CS 295: Statistical NLP: Winter 2018
Homework 2: Language Modeling

Sameer Singh
http://sameersingh.org/courses/statnlp/wi18/

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 **February 8, 2018**.

1 Task: Language Modeling of Different Datasets

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

1.1 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 are available at http://www.hit.uib.no/icame/brown/bcm.html.
- *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.

1.2 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 can extend to implement your 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 perplexity of all the models on each other's data. 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?

Prepare and submit a single write-up (**PDF, maximum 5 pages**) and relevant source 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. **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 a Language Model (20 points)

The primary task of the homework is to implement a non-trivial language model. You are free to pick the type of the model, such as discriminative/neural or generative. If you decide to implement an n-gram language model, it should *at least* use the previous two words, i.e. a trigram model (with appropriate filtering and smoothing). 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. In order to make things efficient for evaluation and analysis, it might be worthwhile to implement serialization of the model to disk, perhaps using `pickle`.

In the write up, define and describe the language model in detail (saying "trigram+laplace smoothing" is not sufficient). Include any implementation details you think are important (for example, if you implemented your own sampler, or an efficient smoothing strategy). Also describe what the hyper-parameters of your model are and how you set them (you should use the dev split of the data if you are going to tune it).

2.2 Analysis on In-Domain Text (40 points)

Here, you will train a model for each of the domains, and anayze only on the text from their respective domains.

- *Empirical Evaluation:* Compute the perplexity of the test set for each of the three domains (the provided code would do this for you), and compare it to the unigram model. If it is easy to include a baseline version of your model, for example leaving out some features or using only bigrams, please do so. Provide further empirical analysis of the performance of your model, such as the performance as hyper-parameters and/or amount of training data is varied, or implementing an additional metric.
- *Qualitative:* Show examples of sampled sentences to highlight what your models represent for each domain. It might be quite illustrative to start with the same prefix, and show the different sentences each of them results in. You may also hand-select, or construct, sentences for each domain, and show how usage of certain words/phrases is scored by all of your models (function `lm.logprob_sentence()` might be useful for this).

2.3 Analysis on Out-of-Domain Text (40 points)

In this part, you have to evaluate your models on text from a domain different from the one it was trained on. For example, you will be analyzing how a model trained on the Brown corpus performs on the Gutenberg text.

- *Empirical Evaluation:* Include the 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, and your baselines if any, and discuss the results (e.g. if unigram outperforms one of your models, why might that happen?). Include additional graphs/plots/tables to support your analysis.
- *Qualitative Analysis:* Provide an analysis of the above results. 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? Provide graphs, tables, charts, examples, or other summary evidence to support any claims you make (you can reuse the same tools as the qualitative analysis in § 2.2, or introduce new ones).

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 Piazza) 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, and, as a guiding principle, you are not allowed to take anything written or drawn away from these discussions (i.e. no photographs of the blackboard, written notes, referring to Piazza, etc.). Especially *after* you have started working on the assignment, try to restrict the discussion to Piazza 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 Piazza to post your results and comparing them with others.

Acknowledgements

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