# Search Engine by RuiLi

Implement the simplified Search Engine described in Section 23.5.4 for the pages of a small Web site. Use all the words in the pages of the site as index terms, excluding stop words such as articles, prepositions, and pronouns.

#Algorithm description

1. use scapy to crawl several pages

2. for each page,remove the words containing punctuation and stopwords

3. use inverted index to implement a dictionary, storing key-value pairs (w, L) where w is a word and L is a collection of references to pages containing word w.

4. if a word occurs in the specified page reference ,then find the reference in the collection and add 1 into the reference counter.

5. Also insert the (w,ref) into tries where w is a word and ref is the total reference number related to the matching occurrence lists,

6. when doing search, first get the reference number in the tries ,then get a collection of references to pages containing word w in the reference list.

each contains the references and the hit nums: for example ,key words company appears in "http://paulgraham.com/articles.html" for 1 time:

{'key': 'company', 'http://paulgraham.com/articles.html': 1, 'http://paulgraham.com/invtrend.html': 17, 'http://paulgraham.com/swan.html': 4, 'http://paulgraham.com/growth.html': 34, 'http://paulgraham.com/startupideas.html': 8}

{'http://paulgraham.com/articles.html': 1, 'http://paulgraham.com/invtrend.html': 17, 'http://paulgraham.com/swan.html': 4, 'http://paulgraham.com/growth.html': 34, 'http://paulgraham.com/startupideas.html': 8}

7. finally, rank the result by the value of appearance times of each references.

## directory structure

.

├── README.txt

├── InvertedIndex.py -- a dictionary implemented by Trie

├── SearchEngine.py -- main class

├── Tries.py -- tries structure index

├── AppStartup.py -- Initial script which will start up the Flask web server

├── MySpider -- spider program,which include a spider to crawl from paulgraham's blog and output 10 html files

├── file\_link\_map.json -- a mapping of the output html files and its link

├── input -- the website output by the spider and used to be the input of search engine

├── MySpider -- spider program

├──scrapy.cfg --spider config file

├──\_\_init\_\_.py --the auto-gen files by scrapy

├──items.py

├──pipelines.py

├──settings.py

├──spiders -- the specified webpage need to crawl.

├── \_\_init\_\_.py

├── paulgrahamblog.py --the spider which is used to pase the crawl result and generate the file\_link\_map.json

├── templates -- GUI templates

## dependencies

1. python3

2. Scrapy (pip3 install scrapy)

3. beatifulsoup (pip3 install beautifulsoup4)

4. nltk (pip3 install nltk)

5. flask(pip3 install flask)

6. tldextract(pip3 install tldextract)

## deploying

1. download the project under macos or linux

2. go to the spider directory to crawl pages

cd ./SearchEngine/MySpider

3. crawl 10 web pages by following commands from http://paulgraham.com/, You will discover 10 html files in the MySpider/input directory

scrapy crawl paulgrahamblog

4. goto the ./SearchEngine directory and launch the search engine web server app

cd ./SearchEngine && python3 AppStartup.py

5. access http://127.0.0.1:8886/ to do the search task