CMPS 143 - Assignment 1

(10 points)

Due Monday, January 14 11:55PM

1 Getting Started with NLTK

Before going further you should install NLTK 3.0, downloadable for free from http://www.nltk.org/install.html. Follow the instructions there to download the appropriate version for your platform. In this course we will use Python 3.4 *ONLY*, so you **MUST** download that version. Also, you **MUST** download NumPy – this is not optional as the website says.

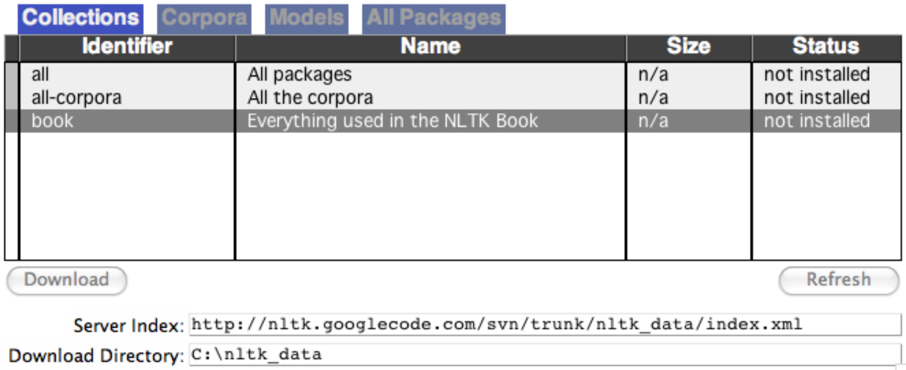
(*Note: Macs with Yosemite or higher have Python 2.7 as default in the terminal. You will need to run "*`sudo pip3 install -U nltk`*" instead of "*`sudo pip install -U nltk`*" in order to install NLTK for Python 3.4. You should make a separate installation rather than updating/replacing your system default Python version.*)

Once you've installed NLTK, start up the Python interpreter:

- Windows: find and open "Python 3.4 (command line - 32bit)" in your start menu.
- Mac: type "python3" and press enter in your terminal.

Install the data required for the book by typing the following two commands at the Python prompt, then selecting the "book" collection as shown in Figure 1 and pressing "Download".

```
>>> import nltk
>>> nltk.download()
```



Identifier	Name	Size	Status
all	All packages	n/a	not installed
all-corpora	All the corpora	n/a	not installed
book	Everything used in the NLTK Book	n/a	not installed

Server Index: http://nltk.googlecode.com/svn/trunk/nltk_data/index.xml

Download Directory: C:\nltk_data

Figure 1: Installing NLTK data.

2 Working with Corpora

In this assignment you will write a program to analyze two different narrative corpora and compare them. The first corpora is a collection of 26 Aesop's fables. The second corpus is a collection of 12 personal narratives told by ordinary people on their weblogs. Both corpora are provided as zip archives with each individual file containing a different fable or blog.

Your program should take one command line argument that specifies which corpus you are going to analyze (**fables** or **blogs**). The input file for the fables corpus is the zip archive containing the 26 fables as text (**fables.zip**). The input for the blogs corpus is the zip archive containing the 12 blogs as text (**blogs.zip**). Use the stub code `assignment1-stub-s15.py` to get started.

For each of the corpora your program should do the following:

1. Tokenization
   (a) Write the name of the corpus to stdout.
   (b) Delimit the sentences for each document in the corpus.
   (c) Tokenize the words in each sentence of each document.
   (d) Count the number of total words in the corpus and write the result to stdout.
2. Part-of-Speech
   (a) Apply the default part-of-speech tagger to each tokenized sentence.
   (b) Write a file named `CORPUS_NAME-pos.txt` that has each part-of-speech tagged sentence on a separate line and a blank newline separating documents. Where `CORPUS_NAME` is either `fables` or `blogs`. The format of the tagging should be a word-tag pair with a / in between. For example: `The/DT boy/NN jumped/VBD ./.`
3. Frequency
   (a) Write the vocabulary size of corpus to stdout. Please note that you should use the **lowercased word**.
   (b) Write the most frequent part-of-speech tag and its frequency to the stdout.
   (c) Find the frequency of each unique word (after **lowercasing**) using the FreqDist module and write the list in **decreasing order** to a file named `CORPUS_NAME-word-freq.txt`.
   (d) Find the frequency of each word given its part-of-speech tag. Use a conditional frequency distribution for this (`CondFreqDist`) where the first item in the pair is the part-of-speech and the second item is the **lowercased word**. Note, the part-of-speech tagger requires uppercase words and returns the word/tag pair in the inverse order of what we are asking here. Use the `tabulate()` method of the CondFreqDist class to write the results to a file named `CORPUS_NAME-pos-word-freq.txt`.
4. Similar Words
   (a) For the most frequent word in the **NN** (nouns), **VBD** (past-tense verbs), **JJ** (adjectives) and **RB** (adverbs) part-of-speech tags, find the most similar words using `Text.similar()`. Write the output to stdout (this will happen by default).
5. Collocations
   (a) Write the collocations to the stdout.

3 Output Results

In conclusion, if you follow the steps in Section 2, your program should output the following information in standard output for each corpus:

1. Corpus name:
2. Total words in the corpus:
3. Vocabulary size of the corpus:
4. The most frequent part-of-speech tag is __ with frequency __
5. The most frequent word in the POS (NN/VBD/JJ/RB, note that you need to output result for each of these POS) is __ and its similar words are: __
6. Collocations:

Your python program should also **output three files for each corpus**. For example, for fables corpus, it should output `fables-pos.txt`, `fables-word-freq.txt` and `fables-pos-word-freq.txt`.

What to Turn In

Turn in on Canvas a single zipped file including your Python program with all the output files.