**Assignment 3**

Hruday Vishal Kanna Anand

S.No- 1006874517

**Q1.2.1** Which class does LanguageModelingDataset inherit from?

**A-**

LanguageModelingDataset inherits from the Dataset class from torch.utils.data.

**Q1.2.2** What does the function lm collate fn do? Explain the structure of the data that results

when it is called.

**A-**

Function lm collate fn pads the batch of data with ones so that all the inputs and targets of that batch have the same size.

Structure- all the inputs are padded with ones at the end so that they all have the same length of the longest input. And the similar process is carried out for the targets

Structure example-

x- [{45,67,89,90}, {78,90,12}] y- [{67,89,90}, {90,12}]

padded-

x\_padded-[{45,67,89,90}, {78,90,12,1}] y\_padded- [{67,89,90}, {90,12,1}]

**Q1.2.3** Looking at the notebook block [6], (with comment “Print out an example of the data”) what does this tell you about the relationship between the input (X) and output (Y) that is sent the model for training?

**A-**

X and Y are of the same length and Y contains the word that will come after the series of words on X. This will train the model to predict the next word given an input series.

**Q1.2.4** Given one such X,Y pair, how many different training examples does it produce?

**A-**

One pair of X,Y gives us N training examples where N is the length of X or Y(they both are of same length.)

**Q1.2.5** In the generate function in model.py what is the default method for how the generated word is chosen?

**A-**

The generate function by default uses the top k method to generate a word with a temperature of 1 and a k value of 1

**Q1.2.6** What are the two kinds of heads that model.py can put on to the transformer model?

Show (reproduce) all the lines of code that implement this functionality and indicate which

method(s) they come from.

**A-**



These are the lines of code that are used to implement the heads of the transformer model. They both come from torch.nn.linear method.

The first line is used to implement the language model which will be used to train the model.

The second line is for the classifier head which is used for classification after the model is trained.

**Q1.2.7** How are the word embeddings initialized prior to training?

**A-**

This is how the word embeddings are initialized prior to training in the init method of class GPT, within self.transformer-

wte = nn.Embedding(config.vocab\_size, config.n\_embd)

where vocab size is size of vocab and n\_embd is the size of the embedding vectors.

**Q1.2.8** What is the name of the object that contains the positional embeddings?

**A-**

The name of the object is pos\_emb-

pos\_emb = self.transformer.wpe(pos) # position embeddings of shape (1, t, n\_embd)

**Q1.2.9** How are the positional embeddings initialized prior to training?

**A-**

This is how the word embeddings are initialized prior to training in the init method of class GPT, within self.transformer-

wpe = nn.Embedding(config.block\_size, config.n\_embd)

where vocab size is size of vocab and n\_embd is the size of the embedding vectors.

**Q1.2.10** Which module and method implement the skip connections in the transformer block? Give the line(s) of code that implement this code.

**A-**

The skip connection is implemented in class block under the forward method. The code to implement is given below.

def forward(self, x):

        x = x + self.attn(self.ln\_1(x))

        x = x + self.mlpf(self.ln\_2(x))

        return x

**Q2.1.** Run the code up to the line trainer.run() and make sure it functions. Report the value of the loss.

**A-**

Code run-

Text

Description automatically generated

Value of loss- 0.68115 after 2900 iterations

**Q2.2** Run the two code snippets following the training that calls the generate function. What is the output for each? Why does the latter parts of the generation not make sense?

**A-**

The outputs of the 2 code snippets are given below-

Text

Description automatically generated

Output- He and I can hold a dog. cat. cat and dog

Graphical user interface, application, Word

Description automatically generated

Output- She rubs a dog and cat. cat. cat. cat

The later parts of the generation don’t make sense as the training data contains sentences that are very small (around 5 words). Since we have forced the generator to come up with a 10 word sequence the words that come later don’t have much meaning to them.

**Q2.3** Modify the generate function so that it outputs the probability of each generated word.

Show the output along with these probabilities for the two examples, and then one of your

own choosing.

