Project Text Analysis – Assignment 2

Malvina Nissim
m.nissim@rug.nl

29 April 2019

The aim of this assignment is to get beyond text manipulation at the word level using `nltk`. You will have to use several `nltk` functions, whose theory we have seen in class. Some of the functions we have seen in the slides, some others you will have to figure out yourself (please, refer to the `nltk` online material). Please, embed your functions in proper python code.

- use Nestor for submission
- **Deadline**: 6 May 2019, 23:59
- hand in three files, two `.py` and one `.txt/.pdf`:
  - one python script that generates all the information required in Exercise 1
  - one python script that generates all the information required in Exercise 2
  - a `.txt` or `.pdf` file (please no `.docx`) with all comments/answers to the questions in Exercise 2. If you want, you can also included comments about Exercise 1. If your scripts generate (almost) all the information you need in a nice informative way you can just copy and paste it from your code's output, of course. Add comments where appropriate.

Exercise 1 – Bigrams and Collocations

For this part of the assignment you will work with collocations, i.e. significant bigrams. The text file you will have to work with as input is loaded on Nestor, and it's the same that you used for Assignment 1. And as you've done for Assignment 1, you will have to start from scratch with the text that it's provided for you. After doing some manipulation, you will have to answer some questions.

Note: all of the tasks that you have to perform for this assignment must be run within a **single** python script. It's nice if your script prints a few headings/comments referring to what it's doing.

1. work with collocations, and find out:
   (a) the most likely 20 collocations using PMI

(b) the most likely 20 collocations using χ^2

(c) just looking at them, there are surely differences in the rankings. Can you provide an explanation? You can also compare what you get to what you got just by taking the top 20 bigrams (without any association measure — `raw_freq`, see also code below)

(d) **extra: calculate the Spearman's coefficient on the different collocation ranks (item 6 at link given in "hint" box here below)**

hint

To execute this part of the assignment you can refer to information provided here: `http://www.nltk.org/howto/collocations.html` (items 1–5)

sample code you can use and modify:

```
>>> import nltk
>>> from nltk.collocations import *
>>> text = open('holmes.txt').read()
>>> tokens = nltk.wordpunct_tokenize(text)
>>> bigram_measures = nltk.collocations.BigramAssocMeasures()
>>> finder = BigramCollocationFinder.from_words(tokens)
>>> scored = finder.score_ngrams(bigram_measures.raw_freq)
>>> sorted = sorted(bigram for bigram, score in scored)
>>> print(sorted)
```

2. Explain why

$$\chi^2 = \sum_{ij} \frac{(O_{ij}-E_{ij})^2}{E_{ij}}$$

(which is the general formula that we have seen in class written as $\chi^2 = \sum \frac{(f_o-f_e)^2}{f_e}$, and where i ranges over rows of the table, j ranges over columns, O stands for observed, and E stands for expected frequencies) can be re-written as:

$$\chi^2 = \frac{N(O_{11}O_{22}-O_{12}O_{21})^2}{(O_{11}+O_{12})(O_{11}+O_{21})(O_{12}+O_{22})(O_{21}+O_{22})}$$

where 1 and 2 stand for $word1$ and $word2$ or more simply, as we have seen, $w1$ and $w2$ (think in terms of contingency tables).

Exercise 2 – Part-of-speech tagging

The first part of this exercise asks you to first get acquainted manually with POS tags, and then to answer a set of questions by writing some code that exploits existing information about POS-tags in the Brown corpus. In the last part you will have to pos tag some raw text.

hint

if you are stuck, you can find some help in 5.2 here: `http://www.nltk.org/book/ch05.html`. Also, note that you can get info about a POS tag by using: `nltk.help.brown tagset(RB)`, also using a regular expression, e.g. `nltk.help.brown tagset(NN.*)`

1. Manually POS-tag this sentence:

   Peter really liked the movies and warm pop-corn . He would never bring Mira with him, though .

   three times, using:

   - The Penn Treebank POS tagset (`http://www.comp.leeds.ac.uk/ccalas/tagsets/upenn.html`)
   - The Brown Corpus tagset (`http://en.wikipedia.org/wiki/Brown_Corpus`)
   - NLTK's *universal* tagset (12 tags) (`http://www.nltk.org/api/nltk.tag.html#module-nltk.tag.mapping` at this link, if you are interested, you can find information about the mapping from other tagsets)

2. Answer the following questions about the POS-tagged "mystery" portion of the Brown Corpus. You can load the data as follows. Make sure to get the tagged words and the tagged sentences.

```
br_tw = nltk.corpus.brown.tagged_words(categories='mystery')
br_ts = nltk.corpus.brown.tagged_sents(categories='mystery')
```

   Make sure you use the Brown Corpus tagset, not the universal one. When citing a POS, make sure to give its **description** (e.g. 'plural noun') along with its **label** ('NNS'). This will help you getting acquainted with notions and terminology. POS information is the base for all further processing, so it's important you really get to know them!

   (a) how many words and sentences are there?

   (b) what is the 50th word and its POS? How about the 75th?

   (c) how many different POS tags are represented in this Brown category?

   (d) what are the top 15 words and their counts?

   (e) what are the top 15 POS tags and their counts?

(f) what is the most frequent POS tag in the 20th sentence? And in the 40th?

(g) what is the most frequent adverb?

(h) what is the most frequent adjective?

(i) consider the word 'so', which is ambiguous in its part-of-speech. In fact, it can take on three distinct parts-of-speech. What are they?

(j) among the three, which POS is most frequent for 'so'?

(k) for each of the three parts-of-speech for 'so', give an example sentence where it is used as the POS.

(l) for each of 'so's potential parts-of-speech, find out:

- the most likely POS preceding it
- the most likely POS following it

3. What if our text isn't tagged for parts-of-speech yet? We can do it ourselves! There is a function in `nltk`, namely `nltk.pos_tag()`, that will let you do that. Remember that first you will have to tokenise your data. Write a few lines that will pos-tag the whole of `holmes.txt`.

   You're creating a text processing pipeline!

4. Get collocations for POS tags, choosing whether you want to use PMI or χ^2, and rank the top 5 significant bigrams you get out (your script should specify this, and your report too). Please, answer these questions:

   - do they look interesting at all?
   - do they differ from the top 5 POS bigrams ordered by raw frequencies?
   - what could they be used for?