SE305 DBST Final Report
Illustration of the Books

Yuqi Yi* Hongjie Chen†

December 2018

Contents

1 Project Description

Books are a rich source of both fine-grained information. In order to improve the readability of the book, many books have illustrations which are strictly illustrative of the text. However the E-book rarely contains illustrations. So in this work you are required to choose illustrations for each chapter in a book. This project can be seen partly as an information retrieval task.

There is a dataset called BookCorpus, but this corpus has not been maintained. Firstly, you are required to crawl all the books in BookCorpus. You can find the booklist on https://www.smashwords.com/books/category/892 and you can download books from https://www.gutenberg.org or other website.

*515030910596, awonderfullife@sjtu.edu.cn
†515030910597, chenhongjie@sjtu.edu.cn

What you also need to crawl is a picture dataset about the books. The requirement is that you need to crawl at least one picture about a named entity or a concept in *each* chapter of every book in the data set.

In this project, you are required to create a relational database to store the entire data, the book text and pictures. You can store the books with separate chapters. In each chapter, you are required to extract some important information which you believe. And then, you can label for pictures and match the pictures to the chapter according to the labels you get from some tools and the important information you extract from the chapter text. For example, I can search "dodo" in Alice in Wonderland , and I will get figure 1.

Figure 1: dodo

Required queries:

- Find the books according to the name, author.
- Given a book and chapter number, you can return some relative pictures.
- Given some words, you can find the relative pictures according to the words.

2 Database Design

In order to implement the required queries mentioned above, we first design a proper Entity-Relation (ER) model accordingly. Then we convert the E-R model to a relational model.

2.1 E-R Model

There should be two entities, "*Book*" and "*Picture*". Since a picture must belong to some book, there should be a "*belong_to*" relation between "*Book*" and "*Picture*". Moreover, a picture also appears in some chapter of a book.

Thus, we need a additional attribute for the "*belong_to*" relation. Figure 2 shows the E-R model.

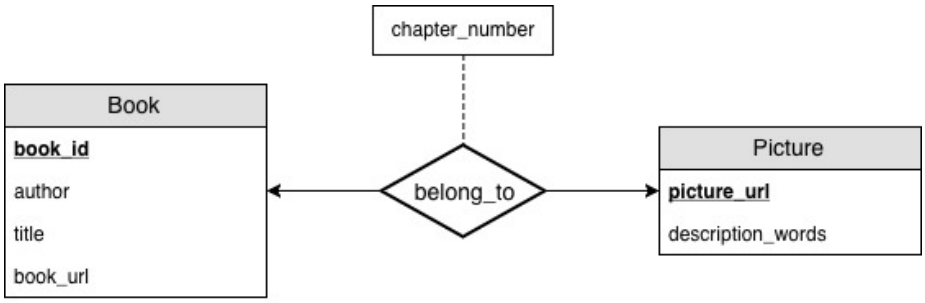


Figure 2: E-R Model

2.2 Relational Model

Based on the above E-R Model, we obtain the following 3 relation schemata naturally.

- **Book** (book_id, author, title, book_url)
- **Picture** (picture_url, description_words)
- **belong**_to (book_id, picture_url, chapter_number)

3 Database Construction

Now we already have relation the schemta we need. Next, we first crawl data online and download useful data to local hard disks as much as possible. Then we retrieve information that we need to

3.1 Crawl Data

In order to build a database, we need to get data first. Therefore, we crawled Project Gutenberg (see figure 3). For each book, we need its meta-information (such as title, author and language) and its content. After careful analysis of the structure of Gutenberg websites, we found that for each book there is a corresponding website containing information about its author, title, and etc (see figure 4). And there is also a link, i.e. "Read this book online: HTML", on that website leading to a new website which contains book content (see figure 4). Thus, we crawled two websites and then downloaded them for each book. Finally, we crawled more than **50,000** books.

Figure 3: Project Gutenberg

3.2 Process Data

Now we have book data on local disks, but we need to process these data before creating the final database. Each book has two HTML files of which one contains its meta-information and the other contains its content. We use BeautifulSoup, a Python library for pulling data out of HTML and XML files, to extract useful data from these HTML files. It's not difficult to extract a book's author, title and URL from its HTML file. However, the case is different for pictures. To find out which chapter a picture belongs to, we need to traverse the whole HTML tree. For each picture, we get its URL form the HTML file, and then we download it to local disks. We also get rid of books whose information is not complete. After processing, we get nearly **20, 000** books and more than **80, 000** pictures.