**A-**

Prompt- He and I

Output-

Text

Description automatically generated

Prompt- She rubs

Output-

Text

Description automatically generated

Prompt- They are

Output-

Text

Description automatically generated

**Q2.4** Modify the generate function, again, so that it outputs, along with each word, the words

that were the 6-most probable (the 6 highest probabilities) at each word output. Show the

result in a table that gives all six words, along with their probabilities, in each column of the

table. The number of columns in the table is the total number of generated words. For the

first two words generated, explain if the probabilities in the table make sense, give the input

corpus.

**A-**

Input prompt-He and I

Output table-

Text, letter

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Generated word | Top 6 words | probabilities |
| can | can hold rub holds and the | [0.516, 0.3306, 0.1488, 0.0035, 0.0004, 0.0002] |
| hold | hold rub can the a holds | [0.7092, 0.2888, 0.0015, 0.0001, 0.0001, 0.0001] |
| a | a the and hold dog rub | [0.5184, 0.4793, 0.0018, 0.0002, 0.0001, 0.0001] |
| dog | dog cat a . the. | [0.5985, 0.4012, 0.0001, 0.0001, 0.0, 0.0] |
| . | . and . cat rub a | [0.9979, 0.001, 0.0009, 0.0, 0.0, 0.0] |
| cat | cat dog a the and. | [0.6004, 0.3947, 0.0018, 0.0014, 0.0013, 0.0003] |
| . | . and . rub cat can | [0.9901, 0.0062, 0.0033, 0.0002, 0.0001, 0.0001] |
| dog | dog cat a the and rub | [0.5295, 0.47, 0.0001, 0.0001, 0.0001, 0.0001] |
| and | and. a the can rub | [0.5187, 0.4788, 0.0008, 0.0007, 0.0004, 0.0003] |
| dog | dog cat and can holds a | [0.8043, 0.1935, 0.0006, 0.0005, 0.0004, 0.0002] |

We can see the table has 10 rows for the number of words created and each row contains the top 6 words for the word chosen and the respective probabilities.

For the first word can- we can see the top 2 words are can, hold which both of fairly high probability as they both can come there are both those options make sense. Rub also has a 14% probability as it also makes sense. The last 3 words – holds, and, the have very low probabilities as they don’t make sense in this place.

For the second word hold- we can see the top 2 words are hold, rub which both have high probability as they both can make sense after ‘He and I can’. Now we can see the last 4 words all have very low probabilities as they will not make sense after ‘He and I can’.

**Q3.1** Report which of these two methods you used - trained yourself, or loaded the saved model.

**A-**

The model was trained with the hyperparameters mentioned in the question. It was trained by myself.

**Q3.2** Check that this model can generate words by seeding the generate function with a few examples different from the ones given. Report the examples you used and the generation results, and comment on the quality of the sentences.

**A-**

Prompt- He and I

Text

Description automatically generated

Prompt- She rubs

Graphical user interface, application, Word

Description automatically generated with medium confidence

Prompt- They are

Graphical user interface, text

Description automatically generated

Prompt- Money is

Text

Description automatically generated

All the prompts have something to do with money/ currency as this was the dataset this model was trained with. But the answers don’t make sense logically or grammatically as the training data is not extensive enough to capture the complete understanding of general prompts.

**Q3.3** Report the training and validation curves for the fine-tuning, and the accuracy achieved on the validation dataset.

**A-**

Training and final validation accuracy-

Text

Description automatically generated

Validation accuracy of- 66.66%

Training curves-

Loss-

Chart, line chart

Description automatically generated

Accuracy-

Chart, line chart

Description automatically generated

**Q4.2** Report the classification accuracy on the validation set. Comment on the performance of this model: is it better than the model you fine-tuned in the previous section?

**A-**

Model used- sshleifer/tiny-gpt2

Batch size- 16

Epochs- 50

Final training loss- 0.6914107259114584

Final validation accuracy- 0.541667

This huggingface model performed worse than the fine-tuned model in the previous section. The fine-tuned model was able to overfit to the training data where as this simple model from hugging face was not able to. This was an indicator the huggingface model does not have enough parameters. If we use a bigger model such as gpt2- medium or gpt2 we would get much better results but due to GPU limitations I ran it on the simplest model.

Hence it ended up with an accuracy of 0.54 whereas the fine-tuned model has an accuracy of 0.66