PYTHON WORKSHOP – SUMMER 2017

ADRIANA PICORAL
ADRIANAPS@EMAIL.ARIZONA.EDU

The main goal of this workshop is to develop the following skills:

- Coding literacy
  - Identifying HTML tags to scrape data from websites
  - Running, reading and modifying Python scripts
- Text file cleanup
  - Grasping broadly the concept of different types of character encoding
  - Writing regular expressions
  - Running command lines to find text excerpts in file batches
- Parts of Speech (POS) tagging
  - Running a tagging python script
  - Formatting the tagger's output so it can be read by AntConc or R.
  - Using POS tags

1. Identifying HTML tags

We are going to scrape blog entries from the China Daily Blog (`http://blog.chinadaily.com.cn/`), a website where bloggers from different L1s (mainly Chinese) post entries in English. Here's a list of users I have selected, who have written more than 15 entries.

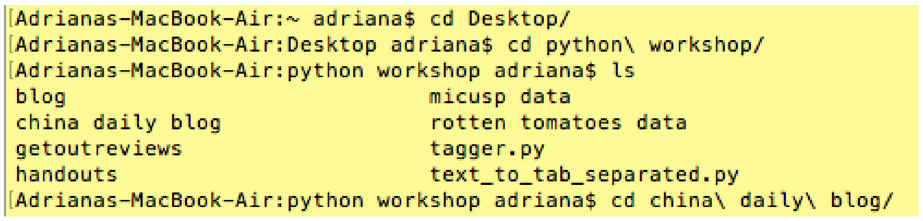
(1) `http://blog.chinadaily.com.cn/space-uid-1026274.html`
(2) `http://blog.chinadaily.com.cn/space-uid-1362846.html`
(3) `http://blog.chinadaily.com.cn/space-uid-728189.html`
(4) `http://blog.chinadaily.com.cn/space-uid-1035253.html`
(5) `http://blog.chinadaily.com.cn/space-uid-2456621.html`
(6) `http://blog.chinadaily.com.cn/space-uid-1348685.html`
(7) `http://blog.chinadaily.com.cn/space-uid-1835901.html`
(8) `http://blog.chinadaily.com.cn/space-uid-1395221.html`
(9) `http://blog.chinadaily.com.cn/space-uid-1433233.html`
(10) `http://blog.chinadaily.com.cn/space-uid-72296.html`

Choose a profile from the list above, find the page where the blog entries are listed, and view page source (right click on blank part of the page, and choose "view page source"). We are looking for HTML tags that contain links to each blog entry. HTML tags are always between angular brackets (i.e., ⟨ ⟩), and usually come in pairs with an opening tag (e.g., `<div class="name"> Content of division </div>`). Attributes (such as class and id) are defined in the opening tag.

2. Running a scraper script

(1) Open command or Anaconda prompt (Windows), or Terminal (Mac).
(2) Find the folder containing the scraping script (i.e., chinadaily_blogscrapper.py) using the `cd` command (hint: you can hit the tab key to autocomplete file and folder names). Use `dir` (Windows) or `ls` (Mac) to list the files and folders of your current directory. Here's a summary of commands you can use:

	Windows	Mac
list contents of folder	dir	ls
move into a folder	cd foldername	cd foldername
move out of a folder	cd ..	cd ..
go to home folder	cd	cd



```
[Adrianas-MacBook-Air:~ adriana$ cd Desktop/
[Adrianas-MacBook-Air:Desktop adriana$ cd python\ workshop/
[Adrianas-MacBook-Air:python workshop adriana$ ls
blog                            micusp data
china daily blog                rotten tomatoes data
getoutreviews                   tagger.py
handouts                        text_to_tab_separated.py
[Adrianas-MacBook-Air:python workshop adriana$ cd china\ daily\ blog/
```

(3) Run the script by typing `python chinadaily_blogscrapper.py`
(4) Open the script on notepad++ (Windows) or Aquamacs (Mac)

3. Character Encoding

There are many different encoding systems, but the most common are ASCII (American Standard Code for Information Interchange, 7 bits) and UTF-8 (Unicode Transformation Format, one to four 8-bit bytes). Web pages are usually encoded in UTF-8, which is used by more than 75% of websites in the entire Internet (`https://trends.builtwith.com/encoding`). For a list of UTF-8 codes, check `http://www.utf8-chartable.de/`.

4. Regular Expressions – Regex

Regular expressions define search patterns, and are usually used in find (or match) and replace operations on strings. This is a useful website to build and test your regular expressions: `https://regex101.com/`

5. Text File Cleanup

Why do text files need cleanup? Most taggers and parsers for English cannot deal with non-English characters. In addition, these text processing tools also expect sentences and punctuation to be normalized, e.g., single space after each punctuation mark, no "smart quotes" or any other type of quote such as guillemets. We also may want to eliminate both repetitive strings such as warning messages and course information, for example, and strings that taggers cannot deal with such as URLs or references.

