iSearch+: A Smart News Search Engine

EEL6761 Cloud Computing and Storage - Midterm Report

Yifan Wang
dept. Computer & Information Science & Engineering
University of Florida
Gainesville, United States
wangyifan@ufl.edu

Jingyang Guo
dept. Computer & Information Science & Engineering
University of Florida
Gainesville, United States
jingyang.guo@ufl.edu

Qi Jiang
dept. Computer & Information Science & Engineering
University of Florida
Gainesville, United States
q.jiang@ufl.edu

***Abstract*—This is the midterm report of the iSearch+: A Smart News Search Engine project. As we proposed in the proposal paper, iSearch+ is a search engine software system aiming to search information from news articles from major news websites and conduct data mining analysis. In section 1 and 2 We will further propose the project and illustrate the motivation. A naive demo has been finished by now, in section 3 we will introduce the system architecture, including basic components and relationships among them. Section 4 shows our system design as well as implementation, illustrating the main ideas and fundamental tools or algorithms we are using. After introducing the system, we performed experiment on the system and the evaluation results are presented in section 5. At last the current progress, our team coordination, along with further plans on the project will be presented in section 6.**

***Index Terms*—search engine, fuzzy search, Spark, data crawling, data mining, inverted index, page ranking**

I. Introduction and Motivation

Nowadays, the amount of information on world wide web is growing even fast. In order to get information efficiently, users need to use search engine. The search engine [1] refers to a system that automatically collects information from the Internet and, after some sorting, provides the user with a result of query. The information on the Internet is vast and unordered. All the information is like a small island on the ocean. The web link is a bridge between these islands, and the search engine draws a clear picture for the user. Information map for users to check at any time. It is obvious that search engine is extremely useful and important in big data field.

II. Related Work

Starting from 1998, when Sergey Brin and Larry Page published their paper "The anatomy of a large-scale hypertextual web search engine.", and introduced Google to the world, an era of search engine has lifted its curtain. Search engine has always been a hot topic because of the high requirement for it, and plenty of search engine products has come out since then.

In 2006, Yuangui Lei and his team proposed SemSearch, which is a search engine for the semantic web. Until the paper was published, most existing tools are mainly focused on enhancing the performance of traditional search technologies, aiming to serve those users who are familiar with semantic data, while to some extend ignoring common users who only care about using search results. So they developed the search engine which has such architecture: take a Google-like user interface as the outermost layer, collecting user input, put the input through the text search layer (semantic search), then query for the processed objects to be searched from the World Wide Web or databases.

The iSearch+ system is partly most like SemSearch system, however, the SemSearch is a purely search engine, lacking the functions of data mining. While our project, iSearch+, adds a data mining layer which can deal with the information queried from the news webs, and is able to present those information after analysis to the user, along with semantic searching results.

In section 3, we will further produce the architecture of iSearch+ search engine system.

III. System Architecture

iSearch+ consists of four main layers (Fig. 1): front end pages, web server (back end), parallel computing layer and storage layer.

Front end is UI for users to input what they want to search and get responses. That is what displayed to users.

Web server, or back end, is pivot of the whole system. It receives input from front end, preprocess the input and send it to lower level components. Then it retrieves search results from the lower level components and respond to front end with the results. Web server is deployed on AWS EC2.

Parallel computing layer has two main parts, searcher and crawler. Searcher plays the main role in searching while crawler takes responsibility of retrieving up-to-date news ma-

terials periodically from the Internet. This layer is deployed on AWS EMR.

Storage layer is a MySQL database located on AWS RDS, which stores news articles, inverted index and other materials we need.

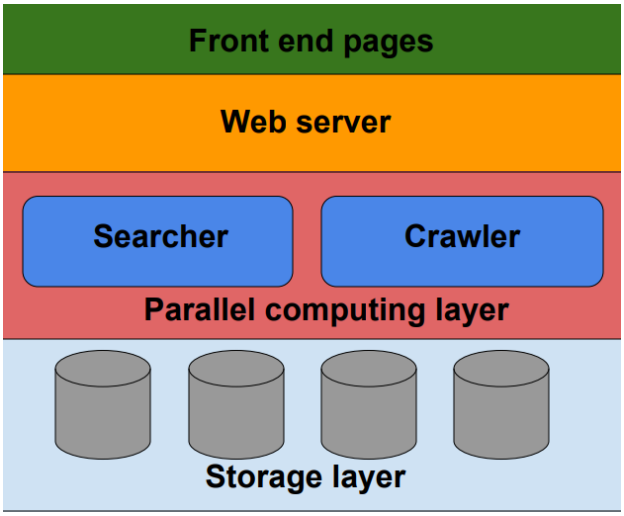

Fig. 1. Main Layers

Basic relationships of each search system component is shown in Fig. 2.

