Ling 282/482 hw6

Due 11pm on November 6, 2024

In this assignment, you will

- Better understand the attention calculation
- Explore subword tokenization with SentencePiece
- Develop an intuition of the flow of recurrent Seq2Seq models
- Implement key pieces of the model architecture

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.

Important Programming Information

This assignment will be more computationally intensive than previous ones. You will need to change your Github Codespace to accommodate this. **After you've created your initial codespace**, follow these steps:

1. Go to `github.com/codespaces`
2. Identify the codespace being used for this homework
3. Click on the three dots symbol at the right edge of the codespace bar
4. Click "Change machine type"
5. Select the option for 4-core 16GB RAM
6. Wait until the status bar for the codespace no longer says "Changing machine type", and displays the new machine type with increased RAM.

1 Understanding Attention [19 pts]

Q1: Padding and Softmax [3 pts]

- What is $e^{-\infty}$? (It's more accurate to think of this as $\lim_{x \to -\infty} e^x$, but no need to calculate the limit).
- Give the equation for softmax over an input vector x for the value x_i
- What would happen to the softmax output if every x_i in x were set to $-\infty$?

Q2: Attention Calculation [8 pts] Complete the computation of attention given the following values for Queries and Keys. Interpret the Queries and Keys as having a hidden dimension of 2, with the other dimension being sequence length. Following the case of RNN Seq2Seq, Keys and Values are the same. You can use a library like Numpy or Pytorch to do the computations, but make sure to do them step by step. When filling in the matrices, round to two decimal places, but do not round in your actual calculations.

$$\underset{\text{Queries}}{\begin{bmatrix} 0.2 & 0.4 \\ 0.8 & 0.6 \\ 1.0 & 1.2 \end{bmatrix}} \times \underset{\text{Keys}^T}{\begin{bmatrix} 0.1 & 0.7 & 0.9 & 1.5 \\ 0.3 & 0.5 & 1.1 & 1.3 \end{bmatrix}} = \underset{\text{Compatibility}}{\begin{bmatrix} - & - & - & - \\ - & - & - & - \\ - & - & - & - \end{bmatrix}}$$

$$+ \underset{\text{Mask}}{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\infty \\ 0 & 0 & -\infty & -\infty \end{bmatrix}} = \underset{\text{Masked Compatibility}}{\begin{bmatrix} - & - & - & - \\ - & - & - & - \\ - & - & - & - \end{bmatrix}} \xrightarrow{\text{softmax}} \underset{\text{Attention}}{\begin{bmatrix} - & - & - & - \\ - & - & - & - \\ - & - & - & - \end{bmatrix}}$$

$$\times \underset{\text{Values}}{\begin{bmatrix} 0.1 & 0.3 \\ 0.7 & 0.5 \\ 0.9 & 1.1 \\ 1.5 & 1.3 \end{bmatrix}} = \underset{\text{Output}}{\begin{bmatrix} - & - \\ - & - \\ - & - \end{bmatrix}}$$

Q3: Attention Questions [8 pts]

- Give the equation for attention (i.e., the Output of Q2) in matrix notation, using Q, K, and V to represent the Queries, Keys, and Values matrices (no need to show actual numbers). [2 pts]
- In the Compatibility matrix, what does each row represent? [1 pt]
- Why do Keys and Values have to be the same length? [2 pts]
- In a recurrent encoder-decoder model, where do the Queries, Keys, and Values come from respectively? [3 pts]

2 Exploring Subword Tokenization [10 pts]

In the starter repository for this assignment, you will find the folder `data/europarl-v7-en-es` which contains traslation training data. In that folder is also the file `europarl.vocab`, which contains the trained vocabulary for a SentenePiece tokenization model. Use that file to answer the following questions.

