Cyberminer: A web search Engine

Tingwei Wu
Computer Engineering
University of Texas at Dallas
Richardson,US
txw190006@utdallas.edu

Churong Zhang
Computer science
University of Texas at Dallas
Richardson,US
cxz173430@utdallas.edu

Feifei Guo
Computer science
University of Texas at Dallas
Richardson,US
fxg180009@utdallas.edu

Qixiang Wang
Computer science
University of Texas at Dallas
Richardson,US
qxw200000@utdallas.edu

Abstract— Cyberminer, which is a simple web search engine, shall accept a list of keywords and return a list of URLs whose description contain any of given keywords. By implementing this project, Object-Oriented Analysis and Design (OOAD) is applied. This document furnishes the Use Case diagrams, Class diagrams, and Sequence diagrams using UML.

Keywords— Cyberminer, Search Engine, OOAD

I. Introduction

A search engine is a software system that is designed to carry out web searches. They search the World Wide Web in a systematic way for particular information specified in a textual web search query.[1] Web search engines get their information by web crawling from site to site. The "spider" checks for the standard filename keywords.txt, addressed to it. The keywords.txt file contains directives for search spiders, telling it which pages to crawl and which pages not to crawl. After checking for keywords.txt and either finding it or not, the spider sends certain information back to be indexed depending on many factors, such as the titles, page content, JavaScript, Cascading Style Sheets (CSS), headings, or its metadata in HTML meta tags. For Cyberminer, the spider sends back the title, URL, and page content. These data will be stored in our MySQL database. When a user enters a query in Cyberminer, the search engine will search the database and return a list of results which either its title or its description contains the given keywords. The result can be shown by pages and ascending alphabetical order.

II. Domain modeling and Requirements

A. Functional requirements

As required, the Cyberminer shall include the below 9 functional requirements.

FR1. Case sensitive search.

FR2. Autofill, while correcting typographical errors

FR3. Specifying OR/AND/NOT Search

FR4. Filtering out symbols that are not meaningful, according to the user configuration

The above four functional requirements are all about the search query, or input keywords. We can combine them into one use case.

FR5. Multiple search engines. Since we use MySQL database, it naturally satisfies this function.

FR6. Setting the number of results to show per page, and navigation between pages.

FR7. Listing of the query result in ascending alphabetical order; most frequently accessed order, or per payment.

FR6 and FR7 are both concern on the search result listing. They can be combined into one use case: search.

FR8. Hyperlink enforcement. When user clicks on one URL, which has been retrieved as the result of a query, the system can take the user to the corresponding web site. To achieve this function, the system's database should contain URL information of each item, and the URL information shall be returned to page to enforced hyperlink. This also can be one use case.

FR9. Deletion of out-of-data URL. We can set check mechanism in a given frequency, such as one day or one week. It will check if the URL stored in database is or not out of date. If it is out of data and cannot reach again, then delete it. This function has nothing to do with user. This is the server-side work. Therefore, this function should be a single use case.

B. Nonfunctional requirements

Cyberminer shall also satisfy these below nonfunctional requirements:

NFR1 Easily understandable/User friendly
NFR2 Portable
NFR3 Responsive
NFR4 Good performance

For NFR1, it means the interface should be simple. Thus, we use a graphical interface to make the user experience better. The UI design is very simple and similar to google search engine.

For NFR2, it means Cyberminer should be machine independent. We use python to implement because python is independent of any Operating system or platform.

For NFR3, it means the system should be interactive, proper error messages should be displayed to enable user understanding the behavior of the system. In case of error, the system must instruct the user to carry out corresponding steps to eliminate or detect the cause of error. For example, to remind user: the input is not valid. We use warnings to interact with user.

For NFR4, the system should have good performance, which means the system should be efficient. Therefore, our algorithm has to minimize the time and space complexity, response time should be as quick as possible. We have done our best, our system has good performance as tested.