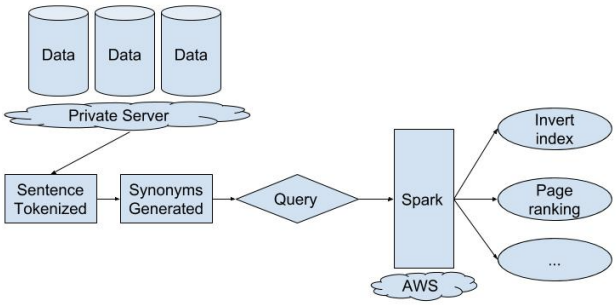

Fig. 2. Architecture of Search System

IV. System Design and Implementation

A. Front End Pages

iSearch+ is about information search. It allows users to explore the news with flexibility and they can find plenty of results which they interested in. The GUI of iSearch+ is mainly built in HTML, CSS and Javascript. We will accomplish following functions in our application and introduce them in details.

1) Main Search Board: First of all, we designed a main search board for our application like Figure 1. For all the users, search and graph function is allow to be used. There is a sign-in button at upper right corner and user could go to sign-in page for more private search history and personal favorite collection. User could search any news when they enter related keywords into search bar. For example, if user is concerned about news of stock, just enter a few keywords like: stock or NASDAQ, the application will go to result dashboard and show all of the news related to news in news database. More detailed function of listed news will be introduced in section search result dashboard.

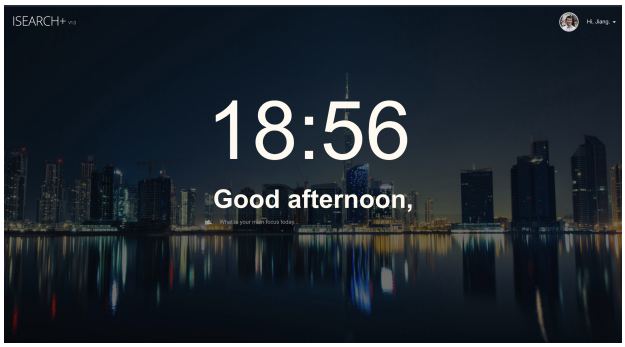

Fig. 3. Main Search Board of Search System

Additionally, a real-time clock and correspond greeting function are implemented in the center of search board for better interface design and usage.

2) Sign-in and Sign-up Page: User need to register at the first and have account for the better usage of our application. As it mentioned above, after registration and sign-in, user is accessible for their personal search history and interested news management. Consequently, sign-in page is designed and implemented as Figure 2.

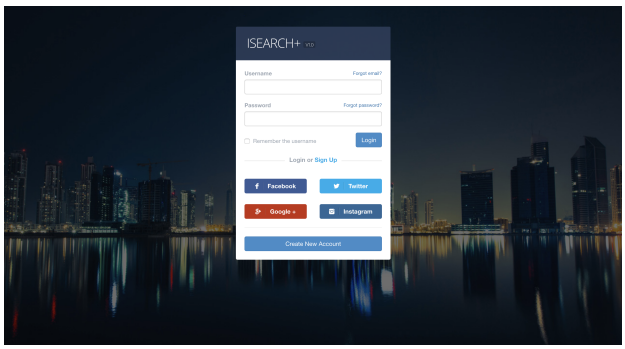

Fig. 4. Sign-In Page of Search System

Also, a sign-up page is made for easy registration like Figure 3. After typing register button on the Sign-In Page, it will direct user to Register Page. Users have to enter their e-mail, password and confirm password while registering. Used e-mail cant not be approved for registering new account. After user type register button, if use cant register successfully, the web-page will show why and user need to fix the problem. If user registered successfully, it

will direct user to Sign-In page and show that user register successfully. User need to type email and password for sign-in.