3.3 Create Database (MySQL)

We use MySQL to create the final database using the above relational model we designed and the above data we processed. The code for creating tables is as follows:

```
TABLES['Book'] = (
    """CREATE TABLE `books` (
        `book_id` varchar(10),
        `author` varchar(100),
        `book_url` varchar(100)
        `book_name` varchar(300)
    )""")

TABLES['Picture'] = (
```

Figure 4: **Up**: Book information. **Down**: Book content

```
    """CREATE TABLE `Picture` (
        `picture_url` varchar(100),
        `description_word` varchar(30)
    )""")

TABLES['belong_to'] = (
    """CREATE TABLE `belong_to` (
        `picture_url` varchar(100),
        `book_id` varchar(10),
```

```
        `chapter_number` int(10)
    )""")
```

We also implemented other basic database operations, such as *insert* and *delete*.

4 User Interface

For realize these needs we are required to implement, we design a simple GUI interface with wxpython. You can see our user interface below (Figure 5).

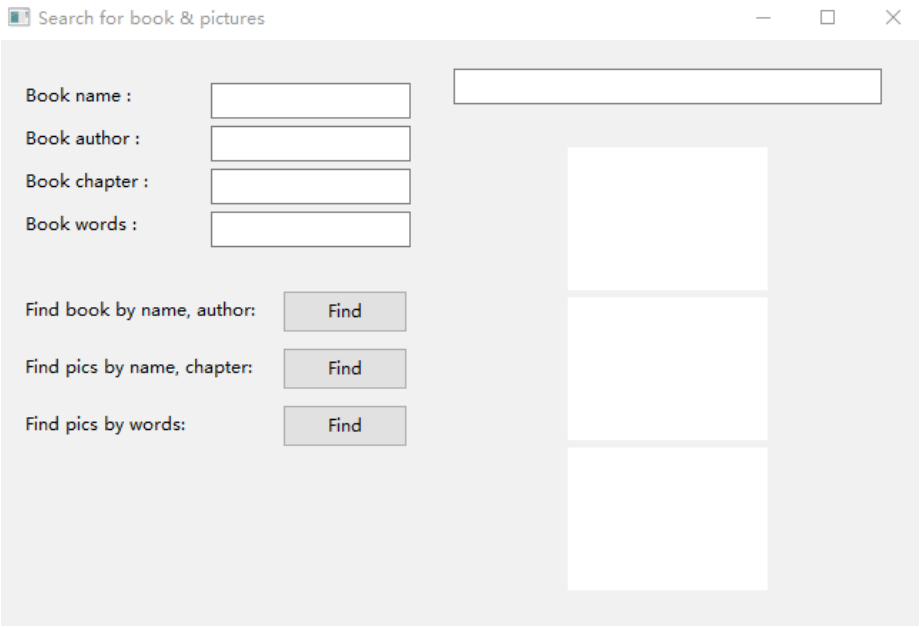


Figure 5: **Interface**: interface with three basic function buttons

there are four input text on the interface: book name, book author, chapter number, and the key words. After you input the corresponding feature that is needed for a specific search. you could click the corresponding button. For finding book through it's name and author, we would return a link for it's contents on the top of the right column. For finding pictures through some book name, chapter number or through key word. we would show the result pictures on the right column too and click same button for mutilate times, you can get for other result pictures for that instead of just three of them. If you change the corresponding input features, you just need to re-click the find button to update the result.

5 Experiments

We build a database with over 100000+ information about books and pictures in it, for simplify here we just show one example that we use the system to find the related book or pictures we wanted.

Figure 6 shows the experiment result we done for finding the corresponding book through book name and author. the result we return as a link for that book on the right top of our interface.

