Web Search and Mining Group Project

Report

Members:

孙浩然 116033910108

徐浩 116033910114

周小舟 116037910018

Option A: Xunyiwenyao

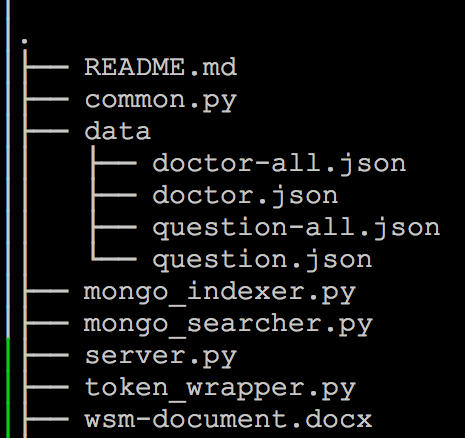
# Introdcution

Our system is a search engine built on the crawled data of Xunyiwenyao webset. It has three main components: crawler, Backend server and front end user interface.

Our system has several dependencies:

* Python3 and flask is required to run the server program
* MongoDB is used to store the pages and indexes
* PyMongo is the python driver of MongoDB
* Jieba to break the document into words

After installing the dependencies, run ./server.sh to start the server.



# Design

## Crawler

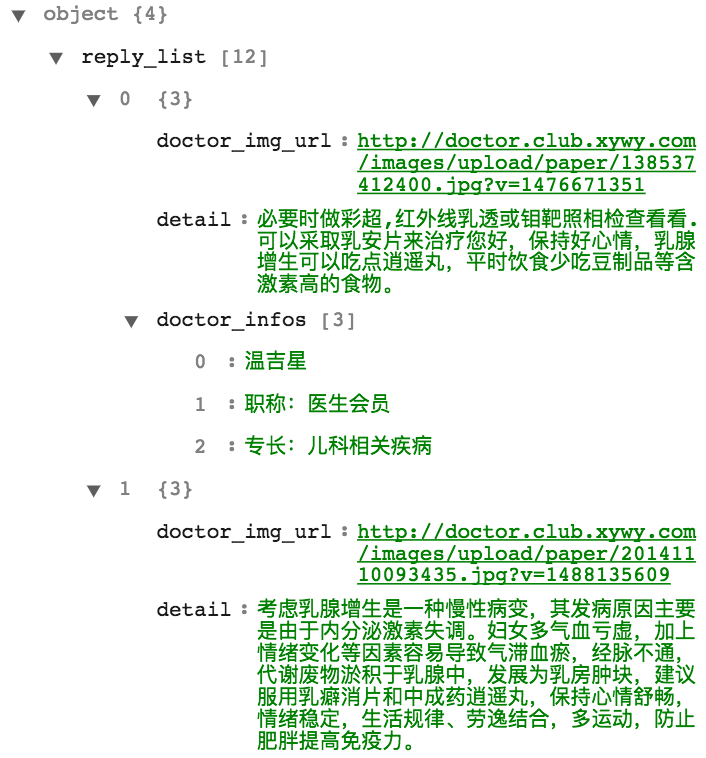
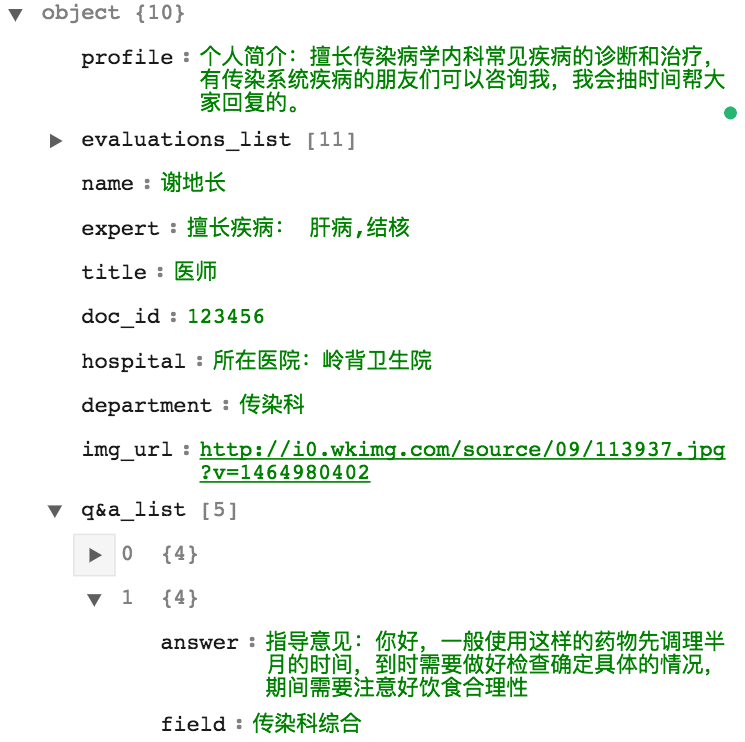


