CS 4650 Fall 2021: Homework 6

November 17, 2021

Instructions

1. This homework has two parts: questions 1–2 are written questions, and Q3 is a programming assignment with some written components within a Jupyter Notebook.

   We will be using Gradescope to collect your assignments. Please read the following instructions for submitting to Gradescope carefully!

   (a) Each subproblem must be submitted on a separate page. When submitting to Gradescope (under HW6 Writing), make sure to mark which page(s) correspond to each problem or subproblem. For instance, Q2 has five subproblems, so the solution to each must start on a new page.

   (b) For the coding problem (Q3), please upload **`HW6_Chatbot.ipynb`**, **`dataset.py`**, and **`model.py`** under the HW6 programming assignment on Gradescope. Write your solutions for Q3 in your writeup, and attach a pdf export of 'HW6.ipynb', including outputs, to your writeup.

   (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.

2. LaTeX solutions are strongly encouraged (a solution template is available on the class website), but scanned handwritten copies are also acceptable. Hard copies are not accepted.

3. 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.

Questions

1. **Open-domain Question Answering** aims to find the answers to questions expressed in natural language from a large collection of documents.

   Traditional methods for Open-domain Question Answering are composed of two stages. In the first stage, a Document Retriever retrieves articles that are likely to be relevant to the question as candidate documents. In the second stage, a Document Reader predicts a text span in the candidate documents as the final answer.

   (a) In the Document Reader, we consider candidate documents as a large paragraph composed of N words $\{w_1, w_2, \cdots, w_N\}$. We use a paragraph encoder to encode all words in the paragraph into vectors: $\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N\}$. We also use a question encoder to encode the question into a vector $\mathbf{q}$. Finally, we compute the probabilities of each token in the paragraph being start and end of the correct span: $P_{start}(i) = \mathbf{p}_i \mathbf{W}_s \mathbf{q}$, $P_{end}(i) = \mathbf{p}_i \mathbf{W}_e \mathbf{q}$.

      i. In the paragraph encoder, the basic input are word embeddings. Besides word embeddings, we want to design a binary feature as additional input for each word. What would you choose as a binary feature to improve the performance most. [2 pt]

      ii. Suppose parts of $\mathbf{p}_i$ is constructed from aligned question embedding. Assume we have a column vector to represent the word "America" in the paragraph: $\mathbf{e} \in \mathbb{R}^d$. We have a matrix composed of three vectors to represent the question "Where is Seattle": $\mathbf{Q} \in \mathbb{R}^{3\times d}$. $f(\cdot)$ denotes a softmax layer. Use dot-product attention mechanism to calculate the aligned question embedding for the word "America". Your answer should be in terms of $\{\mathbf{Q}, \mathbf{e}, f(\cdot)\}$ [3 points].

   (b) In the Document Retriever, articles and questions are compared as TF-IDF weighted bag-of-word vectors in order to return the k most relevant articles. But such term-based sparse representations have limited capabilities in matching questions and passages, e.g., there could be many relevant articles where there is no exact term match.

      Instead, we want a model to learn to retrieve documents. Suppose we have a large collection of documents $\{z_1, z_2, \cdots, z_M\}$. We use a model to compute $p(z_i|x)$, where x is the question. Document Reader can compute the probability of answer a: $p(a|x, z_i)$. To train Document Retriever and Document Reader jointly, we maximize the probability $p(a|x)$.

      i. Write down the probability $p(a|x)$ in terms of $p(z_i|x)$ and $p(a|x, z_i)$. [2 points]

      ii. Computing $p(a|x)$ exactly in such formula is very time-consuming. Explain why [1 point], and propose a solution to approximate $p(a|x)$ [2 points].

