IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

Duplication Free Minimal Keyword Search in Graph using Top-K Algorithm

Miss. Payal P. Thakur[1], Prof. N. R. Borkar[2]

Student, Department of CSE, Dr. Sau. K.G.I.E.T, Amravati, Maharashtra[1]

Department of CSE, Dr. Sau. K.G.I.E.T, Amravati, Maharashtra[2]

Abstract: Keyword search over a graph searches for a subgraph that contains a set of query keywords. A problem with most existing keyword search methods is that they may produce duplicate answers that contain the same set of content nodes (i.e., nodes containing a query keyword) although these nodes may be connected differently in different answers. Thus, users may be presented with many similar answers with trivial differences. In addition, some of the nodes in an answer may contain query keywords that are all covered by other nodes in the answer. Removing these nodes does not change the coverage of the answer but can make the answer more compact. The answers in which each content node contains at least one unique query keyword are called minimal answersin this paper. We define the problem of finding duplication-free and minimal answers, and propose algorithms for finding suchanswers efficiently. Extensive performance studies using two large real data sets confirm the efficiency and effectiveness of theproposed methods.

Keywords: Keyword search, graph data, polynomial delay, approximation algorithm.

I. INTRODUCTION

KEYWORD search is a well-known method for extracting relevant knowledge from a set of documents in information retrieval. Similarity query in multidimensional database is a fundamental research problem with numerous applications in the areas of database, data mining, and information retrieval[1]. Given a query object, the goal is to find similar objects in the database. Recently, querying incomplete data has attracted extensive research efforts. In this problem, the data values may be missing dueto various practical issues. Given a graph where nodes are associated with text, keyword search over the graph finds a subgraph that contains a set of query keywords. Due to the fact that many types of data can be represented by graphs, keyword search over graphs has received much attention in recent years. Most of the work in this area find minimal connected trees or subgraphs that minimize a proximity function (e.g., the sum of distances from the nodes in the answer to a centernode [3]). However, these methods may generate many trees or subgraphs with the same set of content nodes (i.e., nodes containing at least one query keyword) even though these answers may have different intermediate nodes connecting the content nodes. The following example illustrates the duplication problem for a tree-based method. Suppose the nodes in aninput graph are web pages. Two nodes are connected by an edge if there is a link from one page to the other. Consider thefigure1.1 the user is interested in finding pages that contain keywordsk1 and k2. Two nodes mk1 and nk1 contain keyword k1 and another two nodes mk2 and nk2 contain keyword k2. The left graph in the figure contains 4 trees that cover mk1 and mk2, where each branch from mk1 to mk2 is a tree. The right graph contains a single tree that covers nk1 and nk2. Assume that the weight on each edge is the same. According to the ranking function used in the tree approaches, the tree that contains nk1 and nk2 in the right graph is produced afterthe first four trees that cover mk1 and mk2 on the left, because it has more edges than the other four trees.

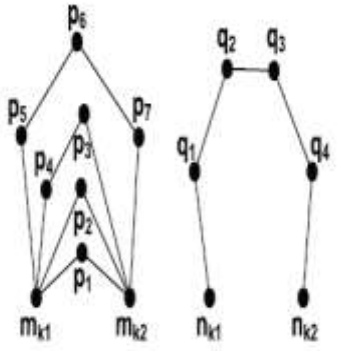

Figure 1: Duplication problems with tree answers.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

However, all the four trees on the left have the same set of content nodes. Since the users usually want to see different groups of content nodes that are close to each other and might not be interested in browsing multiple relations to see how the nodes that contain input keywords are related to each other, the above search results might not be desirable[1]. Producing results with distinct sets of content nodes can prevent the search engine from overwhelming the user with many similar answers[1].

In this, we first propose a new approach to keyword search in that produces duplication-free answers. Each answer produced by our approach has a unique set of content files. We also define minimal answers, in which each file contains at least one input keyword. We propose two algorithms that convert an answer to a minimal answer. We prove that the problem of finding a minimal answer while minimizing the proximity function that we use is NP-hard. Thus, one of the algorithms we propose is a greedy algorithm that searches for a sub-optimal minimal answer. We prove that this greedy algorithm has a bounded approximation ratio. Finally, for finding top-k duplication-free and minimal answers, we propose an Top-K algorithm. Our extensive experiments show the efficiency and effectiveness of the proposed methods.

