Homework 4: Sequence to Sequence Modeling (100 points)

Kathleen McKeown, Fall 2019
COMS W4705: Natural Language Processing

Due 12/4/19 at 11:59pm

The aim of this assignment is to provide you with an understanding of how encoder-decoder sequence to sequence models work, give you practical experience training an encoder-decoder model, and show you problems and open areas of research in this domain.

In the first part of the assignment, you will write code and train a encoder-decoder model on a restaurant description dataset. This will be done in a jupyter notebook, and the training *must* be done on a GPU. In the second part, you will do an error analysis and answer questions about open problems and how the model could be improved.

General instructions

Please post all clarification questions about this homework on the class Piazza under the "hw4" folder. You may post your question privately to the instructors if you wish. If your question includes a partial solution, please post it privately to the **instructors only.**

This is an individual assignment. Although you may discuss the questions with other students, you should not discuss the answers with other students. All code and written answers must be entirely your own.

The submitted written assignment must be typeset, with the except of the model diagram.

Late policy: You may use late days on this assignment. However, you MUST include at the top of your submission how many late days you are using. If you don't tell us or have used all your late days, 10% per late day will be deducted from the homework grade. You may choose to save your late days for harder assignments later in the class.

If you have not already confirmed you can run code on the GPU, budget extra time for this. Getting set up on VMs, especially with GPUs, is notoriously finicky. Give yourself extra time to work out any problems you may encounter. Additionally, be sure to **shut down your VM instance when not in use**, so you don't run out of credits.

1 Coding Portion

Please download the code from the course website. This code includes one jupyter notebook, and a folder of data. The instructions for the coding portion are included in more detail in the jupyter notebook itself.

You will need to do some set-up to be able to use jupyter notebooks on the Google Cloud VM. This tutorial gives a good overview of how to do so. If you have set up a VM with GPUs for a past homework assignment, you should be able to use this machine. If not, you will first need to set up a new machine with GPUs. (Instructions for this can be found here.)

Google Cloud VM Settings

Once you have your VM, you will need to **change two settings**. The **first** is giving your machine a static external IP address. On the Google Cloud Platform, click on the menu icon in the top left, then 'VPC network', then 'External IP addresses'. For your machine, change the type from ephemeral to static. The **second** is to change the firewall settings. Similarly under 'VPC network' you can find 'Firewall rules'. Add a rule that allows the following specified protocols and ports: `tcp:5000`. This will allow you access a jupyter notebook on port 5000.

Set Up Jupyter

Install jupyter and check if you have a config file:

```
ls ~/.jupyter/jupyter_notebook_config.py
```

If not create a config file:

```
jupyter notebook --generate-config
```

Then, add the following lines to this config file:

```
c = get_config()
c.NotebookApp.ip = '*'
c.NotebookApp.open_browser = False
c.NotebookApp.port = 5000
```

Now you should be able to launch jupyter notebooks from the command line of your VM with this command:

```
jupyter-notebook --no-browser --port=5000
```

To access the notebook, go to the following URL:

```
http://<External Static IP Address>:5000
```

Finally–**don't forget to shut down your VM when not in use.**

1.1 Modify the code to run with the E2E dataset

As is, the code imports a number reversal dataset, which is a simple dataset of number sequences and those numbers in reverse. This is a dummy dataset designed to show that the model and training procedure work correctly. You should be able to run the entire notebook as is, and a model will learn to reverse short sequences of numbers.

Information about the E2E dataset is provided in the notebook. It is a dataset of restaurant meaning representations and associated sentences describing the restaurant. You should read about the dataset, and write code to load it in correctly.

1.2 Train a model on this dataset

Once you have imported the E2E dataset, you must train the model on a GPU. You will not need to make any changes to the model or training procedure. In fact, you are not allowed to change the model. You may make some changes to the training procedure if you wish. You will be required to submit a trained model (encoder and decoder). This model should pass an accuracy threshold, determined by a BLEU evaluation. This threshold should not be difficult to achieve; it simply checks that you imported and trained your model successfully.

The threshold is .35 on the development data. That is, if you calculate the BLEU score for every sentence in the development data (using greedy decoding), the average BLEU score should be above .35. This threshold should not be difficult to achieve. See the note below about how to correctly calculate BLEU scores.

1.3 Implement a beam search evaluator

We provide an evaluation function that performs greedy decoding on any input sequence provided. You must write a new evaluation function that implements beam search for an arbitrary beam size.

In greedy decoding, at each decoding step the most likely word is selected, resulting in one decoded sequence output. In beam search, at each decoding step the top k most likely sequences are selected. Each of these k sequences is then used to generate the next step; the top k next words per sequence are considered (for a total of $k * k$ sequences) and the top k sequences are selected to take to the next decoding step. At the end, you have k decoded sequence outputs.

1.4 Implement a BLEU evaluation

Although there exist python libraries that will calculate BLEU for you, you are required to implement your own. This will make you familiar with exactly how it works, which will be useful for understanding how well it measures performance.

BLEU scores require a list of candidate correct outputs. If you look at the development data, you will see that there are multiple data points with the same meaning representation (i.e. input) and different output sentences. When calculating your BLEU score, you must collect all the possible correct outputs.

To understand how BLEU is calculated, please rely on the original BLEU paper, BLEU: a Method for Automatic Evaluation of Machine Translation for how they calculate *sentence level* BLEU scores. The description in Natural Language Processing by Jacob Eisenstein may also be useful.

