ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

A Survey on Fuzzy Based Fast Query for Data Retrieval

T.Nalini[1], J. Rathanaa Ranjeni[2], S.Brintha Rajakumari[3], Sundararajan.M[4], Arulselvi S[5]

[1]Professor, Dept. of CSE, Bharath University, Chennai, Tamil Nadu, India

[2]Professor, Dept. of CSE, Bharath University, Chennai, Tamil Nadu, India

[3]Associate Professor, Dept. of CSE, Bharath University, Chennai, Tamil Nadu, India

[4]Director, Research Center for Computing and Communication, Bharath University, Chennai, Tamil Nadu, India

[5]Co-Director, Research Center for Computing and Communication, Bharath University, Tamil Nadu, India

ABSTRACT: Structured query language (SQL) is a classical way to access relational databases. Although SQL is powerful to query relational databases, it is rather hard for inexperienced users. A search-as-you-type system computes answers on-the-fly as a user types in a keyword query character by character. To support prefix matching, we use auxiliary tables as index structures and SQL queries to support search-as-you-type. We present solutions for both single-keyword queries and multi keyword queries. We extended the techniques in the case of fuzzy queries and proposed various techniques to improve query performance. However, to support fuzzy search, there may be multiple prefixes similar to the keyword. We call the nodes of these similar prefixes the active nodes for a keyword. Now we are going to use the fuzzy search to retrieve images. Data Indexer builds Keyword to Attribute Mapping for mapping keywords to attributes. Given a keyword query, Template Matcher suggests relevant templates based on the index structures.

KEYWORDS: Data Indexer, Fuzzy search, multiple keyword queries, Retrieve images, Single-keyword queries and Template Matcher.

I. INTRODUCTION

MANY information systems nowadays improve user search experiences by providing instant feedback as users formulate search queries. Most search engines and online search forms support auto completion, which shows suggested queries or even answers "on the fly" as a user types in a keyword query character by character. For instance, consider the Web search interface at Netflix, 1 which allows a user to search for movie information. If a user types in a partial query "mad," the system shows movies with a title matching this keyword as a prefix, such as "Madagascar" and "Mad Men: Season 1." The instant feedback helps the user not only in formulating the query, but also in understanding the underlying data. This type of search is generally called search-as-you-type or type-ahead search.

Search-as-You-Type for Single-keyword Queries Exact Search:
As a user types in a single partial keyword w character by character, fuzzy search on-the-fly finds records with keywords similar to the query keyword. We use edit distance to measure the similarity between strings [2]. Formally, the edit distance between two strings s1 and s2, denoted by red (s1, s2), is the minimum number of single-character edits operations (i.e., insertion, deletion, and substitution) needed to transform s1 to s2. For example, Ed (correlation; correlation) = 1 and Ed (correlation, correlation) =2.

Search-as-You-Type for Multi keyword Queries: Exact Search: Given a multi keyword query Q with m keywords w_1; w2;...;w_m, as the user is completing the last keyword w_m, we treat w_m as a partial keyword and other keywords as complete keywords.[1] As a user types in query Q character by character, search-as-you-type on-the-fly finds the records that contain the complete keywords and a keyword with a prefix w_m.[3]

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

Fuzzy Search: Fuzzy search on-the-fly finds the records that contain keywords similar to the complete keywords and a keyword with a prefix similar to partial keyword w_m. For instance, suppose the edit - distance threshold _ ¼ 1.