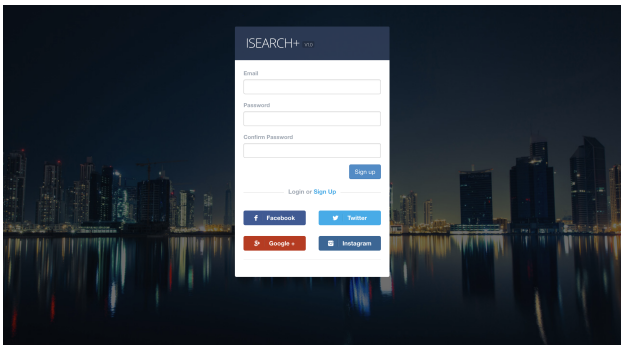

Fig. 5. Sign-In Page of Search System

3) Result Dashboard: Now, we will introduce the most important part of GUI in our application. After type search keywords in search bar, web page will direct into result dashboard. The result dashboard is consist of several important components. First of all, there is a search bar inside the result page in which user can do search immediately without go back to main search board. In the left side menu, user could get access to their favorite news collection in "Favorite" icon, search and view history in "History" icon, there are also some additional information shown in chart in "Stock" and "Chart" icons. In main page, popular news and hot tag keywords are shown. User can just go into popular news or focus on hot keywords to get fresh and important information. In the middle of web, search result news are listed, user can go to next or previous page to check all the result.

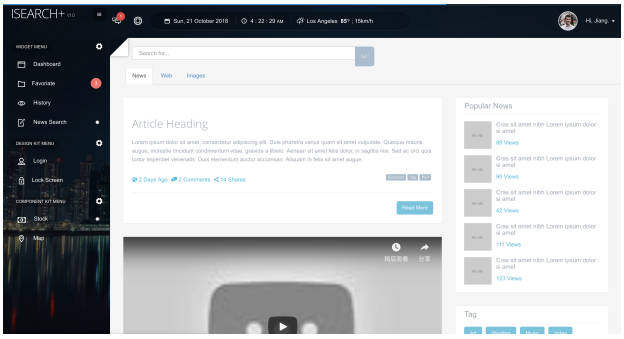

Fig. 6. Result Dashboard of Search System

B. Web Server

1) Web Framework: We choose Python Tornado as web framework for web server. The reason why we choose Python Tornado is that Tornado is an asynchronous framework. The day it started to support asynchronous is even earlier than the time when Python started to support asynchronous officially. This means Tornado can support thousands of connections at the same time, which is a huge advantage comparing to other Python web framework like Flask and Django.

2) Reverse Proxy: We deploy a reverse proxy using Nginx. Nginx is a light weight web server comparing to Apache server. So it is more suitable to act as a reverse proxy.

A reverse proxy is very important for most websites, especially for those which need dense computing, because reverse proxies can provide better isolation, easier maintenance, simpler management and higher level security.

First, we deploy front end page files on the reverse proxy so that the main server needs only to care about back end computing, which means we separate front and back end more clearly.

Second, since we isolate front end from back end, it is much simpler to maintain both ends now.

Third, another critical task of a reverse proxy is balancing load on web servers. When we have more than one server, we need to balance load on each of them to improve their performance. That is, we are supposed to move some work from heavy-loaded servers to light-loaded servers. At this time, the reverse proxy acts as a server cluster manager which monitors all nodes' load status and redirects data stream from those with heavier load to those with lighter load.

Finally, a reverse proxy can offer the system a higher level of security. The data stream from clients will always get through the proxy first and then arrive at actual servers. So proxy can monitor and filter dangerous data stream before it harms our servers. Even if the evil clients attack the proxy successfully, they cannot get access to any valuable data since the data are stored only on actual servers instead of on the proxy.

C. Parallel Computing Layer

This layer is the core of the whole system. Major works are done in this layer.

1) Crawler: Crawler is the tool for retrieving information from the Internet. Users can access online resources by web browser. But if we want to extract main information automatically from web, we need crawlers.

For fetching online information, a crawler will construct request just like what a browser does at the very beginning. But a crawler will not request for trivial files like CSS and JavaScript files. It only request the files containing the texts, images or other information that it needs.Then it parses the fetched files to extract what it needs and store the information.

Crawler is a very typical case of parallel computing. Most of time, the information we want to crawl has no relationship with each other. In our search engine, we need as many news articles as possible. Each news article is independent from other articles. Whether we have gotten one article has nothing to do with other articles.

As a result, multi-threaded or distributed techniques can improve crawlers' performance significantly.

