A SIMPLE PYTHONIC WEBSPYDER

CSCI1001 FINAL PROJECT REPORT

Project Member: Haosong Liu

April.29th, 2018

Haosong Liu
liux3857@umn.edu

Introduction

When doing literature reviews, one may need to download numerous paper of interest and usually these papers are all listed in the result page after you've entered the search criteria. To download all the pdf files of the paper listed can be tedious. A simple web spider can alleviate the pain of manually download all those files provided the user input the right URL link of the page the user wants the papers to be downloaded. This can accelerate the speed for literature review research.

In this project, a simple Python script was written to scrawl out URL links to the downloading page of the arxiv Material Science subject's latest papers. The papers downloaded are stored in a folder of user's choice as long as the user provides the correct path.

Technology employed in this project is mainly consisted of two parts: Python programming language and basic HTML components. Although a real webspyder requires the knowledge of many more knowledge such as JavaScripts, HTTP protocols etc, this project only focuses on simple static-crawling, does not handle JavaScripts responses, does not consider scenarios where log-in or authentications are required.

Project Flow:

The product of this project is some Python scripts which can auto-download papers from arxiv website. The workflow of this program is fairly simple: the user download the scripts to their computer, put the scripts in the same folder, in the 'moduleTest.py' scripts, in line 9, revise the path to the current path they are at and then click compile and run to do the work. For a sample input/output example, refer to Appendix A.

The final codes for this project consist of two parts: A main code named 'PrepareData.py' which consists of all the necessary functions needed for the webspyder, and a test file called 'moduleTest.py' which imports 'PrepareData.py' as a module and called the method 'prepareData()' within that modules to do the actual work. The intention of writing the scripts in this fashion is that it can modulate the code and can let user ignore all the details of the code and just call the method they need and do the work. It also

protects the source code as well, because user can now only modify the code in the driver code and will not damage the source code.

Appendix A: Sample Input/Output Example (Code Usage Example)

For users that don't know anything about Python programming:

Download both 'PrepareData.py' and 'moduleTest.py' file in the same folder. In 'moduleTest.py' script, replace the path on line 9 with the current path 'PrepareData.py' is in. Save the file and click run. See Fig A.1 for Guideline.

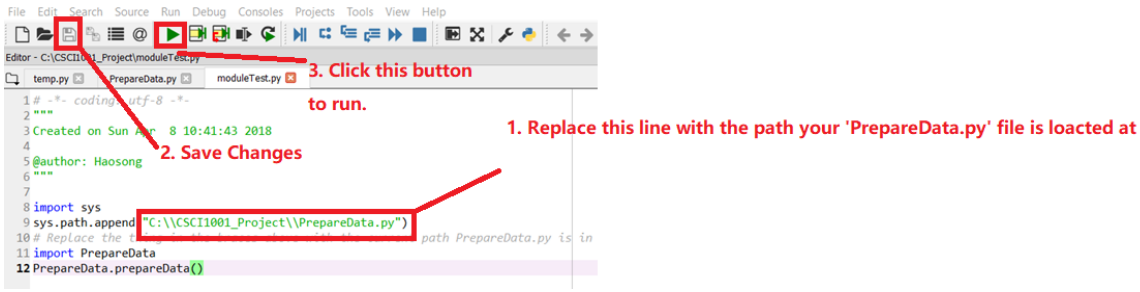

Fig A.1: A guideline for using the code

For users that can write codes:

Download only the main program 'PrepareData.py'. Create a new Python Scripts of your own, import sys module and append the path of 'PrepareData.py', import the package and call the 'prepareData' method in that package. The method does not require any argument.