No-Index Methods: A straightforward way to support search-as-you-type is to issue an SQL query that scans each record and verifies whether the record is an answer to the query. There are two ways to do the checking: 1) Calling User-Defined Functions (UDFs).[4] We can add functions into databases to verify whether a record contains the query keyword; and 2) Using the LIKE predicate. Databases provide a LIKE predicate to allow users to perform string matching [4]. We can use the LIKE predicate to check whether a record contains the query keyword. This method may introduce false positives, e.g., keyword "publication" contains the query string "ic," but the keyword does not have the query string "ic" as a prefix. We can remove these false positives by calling UDFs. The two no-index methods need no additional space, but they may not scale since they need to scan all records in the table.[5]

Index-Based Methods

In this section, we propose to build auxiliary tables as index structures to facilitate prefix search. Some databases such as Oracle and SQL server already support prefix search and we could use this feature to do prefix search [5, 18]. However, not all databases provide this feature. For this reason, we develop a new method that can be used in all databases. In addition, our experiments show that our method performs prefix search more efficiently[6].

Inverted-index table: Given a table T, we assign unique IDs to the keywords in table T, following their alphabetical order [6]. We create an inverted-index table IT with records in the form {lkid,uidi}, where kid is the id of a keyword and rid is the id of a record that contains the keyword. Given a complete keyword, we can use the inverted-index table to find records with the keyword.[7]

Prefix table: Given a table T, for all prefixes of keywords in the table, we build a prefix table PT with records in the form (p,lkid,ukidi) where p is a prefix of a keyword, lkid is the smallest id of those keywords in the table T having p as a prefix, and ukid is the largest id of those keywords having a prefix.[3,19] An interesting observation is that a complete word with p as a prefix must have an ID in the keyword range [lkid,ukid], and each complete word in the table T with an ID this keyword range must have a prefix p. Thus, given a prefix keyword w, we can use the prefix table to find the range of keywords.[8]

The Inverted-Index Table and Prefix Table

a) Keywords

	Keyword
K_1	icde
K_2	icdt
K_3	preserving
K_4	privacy
K_5	publishing
K_6	pvldb
K_7	sigir
K_8	sigmod
K_9	vldb
K_{10}	vldbj

b) -index table

kid	rid
K_1	r_{10}
K_2	r_6
K_3	r_8
K_4	r_{10}
K_5	r_1
K_6	r_9
K_7	r_3
K_8	r_6
K_9	r_8
k_{10}	r_4

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

c)Prefix table

prefix	lkid	ukid
ic	k_1	k_2
p	k_3	k_6
pr	k_3	k_4
pri	k_4	k_4
pu	k_5	k_5
pv	k_6	k_6
pvl	k_6	k_6
sig	k_7	k_8
v	k_9	k_{10}
vl	k_9	k_{10}

II. FUZZY SEARCH FOR SINGLE KEYWORD

No-Index Methods:

Recall the two no-index methods for exact search in Section 3.1. Since the LIKE predicate does not support fuzzy search, we cannot use the LIKE-based method.[9] We can use

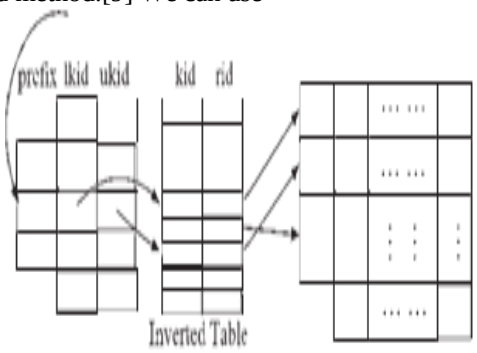


Fig. 1Using inverted-index table and prefix table to support search-as-you-type.[10]

UDFs to support fuzzy search. We use a UDF PED (w, s) that takes a keyword w and a string s as two parameters, and returns the minimal edit distance between wand the prefixes of keywords in s.[11]

Index-Based Methods:

This section proposes to use the inverted-index table and prefix table to support fuzzy search-as-you-type. Given a partial keyword w, we compute its answers in two steps.[12] First, we compute its similar prefixes from the prefix table PT, and get the keyword ranges of these similar prefixes. Then we compute the answers based on these ranges using the inverted-index table IT as discussed in Section 3.2. In this section, we focus on the first step: computing w's similar prefixes.[13]

Using UDF:

