Text Mining Homework - Week 0 - Crawler Manual

Aimee Barciauskas, Felix Gutmann, Guglielmo Pelino, Thomas Vicente

May 16, 2016

Introduction

For this homework we chose to scrape poemhunter.com, which is a website that collects many different kind of poems (either famous ones and amatorial). In particular, we focused on the new poems section, where new amatorial short poems are continuously published, because we thought it would be more interesting. As a general note, we tried to maintain the original structure of the poems, thus preserving line breaks and erasing html tags (more details later on).

Function-Manual:

crawl_poems(url , file_name = "poem", start = 1, end = 10, sleep = None, log = 25)

Function arguments:

- url: the url you want to start from (this for sure works with the new poems section, but we did not check others);
- file_name: the name of the file the poems will be saved into;
- start: the previous url is such that it contains different pages; start is an integer which flags for the number of page you want to start from, so it completes the previous url;
- end: last page you want to consider in the previous url;
- sleep: integer, upper bound on the number of seconds you want to randomly wait for scraping each page;
- og: frequency of the dump in number of pages (every log number of pages you save the scraped poems in a .txt).

Code-Description:

The code follows the following procedure. Each page in the new poem section points to 25 poems. First we get the urls for each poem from each page and use them to accuses the poems. We continue this procedure for each page.
In more detail the code consists of three main functions and two auxiliary

functions: We first use the function get_url_list() to read all URLs from the current page by filtering the relevant tags. From this first function, we get each poem's URL from the current page. We feed the list of gathered urls to the get_poems() function, which scrapes the actual poem from this url. The function notably contains an option for random sleeping time between the scraping of two poems.
The function crawl_poems() is a wrapper function using the former two functions to crawl all poems from all page in the section new poems (arguments see previous section). Furthermore it saves periodically all poems with corresponding title to the hard drive.
Finally we added two auxiliary functions. The function get_titles() extracts the title of a poem from an url. On the other hand the clean() function processes the crawled poems by removing html tags, but preserving line breaks.

Robustness:

To ensure that our crawler doesn't get stuck at a wrong URL, we implemented a try/except chunk in the get_poems() function. The crawler then tries to get a URL, and if not, it just continues. The random wait option can play the same kind of role when pages do not load sufficiently fast. The log is notably used within the try/except chunk. If the try is a success, the URL is added to a list of successfully loaded URLs. Otherwise, it is added to the list of unloaded URLs.
Moreover, the crawler comes with unit tests for all its low-level functions. If changing the source code, ensure tests still pass using the command python test_crawler.py.
Finally, we did not observe any restrictions from poemhunter.com/robots.txt.