CS 4650 (Spring 2021) HW4 – seq2seq

March 2021

Instructions

1. This homework has two parts: Q1 and Q2 are theory problems, and Q3 is a coding problem.
2. We will be using Gradescope to collect your assignments. Please carefully read the following instructions for submitting to Gradescope.
   (a) Each subproblem must be submitted on a separate page. When submitting to Gradescope (under **HW4 Writing**), make sure to mark which page(s) correspond to each problem or subproblem.
   (b) For the coding problem, please upload code pdf (notebook should be transformed to pdf with running output) under the **HW4 Programming** assignment on Gradescope.
   (c) Note: this is a large class and Gradescope's assignment segmentation features are essential. Failure to follow these instructions may result in parts of your assignment not being graded. We will not entertain regrading requests for failure to follow instructions.
3. LaTeX solutions are strongly encouraged (solution template available on the class website), but scanned handwritten copies are also acceptable. Hard copies are not accepted.
4. We generally encourage collaboration with other students. You may discuss the questions and potential directions for solving them with another student. However, you need to write your own solutions and code separately, and not as a group activity. Please list the students you collaborated with on the submission site.

1 Beam Search Decoding for Machine Translation

Beam Search decoding is usually used in machine translation to explore more than one possible translation hypotheses for a given sequence. At any given time, beam search keeps track of k-best options and extends them at each time step. At each step, it prunes the candidates and keeps only the top-k sequences. This process stops when an [EOS] token is generated.

Consider the following Telugu sentence:

"Varu spam mariyu gudlu tayaru chesaru."

We have a Machine Translation system that is trained to translate sequences from Telugu to English. It generates a probability distribution over each word in the vocabulary and then decodes these probabilities using beam search.

For the sake of this problem, let us assume that the generated English sequence is only dependent on the previously generated word and the input sequence.

Therefore,

$$P(w_n|w_1, w_2, ...w_{n-1}, input) = P(w_n|w_{n-1}, input) \tag{1}$$

We can then construct the following two-dimensional table of **log-probabilities**. Rows represent w_{i-1} and columns represent w_i.

(w_{i-1}, w_i)	they	prepared	cooked	he	spam	eggs	of	and	instead	[EOS]
[BOS]	-0.1	-0.9	-0.95	-0.2	-0.35	-0.4	-0.3	-0.3	-0.25	-1.3
they	-1.2	-0.7	-0.5	-1.1	-0.95	-0.95	-0.8	-0.9	-0.85	-1.5
prepared	-2.3	-3.5	-1.6	-1.8	-0.5	-0.4	-1.1	-0.7	-0.85	-0.9
cooked	-2.2	-3.6	-1.4	-1.9	-0.45	-0.6	-1.3	-0.75	-0.8	-0.95
he	-1.4	-0.55	-0.7	-2.1	-1.4	-0.8	-1.7	-0.9	-0.75	-2.1
spam	-1.2	-0.8	-0.75	-1.3	-2.6	-1.6	-1.8	-0.4	-0.6	-0.55
eggs	-1.5	-0.95	-0.8	-1.4	-2.8	-2.7	-0.8	-0.5	-0.6	-0.4
of	-3.2	-2.5	-2.7	-2.1	-1.6	-0.9	-2.8	-1.9	-0.3	-2.7
and	-1.2	-0.9	-0.8	-0.7	-0.5	-0.35	-2.5	-2.9	-0.6	-3.4
instead	-1.7	-1.2	-1.25	-0.95	-1.6	-2.1	-0.4	-0.85	-3.6	-1.2

Table 1: Log Probabilities. [BOS] and [EOS] mark the beginning and end of a sequence respectively.

