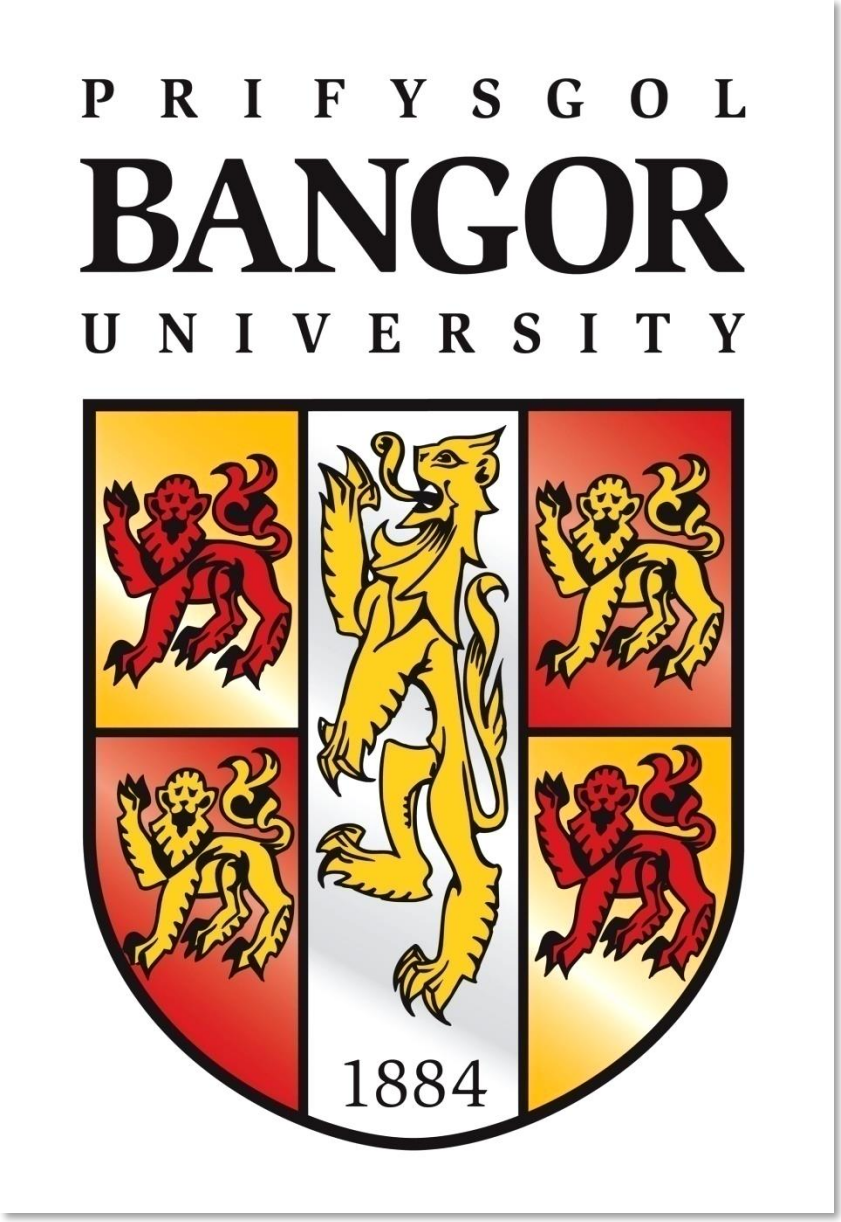


ICP-4721: Natural Language Processing

Laboratory 2: Introduction to the NLTK, Regular Expressions and Part of Speech Tagging

Bill Teahan

EXERCISES

For this module's laboratory sessions, you will need to produce a report that includes the answers to each of the lab exercises and then submit the report to Blackboard. You will have two weeks to complete the report. Marks (out of 100) are shown at the top of each exercise. Try to complete the exercises within the two lab sessions (100 marks for 100 minutes so 1 mark per minute) otherwise you will need to complete the exercises in your own time. There will be four labs for this module, so as the lab weighting is 50%, this means that each lab report is worth 12.5%. Feedback on the previous week's lab will be provided the following week so have your lab report and programs ready for checking.

