INF5830, 2015, Obligatory assignment 1

Deadline: Sept. 18, 18.00
To be delivered in devilry

Exercise 1 – Probabilities and Python

When we apply probabilities and statistics we will mainly use packages with built-in functions. But to get a better understanding of what we are doing, it is good training to implement some of it ourselves. We will in this exercise use basic Python and not packages like math, numpy or scipy. We will then later on compare our implementations with these packages.

a) Implement a function factor(n) which returns the factorial of n when n is an integer ≥ 0. (i.e. f(n) = n! = 1*2*…*n, and f(0)=1.)
b) Implement a function binom(m, n) which to two integers where $n \geq m \geq 0$ returns $\binom{n}{m}$.
c) Implement a function binom_pmf(k, n, p) where k and n are integers, where $n \geq k \geq 0$, and p is a real $1 \geq p \geq 0$. The function returns the probability mass function at k for the binomial distribution of n individuals with probability p.
d) Implement a function binom_cdf(k, n, p) with the same arguments which returns the value of the cumulative distribution function at k.
e) Fix n=8 and p=1/2 and calculate the pmf and cdf for k=0, 1, 2, …, 8. Report the numbers in a table. This corresponds to flipping a fair coin 8 times.
f) Repeat similarly for n=5 and p=1/6. This corresponds to throwing a fair dice and counting 6s as success.

   Observe that 1/6 in python will return 0. You might either use 1.0/6 or
   from __future__ import division
   before you start

Exercise 2 – Python library: math

Python comes with a standard library with several useful modules. One such module is *math* which contains a collection of useful mathematical functions, e.g. the factorial function

a) Try
   - import math
   - math.factorial(13)

   See that you get the same results as with your own
   - factor(n)

   Use math.<tab> to get an impression of which functions are available in math. We will in particular later on make use of exp and the log-functions.
   (No delivery at this point.)

Exercise 3 – Python library: random

Computers are deterministic. They do not act randomly. However, they can simulate randomness and act so-called pseudo randomly. There is a module *random* in the standard library with several useful functions.

a. In particular, the function random.random() returns a real number between 0 and 1. We can use this e.g. as follows.

```
def bernoulli(p):
  if random.random() < p:
    return 1
  else:
    return 0
```

   What does this function simulate? Run bernoulli(0.5) 10 times and record the results.

b. We will then see what happens when we perform n Bernoulli trials (flip the coin or throw the dice n times). Make a function bin_exper(n, p) which performs n random Bernoulli experiments with probability p and return the number of successes. Run bin_exper(10, 0.5) ten times and report the results.
c. We will inspect the effect of running an experiment many times and taking the averages. Make a function bin_freqs(m, n, p) which runs bin_exper(n, p) m many times and returns the relative frequencies of k successes for k = 0, 1, …, n.
d. Fix p=0.5 and n=8 and see what happens when m varies. Run bin_freqs for m= 4, 10, 100, 1000, 10000. Report the results in a (6*9) table where you also include the values for the theoretical distribution binom_pmf(k, n, p) from exercise (1)
e. To familiarize yourself a little more with random try the following

```
>>> a = range(100)
>>> random.choice(a)
>>> random.choice(a)
>>> random.sample(a,10)
>>> random.sample(a,10)
>>> random.shuffle(a)
>>> a
```

Make sure you understand the commands. (No deliveries at this point).

Exercise 4 – Conditional frequency distributions

The NLTK book, chapter 2, has an example in section 2.1 in the paragraph Brown Corpus where they compare the use of modals across different genres. We will conduct a similar experiment, but we will instead inspect the differences in gender. We are in particular interested in to which degree the different genres use the masculine pronouns (he, him) or the feminine pronouns (she, her).

a. Conduct a similar experiment as the one mentioned above with the genres: *news, religion, government, fiction, romance* as conditions, and the words: *he, she, her, him*. Make a table and deliver code and table.

b. Do you see any interesting differences between genres?
c. The experiment reveals not only differences between the genders but also between the cases, where *he, she* have *subjective* case, and *him, her* have *objective* case. We will explore this further. We will consider the whole Brown corpus. We will check how case varies with gender. We will do this by constructing a conditional frequency distribution where we use gender as condition and for each gender count the occurrences of subjective and objective case. Report the results in a two by two table. Deliver table and code.
(Help: You may read more about conditional frequency distributions in section 2.2 of the NLTK book.)
d. Answer in 5-10 lines: What does this experiment reveal about language and culture?

Exercise 5 – Downloading texts and Zipf's law

In this exercise we will consider Zipf's law which is explained in exercise 23 in NLTK chapter 2. You may also consult Wikipedia. We will use the text *Tom Sawyer* as that is also used in the book *Foundations of Statistical Natural Language Processing* for studying Zipf's law.

a. First you need to get hold of the text. It can be downloaded from project Gutenberg as explained in section 3.1 in the NLTK book.
b. Then you have to do some clean up. For example, there might be additional headers in the text which are not part of the text itself.
c. You can then extract the words. Explain the steps you take here and in point (b) above. (You may use nltk.word_tokenize().)
d. Use the nltk.FreqDist() to count the words. Report the 20 most frequent words in a table with their absolute frequencies.
(Observe that in the NLTK book first ed. FreqDist.keys() are sorted by frequencies. Not anymore. Use FreqDist.most_common()).
e. Consider the frequencies of frequencies. How many words occur only 1 time? How many words occur n times, etc. for n = 1, 2, …, 10; how many words have between 11 and 50 occurrences; how many have 51-100 occurrences; and how many have more than 100 occurrences? Report in a table!
f. Let r be the frequency rank for each word and n its frequency. According to Zipf's law r*n should be nearly constant. Calculate r*n for the 20 most frequent words and report in a table? How well does this fit Zipf's law?

The end