Computational Social Science & Ethics. In this section, you'll make some decisions about a model that you've trained to detect "toxic" text on a large annotated dataset your boss has provided you. You've completed your task—training your model for 6 epochs—your Train F-1 score looks great (practically 1, see Table 1). However, you're noticing some odd problems in your model:

2. (a) After experimenting with your model, you've noticed that negative auxiliary inversions (NAI), present in African American Vernacular English (AAVE), are resulting in label flips for your classifier. Here's an example of an NAI:

   i. Nobody wants to procrastinate $\rightarrow$ NOT TOXIC

   ii. Don't nobody want to procrastinate (AAVE with an NAI) $\rightarrow$ TOXIC

   What conclusions can you draw about your model/dataset? [2 points]

   (b) What are some consequences of deploying a model that is biased against NAIs? [2 points]

   (c) You decide to annotate your randomly sampled validation set ($N = 1000$) into two categories: one with AAVE data ($N \approx 100$), and another with all other dialects ($N \approx 900$). Below are tables of your train F-1 and validation F-1 across both dialects.

Dialect	Epoch=1	2	3	4	5	6
Non-AAVE	0.52	0.76	0.82	0.87	0.96	0.99
AAVE	0.26	0.45	0.68	0.72	0.76	0.81

Table 1: Train F-1 over 6 epochs

Dialect	Epoch=1	2	3	4	5	6
Non-AAVE	0.23	0.45	0.79	0.84	0.92	0.87
AAVE	0.12	0.35	0.66	0.69	0.72	0.43

Table 2: Validation F-1 over 6 epochs

   What is going on? Given the specific training behaviour of your model, what is one solution that might mitigate the severity of your problem? [4 points]

   (d) What is one other general strategy you might propose to mitigate this particular NAI problem? [2 points]

   (e) (BONUS) Read this paper on Model Cards, and summarize 3 of the key points it makes. How might these points change the situation you were in earlier? Do you agree/disagree with the authors' proposals, and why? [3 points]

3. For this assignment, we will be building a conversational agent (chatbot) using the familiar template of Seq2Seq framework we implemented in HW5. However, this time we will directly start with the AttnDecoderRNN. Our chatbot will be trained on a manually collected and scraped corpus of dialogues from the popular TV show, Friends.

   You will work with three filed: the main notebook **`HW6_Chatbot.ipynb`**, **`dataset.py`**, and **`model.py`**. Please start with the notebook file and follow instructions there. **<u>All three files</u>** should be submitted to the Gradescope assignment **HW6 Coding**. For your write-up, please answer questions in the following parts and attach a PDF for your final version of the notebook at the end of your write-up.

   The assignment zip can be downloaded here:

   https://www.cc.gatech.edu/classes/AY2022/cs4650_fall/programming/h6_chatbot.zip

   (a) Implement dataloader in ‘dataset.py’ to retrieve data pairs for context and response messages. No write-up is required for this part. [5 points]

   (b) Implement `EncoderRNN` and `AttnDecoderRNN` in ‘models.py‘. No writeup required. [3 + 7 points]

   (c) Pick *two* of the six main characters. Then implement `train` and `evaluate` functions in the notebook and train the model (with context size of 1). You will have to perform some hyperparameter tuning (lower `eval_every` and `eval_samples` argument while tuning to see frequent feedback, note: this will increase noise in the loss reported every iteration). Please make sure to mark appropriate pages to these sub-routines for your write-up Gradescope submission.

   Further, **in your write-up**, include plots (loss and BLEU scores) and final performance numbers. Furthermore, describe the effects of various hyperparameters you tried and attach 10 sample sentences obtained by conversing with the model (obtained by running `evaluateManually` in the notebook) [5 points implementation + 5 points write-up].

   (d) Train the model again but this time with context size of 3 previous messages instead of 1. The data is already preprocessed and stored in `filtered_df`, you simply need to modify the `context_column` variable. **In your write-up**, include plots (loss / BLEU) and final performance numbers, analysis of various hyperparameters you experimented with as well as 10 sample response obtained by conversing with the model. **Compare** your results here with those from previous part (with context size of 1) and **explain** any differences you observe. [5 points write-up]