Please do not include the questions in the lab report – just the answers – as this will affect the similarity index for your report for the TurnItIn plagiarism software.

SUBMISSION DEADLINE

All lab work will be due at midnight on Tuesday of the week when it is due (i.e. the night before the next lab session).

DOCUMENTATION

Official Python website – Python.org:

http://www.python.org/

For further information and documentation, you can also try the Python Wiki:

http://wiki.python.org/moin/

O'Reilly Media – "Learning Python" :

http://oreilly.com/catalog/9780596513986/

THE NLTK MODULE

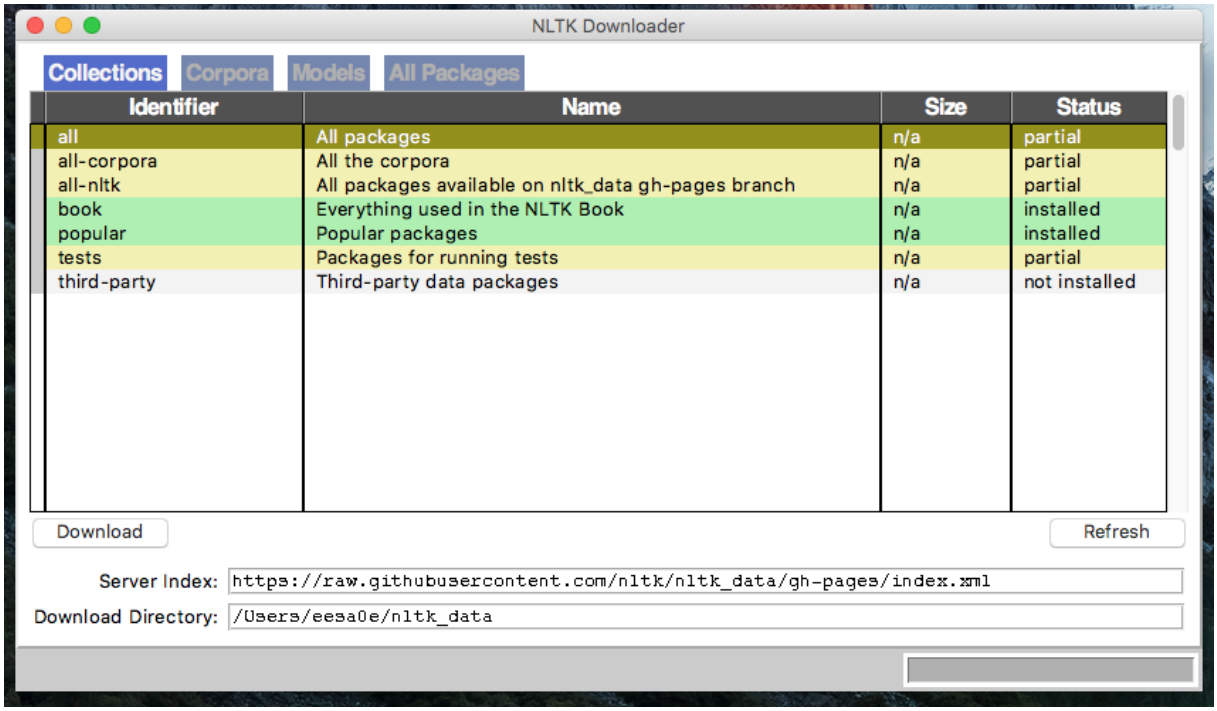
The NLTK is a package that has been installed on the lab machines. To install the package at home, follow the instructions provided at: https://www.nltk.org/install.html

Start up the Python interpreter, and install the data required for the book by typing the following two commands at the Python prompt, then selecting the book collection as shown below:

```
>>> import nltk

>>> nltk.download()
```

The following screen should appear:



