ITCS 4111/5111 Introduction to Natural Language Processing

Assignment 3 **Due: October 13th, 2017 11:59 pm**

Problem 1 (80 points):

This homework is adapted from Dan Jurafsky and Chris Manning's online NLP course. They based their code on Peter Norvig's spelling corrector.

In this assignment you will be training a language model to build a spell checker. Specifically, you will be implementing part of a noisy-channel model for spelling correction. We will give the likelihood term, or edit model, and your job is to make a language model, the prior distribution in the noisy channel model. At test time you will be given a sentence with exactly one typing error. We then select the correction that gets highest likelihood under the noisy-channel model, using your language model as the prior. Your language models will be evaluated for accuracy, the number of valid corrections, divided by the number of test sentences.

Data

We will be using the writings of secondary-school children, collected by David Holbrook. The training data is located in the data/ directory. A summary of the contents:

holbrook-tagged-train.dat: the corpus to train your language models
holbrook-tagged-dev.dat: a corpus of spelling errors for development
count_1edit.txt : a table listing counts of edits x | w

Note that the data files do not contain <s> and </s> markers, but the code which reads in the data adds them.

Instructions

Implement the following language model:
Laplace Unigram Language Model: a unigram model with add-one smoothing. Treat out-of-vocabulary items as a word which was seen zero times in training.

We have provided you with a uniform language model so you can see the basic layout, located in UniformLanguageModel.py.

To implement a language model you need to implement two functions:

train(HolbrookCorpus corpus): takes a corpus and trains your language model. Compute any counts or other corpus statistics in this function. See the example UniformLanguageModel for how to access sentences in the corpus and the words in those sentences.

score(List words): takes a list of strings as argument and returns the numerical score, which should be the log-probability of the sentence using your language model. Use whatever data you computed in train() here.

Evaluation

Your language models will be evaluated on the development data set and a held-out test set. The performance of each language model will be evaluated with respect to the performance of our solution implementation. To help with your implementation, we give you the expected performance of each language model on the development set:

Laplace Unigram Language Model: 0.11

Note that the performance we expect from your language model is not that great! We have provided you with a very simple edit model, not a lot of training data, and the task is rather difficult. You will receive full credit for implementations which meet the stated threshold.

Given Code

The rest of the scaffolding has been provided (reading corpora, computed edit probabilities, computing the argmax correction). A short summary of the remaining files:

SpellCorrect.py: Computes the most likely correction given a language model and edit model. The main() function here will load all of your language model and print performance on the development data, useful for debugging. It may be useful to comment out some of the tests in main() when developing.

EditModel.py: Reads the count_1edit.txt file and computes the probability of corrections. The candidate corrections are all strings within Damerau-Levenshtein edit distance 1. The probability of no correction is set at .9 (P(x|x) = 0.9). Note that the EditModel isn't great, but your language models will be evaluated using this model, so it won't effect your grade.

HolbrookCorpus.py: Reads in the corpus and generates test cases from misspellings.

Sentence.py: Holds the data for a given sentence, which is a list of Datums. Contains helper functions for generating the correct sentence and the sentence with the spelling error.

Datum.py: Contains two strings, word and error. The word is the corrected word, and error contains the spelling error. For tokens which are spelled correctly in the corpus, error = ''.

Additional Files

There are additional files bundled up with this code, other than the ones mentioned above already. Feel free to ignore those.

Running the code

```
$ cd src
$ python3 SpellCorrect.py
```

There's a lot of code here, but the stuff you have to implement is pretty straightforward.

Problem 2 (20 points **): You can opt to do any **one** of the following:

(a) Custom Language Model: implement a more sophisticated language model of your choice (note we haven't discussed these in class!) Ideas include interpolated Kneser-Ney, Good-Turing, linear interpolated, trigram, or any other language model you can come up with. Your goal is for your custom language model to perform better than the language model we asked you to implement.

(b) Use more training data: There really isn't very much training data for the language model here. Maybe you can get better accuracy if you use more text? Questions to answer here include: where to get more text? (Hint: NLTK) What format does it need to be in? Will any English text work, or will some work better than others? How much more accuracy did you get for how big of a bigger training corpus?

What to submit

Turn in your language model file (LaplaceUnigramLanguageModel.py), along with a text file assignment3.txt, describing your accuracies, any difficulties you had, and discussion based on your choice of problem 2.

Note: ITCS 5111 students will be graded out of a 100 points total for all problems in this assignment. ITCS 4111 students will be graded out of 80 points total for all problems in this assignment, except for those marked with **.