C. Domain modeling

Fig.1 is this system's domain model. It contains 4 parts: user, web search engine, server, and database. The main process is like this: a public user opens the web search engine, then inputs the query string into the web search engine, next the sever gets the request and search the database. Data items in the database include title, description, and URL. When the server gets the search results from the database, it returns the results to web search engine. The web search engine gets the results and shows it in pages as designed.

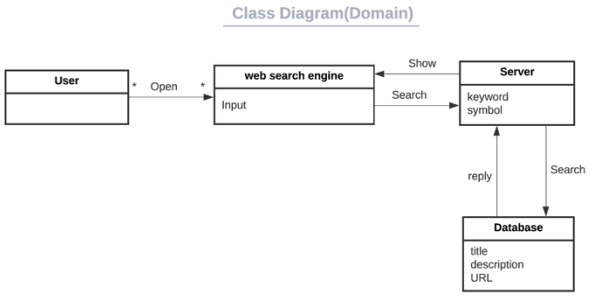


Fig.1 Class Diagram (Domain)

III. Design Specification

A. Use Case Diagram

By analyze the functional requirements, this system shall include 4 use case: press search, show list, open URL, and delete of out-of-date URL. Fig2 shows the main use case diagram of Cyberminer. There are two actors: public users and server. Fig.3-6 are the use cases separately for this system.

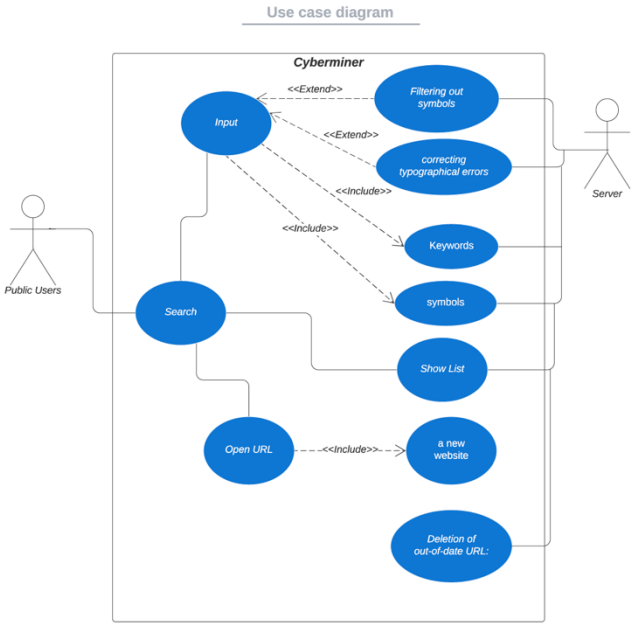
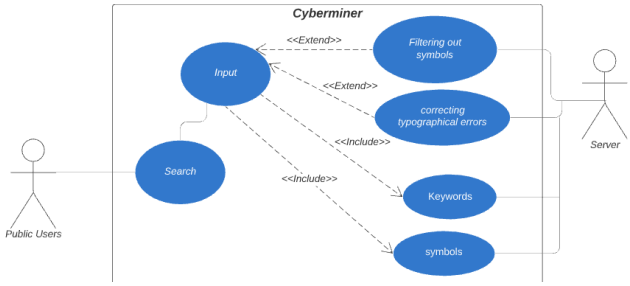
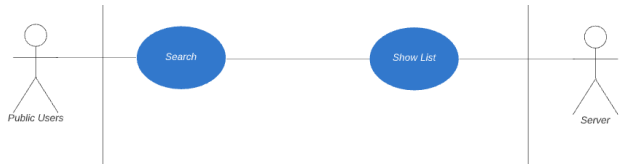


Fig.2 Use Case Diagram



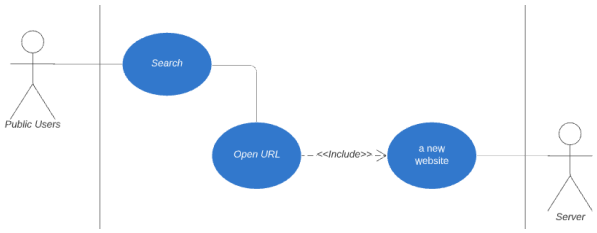
Name	Show List
ID	UC1
Actors	User
Description	Input String in the search bar
Main Course	1.Search all String in the database 2.Show the string list including input string
Alternate Course	AC1:<input wrong string> 1.Show errors and the most similar string AC2:<input symbols without in user configuration> 1.Show errors

