Advances in Computer Science Research, volume 92

International Conference on Mathematics, Big Data Analysis and Simulation and Modeling (MBDASM 2019)

Design of Network Database with Search Functions on Internet

Zhao Li and Jianping Sun*

School of Electrical and Electronic Engineering North China Electric Power University, Beijing 102206, China

*Corresponding author

***Abstract*—Nowadays, with the rapid increase of network resources, traditional search engines can no longer meet people's needs. Therefore, the network crawler as an important part of the search engine is more and more important. This paper introduces the concept and classification of network crawler, how to use Python language to write the basic network crawler, and how to combine the crawler with the database.**

Keywords—Python; database; web crawler; search engine

I. INTRODUCTION

Nowadays, our society has entered the information age. With the rapid development of Internet, the search engine is gradually unable to adapt to the development of society. There are two main problems faced by search engines today: first, how to adapt to the rapid development of the Internet. Two, how to provide users with more accurate search information under the premise of ensuring search speed. The traditional search engine can't solve the above problems perfectly. The new generation of intelligent search engine has become one of the hottest research topics of social development because it involves artificial intelligence, large data mining, database, distributed processing, natural language processing and so on.

Web crawler is a program or script that automatically grabs the World Wide Web information according to certain rules. It can be roughly divided into the following types: General Purpose Web Crawler, Focused Web Crawler, Incremental Web Crawler and Deep Web Crawler.

Python is an object-oriented programming language. Python has a wide range of applications in cloud computing, but also has great advantages in large data processing, such as relatively small code, faster development speed, rich data processing packages, low use of internal types, no need for additional operations, with a framework for processing large data, in coding Handling problems in the process is relatively convenient. It not only has a wide range of applications in cloud computing and data processing, many large websites are also developed with Python, such as YouTube, Instagram, Python is also the main language used by many large foreign companies (such as Google).

We will use Python language to write the basic network crawler, so that we write the network crawler and database together, to achieve our goal of accurate data capture..

II. WEB CRAWLER DESIGN, WRITING, DATABASE COMBINATION

A. Compilation of Web Crawler

1) Principle

The network crawler is a piece of computer code segment that automatically extracts web pages. It downloads Web pages from the internet Web pages and is an important component of search engines. A crawler starts with a starting seed link, sends an HTTP request for the link, gets the content of the link, then mostly regularly matches the valid links in the page, and then saves the links to the queue to be visited, waiting for the crawler thread to fetch the queue to be visited. Once the link has been visited, in order to effectively reduce the unnecessary network We have placed the visited links in the visited Map to prevent repeated fetching and dead loops. This is just a relatively simple crawler implementation, and there are more complex crawler implementations, such as proxy servers, masquerading as browsers, logging in and extracting authentication codes.

Common techniques for writing Crawler

2) Basic Methods

Using the request command in the urllib module of Python.

```
from urllib import request
response=request.urlopen
("http://www.cnki.net/")
content = response.read().decode
('utf-8')
print(content)
```

3) Using Proxy Server

For the anti-crawl mechanism of the website, you need to use a proxy server to continue to crawl the required information by using different proxy servers.

4) Cookie Processing.

For sites with a slightly higher security level, the first two methods are unable to crawl data. These websites need to provide cookie information when sending URL requests, otherwise they will not be able to request success

5) Disguised as Browser

When accessing the website has anti-reptile mechanism, we set the browser information (disguised as a browser) in the request, by modifying the http packet header.You can disguise as a browser by setting up browser information in headers and placing headers in request requests.

6) Sign In

When using a crawler to log in, it is actually a simulation of the browser to send a login request, putting the user name and password needed for login into the request data.

B. Design of Reptiles

1) Determining URL Format

We generally divide the URL into two parts, one is the basic part, the immutable part; the other is the parameter part, the variable part. For example, the address we see is: http://kns.cnki.net/kns/brief/default_result.aspx.The URL can be divided into the base part http://kns.cnki.net/kns/brief, and the parameter part: default_result.aspx.

2) Page Grabbing

Write page grabbing program in object-oriented coding mode. To get a page, we need to know how many records to look for each time, starting with which records. In this method, a cycle is needed, and the required records are fetched through the loop. The code is as follows:

Class Paper(object):

```
def__init__(sellf):
sellf.start = 0
sellf.param='default_result'
Self.headers={'UserAgent':'Mozilla/5.0
(Windows NT 6.1;WOW64)'}
Def get_page(self):
page_content=[]
try:
while sellf.start<= 225:
url='http://kns.cnki.net/kns/brief/
default_result.'+str(sellf.start)
req=request.Request
(url,headers=sellf.headers)
response=request.urlopen(rep)
page=response.read().decode('utf-8')
page_num=(sellf.start+25)//25
print('Grabbing'+str(page_num)+'Page
 data...')
Sellf.start+=25
Page_content.append(page)
Return page_content
except request.URLError as e:
if hasattr(e,'reason')
print('Crawl failure,
failure reason: ',e.reason)
def main(sellf):
print(
'Start to grab data from cnki.net.')
sellf.get_page()
print('Data crawling completed')
```

3) Extract Relevant Information

The re module in Python provides us with a compile function that helps us convert regular expression syntax into regular expression objects. In the Paper class, we can use an object-oriented approach to extract a method for HTML text parsing, so as to parse out what we care about from the text. In this class we define a method in the form:

```
def get_movie_info(sellf):
pattern = re.compile
(u'<div.*?Class="item">.*?'....)
```

4) Write File

Write a method dedicated to the write operation of the file, making the program object-oriented. Define a method in the Paper class:

```
def write_text(sellf):
print(
'Start writing data to the file.....')
file_top=open(
sellf.file_path,'w',encoding='utf-8')
```

C. Combination of Database and Crawler

Design database search function. Here we will introduce the Bloom Filter.The Bloom Filter is a common algorithm. Its purpose is to filter out those elements that are not targets. That is to say if a word to be searched does not exist in my data, then it can return to the target at a very fast rate does not exist.