(a) Use beam search with $k = 2$ to find the most likely English translation. When you encounter an [EOS] tag, **keep** that sequence in the beam and continue decoding the remaining $k - 1$ beams. Draw a figure similar to Figure 11.11 in Jurafsky and Martin decoding the likely sequences. [5 points]

(b) Beam search is biased towards shorter sequences and sometimes picks a shorter, less accurate sequences over a longer, more accurate one. One way to mitigate this, is to normalize the log-likelihood of the sequence by its length. Apply this method to the above sequence and show your results. Include the [EOS] in your sequence length count, but skip the [BOS] token. [5 points]

2 N-gram models and Smoothing

The Kneser-Ney smoothing method approximates the probability of an n-gram that has not been seen using the likelihood of the (n-1)-gram to occur in diverse contexts. If we want to finish the sentence "I want to go to the movie" with the word "theatre" but we have not observed "theatre" very often, then we want to make sure that we can guess "theatre" based on its likelihood of combining with other words. Section 3.5 of Jurafsky and Martin has an in-depth explanation of the intuition behind Kneser-Ney smoothing.

The full formula for the bigram version of Kneser-Ney smoothing follows:

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(C(w_{i-1}w_i) - d, 0)}{C(w_{i-1})} + \lambda(w_{i-1}) \frac{|\{v : C(vw_i) > 0\}|}{\Sigma_{w'} |\{v : C(vw') > 0\}|}$$

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} |\{w' : C(w_{i-1}, w') > 0\}| .$$

Assume that you have collected the data in the following tables, and assume that all other observed counts are 0. In this bigram table, rows represent w_{i-1} and columns represent w_i, e.g., $C(spam, ham) = 4$.

$C(w_{i-1}, w_i)$	clothing	store	retailer	website
clothing	0	4	2	4
store	3	0	0	3
retailer	1	5	0	0
website	4	4	0	0

Table 2: Bigram Frequency

clothing	12
store	12
retailer	8
website	12

Table 3: Unigram Frequency

Consider the following sentence fragment S : "Monty Python shopped at the clothing _____". You need to determine whether the sentence is more likely to end with "store" or "website".

(a) Compute the raw bigram probabilities for the candidate words $\{store, website\}$ to complete the sentence S, i.e., $P(store|clothing)$ and $P(website|clothing)$. Is one word more likely than the other, and if so which one? [2 pts]

(b) Compute the Kneser-Ney smoothed bigram probability of the candidate words $\{store, website\}$ to complete the sentence. Use $d = 0.5$ as the discount term. Is one word more likely than the other, and if so which one? If the result has changed, why do you think it changed? [5 pts]

(c) Change the discount term to $d = 0.1$ and re-compute the Kneser-Ney smoothed bigram probability of the candidate words $\{store, website\}$ to complete the sentence. Is one word more likely than the other, and if so which one? If the result has changed, why do you think it changed? [3 pts]

3 Programming: Machine Translation (30 points)

In this problem, you will implement a basic LSTM Seq2seq machine translation model, and carry out hyper-parameter tuning. You need to first copy the skeleton ipython notebook to your own Google Drive. After completion, you need to transform the notebook to pdf and submit your pdf file under HW4 programming on Gradescope. When making a submission, make sure all the outputs are present.

(a) Implement a basic LSTM Seq2seq machine translation model by completing the skeleton code provided in `Seq2seqBaseline` Class in the notebook. Also, you need to implement `make_batch` function and `predict_greedy` function.

Under the default hyper-parameter setting, after 10 epochs you should be able to get at least a validation token accuracy above 53%. Using your implemented greedy search function, you should get above 18 BLEU on the validation set. Note that it takes around 4 minutes to train the model for 10 epochs with default setting on GPU. (20 points)

(b) In order to improve your model performance, try the hyper-parameter tuning. Specifically, you are expected to make some modifications on embedding and hidden dimensions, number of LSTM layers, dropout rate, learning rate, batch size and training epochs, where training epochs should be no larger than 20. You will receive 3/5 pts if your model is able to achieve 54%/55% accuracy on the validation set. You will receive another 3/5 pts if your model is able to achieve above 20/21 BLEU on the validation set. (10 points)