Be aware: This may not work (if you are using a Mac)! If you get an SSL error, use the following link to download a workaround:

https://stackoverflow.com/questions/38916452/nltk-download-ssl-certificate-verify-failed

The specific code you need to use is as follows:

```
import nltk
import ssl

try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context
```

```
nltk.download()
```

Once you get the `download` app to work, select to "book" collection to download. Once the data is downloaded to your machine, you can load some of it using the Python interpreter. To load some already loaded texts:

```
>>> from nltk.book import *
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
>>>
```

Any time we want to find out about these texts, we just have to enter their names at the Python prompt:

```
>>> text1
<Text: Moby Dick by Herman Melville 1851>
>>> text2
<Text: Sense and Sensibility by Jane Austen 1811>
>>>
```

There are many ways to examine the context of a text apart from simply reading it. A concordance view shows us every occurrence of a given word, together with some context.

```
>>> text1.concordance("monstrous")
Building index...
Displaying 11 of 11 matches:
ong the former , one was of a most monstrous size . ... This came towards us ,
ON OF THE PSALMS . " Touching that monstrous bulk of the whale or ork we have r
ll over with a heathenish array of monstrous clubs and spears . Some were thick
d as you gazed , and wondered what monstrous cannibal and savage could ever hav
that has survived the flood ; most monstrous and most mountainous ! That Himmal
they might scout at Moby Dick as a monstrous fable , or still worse and more de
th of Radney .'" CHAPTER 55 Of the monstrous Pictures of Whales . I shall ere l
ing Scenes . In connexion with the monstrous pictures of whales , I am strongly
ere to enter upon those still more monstrous stories of them which are to be fo
ght have been rummaged out of this monstrous cabinet there is no telling . But
of Whale - Bones ; for Whales of a monstrous size are oftentimes cast up dead u
>>>
```

Now, just for fun, let's try generating some random text in the various styles we have just seen. To do this, we type the name of the text followed by the term generate. (We need to include the parentheses, but there's nothing that goes between them.)

```
>>> text3.generate()
In the beginning of his brother is a hairy man , whose top may reach
unto heaven ; and ye shall sow the land of Egypt there was no bread in
all that he was taken out of the month , upon the earth . So shall thy
wages be ? And they made their father ; and Isaac was old , and kissed
him : and Laban with his cattle in the midst of the hands of Esau thy
first born , and Phichol the chief butler unto his son Isaac , she
>>>
```

A text in NLTK is a list of words.

```
>>> text4[173]
'awaken'
>>>
```

We can do the converse; given a word, find the index of when it first occurs:

```
>>> text4.index('awaken')
173
>>>
```

EXERCISE 1 – BOOK WORM (10 MARKS)

Play around with the NLTK Book Collection. Find out what's available and use several methods (e.g. slices) to extract the text in one or more books. In particular, also find out the opening 100 words to the book *Monty Python and the Holy Grail*.

Post your code, and sample output along with a description into the report.

Note: On some computers, it may take a while to download the collection. If this is the case, please ask for help or try installing the NLTK and loading the collection on your own machine.

PLAYING WITH WORDS IN THE NLTK

The NLTK includes some corpora that are nothing more than wordlists. The Words Corpus is the /usr/share/dict/words file from Unix, used by some spell checkers. You can load it using the following command:

```
>>> import nltk
>>> words = nltk.corpus.words.words('en')
>>> words[:10] # prints out first 10 words in words dictionary
['A', 'a', 'aa', 'aal', 'aalii', 'Aani', 'aardvark', 'aardwolf',
'Aaron']
```

We can start to have some fun with words if we combine the words dictionary with Python's regular expression module, re, along with using list comprehensions. For example, if we wish to find all words ending in "ed", we can use the regular expression "ed$", where the "$" is a meta-character indicating the end of the string. (When we have loaded the words dictionary, the end of the string is simply the end of each word.)

```
>>> edwords = [w for w in words if re.search('ed$', w)]
```

```
>>> edwords
['abaissed', 'abandoned', 'abased', 'abashed', …]
```