Figure 1 question example Figure 2 doctor example

## Indexer and Searcher

To index the documents, we process the input JSON files and break each field of document into tokens. We use Jieba (<https://github.com/fxsjy/jieba>) to do the Chinese text segmentation.

|  |
| --- |
| Example of Chinese text segmentation |
| 乳房肿块可以采用中药回阳玉龙膏外治的方法，直接作用于乳房特殊部位，让药物吸收更加充分，药物效果更加持久。采用柴胡、红花、鳖甲、川芎、当归、贝母、黄芪等30多味纯天然中药材，将药物磨粉，并且因为乳房的穴位比较多也利用磁粉，药粉和磁粉同时作用。 |
| ['乳房', '肿块', '可以', '采用', '中药', '回阳', '玉龙', '膏外治', '的', '方法', '，', '直接', '作用', '于', '乳房', '特殊', '部位', '，', '让', '药物', '吸收', '更加', '充分', '，', '药物', '效果', '更加', '持久', '。', '采用', '柴胡', '、', '红花', '、', '鳖甲', '、', '川芎', '、', '当归', '、', '贝母', '、', '黄芪', '等', '30', '多味', '纯天然', '中药材', '，', '将', '药物', '磨粉', '，', '并且', '因为', '乳房', '的', '穴位', '比较', '多', '也', '利用', '磁粉', '，', '药粉', '和', '磁粉', '同时', '作用', '。'] |

Then we build inverted index. Because the size of dataset is larger than the memory size, we store the inverted index in MongoDB. Each item in posting list has three fields: document id, region, term frequency. The region field can support regional search.

For the first three kind of queries, we use a generic interface search\_all(query\_string, region, type). For the fourth kind of queries, we build a term to doctor index to find the best doctor for a disease.

Question

Doctor

Answer

Terms in answer:

term -> question\_id in tbl\_index

term -> doctor\_id in tbl\_answer

Our database has Three tables:

* **tbl\_pages**: stores the crawled page. Searcher use this table to returns the demanding page
* **tbl\_index**: regional inverted index.
* **tbl\_answer**: map tokens in answers to the doctor who wrote the answer. This table is used to find the doctors related to a specific disease.

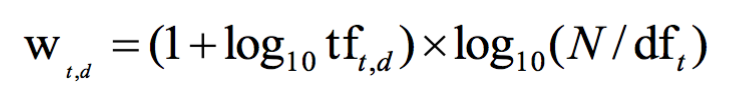
To gain better performance, stop words are removed from the index.

See mongo\_indexer.py for more details.

## Searcher

Because we used a regional inverted index and the word frequencies in each regional is not the same, the df is not precomputed by the indexer, but the searcher.

The traditional tf.idf index is used in our system.



Posting list is retrieved from the database, and the results are sorted by the decreasing tf.idx index.

The web server provides two API for searching:

|  |
| --- |
| post /search\_all  parameter is an object, it contains three fields  s is the query string  region is an array of regions:  e.g. ("title" "detail" "answer" "name" "expert" "profile")  type in ["doctor", "question"], or omit it for all pages  => return all related pages in json |
| post /search\_doctor  parameter contains only one field  s is the query string  => return all related doctor pages in json |

See server.py, mongo\_searcher.py and searcher.py for more details.

## Front-end

The front-end part of the project is developed by html + javascript + css, using the jQuery library for rapid HTML element selection, and call the query API by initiating HTTP requests to back-end through the Ajax query interface, then display the query results.

The site has three pages. The first page is the home page, including a text box, a list of function tags, an advanced search hyperlink, and user search history. Users can enter the keyword to be queried directly in this page, and select the search function needed in the tag to search. These tags are corresponding to the first, second and fourth problem requirements respectively.

The second page is advanced search page, including search type radio boxes and search field check boxes, corresponding to the third requirement of the problem. Users can customize the search on this page, freely specifying the fields in which they want to search, and the type of search results (Q&A record, or doctor information).

The third page is the search results page, which displays all the query results in the middle of the page and contains the same search function tags as the home page, the search field filtering button, and the user search history. The user can easily switch to the desired search function and the fields to search for in this page.