# HW Assignment 1 (Due date: Tue, Feb 16, by 9:00am)

## 1 Lidstone's Law (from M&S) [100 points]

Using the word and bigram frequencies within Austen's Persuasion test corpus given below, compute the Jeffrey-Perk's estimate for the test clause *she was inferior to both sisters*, using the fact that the word before *she* in the corpus was *person*. Assume the Austen training corpus results in a vocabulary $V$ of 14,585 word types and $N = 617,091$ words of text.

| $w$ | $C(w)$ | $w_1w_2$ | $C(w_1w_2)$ |
|---|---|---|---|
| person | 223 | person she | 2 |
| she | 6,917 | she was | 843 |
| was | 9,409 | was inferior | 0 |
| inferior | 33 | inferior to | 7 |
| to | 20,042 | to both | 9 |
| both | 317 | both sisters | 2 |

## 2 Good-Turing (from J&M) [100 points]

Exercise 4.8, page 121.

## 3 Language Models [100 points]

The Brown corpus is a selection of 500 American English documents containing a wide diversity of narratives. It is installed in `/usr/share/nltk_data/corpora/brown`.

(a) Compute the percentage of *happax legomena* and *dis legomena* occurrences in the entire Brown corpus.

(b) What are the longest *happax legomena* and *dis legomena* in the entire corpus?

(c) Plot the frequency ($N_r$) vs. rank ($r$) graph for all words in the Brown corpus, similar to the graph on slide 6, lecture 2. Also plot the $log(N_r)$ vs. $log(r)$ graph.

(d) Bonus points: Use linear regression to find the constants $a$ and $b$ such that $log(N_r) = a + blog(r)$ best approximates the graph above. Plot the resulting line together with the previous graph.

As detailed in `cats.txt`, the Brown documents are spread across 15 categories, such as *news*, *government*, *fiction*, *romance*, etc. Create a train-test partition of the corpus as follows:

**Test:** The test corpus contains the first document that is listed in each category according to the classification in `cats.txt`. For example, *ca01*, *cb01*, *cc01* will belong to the test corpus.

**Train:** The train corpus contains all the remaining documents from the Brown corpus.

Use the first method that was discussed in class for dealing with out of vocabulary words. Preprocess the train and test corpora as follows:

1. Create a vocabulary $V$ containing the words that appear at least twice in the training corpus.
2. Find all *happax legomena* in the training corpus and replace them with a special <*unk*> token. Do the same for out of vocabulary words in the test corpus.
3. Convert all numbers to <*num*>.

Next, create three language models – a trigram model $LM_3$, a bigram model $LM_2$, and a unigram model $LM_1$ – using Katz backoff, where the discounted probabilities are computed based on add-$\delta$ smoothing (Lidstone's Law), choosing $\delta = 0.2$. The language models' probabilities are to be estimated on the training data.

(e) Let $mix$ be a mixing parameter, where $mix \in \{0\%, 0.01\%, 0.1\%, 1\%, 5\%, 10\%\}$. Given a $mix$ value, each word in the test data is chosen with likelihood $mix$ to be changed. If the word is chosen to be changed, change it with a randomly chosen word from the vocabulary. For each $mix$ value, run the experiment 10 times, and report the average perplexity and its standard deviation.

## 4 Authorship Attribution [100 points]

In this exercise, you will be using a subset of the literature from the Gutenberg project, available in `corpora/gutenberg`. Create the following training and test data:

1. AustenTrain = {austen-persuasion.txt, austen-sense.txt}
   AustenTest = {austen-emma.txt}
2. ChestertonTrain = {chesterton-thursday.txt, chesterton-brown.txt}
   ChestertonTest = {chesterton-ball.txt}
3. ShakespeareTrain = {shakespeare-macbeth.txt, shakespeare-caesar.txt}
   ShakespeareTest = {shakespeare-hamlet.txt}
4. BibleTest = {bible-kjv.txt}

Train a trigram language model for each training corpus, using Katz backoff, where the discounted probabilities are computed based on add-$\delta$ smoothing (Lidstone's Law), choosing $\delta = 0.2$. Create a vocabulary for each training dataset, using <*unk*> for hapax legomena. Report in a table the perplexity that each of the 3 language model obtains for each of the 4 test corpora. Is this approach useful for authorship attribution? If not, elaborate on possible changes that would improve its authorship attribution performance. Repeat the experiments, this time using the singleton unigrams, as suggested in exercise 4.10, page 122 in J&M.

## 5 Language Detection [100 points]

In this exercise, you will be using the 11 language corpora available in `corpora/europarl_raw`. For each language corpus, create training data from the first 9 documents, and test data from the last document (i.e. `ep-00-02-16.xx`). Tokenize the text and split into sentences using the Punkt sentence tokenizer implemented in NLTK. Convert to lowercase, erase numbers, and replace multiple spaces with just one space. Use the training data to build a bigram language model for each language – make sure that you use characters in n-gram events. Use a confusion matrix $P$ to report the language detection performance as follows:

- The entry $M(L_1, L_2)$ contains the percentage of sentences from the test corpus of language $L_2$ that were classified to belong to language $L_1$.
- To classify an arbitrary sentence $S$, compute the perplexity of each language model, and select the language that results in the lowest perplexity. Break ties randomly.

To deal with unseen characters: create an alphabet for each language from its training data; if the test sentence contains an out of alphabet token, report an infinite perplexity. Elaborate on other, better methods for dealing with unseen characters.

## 6 General Requirements

Make your language models case insensitive. Make sure that you include a detailed discussion of the results and your insights. Additional work is very likely to results in bonus points. Examples:

1. Any approach that leads to improvement in the performance;
2. Making the language models work with other smoothing techniques;
3. Running experiments with different orders for the N-gram models;
4. Experiment and compare with **neural language models**.

## 7 Submission

Please turn in a hard copy of your homework report at the beginning of class on the due date.

Electronically submit a directory that has your working code, and a concise README file describing these before class. Do not send NLTK code! Create a gzipped, tar ball archive of your directory, and upload it on Blackboard.

For example, if the name is John Williams, creating the archive can be done using the following commands:

```
> tar cvf williams_john.tar williams_john

> gzip williams_john.tar
```

These two steps will create the file 'williams_john.tar.gz' that you can upload on Blackboard.

Please observe the following when handing in homework:

1. Structure, indent, and format your code well.
2. Use adequate comments, both block and in-line to document your code.
3. Type and nicely format the project report, including discussion points, tables, graphs etc. so that it is presentable and easy to read.
4. Working code and/or correct answers is only one part of the assignment. The project report, including discussion of the specific issues which the assignment asks about, is also a very important part of the assignment. Take the time and space to make an adequate and clear project report. On the non-programming learning-theory assignment, clear and complete explanations and proofs of your results are as important as getting the right answer.