DATABAE PROJECT

Wikipedia

2015-12-1

仇馨婷 5120409692
张哲慧 5120309056

1. INTRODUCTION

The main task is to build a database which contains specific information like link, anchor, title and content of wiki pages, so that user can search upon the database. It contains three queries, 1. Given an anchor, return all the link and their sentence -level co-occurrence count; 2. Given a wiki title, return all the anchor linked to it and their sentence -level co-occurrence count; 3. Given a link and word pair, return their sentence -level co-occurrence count. User can search certain content to get corresponding things.

In this project, all the codes are based on python. With the help of python regular expression, we process the raw data. Get access to database on disk through python API MySQLdb and execute mysql command.

The GUI is implemented based on Web front. We utilized Web Development API based on Flask, a lightweight Python web framework based on Werkzeug and Jinja 2. In addition, the template rendering is based on CSS and JavaScript.

2. ENVIRONMENT

- linux 14.04
- mysql 5.5.46
- python 2.7.6

3. DESIGN

Since link and title of one page are the same in raw data, we make link representing both link and title in the design below.

For the first two queries, we found that link and anchor are in many-to-many relation, since one link can be represented by many anchors and one anchor can point to different link content. So in E-R model, both link and anchor table have id as primary key and corresponding content. Since they are in many-to-many relation, in this relation table the primary key will contain both id from link and anchor.

For the last query, first we think if storing every link and word would be possible, because it will be so convenient when searching for a certain link and word pair, but

obviously it will occupy extremely large storage. So the whole sentence instead of single word will be stored. Every row contains a link and the sentence which the link is in. Since link and sentence is also many-to-many relation, same as above, a table named sentence is created with sentence id and sentence content pairs. To connect link and sentence, a relation table is also needed with both id of link and sentence.

To implement above design, 5 tables will be created in database. Although the design is "correct", it still has duplicated information. To save the memory, we try to use only two tables -- one containing link and anchor pair and the other link and sentence pair.

