b)

Input: "THE RAT AND THE ALLIGATOR"

Output: A CAVALRY officer took the greatest of pains with his charger. As long as the war lasted, the Horse was fed with chaff and whatever he might find by the wayside. He was made a drudge too, and was often forced to carry loads much too heavy for his strength.

c)

Input: The rats are pretty

Output: The rats are pretty the following which he thought was inside her. When he had laid her open, he found that she was exactly like all other geese.

d)

Input: When I was young I

Output: When I was young I never let him.

“Well,” said the Crab, when he found that his old enemy could not move.

Talking about the results I can say that for most of the generations the first characters are meaningful, but trying to create more characters I get results that don’t follow a logic. If I pass a title, the model returns the first sentence correctly, and then it generates another wrong fable. Instead, providing a new title that follows the style, it generates a new fable merging together different pieces of different fables. To summarize the results I can say that generally, it is not able to always produce new novels, most of the time it wrongs and produce sentence without a meaning in the context and then continues with sentences already seen.

1.3.8 Trying the decoding with sampling using prompt a) and b) I obtain worster results respect not using sampling. The sentences generated don’t have a meaning, in most of the cases, they are only pieces of words and sentences merged.

1.3.9 For this experiment, I used a collection of speeches of the USA presidents. I wanted to try this kind of text to see if the model was able to generate a little piece of speech. To allow the training I had to cut the original file making it smaller. The original dimension did not allow the reaching of 1.03 of perplexity during the training phase in a reasonable time.
Down below you can see three outputs of three different sentences predicted:

Input: I am again called upon by the voice

Output: I am again called upon by the voice of my country to execute the functions of its Chief Magistrate. When this corruphican peace while

Input: I have to reduce

Output: I have to reduce within limits too narrow that this Government, and that both have required a liberal indulgence

Input: I love my country, it is the

Output: I love my country, it is the people only that early liberties in relation to this these obligations, and in a spirit of amity

The first input is a sentence present in the text. Looking at the prediction, the model was able to reproduce the original sentence and it has generated nothing different or new.
The second input is a sentence not present in the text. As you can see the model can predict only two words with meaning, but the rest it’s meaningless in the context.
The third input is a sentence not present in the text, but interesting. Also in this case the model was not able to generate new words that follow a logic in the context. To summarize I can say that the behavior is the same as using the fables file. The model can predict a few words but for the most, it can’t generate

sentences that follow the context. However, for all the experiments done without sampling decoding, the model was able to respect the syntax of the language.

2 Questions

2.1 V

2.2 The first example of sequence prediction could be Stocks prediction where we try to predict a sequence of prices of stocks. The second could be a sequence prediction of products bought by a user in a store.

2.3 The Vanishing gradient is a problem where small gradients updates can't propagate useful information through the network leading the network to not learn. The Exploding gradient is a problem where large gradient updates accumulate and result in bigger updates to the weights of the neural network in the training phase. This problem can lead the neural network to unstable learning. Because of these situations, the ability of a model to learn long-term dependencies decreases drastically due to the gradient is not able to update the weights of the previous steps leading the network to not consider important changes to previous dependencies.