Oscar Search

fanfaredash, 2019201407

July 2020

1 Introduction

This project is a simple *offline web-search program* implemented with web-crawler technology, inverted index, and TF-IDF.

With this project, you can first get all web-pages under the root URL, which is specified, then use search-engine module to query web-pages.

2 Usage

- Requirements
- Installation
- Initialization
- Searching
- Settings

2.1 Requirements

Python3 ($\geq$ 3.5.2)

2.2 Installation

Run following code in *bash* to install basic packages:

```
sudo pip3 install -r requirements.txt
```

2.3 Initialization

That is, to download and prepare data for search-engine.
First, go to *arguments.py*, and change argument *root* to your target website.
For example, `root = "http://info.ruc.edu.cn/"`.
Then start *run.sh*:

```
./run.sh
```

Or you can use `python3 [filename]` to initialize data step by step:

crawler_mt.py	getting html data from website
parser.py	parsing html data for pure text
cutter.py	split documents for terms
index.py	building inverted index

2.4 Searching

Start search using: `python3 search.py [query_text]`.
This will display result in console.

Or use Flask to deploy user interface: `flask run`
After that, use browser to access your web server.

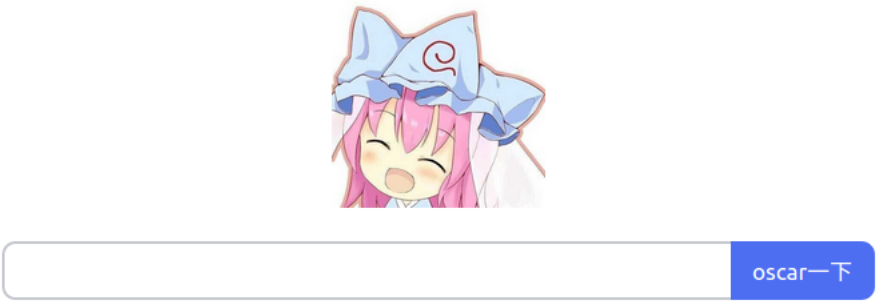

Figure 1: Oscar search: main page

Figure 2: Oscar search: sample

2.5 Settings

Customize arguments in *arguments.py*.

root	web-crawler target
resultDisplay	maximum results on display
abstractDisplay	maximum length of abstract (UI only)
directory arguments	modify file path(not recommended)

Default file path:

html file	/raw/html
page text	/raw/text
page title	/raw/title

File-names are encrypted by base64 method.
Use `python3 ess.py -d [filename.html]` to decrypt for web-page URL.
Or use `python3 ess.py -e [web-page URL]` to encrypt for filename.

Customize stop word list in *stopwords.txt*, word per line.
Customize user dictionary in *userdict.txt*, word per line.

3 Features

- Flexibility
- Base64 encryption
- Offline index building
- Process-Oriented Programming
- Multi-Thread crawling

3.1 Flexibility

Compared to other projects in SPP2020 homework, this project includes *no* algorithm or method related to "https://info.ruc.edu.cn/". The whole project works well with other root URLs.

Besides root URL, user can also customize number of search results and length of abstract displayed, though it's much more trivial than the former one.

3.2 Base64 encryption

As we know, there should be no character "/" in filename.

This project uses base64 encryption to resolve URL string. Base64 builds a bijective function between URL string and filename, which is much safer than to replace "/" to other substrings.

3.3 Offline index building

This project uses offline index building, which means the program won't have to build inverted index every time when the user search for result.

Offline index building also requires saving data to files, that might rise the time complexity of the program. Therefore, *BKDR-hash* algorithm and *binary-search* algorithm are used in term searching.

Yet due to the data scale of "https://info.ruc.edu.cn/" is only about 7000 (web-pages), this program didn't show its speed advantage in automatic test.

3.4 Process-Oriented Programming

The source code of this project are well process-oriented. Developers who are interested in this project can easily customize their own process and replace its counterpart in the source code.

3.5 Multi-Thread Crawling

This project uses multi-thread technique to speed up web-crawling process.