Additionally, we outline it for you here below. At a high level, BLEU is calculated as:

$$\text{BLEU-N} = \text{BrevityPenalty} * \exp(\sum_{n=1}^{N} w_n \log p_n)$$

BLEU-N is a weighted sum (w_n) of modified precision scores (p_n) for a set of n-grams (n). That is, BLEU-4 is the weighted sum of the modified precision scores for unigrams, bigrams, 3-grams, and 4-grams. A geometric mean is used; in the equation above this amounts to $w_n = 1/N$.

Additionally, a brevity penalty is applied. This brevity penalty is calculated using the length of the output (often called 'candidate') sentence and the length of the reference sentence that has the closest length to the candidate sentence. If c is the length of the candidate sentence and r is the length of the closest reference sentence, then:

$$\text{BrevityPenalty} = \begin{cases} 1 & \text{if } c > r \\ e^{1-r/c} & \text{if } c \leq r \end{cases}$$

So how do we calculate p_n, the modified precision? For each unique ngram in the candidate, count the maximum number of times it occurs in a reference sentence. Sum these across all unique ngrams in the candidate, and divide by the total number of ngrams in the candidate. Sometimes p_n is 0 for higher ngrams. In this case, back-off and use a lower N. If it is 0 for all ngrams, report BLEU as 0.

Please calculate BLEU-4 for this homework assignment.

2 Written Portion

For each question, write at least 4 sentences.

2.1 Model architecture

Draw a diagram of the model used in this homework. It should include at least: individual input tokens, individual output tokens, a start-of-sequence token, an end-of-sequence token, and the final hidden state of the encoder. In words, describe how this model works. (This is the only portion of the written homework that may be hand-drawn, and only the diagram. The description must be typeset. And we would prefer you typeset the diagram if possible.)

2.2 Error analysis

Describe three kinds of errors the system makes, with examples (input and output) from the development dataset. You may use greedy decoding for this question. Describe each error in at least 2 sentences.

2.3 Beam search analysis

Show your BLEU scores on the development set for greedy decoding and beam search (beam size = 3). For beam search, use the top scoring sequence (i.e. with the highest probability) to calculate the BLEU score for that datapoint. Provide and discuss three examples of beam search. That is, for three input sequences, show the input sequence and multiple outputs from the beam search. How do the multiple outputs differ? Will beam search always include the greedily decoded output? (Why or why not?) What are the advantages and disadvantages of beam search? Discuss the differences in BLEU score for the greedy and beam search decoding.

2.4 What's wrong with BLEU?

Although several new metrics have been proposed, BLEU remains a common evaluation metric used to evaluate models. What are the problems with BLEU? Does it tend to over- or under-estimate performance? Use examples. What are the benefits of BLEU?

2.5 Pyramid scoring for diverse outputs

Pyramid scoring can be used to score generated text outputs, traditionally for summaries. Show how to build the pyramid for the 3 reference sentences shown below. You don't have to build the whole pyramid, but show **two** Summary Content Units (SCUs) that could be added to the pyramid of weight 3, **two** SCUs that could be added to the pyramid of weight 2 and **two** SCUs that could be added to the pyramid of weight 1. Then, assuming that this is the full pyramid, show how you would compute the pyramid score for the generated sentence.

Reference sentences:

1. If you're in the riverside area and are looking for somewhere family-friendly, I've heard that The Cambridge Blue coffee shop has some decent reviews. It's just by Burger King.
2. A low-priced, children-friendly coffee shop, The Cambridge Blue has average customer ratings and is located near Burger King by the riverside.
3. The Cambridge Blue is a family friendly coffee shop providing Italian food with an average customer rating.

Generated sentence:

- Near Burger King at the riverside, there is a family-friendly coffee shop called The Cambridge Blue.

2.6 How could we deal with unseen restaurant names?

Try generating a sentence with a restaurant name that the model has never seen before. Show the result. How could you change the model to deal with this? Be specific enough that one could try to implement this. (You will not actually implement this, so it could be quite complicated. As long as it's within the realm of possibility, dream big.)

2.7 Bonus: What problems might occur in different languages?

Recently there has been a push to try the E2E challenge in other languages. What problems do you think we might encounter in other languages? Pick at least one language to discuss. Would this challenge be harder or easier in this other language? Why? (You do not need to know another language to answer this question, but you must know enough *about* the language to discuss the impact of its differences.)

3 Submission Instructions & Grading

You must submit the following:

1. A compressed file named 'hw4_your-uni.zip' uploaded to Courseworks. This should contain one jupyter notebook, one encoder model, and one decoder model. The jupyter notebook must have the outputs as specified in the notebook. The models must be trained on a GPU. The names of these files must be:

   HW4.ipynb encoder.mdl decoder.mdl

2. A pdf with your answers to the written portion on Gradescope.

3.1 Coding Portion (45 points)

- Model that runs on GPU and passes threshold (10 points)
- Beam search implementation (20 points)
- BLEU evaluator (15 points)

3.2 Written Portion (55 points)

- Q0: Model architecture (5 points)
- Q1: Error analysis (10 points)
- Q2: Beam search analysis (10 points)
- Q3: Problems with BLEU (10 points)
- Q4: Pyramid scoring (10 points)
- Q5: Unseen restaurant names (10 points)
- Bonus Q: Other languages (5 points)