You began using the Brown Corpus last week when you built your sentence segmenter and investigated coverage of \(n\)-grams collected from one genre when applied to another genre. This week, you will use the entire Brown Corpus to build a part-of-speech tagger.

To use NLTK, you first need to import nltk. To see the list of categories in the Brown corpus:

```python
nltk.corpus.brown.categories()
```

To see the list of files in a specific category or categories of the Brown corpus:

```python
nltk.corpus.brown.fileids(categories=['news'])  # one category
nltk.corpus.brown.fileids(categories=['government', 'lore'])  # multiple categories
```

To read the words and part-of-speech tags in a specific category or file:

```python
nltk.corpus.brown.tagged_words(categories=['news'])
nltk.corpus.brown.tagged_words(fileids=['cf26','cf27'])
```

For the first five questions, you will be using only the news category. However, question 6 will ask you to repeat the first five questions using the science fiction category, so be sure to make your code modular and don’t delete solutions to a previous question when moving onto the next question.

You should write a single Python program, stats.py, that will solve the first six questions. You will provide a writeup, writeup.txt, that contains the answers to each of these questions. (If you want, the output of your stats.py program can be your writeup for the first part. In that case, just paste the output into writeup.txt.)

As you begin answering the questions, you may want to maintain a diary of what you’re doing which will serve you well when it’s time to do your writeup. Be sure that your writeup specifies how to get the answer to each question, the answer to each question, and any additional questions or insights you had as you complete the assignment.

**Note:** Do not lowercase the text for any of the questions in this lab.

1. Ignoring part-of-speech tags for the moment, how many types and tokens, including punctuation, are there in the news category?

2. Before you solve this question computationally, write down an educated guess about what the ten most frequent words (including punctuation) will be. Once you’ve done that, report the ten most frequent words (inc. punctuation), along with their frequencies, as found in the news category. Does this list match your prior expectations? If not, why not?

3. List the most frequent part of speech tags, along with their frequencies, as found in the news category. NOTE: Many tags have -TL, -HL, -NC and/or FW- attached to them (e.g. RB-TL or FW-NN). You should remove these suffixes and prefixes from all tags (e.g. yielding just RB and NN) before continuing with the rest of the questions. It is easier to use the replace method on strings, but you are welcome to use regular expressions.
4. What are the five most frequent bigrams in the text? (HINT: You will find the nltk.bigrams function to be very helpful.) Note that some of the bigrams will include punctuation. Repeat this experiment by excluding all bigrams where one or both words contain punctuation. Describe what you found and whether or not this matched your expectations.

5. For each word type (not token), compute a mini-histogram of the parts-of-speech it was assigned. For example, the word stand is used as NN nine times, and as VB seven times. You should represent this in Python as a dictionary of dictionaries, so that hist['stand'] = {'NN': 9, 'VB': 7}. What percentage of the types and what percentage of the tokens have only one tag? What percentage of types (and tokens) have more than two tags? (Do not print the histogram, but you will need to use this histogram dictionary to answer question 7.)

6. Repeat the above questions using the science_fiction category. How similar or dissimilar are your results? What does this tell you?

For this next set of questions, you will be using two texts: a training text and a test text. The training text will be the first 33 files in the news category ('ca01'...'ca33'), and the test text will be the last 11 files in the news category ('ca34'...'ca44'):

from nltk.corpus import brown

fileids = brown.fileids(categories='news')
training_ids = fileids[:33]
test_ids = fileids[33:]
training = brown.tagged_words(fileids=training_ids)
test = brown.tagged_words(fileids=test_ids)

You are going to try and “tag” the test text using statistics gathered from the training text. Of course, the test text already has part of speech tags, but you’ll pretend they aren’t there, try to predict them, and then see how well you did. Put your answers to these questions in pos.py. You may want to import the stats.py file you already created so that you can re-use your code as necessary. In particular, you’ll want to use the tag-cleaning code from question 3 and the histogram code from question 5. You will continue to add to the writeup.txt file.

7. For each word type in the training text, compute the maximum likelihood estimation of $P(T = t | W = w)$. This represents the probability that the tag $t$ is the correct tag for the word, given that the word is $w$. You can calculate this by dividing the number of times word $w$ is used as tag $t$ by the number of times word $w$ occurs. So, $P(T = "NN" | W = "stand") = 9/16$. You don’t need to do anything with this result just yet, but you’ll need it for question 8. You should be able to derive this result from a dictionary like the one you created in question 5. (Just be sure the dictionary you are using for this question only contains words and tags from the first 33 files, not the entire news category.)

8. For each word token in the test text, assign the most likely part of speech tag using $P(T = t | W = w)$. You can break ties arbitrarily. Some words will occur in the test text that did not occur in the training text.

- What tag should assign to these previously unseen words? Should you assign the most frequent tag from the training text? Should you assign the tag randomly? What other choices might you have? (Add your to your writeup, of course. You’ll explore this further in question 10.)

The output of your program should be four tab-separated columns of the following format:

    word  guess  truth  result
Here, word is a word from the test corpus, guess is your guess of the part of speech tag, truth is the actual part of speech tag (taken from the test corpus), and result is either 0 or 1, depending on if you were incorrect or correct.

You should also output the percentage of the words that you tagged correctly.

9. Look through the resulting output file (and/or write code to analyze the results of your tagger). Comment on your errors in your writeup. Did you do better or worse than you expected? Why do you think this is? What were your most common mistakes (e.g. Did you get a particular word wrong a lot? Did you mistake one part-of-speech tag for another a lot?)

10. How well did your tagger perform on words in the test corpus that didn’t occur in the training corpus? What method did you use originally to choose the correct tag? Try some other methods and see how well they work. (Note that this will require you to pick out those words from your results file, but you can do that, right?)

11. Repeat question 9, replacing only the test text with texts from another category of your choosing. You can try something you think will do well, or something you think won’t do well (or you can try both). Be sure you say what you think will happen in your writeup, and then say how well the results matched your expectations, and why.