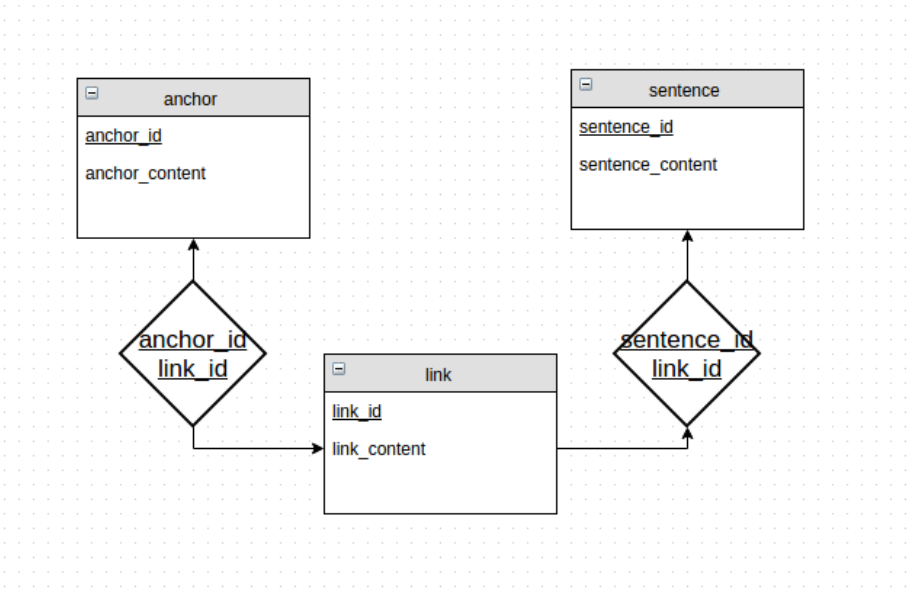


Figure 1 ER Model

link	anchor

link	sentence

4. PROCEDURE

a) Process raw data

What we need to do is to extract useful information from the large scaled xml file. Every page in Wikipedia begins with <doc> and ends with </doc> which can be easily detected. Each link-anchor pair is contained in a same pattern like “anchor”. Use python regulation, these pairs can be easily found. Since every sentence which contains links will be stored in database, separation of the paragraph is needed. Sentence always ends with period, question mark or exclamation mark, and some special cases should be paid attention like “Mr. Brown” since it also has period.

```
sen_pattern = re.compile('<a href="(.*?)">(.*?)</a>', re.S)
items = re.findall(sen_pattern, sentence

p = re.compile
    (r'((?:M(?:R|r|S|s|rs)\.|[^\.\?!])+[\.\?!]+)(?:$|[ \n]+)')
    contents = p.findall(content)
```

```
SCTV (Indonesia)        SCTV
Indosiar        Indosiar
RCTI    RCTI
MNC International       MNC International
MNCTV   MNCTV
Global TV (Indonesia)   Global TV
SINDOtv SINDOtv
Media Nusantara Citra   MNC Media
Media Nusantara Citra   Media Nusantara C
MediaCorp TV12 Suria    MediaCorp TV12 Su
MediaCorp TV12  MediaCorp TV12
MediaCorp TV Channel 5  MediaCorp TV Chan
MediaCorp TV HD5        MediaCorp TV HD5
MediaCorp TV12 Suria    MediaCorp TV12 Su
Astro Aruna     Astro Aruna
Sensasi Sensasi
MediaCorp TV    MediaCorp TV
MediaCorp TV12  MediaCorp TV12
mio TV  mio TV
StarHub TV      StarHub TV
Singapore       Singapore
SCTV (Indonesia)        SCTV
Indosiar        Indosiar
RCTI    RCTI
MNCTV   MNCTV
Global TV (Indonesia)   Global TV
Media Nusantara Citra   MNC Media
Media Nusantara Citra   Media Nusantara C
```

Figure 2 link anchor pair

Finally, we got two txt files. In the first file, every link and anchor pair is in one line, and separated by ’\t’. Same as the first file, the second file contains link and sentence pair separated by “@@@@@@”.

b) Mysql

Create two tables link_anchor and link_sentence. For link_anchor, it contains two fields – link and anchor, and the type is both VARCHAR(20). 20 is a reasonable length. For link_sentence, it contains two fileds – link and sentence, and the type of link is also VARCHAR(20) where TEXT for sentence. A TEXT column can contain a maximum length of 65535 (2^16 - 1) characters which also makes sense.

Next, those processed data should be loaded into our database. Thanks to powerful mysql, it's quite easy and convenient to do this. To get the permission to load local data, a certain flag should be added when entering mysql.

```
mysql -u root -p --local-infile=1

USE wiki;

LOAD DATA LOCAL INFILE '/path/to /link_anchor.txt' INTO TABLE anchor_link
COLUMNS TERMINATED BY '\t';

LOAD DATA LOCAL INFILE '/path/to /link_sentence.txt' INTO TABLE
link_sentence COLUMNS TERMINATED BY '@@@@@@';
```

Actually, a database without index is terribly slow, so here we add some indices to help improve our database. Since link and anchor are two main search object, they need indices. To save space, only half of the content in link and anchor is considered when indexing.

```
ALTER TABLE link_sentence ADD INDEX link_index (link(10));
```

c) Post-process

As mentioned before, our system is based on python. Here we use python mysql API –MySQLdb to connect to our database, execute mysql commands and get output.

```
db = MySQLdb.connect(host="localhost", # your host, usually localhost
                     user="root", # your username
                      passwd="123456", # your password
                      db="test") # name of the data base

cur = db.cursor()
```

```
cur.execute("select anchor from link_anchor where link = 'Indonesian
language'")
for row in cur.fetchall() :
    print row[0]
```

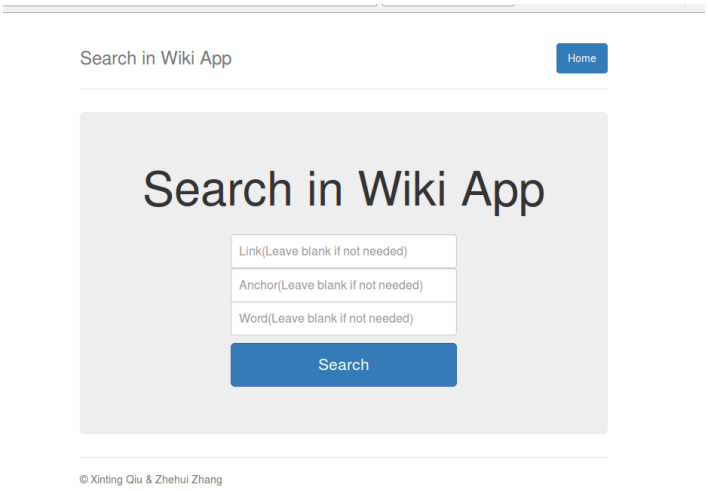
Since title and link are considered same in creating database, here we need to process users' input of link in case it is a url. For every link or anchor, we need to find the corresponding content it points to and count for each link anchor pair which also called sentence -level co-occurrence count.

