Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 3, March 2015, pg.323 – 327

RESEARCH ARTICLE

INCORPORATING REAL TIME WEB SEARCH IN LOCAL SYSTEM

Mrs. S. UMADEVI@YASODHEI[1]
Senior Assistant Professor, Computer Science and Engineering, IFET College of Engineering, Villupuram
mamahima@gmail.com

Miss. A. GAYATHRI[2]
UG Student, Computer science and Engineering, IFET College of Engineering, Villupuram
gayu077@gmail.com

Abstract*— Search engine helps user to locate information from large storage media of content. So I propose search engine to reduce the work load of server. User search query in web its collect all web pages from server i.e. linking open data. All open data are copied temporarily in local system because of that user does not depends on server to view next web page so it reduce the work load of server. Local data search is an emerging information-retrieval paradigm in which a system finds answers to a query instantly while a user types in keywords character-by-character. In that, fuzzy search method further improves user search experiences by finding relevant answers in system and filtering keyword similar to query keyword. A main computational challenge in this paradigm is high speed requirement i.e. each query needs to be answered within seconds to achieve an instant response and a high query throughput. To overcome the space and time limitation, fuzzy search method improves the user user to fetch relevant record in server it temporarily stored in PC and further record search can be done without depending on server.

***Keywords*— search engine, information-retrieval, fuzzy search**

I. INTRODUCTION

Search engines is tool of searching are extremely popular and recurrently used sites. In web search there are billion of pages on web it is difficult to search all pages manually. Local search is the use of specialized Internet search engines that allow users to submit geographically constrained searches against a structured database of local listings. Typical local search queries include not only information about "what" the site visitor searching for (such as keywords, a business category) but also "where" information, such as a street address, city name, postal code. Local search sites are primarily supported by advertising from businesses that wish to be prominently featured when users search for specific products and services in specific locations. Local search advertising can be highly effective because it allows ads to be targeted very precisely to the search terms and location provided by the user. In this paper, Instant is a search enhancement that shows results as you type. The limits of the technology and infrastructure to help you get better search results, faster. The search key was that people type slowly, but read quickly. This means that you can scan a results page while you type.

II. EXISTING SYSTEM

Existing methods only identify a single tuple unit to answer keyword queries. While user enter search text the client system send request to server every time. It takes transaction time and data sharing between client and server. However, they neglect the fact that in many cases, we need to integrate multiple related tuple units to answer a keyword query. To address this problem, in this paper, we propose a structure-aware-index-based method to integrate multiple related tuple units to effectively answer keyword queries. The disadvantage of existing system is Single-keyword search without ranking , keyword search without ranking, Single-keyword search with ranking.

III. PROPOSED SYSTEM

The proposed to improve search efficiency by indexing structural relationships, and existing methods identify a single tuple unit to answer keyword queries. However, in many cases, multiple tuple units should be integrated to answer a keyword query. Thus, these methods will involve false negatives. To address this problem, integrating multiple related tuple units to effectively answer keyword queries. To achieve a high performance, an approach that focuses on common phrases in the data and queries, assuming records with these phrases are ranked higher. To index these phrases and develop an incremental-computation algorithm for efficiently segmenting a query into phrases and computing relevant answers. We conducted a thorough experimental study on real data sets to show the tradeoffs between time, space, and quality of these solutions. The benefits are:

Faster Searches: By predicting your search and showing results without opening multiple search pages.

Smarter Predictions: Even when you don’t know exactly what you’re looking for, the application helps the user to get the actual data which the user need.

Instant Results: The application stores the retrieved records locally and then it searches consecutive times locally without searching the website.

A.Query Upload

In this module the user has to enter a query in order to track the user requirements such as the resultant data. The user has to enter the information such as keyword, description and website URL into the database SQL server. The details validated and stored in the database.

B. Search by keyword

In this module the user has to enter a keyword and the application matches with the existing database table whether any information is found in the database. If any record matches the user criteria it shows the details in grid view.