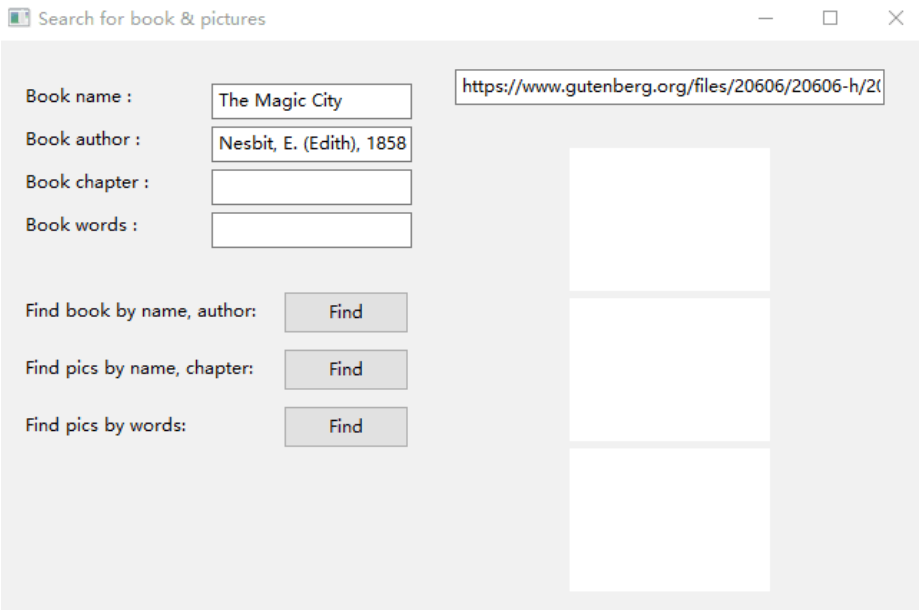


Figure 6: **Interface**: find book through book name and author

Figure 7 shows the experiment result we done for finding pictures through book name and chapter number. We finding there are three related picture and we present then on the right column of our interface.

Figure 8 shows the experiment result we done for finding pictures through book name and chapter number. the result of these also shown as above.

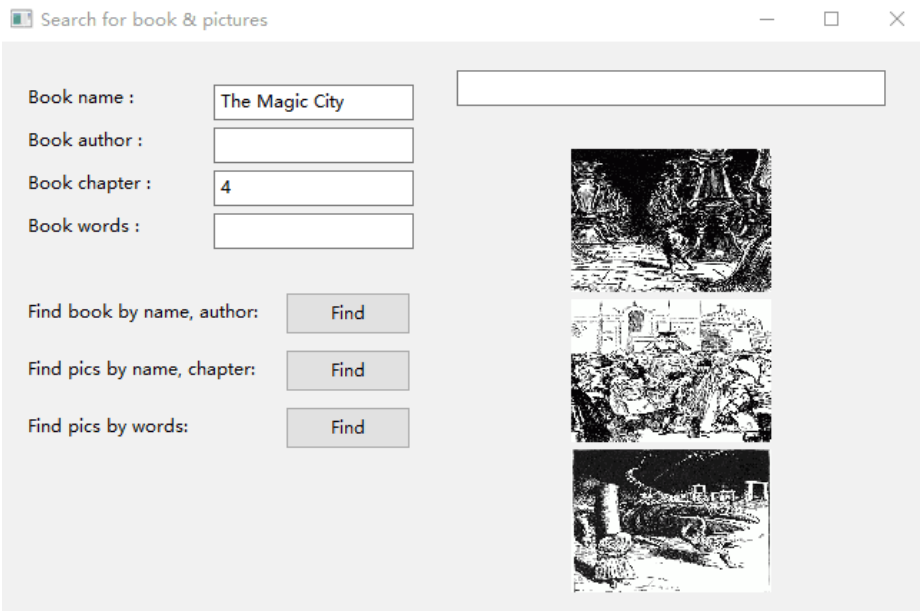
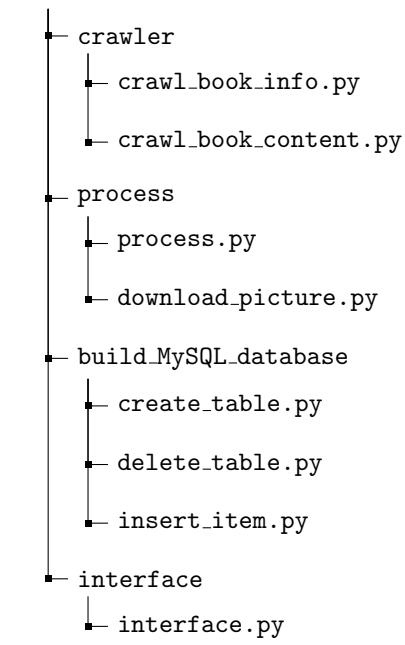


Figure 7: **Interface**: find pictures through book name and chapter number

Figure 8: **Interface**: find picture through book name and key words

6 Code

All the code is available online[1]. We also give the structure of our code here.



[1] https://github.com/CoolPhilChen/SE305_DBST_Project