Ling 282/482 hw5

Due 11pm on October 23, 2024

In this assignment, you will

- Develop understanding of recurrent neural networks, especially as used for language modeling
- Implement components of data processing
- Implement masking of losses for an RNN language model

Submission Instructions

This assignment contains both written and programming portions. The answers to written questions must be submitted in a *.txt or *.pdf file to Blackboard. You will receive an invitation link to complete the programming portion via Github Classroom. This will open a Github repository with starter code and missing portions for you to complete. When you are finished with implementation, simply commit and **push** your changes to the online repository that was created for you. Unless you request otherwise, I will grade your work **as of the most recent commit** in your repository, subject to any applicable late penalties.

1 Recurrent Neural Network Decoders/Taggers [24 pts]

Q1: Understanding Masking [10 pts] Suppose that we want to train a (word-level) language model on the following two sentences:

<s> the cat sits </s>
<s> the model reads the sentence </s>

We saw in hw4 that padding is necessary to make these sentences have the same length so that they can be batched together, as:

<s> the cat sits </s> PAD PAD
<s> the model reads the sentence </s>

Please answer the following questions about these sequences:

a. In a recurrent language model, what would the input batch be? What would the target labels be? Hint: think about what the *input* and *output* tokens are at each time-step. [2 pts]

b. Recurrent language models use a *mask* of ones and zeros to 'eliminate' the loss for PAD tokens. What would the mask be for this batch? Your answer should be a matrix. [2 pts]

c. Suppose that we have the following per-token losses:

$$\begin{bmatrix} 0.1 & 0.3 & 0.2 & 0.4 & 0.7 & 0.5 \\ 0.2 & 0.6 & 0.1 & 0.8 & 0.9 & 0.4 \end{bmatrix}$$

What is the corresponding *masked* loss matrix? [2 pts]

d. Why is it important to mask losses in this way? What might a model learn to do if the loss is not masked? Answer in a few sentences. [4 pts]

Q2: Evaluating Language Models [14 pts] Given a corpus $W = w_1 w_2 \dots w_N$ (with N as the number of tokens in the corpus), a common (intrinsic) evaluation metric for language models is *perplexity*, defined as

$$PP(W) = P(w_1 \dots w_N)^{-\frac{1}{N}}$$

This can be thought of as the inverse probability that the model assigns to the corpus, normalized by the size of the corpus.

a. Is a lower or higher perplexity better? Why? [2 pt]

b. For a recurrent language model, write an expression for $P(w_1 \dots w_N)$ using the chain rule of probability. How is this different from the expression for a feed-forward language model? [2 pts]

c. Show that

$$PP(W) = e^{-\frac{1}{N}\sum_{i=1}^{N} \log P(w_i|w_{<i})}$$

where $w_{<i} = w_1 w_2 \dots w_{i-1}$ and log is the natural (base e) logarithm. [4 pts]

d. What is another name for the exponent $-\frac{1}{N}\sum_{i=1}^{N} \log P(w_i \mid w_{<i})$ in the above expression? Hint: it appears in training as well. [2 pts]

e. Suppose that the same text corpus were tokenized with two different vocabularies of different sizes (perhaps, e.g., one replaces infrequent tokens with an UNK token) and two language models were trained on the resulting tokenized text. All else being equal, would you expect perplexity to be lower or higher for the model with a smaller vocabulary? What consequences does this have for comparing different language models? [4 pts]

2 Implementing an RNN Character Language Model [10 pts]

In the coding portion of this assignment, you will implement (components of) an LSTM character-level language model for the Stanford Sentiment Treebank.

Q1: Data processing [4 pts] Recall from the lectures that we can view language modeling as a sequence tagging task. That is, each element of an input sequence is tagged with a certain target. This gets operationalized in the data processing pipeline; you will generate the inputs/targets for one line of text. In `data.py`, please implement the `example_from_characters` method. Please read the method signature and docstring carefully for details on the input and output.

Q2: Masking the Loss [6 pts] In the written portion above, you explained why masking the loss is important for a recurrent language model. Now, you will implement said masking. In `run.py`:

- Implement `get_mask`, which generates the mask to apply to the losses. [2 pts]
- Implement `mask_loss`, which takes per-token losses, masks out the 'bad' ones, and then returns a mean. See the doc-string for details. [4 pts]

3 Running the Language Model [12 pts]

`run.py` contains a basic training loop for SST language modeling. It will record the training and dev loss (and perplexity) at each epoch, and save the best model according to dev loss. Periodically (as specified by a command-line flag), it also outputs generated text from the best model. **Note:** the reference implementation takes about 1 minute per epoch with default hyper-parameters. **Make sure you plan enough time to complete all three training runs in this part of the homework** (each will be at least 20 epochs).

Q1: Default parameters [4 pts] Execute `run.py` with its default arguments. Paste below the texts that are generated every 4 epochs, as well as the epoch with the best dev loss and the dev perplexity from that epoch. In 2-3 sentences, describe any trends that you see. (Note that generated text will not necessarily be completely coherent: recall that this is a *character-level* language model).

Q2: Modify hyper-parameters [4 pts] Re-run the training loop, modifying some combination of the following hyper-parameters, which are specified by command-line flags:

- Hidden layer size
- Embedding size
- Batch size
- Learning rate
- Number of epochs (in particular: making it larger)
- Softmax temperature
- L_2 regularization coefficient
- Dropout (probability with which neurons are dropped during training)

Include your model's generated texts here. In 2-3 sentences, state exactly what hyper-parameter change(s) you made, and what effects (if any) you see in terms of the dev set perplexity and text that the model generated.

Q3: Modify hyper-parameters again [8 pts] Based on your results from Q1-Q2, pick another *different* set of hyper-parameters to try. Before you re-run training, write down your prediction/hypothesis for what will happen. Re-run the training loop one more time, with the new hyper-parameters. In 4-6 sentences, state what hyper-paremeter change(s) you made. Was the prediction you made supported? Report any trends and why you think your prediction was or was not born out.

4 Testing your code

In the starter code for this assignment, you will find a file `test_hw5.py` with a few very simple unit tests for the methods that you need to implement. You can verify that your code passes the tests by running `pytest` from your code's directory, with the course's conda environment activated. Remember that passing these unit tests is *necessary* but *not sufficient* for full credit on the coding portion.