Our goal is to search exact file that we want with their exention, according to our keyword search that we enter and also show the graph of keyword search. We also calculated the time required for searching, frequency, and the size of file. Keyword searches are an alternative means for querying databases, which are simple and yet familiar to most internet users since they only require the input of some keywords. While keyword searches have proven effective for text documents (e.g., hypertext markup language (HTML) documents), the problem of keyword searches on structured data (e.g., relational databases) or the semi-structured data (e.g., XML databases) is not straightforward and well studied. Keyword searches in text documents find the documents that are more closely related to the input keywords, while in relational databases it searches the correlative tuples in the database that contains all or some the keywords. However, defining the results of keyword searches in XML documents is more complex.

Keyword search on graph data usually returns a set of connected sub-structures, such as sub-trees or sub-graphs, showing that which nodes include query keywords and how they are inter-connected in the graph database[4]. Many approaches find minimal connected sub-trees containing query keywords as succinct answers to a given query [7][8]. Since there can be a significant number of answer sub-trees in a large graph database, a relevance scoring function is often used to rank candidate answers and select top-k ones having the highest relevance. There have been proposed several approaches based on distinctrootsemantics, where for each node in the graph, at most one sub-tree rooted at the node is considered a possible answer to the query [9]. The answer tree consists of a set of content nodes containing all the query keywords as well as the nodes and edges on the shortest paths from the root to each content node. Its relevance is usually computed by a function of the shortest paths, such as the sum of the path lengths. By reducing the number of sub-trees to be explored in the graph significantly, the search methods based on the distinct root semantics can process keyword queries over a large volume of data more efficiently than other approaches. It also facilitates exploiting indexes on graph data to improve query performance [9].

II. LITERATURE REVIEW

Recently, graph-structured data is widely used in various fields such as social networking, semantic web, linked open data, knowledge management and bio-informatics. Keyword search has been attracting a lot of attention since it provides a simple and user-friendly interface to querying graph data and allows users to express their information need using only a set of keyword terms[1]. Keyword search on graph data usually returns a set of connected sub-structures, such as sub-trees or sub-graphs, showing that which nodes include query keywords and how they are inter-connected in the graph database[4]. Many approaches find minimal connected sub-trees containing query keywords as succinct answers to a given query . Since there can be a significant number of answer sub-trees in a large graph database, a relevance scoring function is often used to rank candidate answers and select top-k ones having the highest relevance. There have been proposed several approaches based on distinct rootsemantics, where for each node in the graph, at most one sub-tree rooted at the node is considered a possible answer to the query[4]. The answer tree consists of a set of content nodes containing all the query keywords as well as the nodes and edges on the shortest paths from the root to each content node. Its relevance is usually computed by a function of the shortest paths, such as the sum of the path lengths. By reducing the number of sub-trees to be explored in the graph significantly, the search methods based on the distinct root semantics can process keyword queries over a large volume of data more efficiently than other approaches.

➢ EXISTING METHOD FOR KEYWORD SEARCH

There are various approaches proposed by various researchers for keyword search in graphs. In this chapter a brief description of these approaches and a comparison between them are given.

Chang-sup[4] park defines most approaches searching for sub-trees in the graph are based on two different semantics, namely Steiner tree semantics and distinct root semantics. The Steiner tree semantics defines the weight of an answer tree by the total weight of the edges in the tree. Search methods based on this semantics aim to find answer sub-trees

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

with the smallest weights. However, finding only an optimal answer sub-tree with the smallest weight, called a group Steiner tree, is known to be NP complete [7]. Under distinct-root semantics, sub-trees returned as query results must be rooted at a distinct node. Thus, for each potential root node in the graph, only a single sub-tree having a minimal weight is considered a candidate answer to the query, where the weight of a sub-tree is typically de-fined by the sum of the weights of the shortest paths from its root to each keyword node. This semantics can deal with queries over a large graph database more efficiently than Steiner tree semantics[4].

