CS4740: Introduction to NLP
Project 1: Language Modeling

Due electronically by 11:59 PM, Monday, Feb. 28, 2012

1 Overall Goal

This project is an open-ended programming assignment in which you are to implement a collection of n-gram-based language models. We again 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.

2. Implement any smoothing method you would like as well as an approach for handling unknown words in the text.

3. Apply your language model to the following two tasks:

   (a) **Random Sentence Generation**: generate random sentences based on the unigram or bigram language model. There are training, validation, and test sets provided via CMS. Chooses any two datasets (from Dataset1 to Dataset4) for your analysis. Experiment with the random sentence generator after training on each of the corpora.

   Implement code that computes the perplexity of a test set. Compute the perplexity of each of the language models (trained on each of the corpora) on the two test sets.

   (b) **Email Author Prediction**: 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 only be using the training data and the validation data for developing your language model. The test data should only be used at the end for producing the evaluation results.

In order to evaluate your model, you must go to `http://inclass.kaggle.com/c/email-author-prediction` and submit your predictions to the Kaggle system. You need to use your Cornell email address (@cornell.edu) to sign up an account (in `http://inclass.kaggle.com/account/register`) to use the system. 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.

The format of the 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).

Once you make a submission in Kaggle, the system will compute a classification accuracy score for you. Note that you will not have access to the solutions of the test set until Sunday, Feb. 26, 11:59PM (currently the *author* field in the test set is filled with a dummy value *beck-s*). Before the solutions are released, you can only see the accuracy of your model on the validation set in Kaggle. You can make multiple submissions and the system will keep track of all of them. But you should only choose one as your final submission (one for each team). After the solutions are released, you cannot make any submissions in Kaggle any more. You will see the final score of your model evaluated on the test set, and you should include this score in your report.

4. 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 and email author prediction. Section 3 below provides some ideas.

3 Menu of extensions

You must implement one item from this list of extensions or come up with your own extension. Whatever you do, explain it clearly in your writeup.

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, and include an analysis of the prediction 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(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; clarity and quality of the report (50%)

6 What to submit

By anytime Monday, Feb. 20, submit the proposal for Part 1. 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