Given a keyword w, we can use a UDF to find its similar prefixes from the prefix table PT. We issue an SQL query that scans each prefix in PT and calls the UDF to check if the prefix is similar to w.[14]

Gram-Based Method:

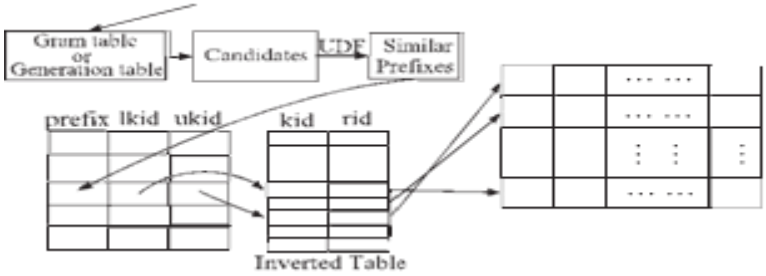
There are many q-gram-based methods to support approximate string search [17]. Given a string s, its q-grams are its

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015



To find similar prefixes of a query keyword w, besides maintaining the inverted-index table and the prefix table, we need to create a q-gram table GT with records in the form (p, gram
), where p is a prefix in the prefix table and q gram is a q-gram of p. Given a partial keyword w, we first find the Prefixes in GT with no smaller than |w| +1 – q- τ*q grams in G^o (w).

Neighborhood-Generation-Based Method:

Ukkonen proposed a neighborhood-generation-based method to support approximate string search. We extend this method to use SQL to support fuzzy search-as-you-type[14].

Given a keyword w, the substrings of w by deleting i characters are called "i-deletion neighborhoods" of w. Let D_i(wÞ)denote the set of i-deletion neighborhoods of w and $D_\tau(w)=u_i.^{\tau}{}_{=}D_i(w)$.

.

Prefix	i-deletion	i
Vldb	Vldb	0
Vldb	ldb	1
Vldb	vdb	1
Vldb	vlb	1
Vlbd	vld	1

Table-3 Neighborhood generation table (τ=1)

We can use deletion operations to replace the substitution and insertion operations as follows: suppose wecantransforms1 tos2 with d deletions, I insertions, and r substitutions, such that ed(s1,s2) = d + i +r<=τ We can transform s1 and s2 to the same string by doing d þ r deletions on s1 and i þ r deletions on s2, respective.[15]

Supporting multikeyword queries:

In this section, we propose efficient techniques to support multi keyword queries.

Computing Answers from Scratch:

Given a multi keyword query Q with m keywords w_1, $w_{2,}$,...;w_m, there are two ways to answer it from scratch. 1) Using the "INTERSECT" Operator: a straightforward way is to first compute the records for each keyword using the previous methods, and then use the "INTERSECT" operator to join these records for different keywords to compute the answers. 2) Using Full-text Indexes: we first use full-text indexes (e.g., CONTAINS command) to find records matching the first m _ 1 complete keywords, and then use our methods to find records matching the last prefix keyword. Finally, we join the results. These two methods cannot use the pre computed results and may lead to low performance.

Word-Level Incremental Computation:

We can use previously computed results to incrementally answer a query. Assuming a user has typed in a query Q with keywords w_1,w_2,...;w_m, we create a temporary table CQ to cache the record ids of query Q. If the user types in a new keyword wmþ1 and submits a new query Q0 with keywords w_1,w_2,...;w_m;w_{m+1}, we use temporary table CQ to incrementally answer the new query.[13]

Exact search. As an example, we focus on the method that uses the prefix table and inverted-index table. As CQ contains all results for query Q, we check whether the records in CQ contain keywords with the prefix wmþ1 of new query Q0.[12]

Fuzzy search. As an example, we consider the character-level incremental method. We first compute $S_t^{wm+1sing}$ the character-level incremental method for the new keyword W_{m+1}, and then use S_T^{m+1} to answer the query.[11]