For word and link pair, first get all sentences where the link in, and count how many these sentences contain the word. That's their sentence -level co-occurrence count.

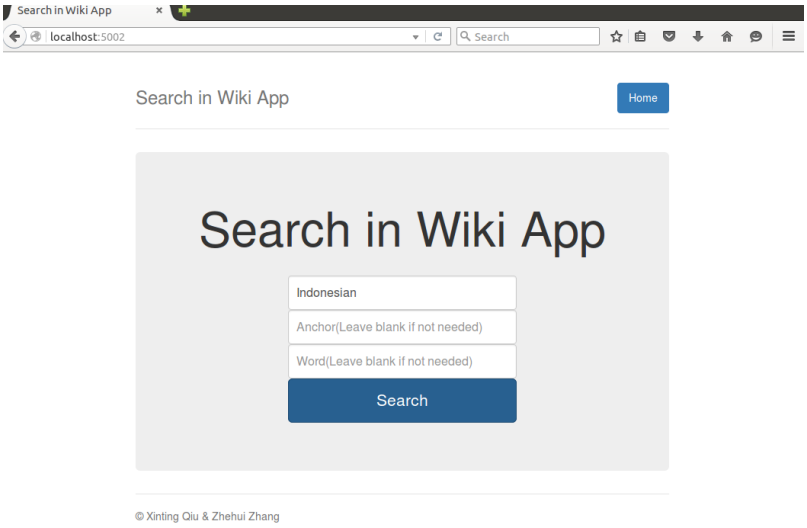
Return all these result to gui part.

d) GUI design.

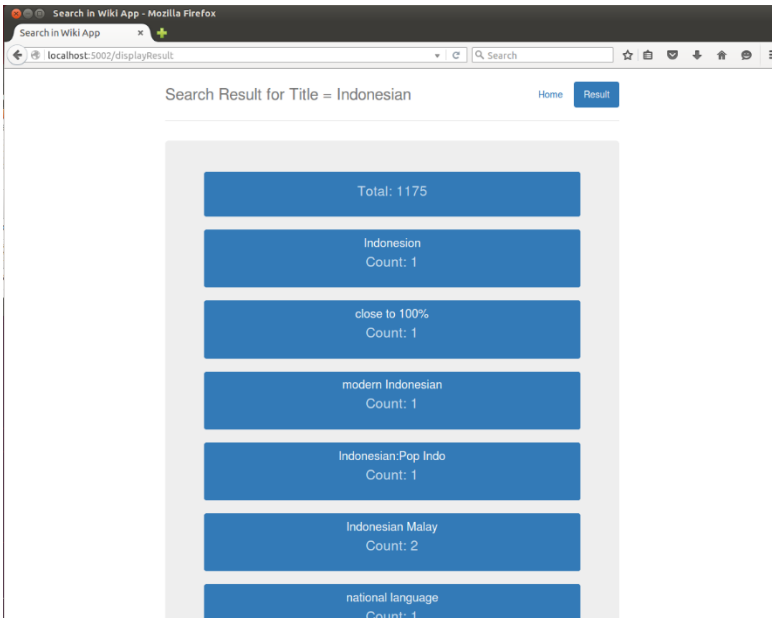
The whole project consists of two main template, home page and displayResult page. The home page is only for inputting, shown as:



Note here we only have one button, so we judge the function by whether the user input values in the input box. If the user enter 'Indonesia' like following, then he will get the result of finding corresponding anchor.

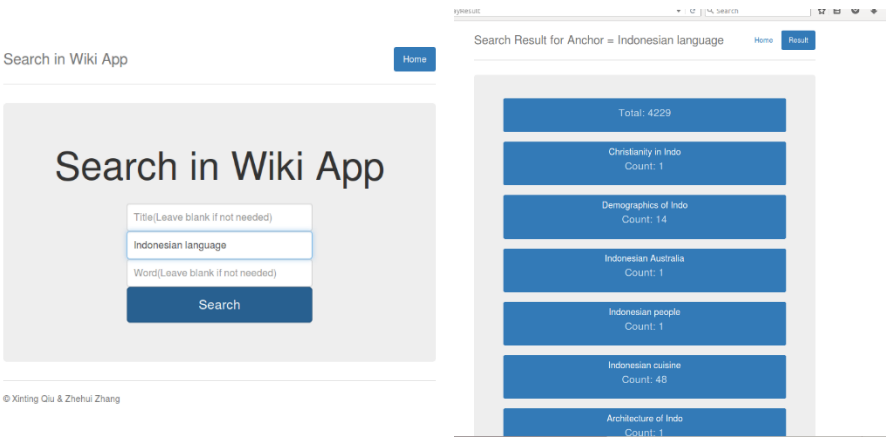


The following result shows the displaying part:



If the user want to go back and proceed to another query, he can click the home button on the right-top of the page and thus get the new result.

