# Part I

## Problem 21

Write a function `unknown()` that takes a URL as its argument, and returns a list of unknown words that occur on that webpage. In order to do this, extract all substrings consisting of lowercase letters (using `re.findall()`) and remove any items from this set that occur in the Words Corpus (`nltk.corpus.words`). Try to categorize these words manually and discuss your findings.

Code:

```
def calc_unknown(url):
    ssl._create_default_https_context =
        ssl._create_unverified_context
    response = request.urlopen(url)
    raw = response.read().decode('utf8')
    tokens = re.findall(r'/\b[a-z]+\b/', raw)
    print(tokens)
    known_words = [w for w in nltk.corpus.words.words('en') if
        w.islower()]
    unknown = list(set(tokens).difference(known_words))

    print("Unknown words: %s" % unknown)
    print("Length of raw response: %d" % len(raw))
    print("Number of unknown words in of raw response: %d" %
        len(unknown))
```

Output:

```
$ python3 unknown.py https://www.nltk.org/book/ch03.html
['/widget/', '/widget/', '/doc/', '/catalog/', '/files/',
    '/hi/', '/software/', '/pypi/', '/nll/', '/gutenberg/',
    '/images/', '/images/', '/images/', '/images/',
    '/images/', '/images/', '/wiki/', '/images/', '/images/',
    '/pipermail/', '/images/', '/images/', '/images/',
    '/images/', '/python/', '/regex/', '/text/', '/howto/',
    '/video/', '/articles/', '/wiki/', '/wiki/',
    '/languagelog/', '/ewan/', '/licenses/', '/us/',
    '/licenses/', '/us/']
```

```
Unknown words: ['/articles/', '/ewan/', '/images/',
    '/gutenberg/', '/regex/', '/files/', '/video/',
    '/software/', '/nll/', '/howto/', '/doc/', '/pypi/',
    '/catalog/', '/languagelog/', '/pipermail/', '/hi/',
    '/wiki/', '/text/', '/widget/', '/us/', '/licenses/',
    '/python/']
Length of raw response: 292794
Number of unknown words in of raw response: 22
```

Observations: the unknown words in Chapter 3 of the NLTK text book appear to be words related to python or coding in general. A few words (specifically 'software', 'us', and 'images') surprised be when they were categorized as 'unknown', because they seem common enough to appear in the words corpus.

## Problem 22

Examine the results of processing the URL http://news.bbc.co.uk/ using the regular expressions suggested above. You will see that there is still a fair amount of non-textual data there, particularly Javascript commands. You may also find that sentence breaks have not been properly preserved. Define further regular expressions that improve the extraction of text from this web page.

Results:

```
$ python3 unknown.py http://news.bbc.co.uk/
Length of raw response: 433939
Number of unknown words in of raw response: 50
```

Observations: Most uknown words seem related to javascript libraries named by unique, coding specific words.

## Problem 29

Readability measures are used to score the reading difficulty of a text, for the purposes of selecting texts of appropriate difficulty for language learners. Let us define `w` to be the average number of letters per word, and `s` to be the average number of words per sentence, in a given text. The Automated Readability Index (ARI) of the text is defined to be: 4.71 w + 0.5 s - 21.43. Compute the ARI score for various sections of the Brown Corpus, including section f (lore) and j (learned). Make use of the fact that `nltk.corpus.brown.words()` produces a sequence of words, while `nltk.corpus.brown.sents()` produces a sequence of sentences.

Code:

```
def readability(input):
   letters = brown.raw(categories=input)
   words = brown.words(categories=input)
   sentences = brown.sents(categories=input)

   letters_per_word = len(letters) / len(words)
   words_per_sentence = len(words) / len(sentences)

   ari_score = 4.71 * letters_per_word + 0.5 *
      words_per_sentence - 21.43
   print("Letters per word: %s" % letters_per_word)
   print("Words per sentence: %d" % words_per_sentence)
   print("ARI Score: %d" % ari_score)
```

Output:

```
$ python3 readability.py lore
Letters per word: 8.60259839164453
Words per sentence: 22
ARI Score: 30
```

```
$ python3 readability.py learned
Letters per word: 8.828718771991555
Words per sentence: 23
ARI Score: 31
```

### Problem 42

Use WordNet to create a semantic index for a text collection. Extend the concordance search program in 3.6, indexing each word using the offset of its first synset, e.g. `wn.synsets('dog')[0].offset` (and optionally the offset of some of its ancestors in the hypernym hierarchy).

## Part II

Perform exercise 42 (above) on the raw text of the TEI XML file. You can find any TEI XML text but you could start with an English translation.

## Part III

Identify a possible available textual source that you might use for a project and identify the possible challenges that you will face in preprocessing the text.

I think the Kings James Bible would be an interesting textual source for one of the class projects. This version is freely available online, and would not require any Copyright or License-based hoops to jump through. However, I envision the source needing some processing regarding chapters, verses,, sections, books, natural breaks within the source that would allow for interesting comparisons and visualizations, much like the "Readability visualization" we saw in class two weeks ago.