# LIN 353C: Introduction to Computational Linguistics, Spring 2017, Erk

## Homework 2: Finite state transducers, and first steps in Python

## Due: Thursday Feb 9, 2017

This homework comes with a file, `lastname_firstname_hw2.py`. This is a (basically empty) file. Please record all the answers in the appropriate places of this file. Please make sure that you save the file in *plain text*, not something like the Word format. Ask your instructor if you are having trouble saving to plain text from your text processing software.

For submission, please rename the file such that it reflects your last name. So for example, if your name was Antonio Zampolli, you should rename it to

```
zampolli_antonio_hw2.py
```

Please also remember to put your name in the 2nd line of the file, so that we know whose homework we're looking at when we print out your answer.

For the part of the homework that requires you to write Python code, we need to see the code, but not necessarily the output that the code produced (unless the problem explicitly asks for it). Please put your Python code into `lastname_firstname_hw2.py`. You can omit statements that produced an error or that did not form part of the eventual solution, but please include all the Python code that formed part of your solution.

**If any of these instructions do not make sense to you, please get in touch with the instructor right away.**

The Python Library Reference contains lists of methods that are available for different datatypes. For strings, you find the relevant information at `https://docs.python.org/3.5/library/stdtypes.html#text-sequence-type-str`

Alternatively, you can type

```
help(str)
```

in a Python shell to see a list of methods that are available for strings. Note that there are more methods than fit on a screen. Hit "space" to see the next screen of methods. Hit "q" to exit the help listing. Here is how to read, for example, the entry for `count`:

```
help(str.count)
```

Please use Python 3, and in particular the Python string methods, to solve the following problems.

A perfect solution to this homework will be worth *100* points.

1. **A finite state transducer (20 points)**

   In class, we discussed a finite state transducer that handles the orthographic rule that for nouns ending in -s, -x or -c, an "e" is inserted before the plural morpheme "ˆs". (Remember that in the finite state transducers we discussed, the symbol ˆwas used to indicate a morpheme boundary, that is, a boundary between an affix and a stem, or between two affixes, and the symbol # was used to mark the end of a word.) That transducer is given in the Jurafsky and Martin book in Figure 3.17, page 64.

   For this problem, you will write a finite-state transducer for another orthographic rule: In English, if a verb ends in ⟨vowel⟩*c*, add a -k before suffixes -ed and -ing (but not before -s). So from "panic" you go to "panicked", not "paniced". And you get "panics" and not "panicks".

   Write a finite state transducer that takes an intermediate-level representation, either

   panic^ed#

   or

   panic^ing#

   or "panic" followed by anything else, and maps it to a surface-level representation. In the first case, it should produce "panicked", in the second case "panicking", and if it is "panic" followed by anything else, it should be left unchanged (except that ^ should be removed).

   Your transducer can be nondeterministic. I would suggest that you build your transducer based on the example of the one in Figure 3.17 of the Jurafsky and Martin book.

   As in the transducer in Figure 3.17, you can leave the "#" symbol undeleted, if you like.

2. **Variable names in Python (10 points).**

   Which of the following are *not* valid variable names in Python? Give a short explanation why.

   (a) this_is_a_variable

   (b) mylist123

   (c) my-string

   (d) 123list

   (e) def

   (f) sum

   (g) VAR

   (h) my string

   (i) another_variable

   (j) var_1005

   **Extra credit question (2 points).** There is also one that is a valid variable name in Python, but you shouldn't use it anyway. Which is it, and why is it a good idea not to assign something to it?

3. **Removing punctuation (20 pts.)**

   Here is a text passage from H.G. Wells, *The War of the Worlds* (from Project Gutenberg, `http://www.gutenberg.org/etext/36`). This is also available with this homework as `wells.txt` in case that should be easier for copying and pasting.

   > After the glimpse I had had of the Martians emerging from the cylinder in which they had come to the earth from their planet, a kind of fascination paralysed my actions. I remained standing knee-deep in the heather, staring at the mound that hid them. I was a battleground of fear and curiosity.
   >
   > I did not dare to go back towards the pit, but I felt a passionate longing to peer into it. I began walking, therefore, in a big curve, seeking some point of vantage and continually looking at the sand heaps that hid these new-comers to our earth. Once a leash of thin black whips, like the arms of an octopus, flashed across the sunset and was immediately withdrawn, and afterwards a thin rod rose up, joint by joint, bearing at its apex a circular disk that spun with a wobbling motion. What could be going on there?