Fig.3 Use Case 1



Name	Press Search
ID	UC2
Actors	User
Description	Click Search after input String
Main Course	1.Open a new page 2.Show the string list including input string 3.Show number of results and have previous page and next page buttons.
Alternate Course	NA

Fig.4 Use Case 2



Name	Open URL
ID	UC3
Actors	User
Description	Open URL
Main Course	Open a new websit
Alternate Course	NA

Fig.5 Use Case 3

Name	Deletion
ID	UC4
Actors	Server
Description	Deletion of out-of-date URL
Main Course	Deletion of out-of-date URL
Alternate Course	NA

Fig.6 Use Case 4

B. Class Diagram

The system has four classes: user, web search engine, server, and database. Fig.7 shows the whole class diagram of Cyberminer.

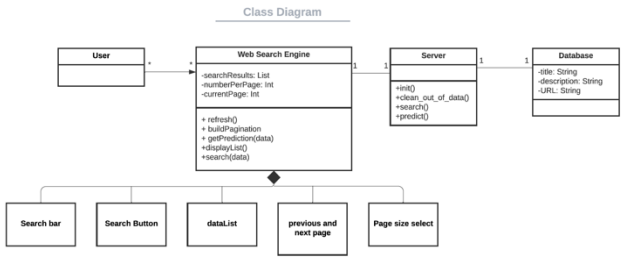


Fig.7 Class Diagram

For class web search engine, it has three attributes: list which is the whole search result, a integer to control the number items of per page, and list of search result for current page. Web search engine mainly has five operations: refresh, build pagination, get Prediction (when input query string), display result, search. This web search engine class include 5 parts: search bar for user to type query string, search button to click to search, dataList to show search result, previous and next page to control page browse function, page size to control the number of items of each page.

The sever class has 4 operations, initiation, clean out-of-date data, search, predict. For initiation, it can check the database whether it is empty or not, if no, it can upload data into the database. Clean out-of-date data can delete the out-of-date URLs from the database regularly.

C. *Sequence Diagram*

From the use case diagram and class diagram, we have the sequence diagram, shows in Fig.8. Sequence diagram displays object interactions arranged in a time sequence. In fig.8 we can see flow of events and the interaction between user, cyberminer, server, and database.

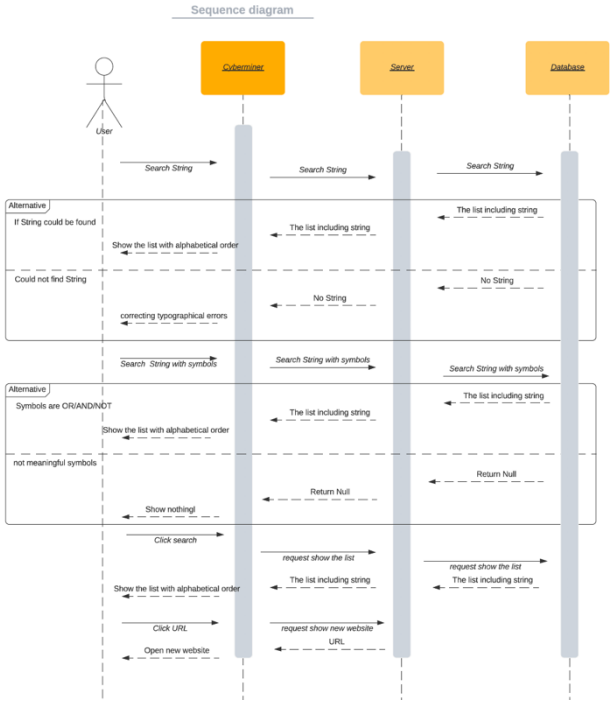


Fig.8 Sequence Diagram

D. *State Transition Diagram*

The state transition of Cyberminer is shown in fig.9. When user open Cyberminer, its state is our system UI initial interface, waiting text input. When user is ready to search or already done, user can still re-enter keywords to start a new search loop.