C. Local Search

Local search, also known as type-ahead search. The studies in proposed indexing and query techniques to support local search. While the user enters a keyword it matches with the table and found records are displayed in a grid view. A session key is created and the detail related to it are grouped under this session key. These details are saved in a new table which is dynamically created.

D. Data Retrieval

Querying is the capability to retrieve data, usually a data subset, based on some user defined formula. These data subsets are often referred to as logical views. Data can be retrieved from the linking open data. The retrieved records are displayed in a grid view.

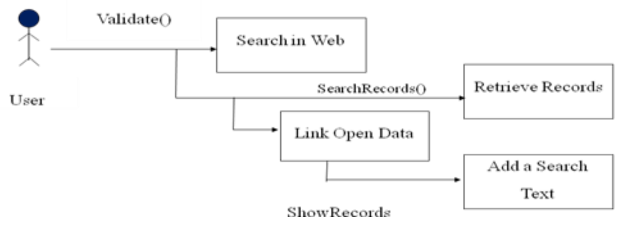

FIG.1 ARCHITECTURE DIAGRAM

IV. METHODOLOGY

1. Make a search from server
2. str = "SELECT * FROM KEYWORDS WHERE KEYWORD LIKE '%" + search string + "%'";
3. Create a Session Key
4. string SessionKey = Session["user"].ToString().Substring(0, 2);
5. Save the SessionKey in Table
6. Save the Open Data
7. Make a client search by entering a new search text

FUZZY SEARCH

A fuzzy search is a process that locates Web pages that are likely to be relevant to a search argument even when the argument does not exactly correspond to the desired information. A fuzzy search is done by means of a fuzzy matching program, which returns a list of results based on likely relevance even though search argument words and spellings may not exactly match. Exact and highly relevant matches appear near the top of the list. The fuzzy search can be done by trie-based approaches. The fuzzy reasoning, described as “If *X* is A then Y is B” is said to be simple and conforms to human language. However in cases where the system has multi inputs and multi outputs, one has to build the fuzzy reasoning rules in multi dimensional input and output spaces in order to describe the behavior of the system. It is considered that the difficulty is caused by a mismatch between the description of the fuzzy reasoning and the actual records of inference rules which humans have. Fuzzy searching is much more powerful than exact searching when used for research and investigation. Fuzzy searching is especially useful when researching unfamiliar, foreign-language, or sophisticated terms, the proper spellings of which are not widely known. Fuzzy searching can also be used to locate individuals based on incomplete or partially inaccurate identifying information.

A fuzzy matching program can operate like a spell checker and spelling-error corrector. For example, if a user types "Misissippi" into Yahoo or Google (both of which use fuzzy matching), a list of hits is returned along with the question, "Did you mean Mississippi?" Alternative spellings, and words that sound the same but are spelled differently, are given. The steps are,

- Make a search by keyword
- Access Centralized Server and fetch records
- Create a Open Data
- Save Open Data
- Search from Open Data
- Compare and Filter Records
- Fetch Results and display in a gridview.

V. IMPLEMENTATION

This project is implemented this in ASP.NET using Session key web pages can be temporarily stored in PC. Session variables are created on a per-user basis. By default, they are maintained in the Web server's memory. Imagine a Web site with thousands of users. Because of the huge number of users, the number of active sessions on the Web server also will be vary high. That means you are storing too much data in the Web server's memory. If the load on the server keeps of increasing, it may reach saturation and cause trouble for overall scalability of your application. ASP.NET 2.0 allows you to store session variables at three distinct locations:

In the memory of the Web server (in process)
1. In the memory of a machine dedicated to storing session variables (state server)
2. In an SQL Server database

The first mode is the default. Modes 2 and 3 are often called "out-of-process" modes because the session store is independent of the Web site. In this article, you will restrict yourself to exploring the third mode.

Storing session variables in the SQL server has the following advantages:

- **Scalability:** If you are looking for a highly scalable option to store your session variables, the SQL Server option is for you. It is a much more scalable option than the others. Web farm architecture can very easily access the session variables because they are stores in an independent database.
- **Reliability:** Because the data is physically persisted in a database, it is more reliable than the other options. It has the ability to survive server restarts.
- **Security:** SQL Server is more secure than the in-memory or state server option. You can protect your data more easily by configuring SQL Server security.

