Programming assignment 1: Language Modeling

CSE 256: Statistical NLP: Fall 2019

University of California, San Diego

Released: October 1, 2019

Due: October 14, 2019

In this programming assignment, you will build probabilistic language models. Your task is to analyze texts from different domains using your language models.

Your final writeup for this assignment, including the problem and the report of the experimental findings of the programming part, should be no more than four pages long. You should submit it as a pdf. We strongly recommend typesetting your scientific writing using LaTeX. Some free tools that might help: TexStudio (Windows, Mac), MacTex (Mac), TexMaker (cross-platform), and Detexify (online).

1 Preliminaries

1.1 Data and Code

We provide three corpora to conduct the evaluation (`Brown, Gutenberg, and Reuters`), summarized below, you are encouraged to examine the them.

- **Brown Corpus:** The objective of this corpus is to be the standard corpus to represent the present-day (i.e., 1979) American English. More details are available at `http://www.hit.uib.no/icame/brown/bcm.html`.
- **Gutenberg Corpus:** This corpus contains a selection of text from public domain works by authors including Jane Austen and William Shakespeare. More details about Project Gutenberg is available at `http://gutenberg.net/`.
- **Reuters Corpus:** A 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`.

Some initial source code is provided. 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*, we provide a 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 the different corpora. The README file provides a little bit more detail. The code has been tested on **Python 3**. Feel free to ignore the code if you do not find it useful.

2 Implement a Language Model (30%)

Implement a language model. You are free to pick any type of language model covered in class. We recommend implementing an n-gram language model, it should at least use the previous two words, i.e. a trigram model (with appropriate filtering). Use appropriate smoothing to ensure your language model outputs a non-zero and valid probability distribution for out-of-vocabulary words as well.
In your 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).

3 Analysis on In-Domain Text (30%)

Train a language model for each of the domains, and evaluate the language model on the text from the respective domain.

- **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, but this is not required. 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 Analysis:** 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).

4 Analysis on Out-of-Domain Text (30%)

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 $3\times$ 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.

5 Adaptation (10%)

Suppose you have trained a model on corpus A and wish to test it on the test set for corpus B. Design an approach for using a small fraction of corpus Bs training data to adapt the model trained on corpus A. How does it influence performance (for example, does it outperform the initial model trained just on corpus A?) How close can you get to performance when training on corpus Bs full training set?

6 Submission Instructions

Submit on Gradescope (more details on Gradescope to be provided.).

- **Code:** You will submit your code together with a neatly written README file with instructions on how to run your code. We assume that you always follow good practice of coding (commenting, structuring), and these factors are not central to your grade.
- **Report:** As noted above, your writeup should be *four pages long, or less, in pdf* (reasonable font sizes). Part of the training we aim to give you in this class includes practice with technical writing. Organize your report as neatly as possible, and articulate your

thoughts as clearly as possible. We prefer quality over quantity. Do not flood the report with tangential information such as low-level documentation of your code that belongs in code comments or the README. Similarly, when discussing the experimental results, do not copy and paste the entire system output directly to the report. Instead, create tables and figures to organize the experimental results.

7 Report Details

Make sure your report includes the following items:

7.1 Language Model Implementation

a. Clear Description of model

b. Clear description of smoothing method

c. Mention hyperparameter tuning and data used for tuning

7.2 Analysis of In-Domain Text

a. Report your perplexity values

b. Compare with unigram model and discuss your observations

c. Show performance w.r.t hyperparameters

d. Show examples of sampled sentences

7.3 Analysis of Out-of-Domain Text

a. Report your perplexity values

b. Compare with unigram model and discuss your observations

c. Compare performance on different corpora and discuss your observations

7.4 Adaptation

a. Discuss your approach

b. Show relevant comparisons

8 Acknowledgments

Adapted with minor changes from similar assignments by Yejin Choi, Noah Smith and Sameer Singh.