EXERCISE 2 – CHEATING AT SCRABBLE (20 MARKS)

Use the above method to produce lists of words that match the following patterns:

- three letter words that contain only vowels **[3 marks];**
- three letter words that start with a vowel and end with a vowel, but have a consonant in the middle **[3 marks];**
- four letter words that have a consonant at the second letter but only vowels elsewhere **[3 marks];**
- five letter words that have a consonant at the second letter but only vowels elsewhere **[3 marks];**
- words of up to length 7 that contain the letters a, e, f, g, h, i or n **[4 marks];**
- words (called 'textonyms') that you can produce on a mobile phone by pressing the 4 digit sequence 3456 **[4 marks].**

In your report, include all your Python code and the output produced.

Note that for your assignment one of the ways you can recognize what is being typed in is using regular expressions.

USING REGULAR EXPRESSIONS FOR YOUR ASSIGNMENT

EXERCISE 3 – ALEXA, SIRI AND OK GOOGLE (35 MARKS)

Devise regular expressions for recognizing simple commands that could be recognized by a virtual assistant (e.g. like Alexa, Siri and Google Assistant):

- "Alexa: …", "Siri: …", "OK Google: …"
- "Alexa: Add to my shopping list: catfood, shampoo, …"
- "Siri: Play the following song: Halo by Beyonce"
- "OK Google: Multiply the following numbers: 2345 and 453456"

In your report, include all your Python code, sample input used in testing and the output produced.

PART OF SPEECH TAGGING IN THE NLTK

For your assignment, you will also have to perform more sophisticated text processing using part of speech tagging. A part-of-speech tagger, or POS-tagger, processes a sequence of words, and attaches a part of speech tag to each word (don't forget to `import nltk`):

```
>>> text = nltk.word_tokenize("And now for something completely different")
>>> nltk.pos_tag(text)
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'),
('completely', 'RB'), ('different', 'JJ')]
```

Here we see that 'and' is CC, a coordinating conjunction; 'now' and 'completely' are RB, or adverbs; 'for' is IN, a preposition; 'something' is NN, a noun; and 'different' is JJ, an adjective.

Let's look at another example, this time including some homonyms:

```
>>> text = nltk.word_tokenize("They refuse to permit us to obtain the refuse permit")
>>> nltk.pos_tag(text)
[('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'), ('us', 'PRP'),
('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'), ('refuse', 'NN'), ('permit', 'NN')]
```

Notice that 'refuse' and 'permit' both appear as a present tense verb (VBP) and a noun (NN). E.g. 'refUSE' is a verb meaning "deny," while 'REFuse' is a noun meaning "trash" (i.e. they are not homophones). Thus, we need to know which word is being used in order to pronounce the text correctly. (For this reason, text-to-speech systems usually perform POS-tagging.)

By convention in NLTK, a tagged token is represented using a tuple consisting of the token and the tag. We can create one of these special tuples from the standard string representation of a tagged token, using the function str2tuple():

```
>>> tagged_token = nltk.tag.str2tuple('fly/NN')
>>> tagged_token
('fly', 'NN')
>>> tagged_token[0]
'fly'
>>> tagged_token[1]
'NN'
```

EXERCISE 4 – NOUN PHRASES (35 MARKS)

Using regular expressions and part of speech tagging, define a function in Python that will take a text string as an input parameter, and return the first noun phrase contained within the string as output. E.g.

Input = “Should all good men to come to the aid of their party?”

Output = “all good men”

Hint: Simple regular expressions of tag sequences such as DT* (JJ*) NN* will match most noun phrases.

In your report, include all your Python code, sample input used in testing and the output produced. Be sure to test your function with at least the following strings:

- “Should all good men to come to the aid of their party?”
- “And now is the time for all good men to come to the aid of their party.”
- “… and for all good men to come to the aid of their party.”
- “Come to the aid of your party.”
- “Your party needs you.”