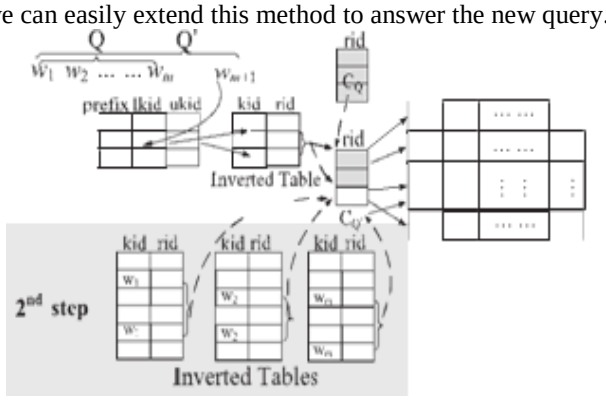
ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

If the user modifies the keyword w_m of query Q to w_m and submits a query with keywords $w_1,w_2,...,w_{m\_1},w_m$, we can use the cached result of query $w_1,w_2,...,w_{m\_1}$ to answer the new query using the above method. Similarly, if the user arbitrarily modifies the query, we can easily extend this method to answer the new query.[10]



III. SUPPORTING FIRST-N QUERIES

The previous methods focus on computing all the answers. As a user types in a query character by character, we usually give the user the first-N (any-N) results as the instant feedback. This section discusses how to compute the first-N results.[9]

Exact first-N queries. For exact search, we can use the "LIMIT N" syntax in databases to return the first-N results. For example, MYSQL uses "LIMIT n1;n2" to return n2 rows starting from the n1th row. As an example, we focus on how to extend the method based on the inverted-index table and the prefix table (Section 3.2). Our techniques can be easily extended to other methods.[8]

Literature survey:

1. Efficient mergining and filtering algorithm for approximate string search (Chen Li, Jiaheng Lu, Yiming Lu)
How to efficiently find a collection string which is similar to the given query? Text data is ubiquitous. Management of string data in databases and information systems has taken on particular importance recently.[7] First we develop several algorithms that can greatly to improve the performance. Here we are using three algorithms for answering approximate string search queries, called Scan Count, Merge Skip, and Divide Skip.[16][

2. Ed-join: an efficient algorithm for similarity joins with editing distance constraints (Thai Ngoc Thuy Huong)
In this project, we implement an efficient algorithm for similarity join with editing distance constraints. Current approaches are mainly that the edit distance constraint is converted to a weaker constraint on the number of matching q-grams between a pair of strings [7, 17]. A new algorithm, Ed-Join, is proposed that exploits the new mismatch-based filtering methods. it achieves substantial reduction of the candidate sizes and hence saves computation time.[18]

3. Efficient Exact Set-Similarity Joins. (Arvind Arasu, Venkatesh Ganti, Raghav Kaushik)
Given two input collections of sets, a set-similarity join (SSJoin) identifies all pairs of sets, one from each collection, that have high similarity[18]. Recent work has identified SSJoin as a useful primitive operator in data cleaning. We propose new algorithms for SSJoin. Our algorithms have two important features: They are exact, i.e., they always produce the correct answer, and they carry precise performance guarantees[19].

4. DISCOVER: Keyword Search in Relational Databases (Vagelis Hristidis, Yannis Papakonstantinou)
DISCOVER operates in relational databases and facilitates information discovery on them by allowing its user to issue keyword queries without any knowledge of the database schema or of SQL. DISCOVER proceeds in two steps[20]. First the Candidate Network Generator generates all candidate networks of relations, that is, join expressions that generate the joining networks of tuples.[21]

5. Efficient Interactive Fuzzy Keyword Search (Shengyue Ji, Guoliang Li, ChJianhua Fengen Li)
In this paper, we study a new information-access paradigm, called "interactive, fuzzy search," in which the system searches the underlying data "on the fly" as the user types in query keywords.[22] It extends autocomplete interfaces by allowing keywords to appear in multiple attributes (in an arbitrary order) of the underlying data; and finding relevant records that have keywords matching query keywords approximately.[23]

