CS4740: Introduction to NLP
Project 1: Language Modeling

Due electronically by 11:59 PM, Tuesday, Feb. 28, 2012
(Part 1 due: 11:59 PM, Tuesday, Feb. 21, 2012)

1 Overall Goal

This project is an open-ended programming and research assignment in which you are to implement a collection of n-gram-based language models. We suggest that you work in groups of 3–4 students. Students in the same group get the same grade. Please form groups using CMS.

2 Programming Portion

1. Write a program that computes unsmoothed unigrams and bigrams for an arbitrary text corpus. You must write all of the code yourself, but you may use any programming language(s) you like. You can also use packages (e.g., NLTK) for low-level tasks like tokenization, stemming, etc. Keep track of all of the key implementation and modeling decisions that you made to include in the report.

2. Use your unigram and bigram language models for **Random Sentence Generation** as described in class. For training data, use two of the four datasets available on CMS. You will only need to use the "train" portion of each dataset.

   Dataset1 : Foreign Broadcast Information Service news. English. Mostly foreign news from abroad.

   Dataset2 : Wall Street Journal. English. News, especially business-related news.

   Dataset3 : Works of Shakespeare. (Olde) English.

   Dataset4 : Excerpts from the novel "War and Peace."

   Be sure to save some sample sentences for inclusion in the report.

3. Implement any **smoothing** method you would like as well as an approach for handling **unknown words**. Implement code to compute

the **perplexity** of a document with respect to a language model. Again, keep track of any decisions made during the implementation to include in the project report.

4. Using the same two datasets, compute the perplexity of each of the language models on the **test** corpus for both datasets. A table of the results should be included in the report.

5. **Email Author Prediction**: In this part of the project you will predict email authors based on the unigram or bigram language model. Use the EnronDataset in CMS for training, validation and testing. The data files have the following format:

   author word1 word2 ...

   where *author* represents the unique id of the email author and *word1 word2 ...* is the text of the email. You should use only the training data and potentially the validation data for developing your language model. The project report should describe the approach that you employed for Author Prediction and should include a table that shows the accuracy of the approach on the validation data. The test data should be reserved for use only at the end for producing the evaluation results.

   We'll use Kaggle to evaluate your system on the validation set and, in the end, the test set. You need to use the Cornell email address (@cornell.edu) of **one** of your team members to sign up for a Kaggle account (at `http://inclass.kaggle.com/account/register`). You can only submit results as a part of a team (form your team in CMS) and use the NetID of one team member to submit the results.

   To submit your system's predictions for evaluation against the gold standard, go to `http://inclass.kaggle.com/c/email-author-prediction`. For this assignment, Kaggle only computes accuracy. (The training and validation data, however, includes the correct predictions for each instance; you can use these if you'd like to compute other scores (outside of Kaggle).)

   The format of the Kaggle submission is as follows:
   beck-s

beck-s
farmer-d
...
where each line is the predicted author of the corresponding email in the validation set and the test set (the first 2024 lines are for the predictions on the validation set and the next 2024 lines are for predictions on the test set). It might seem weird to ALWAYS include predictions for both data sets, but you can use "dummy" prediction values for the test set until you are ready to run on it. (Currently the *author* field in the test set is filled with a dummy value *beck-s*.)

You can make multiple submissions of your predictions for evaluation as you develop the system; Kaggle will keep track of all of them for you, showing you the accuracy of your model on the validation set. You will not be able to see your scores on the test set until the Kaggle competition closes on **Monday, Feb. 27, 5:00PM**. (Note that Kaggle uses UTC time, not Eastern Standard Time.)

Sometime before the competition closes, you will need to select one of your sets of predictions as your final submission (one for each team); the default final submission is the most recent set of predictions (i.e., the most recent submission). Once the Kaggle competition closes, you cannot make additional submissions to Kaggle, and the accuracy of your final submission evaluated on the test set will be posted on the competition scoreboard. You should include this score in your report.

The gold standard predictions for the test set will be released via CMS at the close of the competition. These could be of use as you analyze system performance or characterize the kinds of errors that your system made for inclusion in the project report.

6. Decide on at least **one additional extension to implement**. The idea is to identify some aspects of the language model to improve or generalize, and then to implement an extension that will fix this issue or problem for the tasks of random sentence generation, perplexity computation, or email author prediction. Section 3 below provides some ideas, but be creative(!) — you can experiment with any aspect of language modeling that you'd like. Your report will need to clearly describe the extension you implement and include an evaluation of how well it did or didn't work. (It's ok if it doesn't produce improved results.)

3 Menu of extensions

1. Implement a trigram (or 4-gram, or general n-gram) model.
2. Smoothing. Implement a second smoothing method.
3. Interpolation. Implement an interpolation method, e.g. Linear interpolation, Deleted interpolation, Katz's backoff.
4. Nontrivial unknown word handling. Develop and implement a method for better handling of unknown words.
5. Employ the language model in the service of another NLP or speech application.
6. Implement a modification that makes use of the validation set.

4 The Report

You should submit a short document (5-6 pages will suffice) that describes your work. For the sentence generation task, you should include examples of the random generator and discuss the results of the perplexity experiments. For the email author prediction task, you should report the accuracy score of your system on both the validation and test sets, and include an analysis of the results. (In addition to the Kaggle results, you can also write your own code for further evaluation and include your own results, but it is optional.) Be sure to indicate which smoothing method you implemented and how you handled unknown words.

Finally, describe the extension(s) that you decided to make and why. What experiments did you run, or analysis can you provide, to show whether or not your extension had the desired effect.

5 Grading Guide

- Part 1: progress on unigram and bigram table construction algorithms and the two tasks; plans for the extension and how you will evaluate it (5%)
- design and implementation of the unigram and bigram table construction algorithms (10% of the grade)
- design and implementation of the random sentence generator (5%)

- design and implementation of the email author predictor (5%)
- design and implementation of your selected extension (25%).
- experiment design and methodology; analysis of results; clarity and quality of the report (50%)

6 What to submit

Part 1: Progress Report / Proposal. By anytime **Tuesday, Feb. 21**, submit a short progress report including the team plans for the extension and its evaluation. Just a paragraph is required.

Full assignment. By **Tuesday, Feb. 28, 11:59PM**, submit the full assignment to CMS. This consists of a .zip or .tar — one submission per group, containing the following:

- source code and executables
- a README file that explains how to run your system
- the report