The next thing we want to achieve is the word segmentation. The purpose of the word segmentation is to divide our text data into the smallest unit that can be searched. Here we mainly focus on English, because Chinese word segmentation involves natural language processing, which is more complicated, and English is basically just punctuation.

The main segmentation uses spaces to separate words. In the actual word segmentation logic, there are other separators. The secondary segmentation is similar to the logic of the main segmentation, except that the results from the beginning to the current segmentation are also added. The logic of word segmentation is to divide the text first, and to divide each of the main segments. Then return all the words that were separated.

The last thing we have to do is the search function.

Splunk represents a collection of indexes with search capabilities. Each collection contains a Bloom Filter, an inverted vocabulary (dictionary), and an array of all events. When an event is added to the index, the following logic is done:

(1) Generate a unique ID for each event, here is the serial number.

(2) Segmentation of the event, adding each word to the inverted vocabulary, that is, the mapping structure of the ID of the event corresponding to each word. Note that a word may correspond to multiple events, so the value of the inverted table is a set. The inverted list is the core function of most search engines.

When a word is searched, the following logic is done:

(1) Check the Bloom Filter, if it is false, return directly.

(2) Check the vocabulary, if the searched word is not in the vocabulary, return directly.

(3) Find all corresponding event ids in the inverted list, and then return the contents of the event.

The code is as follows:

```
class Spllunk(object):
def __init__(sellf):
sellf.bf = Bloom filter(64)
sellf.terms = {}
# the term Dictionary to set
sellf.events = []
def add_event(sellf, event):
 """Adds an event to this object"""
 # build and save a unique event ID
event_id = len(sellf.events)
sellf.events.append(event)
 # Add terms to the bloom filter,
and track the event by each term
for term in segments(event):
sellf.bf.add_value(term)
if term not in sellf.terms:
sellf.terms[term] = set()
sellf.terms[term].add(event_id)
def search(sellf, term):
"""Search for every term, and yield
all the events that contain it"""
 # In Spllunk this runs in O(1),
and is possible to be in filesystem
cache (memory)
if not sellf.bf.might_contain(term):
 return
 # In Splunk this possibly runs in
O(log N)where N is the number of
terms in the tsidx
if term not in sellf.terms:
return

for event_id in sorted
(sellf.terms[term]):
yield sellf.events[event_id]
```

III. SUMMARY

This design is based on the Python language, using the MySQL database as the underlying database, using a Bloom filter and Splunk collection technology to complete the design of a database with secondary search capabilities.

After the design is completed, the database is tested and has basic add, delete, update, query and search functions. The design result has achieved the goal set in advance.

ACKNOWLEDGMENT

The authors acknowledge the support of graduate student course project and double-first-rate construction project in NCEPU.

REFERENCES

[1] Mark Lutz.Python Study manuals [M]. Li Jun, Liu Hongwei, translated version.4. Beijing: Machinery Industry Press,2011.

[2] Doug Hellmann.Python Standard Library [M]. Liu Chi, translated. Beijing: Mechanical Industry Press, 2012.

[3] Paul Barry.Head First Python[M].Lin Qi, Claire, translated. Beijing: China Electric Power Press, 2012.

[4] Qi Peng,Li Yinfeng,Song Yuwei.Web Data Acquisition Technology Based on Python[J]. Electronic Science and Technology, 2012, 25 (11):118-120.

[5] Zhang Chao,Yan Hongyin.Design and Implementation of Multi-threaded Web Crawler [J].Computer Development & Applications, 2012, 25 (6): 65-70.

[6] Du Yajun,Yan Bing,Song Liang.Design and program implementation of reptile algorithm[J].Computer Applications, 2004, 24 (1):33-35.

[7] Ni Xiangui,Cai Ming.Focused crawler system based on link structure and content similarity[J].Computer Engineering and Design, 2008, 29 (7): 1709-1763.

[8] Hu Wei. Web mining application based on web crawler [J]. Software,2012,33 (7): 145-147.

[9] Zhou Lizhu,Lin Ling.Summary of research on focusing reptile technology[J]. Computer Applications, 2005, 25 (9): 1965-1969.

[10] Ritika Hans, Gaurav Garg.Web Crawlers and Search Engines[J]. International Journal of Engineering Sciences and Research Technology, 2013, 2 (6): 1545-1551.

[11] Prashant Dahiwale, M.M. Raghuwansh, Latesh Malik.Design of Improved Web Crawler By Analysing Irrelevant Result[J]. International Journal of Computer Science and Mobile Computing, 2013,2 (8): 243-247.

[12] Dhiraj Khurana, Satish Kumar.Web Crawler: A Review[J]. International Journal of Computer Science and Management Studies, 2012, 12(1): 401-405