CS288 HW1: Language Modeling

Nicholas Tomlin and Dan Klein

Due: 4 February 2022, 5:00PM PST

Overview

The first homework will be focused on language modeling. We'll cover classical n-gram language models, smoothing techniques, sequence modeling in Pytorch, tokenization schemes, and how to inference on large pre-trained language models. Please note that this homework is substantially harder than in previous years, as it combines HW0 and HW1 from the Spring 2021 offering, as well as some additional material. This homework is intended to be approximately representative of the difficulty of future course homeworks.

Background Reading

Please checkout the following resources before beginning this assignment:

- PyTorch: `https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html`
- N-Gram Language Models: `https://web.stanford.edu/~jurafsky/slp3/3.pdf`
- Neural Language Models: `https://web.stanford.edu/~jurafsky/slp3/7.pdf`

Assignment

This homework consists of three Google Colaboratory notebooks, as well as a written report. Below, you'll find a handful of questions. Please answer these questions in LaTeX and save them to a file named `report.pdf`. Submit this file and all outputs from the notebooks into a single `.zip` file for submission to Gradescope.

1. Getting Started with PyTorch
   (a) Complete the notebook and save the following files:
      - `hw1a.ipynb`
      - `predicted_test_outputs_pooling.txt`
      - `predicted_test_outputs_improved.txt`

   (b) Report: please list at least one correct and one incorrect prediction from your improved network, and give a proposed explanation for why the model might have gotten it wrong. Did the pooling network get these examples right?
2. Training Language Models
   (a) Complete the notebook and save the following files:
      - `hw1b.ipynb`
      - `bigram_predictions.npy`
      - `trigram_kn_predictions.npy`

        - `neural_trigram_predictions.npy`
        - `lstm_predictions.npy`

    (b) Report: please describe the modifications you made to your LSTM and its corresponding perplexity. Include (1) a concise and precise description of the extension that you tried, (2) a motivation for why you believed this approach might improve your model, (3) a discussion of whether the extension was effective and/or an analysis of the results, and (4) a bottom-line summary of your results comparing validation perplexities of your improvement to the original LSTM. This should involve some combination of tables, learning curves, etc. and be at least half a page in length.

3. Language Model Inference

    (a) Complete the notebook and save the following files:
        - `word_probs.npy`
        - `sst_predictions.npy`

    (b) Report: the uniform information density hypothesis states that speakers aim to distribute surprising content as much as possible across the duration of a sentence. Section 5 of the following paper proposes some metrics of uniform information: `https://arxiv.org/pdf/2010.02650.pdf`. Implement at least one of these metrics and find one sentence which has high uniformity under the metric and another sentence which has low uniformity, and plot their surprisal graphs (using the provided `plot_gpt_probs()` function or similar).

    (c) Report: what prompt did you use for the final part of this assignment? How did you choose it, and did you make any other modifications to the prediction function?

Debugging and Bug Reports

We unfortunately do not have the staffing capacity to help with debugging student code. If you believe you have encountered a legitimate error in one of the notebooks, please post it to the course forum and we'll do our best to correct the issue as soon as possible: `piazza.com/berkeley/spring2022/cs288`

Submission to Gradescope

Please submit the assignment to: `https://www.gradescope.com/courses/361823/` (code: 4PBP57)

When you upload your submission to the Gradescope assignment, you should get immediate feedback that confirms your submission was processed correctly. Be sure to check this, as an incorrectly formatted submission could cause the autograder to fail. For this project, you should be able to see your test set accuracies and a confirmation that all required files were found, but you will not be able to see your score until later. Most assignments will be graded primarily on your test set accuracies and written report.

Note that Gradesope will allow you to submit multiple times before the deadline, and we will use the latest submission for grading. Make sure you have the following files (with correct names and extensions):

- `predicted_test_outputs_pooling.txt`
- `predicted_test_outputs_improved.txt`
- `bigram_predictions.npy`
- `trigram_kn_predictions.npy`
- `neural_trigram_predictions.npy`
- `lstm_predictions.npy`
- `word_probs.npy`
- `sst_predictions.npy`
- `hw1a.ipynb`
- `hw1b.ipynb`
- `hw1c.ipynb`
- `report.pdf`