   For this problem, please split the passage into words (you can use the Python string method `split` for this), then remove punctuation from each word in the passage. That is, a string "planet," should be converted to "planet". You do not need to keep the punctuation in a separate string. For this problem it is fine if you just remove it. You do not need to remove punctuation from the middle of strings, that is, you do not need to transform the string "new-comers" in any way. Your result will be a list of strings.

4. **Counting words (10 pts.)**

   In that same passage from the *War of the Worlds*, how many words are there that end with the letters "ing"? Note that now we are looking for words ending in "ing", not just occurrences of the letter sequence "ing" anywhere in a word (so the string "brings" would not count). Please make sure that you also catch words that end in "ing" but are followed by punctuation, for example by using the result of the previous problem. (But if you did not solve the previous problem, it will be fine if you find some other way of taking punctuation into account.)

5. **Stemming (20 pts.)**

   A *stemmer* is a piece of software that removes affixes from words to get you a rough approximation of their stems. For example, the NLTK implementation of the Porter stemmer changes the War of the Worlds passage from above to this (after removal of punctuation):

   > After the glimps I had had of the Martian emerg from the cylind in which they had come to the earth from their planet a kind of fascin paralys my action I remain stand knee-deep in the heather stare at the mound that hid them I wa a battleground of fear and curios I did not dare to go back toward the pit but I felt a passion long to peer into it I began walk therefor in a big curv seek some point of vantag and continu look at the sand heap that hid these new-com to our earth Onc a leash of thin black whip like the arm of an octopu flash across the sunset and wa immedi withdrawn and afterward a thin rod rose up joint by joint bear at it apex a circular disk that spun with a wobbl motion What could be go on there

   For this problem, you will need to do an approximation of a stemmer, as follows: In the War of the Worlds passage, after removing punctuation, please remove the suffixes

   > -s or -ing or -ed

   from all words that have them. Your results will be a list of strings. If this list was called `stemmedwords`, then the result of the command

   ```
   " ".join(stemmedwords)
   ```

   should look like this:

   > 'After the glimpse I had had of the Martian emerg from the cylinder in which they had come to the earth from their planet a kind of fascination paralys my action I remain stand knee-deep in the heather star at the mound that hid them I wa a battleground of fear and curiosity I did not dare to go back toward the pit but I felt a passionate long to peer into it I began walk therefore in a big curve seek some point of vantage and continually look at the sand heap that hid these new-comer to our earth Once a leash of thin black whip like the arm of an octopu flash acros the sunset and wa immediately withdrawn and afterward a thin rod rose up joint by joint bear at it apex a circular disk that spun with a wobbl motion What could be go on there'

6. **Removing fine print (20 points)**

   From Project Gutenberg, please download *Wuthering Heights* by Emily Brontë. Please make sure to use the plain text version. This will result in a file called `pg768.txt`. Load the file into your Python shell using the following commands:

   ```
   f = open('pg768.txt')
   bronte_string = f.read()
   f.close()
   ```

   Note: Depending on where on your system you put the file, you may need to put a path before `pg768.txt`, for example

   ```
   f = open('/Users/katrinerk/Downloads/pg768.txt')
   ```

   If you cannot determine where on your system the file is located, you can also copy and paste the whole text into a single string `bronte_string` from the web page.

   If you look at the text, you will note that there is "small print" before the start of the novel, and more additional material after it. The novel starts after a line reading

   > ***START OF THE PROJECT GUTENBERG EBOOK WUTHERING HEIGHTS***

   and ends before

   > ***END OF THE PROJECT GUTENBERG EBOOK WUTHERING HEIGHTS***

   Use the Python string method `index()` to determine the location of both the ***START text and the ***END text. Note that `index()` gives you the *start* rather than the end of the ***START text. What do you need to add to this index to identify the position directly after " ***START OF THE PROJECT GUTENBERG EBOOK WUTHERING HEIGHTS***"?

   Then use a string slice to obtain only that part of the string `bronte_string` that is the actual novel.