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

6. XRANK: Ranked Keyword Search over XML Documents (Lin Guo Feng Shao Chavdar Botev Jayavel Shanmugasundaram)

The XRANK system that is designed to handle these novel features of XML keyword search. Our experimental results show that XRANK offers both space and performance benefits when compared with existing approaches.[24] An interesting feature of XRANK is that it naturally generalizes a hyperlink based HTML search engine such as Google. XRANK can thus be used to query a mix of HTML and XML documents

IV. CONCLUSION

Thus we have made a survey analysing various methods for fuzzy based fast query for data retrieval. And develop an efficient fuzzy match algorithm. The effectiveness of our project is measured by evaluating them on real datasets of images.

REFERENCES

1. C. Li, J. Lu, and Y. Lu, "Efficient Merging and Filtering Algorithms for Approximate String Searches," Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE '08), pp. 257-266, 2008.1)
2. C. Xiao, W. Wang, and X. Lin, "Ed-Join: An Efficient Algorithm for Similarity Joins with Edit Distance Constraints," Proc. VLDB Endowment, vol. 1, no. 1, pp. 933-944, 2008.
3. A. Arasu, V. Ganti, and R. Kaushik, "Efficient Exact Set-Similarity Joins," Proc. 32nd Int'l Conf. Very Large Data Bases (VLDB '06),pp. 918-929, 2006V.
4. Udayakumar R., Khanaa V., Kaliyamurthie K.P., "High data rate for coherent optical wired communication using DSP", Indian Journal of Science and Technology, ISSN : 0974-6846, 6(S6) (2013) 4772-4776.
5. Hristidis and Y. Papakonstantinou, "Discover: Keyword Search in Relational Data Bases," Proc. 28th Int'l Conf. Very Large DataBases (VLDB '02), pp. 670-681, 2002.
6. S. Ji, G. Li, C. Li, and J. Feng, "Efficient Interactive Fuzzy Keyword Search," Proc. 18th ACM SIGMOD Int'l Conf. World Wide Web (WWW), pp. 371-380, 2009.
7. Vijayaprakash S., Langeswaran K., Gowtham Kumar S., Revathy R., Balasubramanian M.P., "Nephro-protective significance of kaempferol on mercuric chloride induced toxicity in Wistar albino rats", Biomedicine and Aging Pathology, ISSN : 2210-5220, 3(3) (2013) pp.119-124.
8. Lin Guo Feng Shao Chavdar Botev Jayavel Shanmugasundaram. XRANK: Ranked Keyword Search over XML Documents
9. H. Bast, A. Chitea, F.M. Suchanek, and I. Weber, "ESTER: Efficient Search on Text, Entities, and Relations," Proc. 30th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '07), pp. 671-678, 2007.
10. H. Bast and I. Weber, "Type Less, Find More: Fast Autocompletion Search with a Succinct Index," Proc. 29th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval (SIGIR '06), pp. 364-371, 2006.
11. H. Bast and I. Weber, "The Complete Search Engine: Interactive, Efficient, and Towards IR & DB Integration," Proc. Conf. Innovative Data Systems Research (CIDR), pp. 88-95, 2007.
12. Udayakumar R., Khanaa V., Kaliyamurthie K.P., "Optical ring architecture performance evaluation using ordinary receiver", Indian Journal of Science and Technology, ISSN : 0974-6846, 6(S6) (2013) pp. 4742-4747.
13. R.J. Bayardo, Y. Ma, and R. Srikant, "Scaling up all Pairs Similarity Search," Proc. 16th Int'l Conf. World Wide Web (WWW '07), pp. 131-140, 2007
14. Sundararajan M., "Optical instrument for correlative analysis of human ECG and breathing signal", International Journal of Biomedical Engineering and Technology, ISSN : 0976 - 2965, 6(4) (2011) pp.350-362.
15. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S.Sudarshan, "Keyword Searching and Browsing in Data Bases Using Banks," Proc. 18th Int'l Conf. Data Eng. (ICDE '02), pp. 431-440, 2002.
16. K. Chakrabarti, S. Chaudhuri, V. Ganti, and D. Xin, "An Efficient Filter for Approximate Membership Checking," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '08), pp. 805-818, 2008.
17. Udayakumar R., Khanaa V., Kaliyamurthie K.P., "Performance analysis of resilient ftth architecture with protection mechanism", Indian Journal of Science and Technology, ISSN : 0974-6846, 6(S6) (2013) pp. 4737-4741
18. S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani, "Robust and Efficient Fuzzy Match for Online Data Cleaning," Proc. ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '03), pp. 313-324, 2003.
19. S. Chaudhuri and R. Kaushik, "Extending Autocompletion to Tolerate Errors," Proc. 35th ACM SIGMOD Int'l Conf. Managementof Data (SIGMOD '09), pp. 433-439, 2009.
20. B.B. Dalvi, M. Kshirsagar, and S. Sudarshan, "Keyword Search on External Memory Data Graphs," Proc. VLDB Endowment, vol. 1,no. 1, pp. 1189-1204, 2008.
21. B. Ding, J.X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding Top-K Min-Cost Connected Trees in Data Bases," Proc. IEEE 23rd Int'l Conf. Data Eng. (ICDE '07), pp. 836-845, 2007.
22. L. Gravano, P.G. Ipeirotis, H.V. Jagadish, N. Koudas, S.Muthukrishnan, and D. Srivastava, "Approximate String Joins in a Data Base (Almost) for Free," Proc. 27th Int'l Conf. Very Large Data Bases (VLDB '01), pp. 491-500, 2001. M. Hadjieleftheriou, A. Chandel, N. Koudas, and D. Srivastava,
23. "Fast Indexes and Algorithms for Set Similarity Selection Queries," Proc. IEEE 24th Int'l Conf. Data Eng. (ICDE '08), pp. 267-276, 2008M. Hadjieleftheriou, N. Koudas, and D. Srivastava, "Incremental