Configuring SQL Server to Store a Session State

Before you can actually store a session state in SQL server, you need to configure it.. You can store the session state in three possible locations within the SQL Server:

- **Temporary storage:** In this case, the session state is stored in the "tempdb" database of SQL Server. The tool creates a database called **ASP State** and adds certain stored procedures for managing session to it. The tool also creates required tables in the "tempdb" database. If you restart the SQL server, the session data is not persisted.

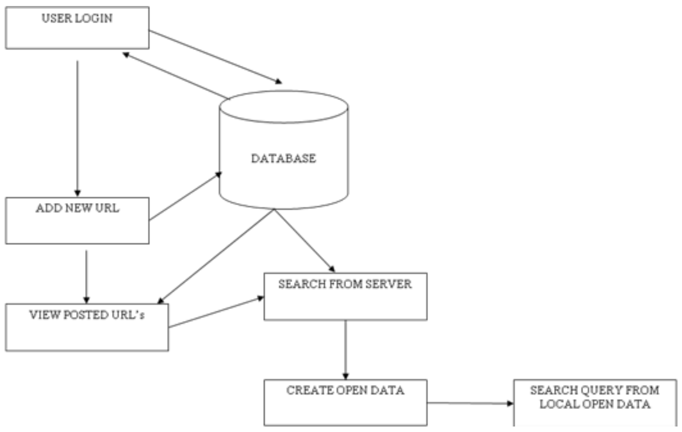

Fig2.Flow Diagram

- **Persistent storage:** The tool creates a database called **ASP State** and adds stored procedures for managing a session to it. The session state is stored in the **ASP State** database. The advantage of this method is that the data is persisted even if you restart the SQL server.
- **Custom storage:** Both the session state data and the stored procedures are stored in a custom database. The database name must be specified in the configuration file.

VI. CONCLUSION

To improve efficiency of an instant-fuzzy search system by considering proximity information when we need to compute top-k answers. To adapt existing solutions to solve this problem, including computing all answers, doing early termination. A technique to index important phrases to avoid the large space overhead of indexing all word grams. The presented fuzzy search algorithm for finding the records in a query efficiently, and studied how to compute and rank the segmentations consisting of the indexed phrases. Through analysis by considering space, time, and relevancy tradeoffs of these approaches. In particular, the experiments on real data showed the efficiency of the proposed technique for 2-keyword and 3-keyword queries that are common in search applications. All the answers for the other queries would give the best performance and satisfy the high-efficiency requirement of instant search.

A main computational challenge in this paradigm is high speed requirement i.e each query needs to be answered within seconds to achieve an instant response and a high query throughput.

REFERENCES

[1] "Development Of A Ranking Algorithm For Search Engine Optimization", Navkirandeep Kaur, Jagroop Kaur, 2014

[2] "Personalized Mobile Search Engine With Multiple Preference", D.Sudha, S.Vijayalakshmi, V.Komathi, K.Amala, 2014.

[3] V. Hristidis and Y. Papakonstantinou. Discover:"Keyword search in relational databases". In *VLDB*,pages 670–681, 2002.
[4] G. Li, J. Feng, and L. Zhou. "Retrieving andMaterializing Tuple Units for Effective KeywordSearch over Relational Databases. In *ER*, 2008.
[5] G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou. Ease:An effective 3-in-1 keyword search methord forunstructured, semi-structured and structured data. In *SIGMOD*, 2008.
[6] "A Survey On Semantic Web Mining Based Web Search Engines",S .Latha .
[7] "Efficient Instant-Fuzzy Search with ProximityRanking", Inci Cetindil, Jamshid Esmaelnezhad, Taewoo Kim and Chen Li"
[8] " Intelligent Elevator Control By Ordinal Structure Fuzzy Logic Algorithm" Tan Kok Khiang,MarZuki Khalid and Rubiyah Yusof.