We are going to follow five steps to prepare our text files for tagging:

(1) Punctuation normalization
(2) Sentence normalization
(3) Replacement of non-English characters
(4) Elimination of repetitive elements
(5) URL tagging (i.e., place URLs inside angle brackets)

I created a python script called `textnormalization.py`. Find it, move it to the folder where your text files are, and then open the script on your text editor.

6. Command Lines

There are command line tools to do quick operations such as counting total number of lines and words in text files, finding strings within text files using regular expressions, and converting batches of Word files to text files. These come already installed in Mac computers, and you can download and install them for free on PCs following the instructions at `https://cygwin.com/install.html`. In this workshop, we are going to work with a command line that uses regular expressions to find strings in a group of text files, i.e., grep for Mac (terminal) and findstr for Windows (command prompt). Here are the commands to search for patterns in all text files in the current directory (〉 indicates a line on command prompt, `$` on terminal; do not include these symbols in your command line):

- `〉 findstr /a:e "http" *.txt`
- `$ grep --color "http" *.txt`

For more information about these command lines, type the following:

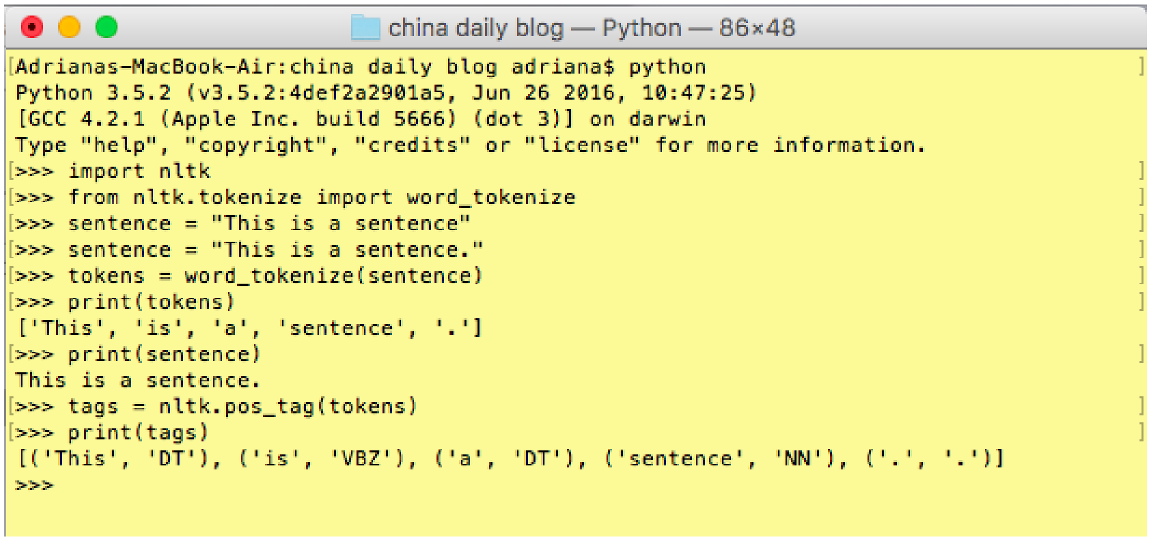
- `〉 findstr -?`
- `$ grep --help`

7. POS tagging

We are going to use Python's NLTK package for POS tagging in this workshop. There are two steps for tagging: 1) tokenization (each token is treated as one separate element), and 2) tagging (each token is then assigned a POS tag). NLTK outputs a list of tuples, e.g., *This is a sentence* becomes `[('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('sentence', 'NN'), ('.', '.')]`

Notice that NLTK uses Penn Treebank POS tags (`https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`). You can test this out in Python on your prompt (command or terminal):

```
import nltk
from nltk.tokenize import word_tokenize
sentence = "This is a sentence."
print(sentence)
tokens = word_tokenize(sentence)
print(tokens)
tags = nltk.pos_tag(tokens)
print(tags)
```



```
Adrianas-MacBook-Air:china daily blog adriana$ python
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 26 2016, 10:47:25)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> from nltk.tokenize import word_tokenize
>>> sentence = "This is a sentence"
>>> sentence = "This is a sentence."
>>> tokens = word_tokenize(sentence)
>>> print(tokens)
['This', 'is', 'a', 'sentence', '.']
>>> print(sentence)
This is a sentence.
>>> tags = nltk.pos_tag(tokens)
>>> print(tags)
[('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('sentence', 'NN'), ('.', '.')]
>>>
```

(1) Find the script named `tagger.py` and move it to the folder where you normalized files are.
(2) Open the POS tagging script on your text editor.
(3) Run the script and inspect your output.
(4) Make the appropriate changes, according to your needs.