Q1: Inspecting a SentencePiece Model [6 pts]

- What is the general purpose of the SentencePiece library? [1 pt]

- Which two main algorithms does it implement? [2 pts]
- Based on `europarl.vocab`, which of the two algorithms was trained here? How can you tell? [1 pt]
- The vocab file consists of a list of strings paired with numbers. What do each represent? [2 pts]

Q2: Tokenizing with SentencePiece [4 pts] In the command line, start an interactive Python session by typing `python` (make sure you're in the main directory of the starter repository first). Next, execute the three lines of code shown below to import the SentencePiece package and load the trained SentencePiece model. You can now tokenize arbitrary sentences with the command `sp_model.EncodeAsPieces("My example sentence.")`. Use this to tokenize at least three sentences or phrases of your choice and post the results here. Describe in your own words what sort of tokenization scheme has been learned by this model and why (3-4 sentences).

```
import sentencepiece as sp
sp_model = sp.SentencePieceProcessor
sp_model.load("data/europarl-v7-en-en/europarl.model")
```

3 Implementing a Recurrent Seq2Seq Model [32 pts]

In the remainder, you will implement key components of a character-level Seq2Seq translation model, based on LSTMs. The dataset we are using is called Europarl, and consists of proceedings of the EU parliament. The data files can be found at `data/europarl-v7-en-es/`. We are translating English to Spanish. All of the methods that you will be implementing are in `model.py`. **Read all docstrings and comments closely for desired behavior**.

Q1: Seq2SeqModel.forward [8 pts] Implement the `forward` method for the `Seq2SeqModel` class. This method should take in batches of input and target sequences, embed them, encode the input sequence and then use its final states and attention to decode the target sequence. **This method will call Seq2SeqModel.encode and Seq2SeqModel.decode.**

Q2: Seq2SeqModel.encode [8 pts] Implement the `encode` method for the module. This method should take in the batch of input sequences and their lengths, and return the position-wise hidden states as well as the final hidden and cell states.

Q3: Seq2SeqModel.decode [8 pts] Implement the `decode` method for the module. This method should take in a batch of target sequences, plus the hidden and final states from the encoder, and use the decoder LSTM + attention to convert to logits over the vocabulary for the target sequence. **This method will call Seq2SeqModel.attention.**

Q4: Seq2SeqModel.attention [8 pts] Implement the `attention` module. This method should take series of vectors acting as Queries, as well as those acting as Keys/Values, plus a padding mask that we will provide for you. **The padding mask is created for you in forward, and must be passed to this method**. [10 pts]

4 Running the Translation Model [8 pts]

`run.py` contains a training loop to use the Seq2Seq translation model, which will record the training loss and generate text every N epochs (controlled by `--generate_every`, set to 1 by default). **Warning: the reference implementation takes ∼2 minutes per epoch to train. It is important that you plan ahead, and only use the default parameters of the script (which are small), until you are sure your code runs end to end without errors and it is time to run your final script.** You may get a message at the end of the training script that says `Segmentation fault (core dumped)`. This is a harmless bug, as long as it only happens after the printout saying `Done`, indicating the script has ended normally.

Q1: Run with Full Parameters [4 pts] Execute `run.py` with the following (full) arguments. Paste below the texts that are generated every epoch, as well as the training and validation loss. In 2-3 sentences, describe any trends that you see.

- `--train_source data/europarl-v7-es-en/train.en.txt`
- `--train_target data/europarl-v7-es-en/train.es.txt`
- `--output_file test.en.txt.es`
- `--batch_size 16`
- `--num_epochs 16`
- `--embedding_dim 16`
- `--hidden_dim 64`

Q2: Evaluate Translations [4 pts] Finally, we will evaluate the translations using Character F-Score (`https://www.aclweb.org/anthology/W15-3049/`). Using the provided script `chrF++.py`, calculate the score of the output Spanish translations using the following command, and report the score for the line that says `c6+w0-F2`.

```
python chrF++.py -nw 0 \
    -R data/europarl-v7-es-en/test.es.txt \
    -H test.en.txt.es > test.en.txt.es.score
```