Another example is about the query of anchor:



If the user search with word input soap:

5. EXPERIMENTS

- Hardware Specifications: ThinkPad T440, 8GB, i7
- Dataset: Wiki

a) word vs sentence

Since we need to search for link and word pair, it's easy to consider storing each link and word pair. But it will waste so much storage, although the response can be quicker. So we decide to store every sentence for a certain link.

Figure 3 word vs sentence

b) index vs without index

It's reasonable to set up an index for every search part: anchor, link.
Test query: SELECT anchor FROM link_anchor WHERE link = 'Indonesian

language';

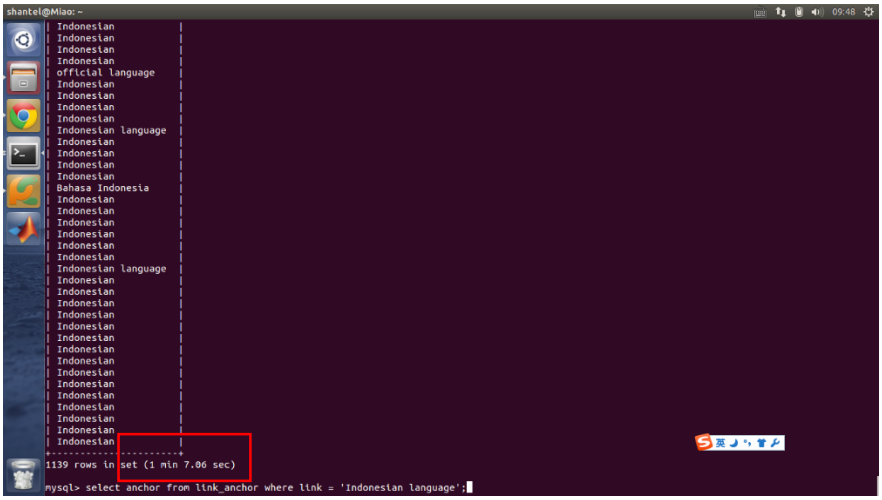


Figure 4 Before indexing

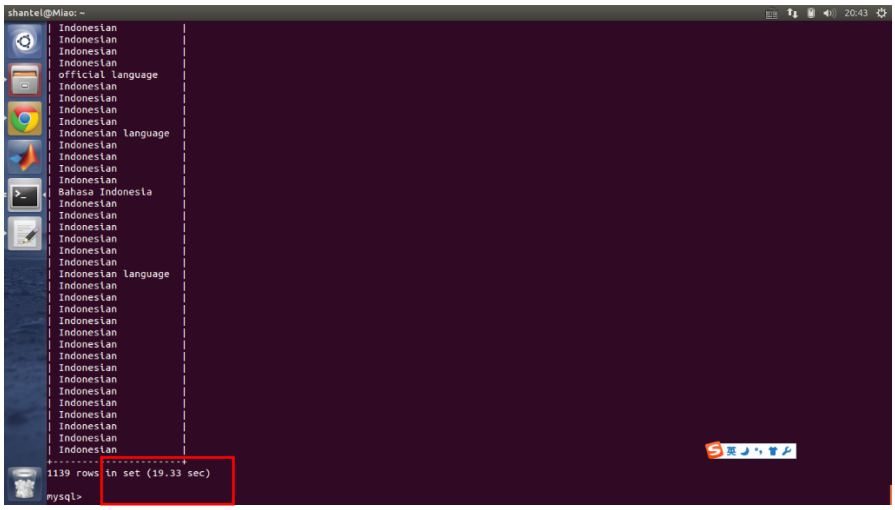


Figure 5 After indexing

6. CONCLUSION

Through this project, we have a deeper understanding of database and learn to use mysql. Mysql is widely used in different platforms such as windows and linux and we can also operate on it in many forms like GUI, command line or through programming language like python, java and so on. We learn how to design a database and ER model and how to optimize our design. Our product is still not perfect, we will keep trying to make it better.

Work allocation:

Xinting QIU: Design, mysql, process data, report

Zhehui Zhang: GUI, process data, report