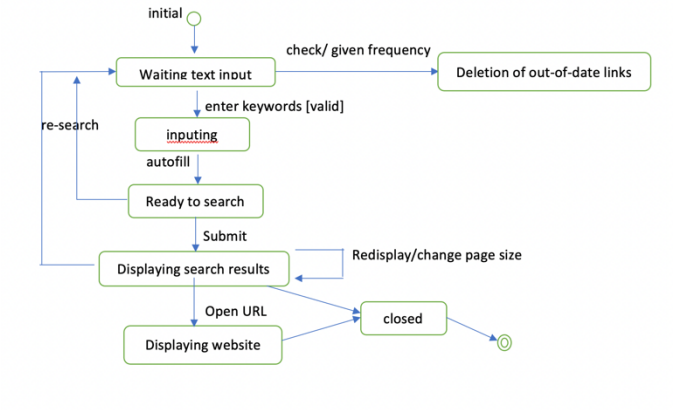


Fig.9 State transition diagram

E. *Activity Diagram*

The activities of Cyberminer is shown in fig.10. We use swim lanes and object flow because each activity is exactly belonging to user or server.

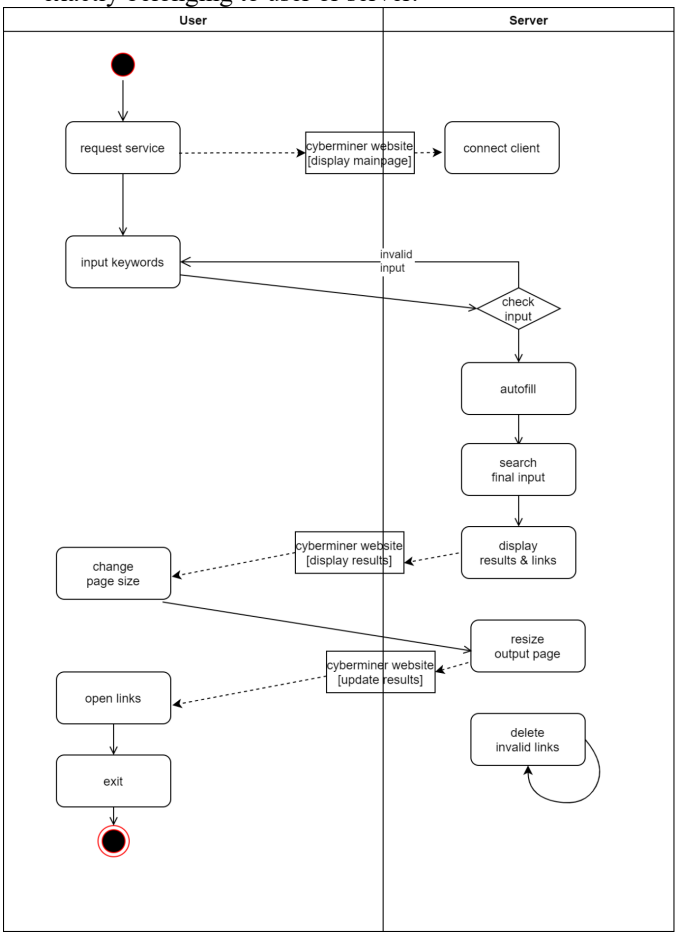


Fig.10 Activity Diagram

IV. Prototype Implementation

We use python flask + MySQL to implement this project. Before my start our main implementation, we web crawling to gather the URL data and saved as .csv files. The program can load the data from csv files to the database which should be created previously. Fig.11 is the traceability matrix for the functional requirements we have achieved.

