CS 272: Statistical NLP: Spring 2022

Homework 2: Language Modeling

Sameer Singh
https://canvas.eee.uci.edu/courses/45812/

One of the fundamental tasks for natural language processing is probabilistic modeling of language, i.e. how can we differentiate between a random sequence of words, and something we might consider an English sentence. Such language models are used in many applications, such as handwriting recognition, speech recognition, machine translation, and text generation. In this second programming assignment, you will perform language modeling of different kinds of text. The submissions are due by midnight on **April 29**.

1 Task: Language Modeling of Different Datasets

Your task is to analyze the similarities and differences in different domains using your language model.

Data

The data archive (available on Canvas) contains corpus from three different domains, with a train, test, dev, and readme file for each of them. The domains are summarized below, but feel free to uncompress and examine the files themselves for more details (will be quite helpful to perform your analysis).

- *Brown Corpus:* Objective of the corpus is to be the *standard* corpus to represent the present-day (i.e. 1979) edited American English. More details at http://clu.uni.no/icame/manuals/BROWN/INDEX.HTM.
- *Gutenberg Corpus:* This corpus contains selection of text from public domain works by authors including Jane Austen and William Shakespeare (see readme file for the full list). More details about Project Gutenberg is available at http://gutenberg.net/.
- *Reuters Corpus:* Collection of financial news articles that appeared on the Reuters newswire in 1987. The corpus is hosted on the UCI ML repository at https://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection.

Source Code

I have released some initial source code, available at https://github.com/sameersingh/uci-statnlp/tree/master/hw2. The interface and a simple implementation of a language model is available in `lm.py`, which you will extend to implement your n-gram models. In `generator.py`, I provide a generic sentence sampler for a language model. The file `data.py` contains the *main* function, that reads in all the train, test, and dev files from the archive, trains all the unigram models, and computes the (adjusted) perplexity of all the models on each other's data. The file `demo.py` is used to launch an interactive demo in your browser for interacting with your trained language models. You will need Python3 for this project, as well as the Python package `streamlit`. The README file provides a little bit more detail. Of course, feel free to ignore the code if you do not find it useful.

2 What to Submit?

Submit a single write-up (**PDF, maximum 5 pages**) and relevant code (compressed in a single `zip` or `tar.gz` file; we will not be compiling or executing it, nor will we be evaluating the quality of the code) to Canvas. Use the ACL style files here: https://github.com/acl-org/acl-style-files. Please remove `[review]` option to generate the final version, as described in the example. **Do not include your student ID number**, since we might share it with the class if it's worth highlighting. The write-up and code should contain the following.

2.1 Implement an N-gram Language Model (25 points)

In `lm.py`, we have implemented a simple unigram language model. The primary task of the homework is to implement an n-gram language model for any arbitrary sized n-gram in the skeleton class `Ngram` in `lm.py`. Your language model should support "start of sentence", i.e. when the context is empty or does not have enough tokens. Use appropriate smoothing to ensure your language model outputs a non-zero and valid probability

distribution for out-of-vocabulary words as well. Describe the design choices, such as what smoothing you used, and any implementation details you think are important (for example, if you implemented your own sampler, or an efficient smoothing strategy), but do not include *any* code.

2.2 Hyper-parameter Search (10 points)

Find the best set of hyper-parameters for your n-gram language model by tuning over the dev data. You do not need to find the best hyper-parameters specifically for each domain, rather, the hyper-parameters that work best on average across the three dev datasets is sufficient. In the write up, describe what hyper-parameters you picked and how you performed the search. Include additional graphs/plots/tables, as needed.

2.3 Empirical Evaluation of Language Models (25 points)

Once you have decided the hyper-parameters and trained your models, you will evaluate perplexity of the models.

- *In-Domain Perplexity:* Present the train *and test* perplexity on each of the three domains using the respective n-gram language models and compare it to the unigram models.
- *Out-of-Domain Perplexity:* Compute the test perplexity of all three of your models on all three domains (a 3×3 matrix, as computed in `data.py`). Compare these to the unigram models as well.

2.4 Language Generation and Scoring (25 points)

Apart from the perplexity results, you will also try to "use" the language models, using the demo. Launch the demo by running `streamlit run demo.py` and open it up in the browser.

- *Generated Examples:* Using the generator in the demo, show generated sentences to highlight the similarities/differences between both the unigram vs your n-gram models and between the three domains. You might want to consider using the same prefix, and show the different sentences each of them results in.
- *Sentence Scoring:* The demo also allows you to *score* a given sentence by perplexity, with lower perplexity indicating a more "natural" sentence. Come up with sentences that unigram scores higher than your n-gram model, and vice versa, to highlight the differences in these models. Similarly, pick any two domains, and generate grammatically correct sentences such that your model for one domain scores it higher than your model for the other domain, and vice versa.

2.5 Discussion (15 points)

Include a discussion on what you observed about the empirical and qualitative results. For example, if unigram outperforms one of your models in any of the settings, why might that happen? Why do you think certain models/domains generalize better to other domains? What might it say about the language used in the domains and their similarity? Support your discussion with insights from the examples.

3 Statement of Collaboration

It is **mandatory** to include a *Statement of Collaboration* in each submission, with respect to the guidelines below. Include the names of everyone involved in the discussions (especially in-person ones), and what was discussed.

All students are required to follow the academic honesty guidelines posted on the course website. For programming assignments, in particular, I encourage the students to organize (perhaps using Ed) to discuss the task descriptions, requirements, bugs in my code, and the relevant technical content *before* they start working on it. However, you should not discuss the specific solutions, such as sharing code, and, as a guiding principle, you are not allowed to take anything written or drawn away from these discussions (i.e. no screenshots/photographs, written notes, etc.). Especially *after* you have started working on the assignment, try to restrict the discussion to Ed as much as possible, so that there is no doubt as to the extent of your collaboration.

Since we do not have a leaderboard for this assignment, you are free to discuss the numbers you are getting with others, and again, I encourage you to use Ed to post your results and compare them with others.

Acknowledgements

This homework is adapted from one by Prof. Yejin Choi of the University of Washington. Thanks, Yejin!