2) Searcher: Searcher works on searching inverted index and sorting searching results.

First, searcher will look up inverted index table to find which documents the keywords are in. Then it sorts the documents and returns them to users.

This is also a typical case of parallel computing. Since different documents have nothing to do with each other, we can search them parallelly. So we choose Spark for the searcher. Spark SQL is a component of Spark which is designed for parallel operations on relational database. The searcher shows a good performance by using Spark and Spark SQL.

D. Storage Layer

This is a MySQL database deployed on AWS RDS. It is a data exchange center for the system, storing inverted index table, documents, crawled web data and users' data to be queried by server back-end.

E. Working Flow

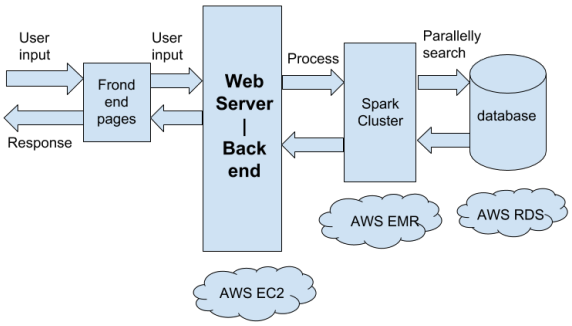

Fig. 7. Main Working Flow

As shown in Fig.6, user inputs search keywords and front end passes them to back end server. Web server preprocesses them such as split query string into separated words, then searchs the keywords in database and returns results.

V. Current Progress and Project Management

A. Current Status

Completed crawling, search engine and front-end parts; functionally tested the project. Data mining algorithms and functions remaining to be completed. First demo has been completed and configured on AWS.

B. Team Coordination

1) Yifan Wang: Data crawling, back-end programming, AWS configuration, documentation.

2) Jingyang Guo: Algorithms, data mining, documentation.

3) Qi Jiang: UI design, front-end programming, documentation.

C. Milestones, Weekly Plans, and Deliverables

1) Milestones: Finish the first stable version demo by midterm, finish data mining part 2-3 weeks later, completely finish the project by the end of term.

2) Weekly Plans: Week Oct. 22 - Oct. 28: Completely develop and implement current functions, prepare for further development. Week Oct. 29 - Nov. 4: Finish the data mining components. Week Nov. 5 - Nov. 12: Completely develop and implement data mining algorithms and relative functions. Week Nov. 13 - Nov. 20: Functional test and performance evaluation experiments. Week Nov. 21 - Nov. 28: Finish documentation and fix minor problems.

3) Deliverables: Codes, documentation, screenshots, slides, report paper.

References

[1] Brin S, Page L. The anatomy of a large-scale hypertextual web search engine[J]. Computer networks and ISDN systems, 1998, 30(1-7): 107-117.

[2] Thelwall M. A web crawler design for data mining[J]. Journal of Information Science, 2001, 27(5): 319-325.

[3] Paramkusham S. NLTK: THE NATURAL LANGUAGE TOOLKIT[J].

[4] Sauls C, Leithead T, Tokumi R. Synonym and similar word page search: U.S. Patent 7,822,763[P]. 2010-10-26.

[5] Zaharia, Matei, et al., "Spark: Cluster computing with working sets.", HotCloud 10.10-10 (2010): 95.

[6] Isele, Robert, et al. "LDspider: An open-source crawling framework for the Web of Linked Data." 9th International Semantic Web Conference (ISWC2010). 2010.

[7] Kreps, Jay, Neha Narkhede, and Jun Rao. "Kafka: A distributed messaging system for log processing." Proceedings of the NetDB. 2011.

[8] Ranjan, Rajiv. "Streaming big data processing in datacenter clouds." IEEE Cloud Computing 1.1 (2014): 78-83.

[9] Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis." Foundations and Trends in Information Retrieval 2.12 (2008): 1-135.

[10] Alexander, Carol. Market models: A guide to financial data analysis. John Wiley & Sons, 2001.

[11] Brin, Sergey, and Lawrence Page. "The anatomy of a large-scale hypertextual web search engine." Computer networks and ISDN systems 30.1-7 (1998): 107-117.

[12] Lei, Yuangui, Victoria Uren, and Enrico Motta. "Semsearch: A search engine for the semantic web." International Conference on Knowledge Engineering and Knowledge Management. Springer, Berlin, Heidelberg, 2006.