LING 575K HW7

Due 11PM on May 20, 2021

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

All files referenced herein may be found in `/dropbox/20-21/575k/hw7/` on patas.

1 Recurrent Neural Network Decoders/Taggers

Q1: Understanding Masking 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 HW6 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:

- In a recurrent language model, what would the input batch be? What would the target labels be? [Hint: <s> is never predicted as a target, and </s> is only ever a target.]
- 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?
- 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 *masked* loss matrix?
- Why is it important to mask losses in this way? What might a model learn to do if the loss is not masked?

Q2: Evaluating Language Models Given a corpus $W = w_1 w_2 \dots w_N$ (so N is 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.

- Is a lower or higher perplexity better?
- 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?
- 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.
[Note: using base e measures perplexity in a unit known as *nats*. Using base 2 would measure it in bits.]
- 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.]
- 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?

2 Implementing an RNN Character Language Model

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

Q1: Data processing 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 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.
- Implement `mask_loss`, which takes per-token losses, masks out the 'bad' ones, and then returns a mean. See the doc-string for details.

3 Running the Language Model

`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.

Q1: Default parameters 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-parameter(s) 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
- Learning rate
- Number of epochs [in particular: making it larger]
- Softmax temperature.
- L_2 regularization coefficient.
- Dropout (probability with which neurons are dropped from the input and to the output 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: Comparison to feed-forward language model In 2-3 sentences, please explain what differences you see in the text generated by this LSTM language model and the feed-forward language model that you trained in HW5. What do you think may be causing these effects (or lack thereof)?

4 Testing your code

In the dropbox folder for this assignment, we will include a file `test_all.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.

Submission Instructions

In your submission, include the following:

- readme.(txt|pdf) that includes your answers to §1 and §3.
- `hw7.tar.gz` containing:
  - run_hw7.sh. This should contain the code for activating the conda environment and your run commands for §3 above. You can use run_hw2.sh from the previous assignment as a template.
  - data.py
  - run.py