Traceability Matrix				
	UC1	UC2	UC3	UC4
FR1	X			
FR2			X	
FR3				X
FR4	X			
FR5		X		
FR6	X	X		
FR7	X			
FR8	X			
FR9	X			
NFR1	X	X	X	
NFR2	X	X	X	
NFR3	X	X	X	
NFR4	X	X	X	

Fig.11 Traceability Matrix

Fig.12-15 is our system snapshots. Fig.12 shows our system can autofill and predict user's query request.

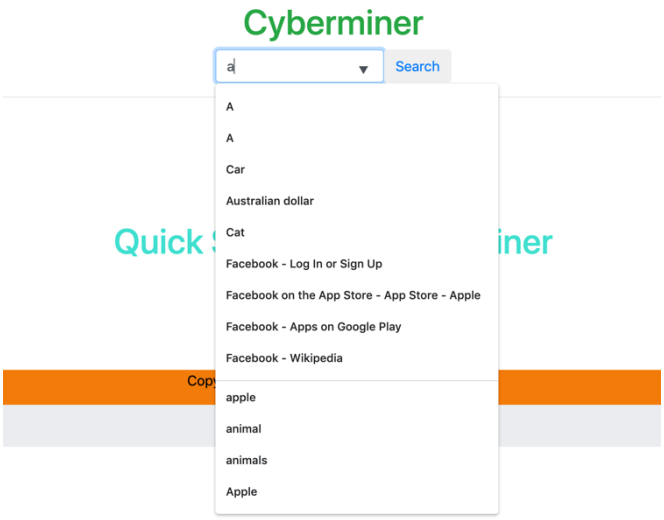


Fig.12 Input query string

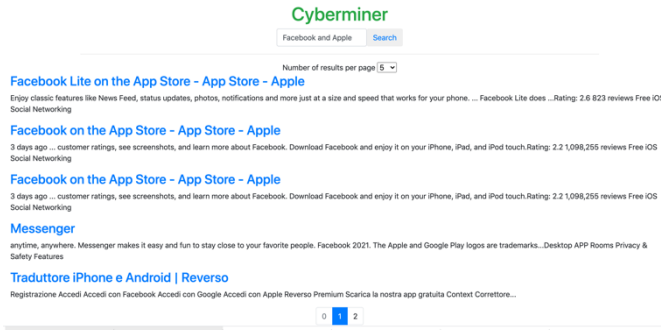


Fig.13 "And" search and results in pages

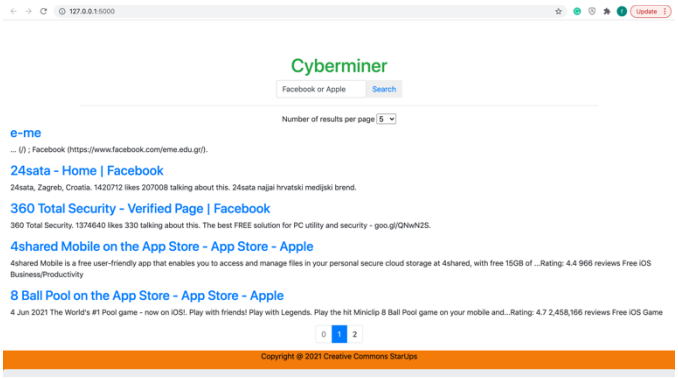


Fig.14 "OR" search and results in pages

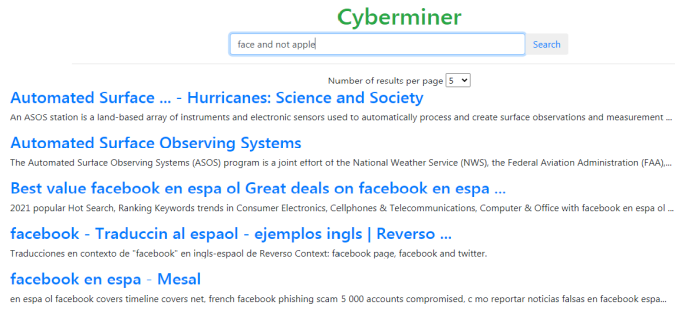


Fig.15 "NOT" search and results in pages

Fig.13-15 shows the results when search query is specified with "And", "OR" and "NOT" respectively. The search results are list in pages, we can change the number of items to list on each page and navigate between pages.

