LIN 353C: Introduction to Computational Linguistics, Spring 2017, Erk

Homework 5: Using Python dictionaries

Due: Tuesday March 28, 2017

This homework comes with a file, `lastname_firstname_hw5.py`. This is a (basically empty) file, called a *stub file*. Please record all the answers in the appropriate places of this file. Please make sure that you save the file in *plain text*, not something like the Word format.

For submission, please rename the file such that it reflects your name. So for example, if your name was "Alan Turing", you should rename it to

```
turing_alan_hw5.py
```

Also do not forget to put your name and EID in the second line of the file.

Please put your Python code into the answers file. You can omit statements that produced an error or that did not form part of the eventual solution, but please include all the Python code that formed part of your solution. **Please submit your homework solution electronically using Canvas.**

If any of these instructions do not make sense to you, please get in touch with the instructor right away.

A perfect solution to this homework will be worth *100* points.

In this homework you will use the "news" part of the Brown Corpus in the Natural Language Toolkit, more precisely a version of this corpus with part-of-speech tags (POS tags). In order to do that, you will need to install the NLTK data (if you have not done this already). Do this as follows:

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

Then follow the further instructions at `http://www.nltk.org/data`

Afterwards, you can access the POS-tagged "news" part of the Brown Corpus as follows:

```
import nltk
brown_news = nltk.corpus.brown.tagged_words(categories="news")
```

This gives you a list of tuples, to be more precise a list of pairs. The first member of each pair is a word, and the second member is its part-of-speech tag. Here are the first ten words and their tags:

```
>>> brown_news[:10]
[('The', 'AT'), ('Fulton', 'NP-TL'), ('County', 'NN-TL'),
('Grand', 'JJ-TL'), ('Jury', 'NN-TL'), ('said', 'VBD'),
('Friday', 'NR'), ('an', 'AT'), ('investigation', 'NN'),
('of', 'IN')]
```

To see a list of the part-of-speech tags used in this corpus, with examples, visit `http://www.scs.leeds.ac.uk/amalgam/tagsets/brown.html`

For all problems below, please map all words to lowercase in order not to distinguish "The" and "the". You can use the method `lower()` for this, as in:

```
>>> "The".lower()
'the'
```

1. **Word frequency and tag frequency (40 points).**

   What is the most frequent word, and what is the most frequent POS-tag in the “news” section of Brown? Do *not* use the NLTK data structures `nltk.FreqDist` and `nltk.ConditionalFreqDist`, or the Python `Counter`, for this problem. Just use Python dictionaries, along with the built-in function `sorted()`.

2. **Ambiguous words (40 points).**

   (a) How many words are ambiguous, in the sense that they appear with at least two different POS tags? Do *not* use the NLTK data structures `nltk.FreqDist` and `nltk.ConditionalFreqDist` for this problem, nor the Python `Counter`. Just use Python dictionaries.

   Note that you only want distinct tags, so a list like `["NP", "NN"]` rather than `["NP","NN","NN","NN","NP"]`. You can either make sure that you only save tags that you have not before seen with the word in question. Or you can use sets instead of lists. A set is similar to a list, but it never contains duplicates. You add to a set using `add()` rather than `append()` as for lists:

   ```
   >>> mylist = ["NP", "NP", "NN", "NN", "NN", "NP"]
   >>> myset = set(mylist)
   >>> myset
   set([’NP’, ’NN’])

   >>> myset.add(’NNS’)
   >>> myset
   set([’NNS’, ’NP’, ’NN’])
   ```

   (b) Which word has the greatest number of distinct tags, and how many distinct tags does it have? Again, do not use `nltk.FreqDist`, `nltk.ConditionalFreqDist`, or `Counter`, but Python dictionaries.

3. **Extracting examples for particular word/tag pairs (20 points).**

   For the word with the greatest number of distinct tags (the one from the previous problem), print an example of each tag: a sentence from the corpus containing that word/tag pair.

   For this problem, you need to access the Brown corpus in a different way than before. `brown_news` just contains a list of word/tag pairs, without sentence boundaries. But for this problem you need the sentence boundaries. You can access the Brown corpus *sentences* as follows:

```
brown_sents = nltk.corpus.brown.tagged_sents(categories="news")
```

   `brown_sents` will be a list of sentences, where each sentence is again a list of word/tag pairs. Here is the first one:

```
>>> brown_sents[0]
[('The', 'AT'), ('Fulton', 'NP-TL'), ('County', 'NN-TL'),
('Grand', 'JJ-TL'), ('Jury', 'NN-TL'), ('said', 'VBD'),
('Friday', 'NR'), ('an', 'AT'), ('investigation', 'NN'),
('of', 'IN'), ("Atlanta's", 'NP$'), ('recent', 'JJ'),
('primary', 'NN'), ('election', 'NN'), ('produced', 'VBD'),
('``', '``'), ('no', 'AT'), ('evidence', 'NN'), ("''", "''"),
('that', 'CS'), ('any', 'DTI'), ('irregularities', 'NNS'),
('took', 'VBD'), ('place', 'NN'), ('.', '.')]
```