CMSC 723: Computational Linguistics I

Assignment 4: Is all smoothing good smoothing?

Jimmy Lin (Instructor) and Melissa Egan (TA)

Due: **November 4, 2009**

Problem 1 (10 points)

Show that using ELE (Expected Likelihood Estimator) for unigrams, as explained in class, yields a well-formed probability distribution, i.e.,

$$\sum_{w_i} P_{\text{ELE}}(w_i) = 1$$

Problem 2 (30 points)

Assume that we have the following scenario: 100 samples have been seen from a potential vocabulary of 1000 items, and in that sample, 9 items were seen 10 times, 2 items were seen 5 times and the remaining 989 items were unseen. Work out the expected frequency estimates for each of the three kinds of items according to Laplace's Law. Also calculate the total probability mass that will be assigned to the unseen items by this law.

Note: It is obvious that this is *not* a language modeling problem. However, it has been to designed to show you that smoothing probability distributions using Laplace's law moves an unacceptable amount of mass to the unseen events.

Problem 3 (60 points)

(a) Write a Python program using NLTK that computes the expected frequency estimates for bigrams of rank[†] 0 through 10, according to the following discounting techniques:

1. Laplace's Law

[†]A bigram is of rank r if it occurs exactly r times in the training data. Bigrams with rank 0 are obviously the unseen bigrams.

2. Lidstone's Law of Succession ($\gamma = 0.5$)

3. Good-Turing Discounting

Use Jane Austen's *Persuasion*, *Sense & Sensibility* and *Emma* together as the training corpus. These corpora come bundled with NLTK as different files in the Gutenberg corpus collection[‡]. Your program should output a table with the rank as the first column and the three expected frequency estimates as the subsequent columns.

(b) For each of the discounting techniques, calculate:

- The total probability mass that is assigned to unseen bigrams.
- The probability assigned to *an* unseen bigram assuming that the mass is uniformly distributed among them.

Notes:

1. The expected frequency estimate for n-grams of rank r is defined in the lecture notes.
2. Lowercase each word in the training corpus before counting it.
3. Do *not* filter any punctuation. Just treat it as a regular word.
4. You do *not* need to add the `<s>` and `</s>` markers to the sentences for this problem.
5. Do *not* count any n-grams across sentence boundaries. Remember that for n-gram language modeling, we always work sentence by sentence.
6. You may find NLTK's built-in `FreqDist` class to be useful for this problem.
7. The reason we are not using the method `gutenberg.words('...')` is because this method tokenizes the sentences differently. It's better to stick to the `sents()` method as suggested.

[‡]Use the `fileids` parameter with the `sents()` method and note that this parameter can take a list.