A search engine for classical Chinese poems

Xiwei Wu & Ning Yang & Hui Ma

Abstract: This is a group-based project for Web Search & Mining. In this project, we implemented a poetry crawler that crawled more than 10,000 poems from 中华诗词网 from pre-Qin to modern times, then implemented a search engine that can support a variety of different search requirements, and finally provided a web front-end that can be easily used.

Keywords: Scrapy, Chinese poems, search engine

1 Project Requirements

In this project, we are required to crawl classical Chinese poems onlineand building a simple search engine. Detailed requirements are as follows:

1. **Data Crawling and Preprocessing**.
   You need to crawl classical Chinese poems from websites. After that, you have to do some necessary data preprocessing steps to transform the raw data into structure records, with each having the following attributes: title, author, dynasty, content, and optionally other information if available: translation, annotation, appreciation, background, etc.

2. **Search Engine**. Based in the poems, you need to implement a simple search engine supporting the following functionalities.

   a) **Boolean Search**: Users provide search keys and operations between keys. The system needs to return all the relevant poems (Either title or content meets the requirement of the query). The query language must include operations such as AND, OR, NOT.

   b) **Zone-specific Search**: Allow users to specify the attribute to filter/search.
   For enumerable attributes like author and dynasty, the results should only contain those exactly match the query.
   For other attributes like title and content, a Boolean or ranked search can be performed.
   You can use either designs of UI or query (e.g. author(李白), title(酒)) to support such specification.

c) **Pinyin-based Tolerant(Fuzzy) Search**: Just like SOUNDEX introduced in class, we can use pinyin for Chinese phonetic tolerant search. This can be especially useful when the user enters a wrong word with the same sound, or the poem itself contains 通假字. Therefore, your search engine should support a special query language SOUND(x) which can retrieve poems with any word having the same pinyin as x. For example, we should be able to get " 最爱湖东行不足，绿杨阴里白沙堤。" with SOUND(荫).

d) **Ranked Search**: Return the search results with certain order that may be favorable by the users. The factors to consider may include: semantic relevance, fame of the author and the poem itself, etc. You can use any ranking method and any available information only to achieve this. We will prepare several test cases and score the system according to our search experience.

3. **Nice Web Interface** for demo and **Project Report**.

Outline. In Section 2, we will introduce our crawler based on Scrapy and SQL. In Section 3, we will introduce our design for search engine. In Section 4, we will introduce our design for web front-end. In Section 5, we will give a demonstration of the results of our project.

2 Data Crawling

We use Scrapy for crawling data and SQL for storing data. Here we use 中华诗词网 as our poem source. Our crawler starts with the pages belonging to different dynasties. By analyzing the web code we get the relevant information.

```
def start_requests(self):
    dynastys = {
        "先秦": "https://www.shi-ci.com/dynasty/72057594037927936",
        "汉代": "https://www.shi-ci.com/dynasty/144115188075855872",
        "三国两晋": "https://www.shi-ci.com/dynasty/216172782113783808",
        "南北朝": "https://www.shi-ci.com/dynasty/288230376151711744",
        "隋代": "https://www.shi-ci.com/dynasty/360287970189639680",
        "唐代": "https://www.shi-ci.com/dynasty/432345564227567616",
        "宋代": "https://www.shi-ci.com/dynasty/504403158265495552",
        "元代": "https://www.shi-ci.com/dynasty/576460752303423488",
        "明代": "https://www.shi-ci.com/dynasty/648518346341351424",
        "清代": "https://www.shi-ci.com/dynasty/720575940379279360",
        "近现代": "https://www.shi-ci.com/dynasty/792633534417207296",
      }
    for k,url in dynastys.items():
        yield Request(url,meta={"item":k},callback=self.parse_dynasty)
```

For each poem, we collect dynasty, poet name, poem name, contents, poet description and the associated page url.

```
class PoemItem(scrapy.Item):
  dynasty = scrapy.Field()
  poet_name = scrapy.Field()
  poet_url = scrapy.Field()
  poem_name = scrapy.Field()
  poem_url = scrapy.Field()
  contents = scrapy.Field()
  poet_desc = scrapy.Field()
  crawl_url = scrapy.Field()
```

Based on these data, we created the table via mysql, and the relevant steps can be referred to the poem/READMD.md file. Finally, we output the data in SQL format for later use. You can see our dataset as poem/data/poem.sql.

Due to the problem of website data, we need to process the poems obtained by the crawlers, such as the unification of Chinese and English commas, the addition of missing content in the body, and the appearance of ellipses due to the excessive length of titles. Here we would like to mention the ellipsis in the title, we found that among the more than 50,000 ancient poems we collected, there are more than 1,200 poems with ellipsis in the title, most of them are Tang poems. And we found that the titles of these poems existed in most of the websites in the form of containing ellipses, so we also dealt with only some of the poems for which the full names could be found.

3 Search Engine

3.1 Overall Design

3.2 Boolean Search

3.3 Zone-specific Search

3.4 Pinyin-based Tolerant Search

3.5 Ranked Search

4 Web Interface

5 Conclusion

You can see our codes and dataset on the Github Crawl-for-poems.