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 7, July 2015

24. Maintenance of Length Normalized Indexes for Approximate String Matching," Proc. 35th ACM SIGMOD Int'l Conf. Management of Data (SIGMOD '09), pp. 429-440, 2009.
25. 25) P.JENNIFER, DR. A. MUTHU KUMARAVEL, Comparative Analysis of advanced Face Recognition Techniques, International Journal of Innovative Research in Computer and Communication Engineering, ISSN(Online): 2320-9801, pp 4917-4923 Vol. 2, Issue 7, July 2014
26. 26) Dr.R.Udayakumar, Computer Simulation of Polyamidoamine Dendrimers and Their Complexes with Cisplatin Molecules in Water Environment, International Journal of Innovative Research in Computer and Communication, ISSN(Online): 2320-9801,pp 3729,25-30, Vol. 2, Issue 4, April 2014
27. 27) DR.A.Muthu kumaravel, Mr. Kannan Subramanian, Collaborative Filtering Based On Search Engine Logs, International Journal of Innovative Research in Computer
28. and Communication Engineering, ISSN(Online): 2320-9801,pp 2432-2436, Vol. 2, Issue 1, January 2014
29. 28) Dr.A.Muthu Kumaravel, Mining User Profile Using Clustering FromSearch Engine Logs, International Journal of Innovative Research in Computerand Communication Engineering, ISSN(Online): 2320-9801,pp 4774-4778, Vol. 2, Issue 6, June 2014
30. 29) P.Kavitha, Web Data High Quality Search - No User Profiling, International Journal of Innovative Research in Computerand Communication Engineering, ISSN(Online):2320-9801,pp 2025-2030, Volume 1, Issue 9, November 2013