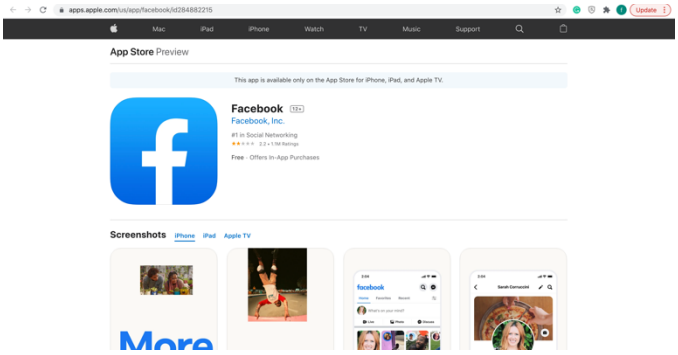


Fig.16 Open URL and hyperlink

Fig.16 shows the hyperlink enforcement requirement. When a user opens one search result, the system shall take the user to the corresponding web site.

V. Conclusions

All of the functional and nonfunctional requirements have been enforced.

In future, we can improve in two aspects. First, enlarge the database. Then the search result will be more effective. Second,

rate the search result in more effective way. Rather than listing results in ascending alphabetical order, it shall be listing in as ascending order of correlation index and user frequency.

VI. User Manual

To start Cyberminer, user should first set the database. User can either set the MySQL in cmd or its panel, or use a MySQL server, shown in fig.17. Then create a database: cyberminer. When all these are prepared, go to the src folder, execute code, run: python app.py, just like in fig.18. Cyberminer will be started. Go to follow link, it opens the homepage of Cyberminer.

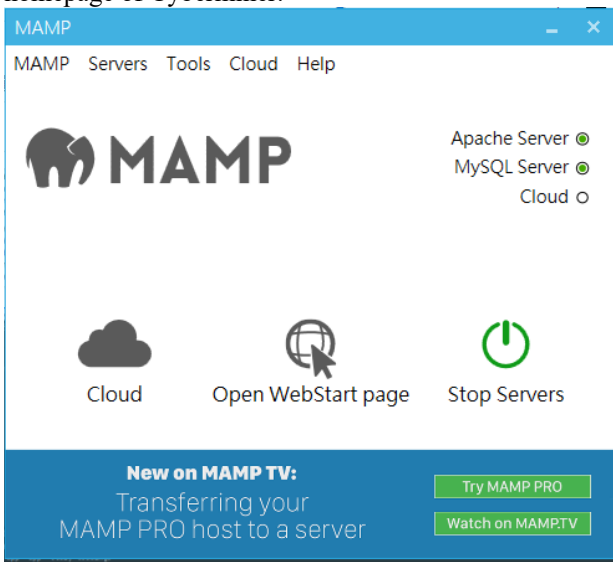


Fig.17 MAMP MySQL Server

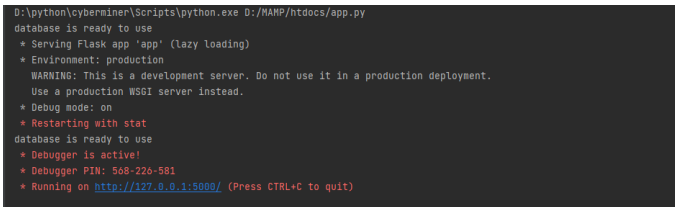
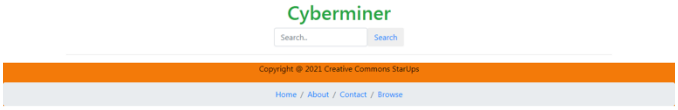


Fig.18 execute app.py

In the Cyberminer homepage, user can input keyword to search. The keyword is case-sensitive. What's more, user can use "AND", "OR"" NOT" to specify logic search.



References

[1] https://en.wikipedia.org/wiki/Search_engine