H. Wang[10] uses an efficient indexing scheme on graph data to speed bi-directional exploration with a good performance guarantee. It pre-computes the shortest paths and their distances from nodes to keywords in the graph and stores them in sorted inverted lists and a hash map. By exploiting indexes, it can avoid a lot of explorations of the graph data and thus can find top-k answers efficiently. A study in suggests creating and utilizing a multi-granular representation of a graph data to minimize disk I/O, and presents search algorithms on multi-granular graphs extended from BANKS[16] and Bi-directional Search. A recent work in has proposed an extended answer tree structure and search algorithm to produce various and effective top-k answers[10].

These approaches, however, have a common drawback of producing sub-trees that are non-reduced or duplicate in content nodes as mentioned in the previous section. Although graph exploration approaches such as BANKS[16] and Bi-directional Search can detect and exclude such redundant answers, an exponential number of answer sub-trees should be probed in the graph, resulting in severe performance overhead. BLINKS[9] does not take redundancy among query answers into account and creates indexes only on the single optimal path from each node to a keyword term in the graph. Therefore, even if a redundant answer tree is detected, it cannot find alternative answer trees rooted at the same node as the root of the redundant answer.

J. Feng[5] show that finding r-cliques are faster and more effective than finding communities. However, all of these approaches might produce duplicate and non-minimal answers. Recently, the BROAD system is proposed to find diversified answers for keyword search on graphs. The system is built on top of a keyword search engine and partitions the answer trees produced by the engine into dissimilar clusters. The dissimilarity between answers is measured based on the structural and semantic information of the given trees. A hierarchical browsing method is further proposed to help users navigate and browse the results[5].

III. PROBLEM DEFINITION

ANALYSIS OF PROBLEM

In previous system duplicated answers are shown for that we take a one data graph whose nodes are associated with text and a query consisting of a set of keywords, the problem of keyword search in a graph is generally to find a subgraph that contains all or part of the keywords. The data graph can be directed or undirected. The edges and/or nodes may have weights on them. In this work, we consider undirected graphs with weighted edges, where two nodes are connected by an edge if there is a relationship between them and the edge weight represents the distance between the two nodes. It should be noted that our approach is adaptable to work with directed graphs[1].

- **Definition 1 (Answer):**

Given a graph G and a set of query keywords (Q = {k1, k2, . . . , kl}), an Answer to Q in G is a set of content nodes in G that together cover all of the input keywords in Q. An Answerhas a weightwhich can be defined accordingto the application need based on the weights of theedges in G that connect the nodes in the Answer. The abovedefinition does not require the nodes in an Answerto beconnected with each other either directly or indirectly inG, but Answers with nodes connected to each other can be preferred over those with disconnected nodes by using a weight function.

- **Problem 1 (Duplication free keyword search).**

Given a graph G, an integer k and a set Q of query keywords, findtop-k uniqueAnswers of Q in G whose weights are optimal.An Answer is unique if it appears at most once in thetop-k list. The next definition deals with the minimality ofthe Answer.

- **Definition 2 (minAnswer):**

Given a graph G and a set of query keywords (Q = {k1, k2, . . . , kl}), a minAnswerof Q in G is an Answer of Q in G in which each content node covers at least one query keyword that other content nodes do not cover.

- **Problem 2(Duplication free and minimal keyword search).**

Given a graph G, an integer k and a set Q of inputkeywords, find top-k unique minAnswers of Q in G whoseweights are optimal.To focus on the generality of the above keyword searchproblems, we intentionally avoided defining the weight of an Answer in Definitions 1 and 2. Below we define a weight function, used in [4], to measure the proximity of the nodes in an Answer. Note that other weight functions can be used with our definitions. Also, most of the algorithms are independent of the weight function.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

- **Definition 3 (sumDistance)**

Suppose that the set of nodes in an Answer in graph G is denoted as V ={v1, v2, . . . , vl}. The sumDistanceof the Answer is defined as

$$\text{sumDistance} = \sum_{i=1}^{l} \sum_{j=i+1}^{l} \text{dist}(vi, vj)$$

wheredist (vi, vj) is the shortest distance between vi and vj in G, i.e., the sum of weights on the shortest path between vi and vj in G [4].
When using sumDistanceto define the weight of an Answer, Answers with smaller weights are considered to be better because the nodes in an Answer are closer to each other when its weight is smaller.

IV. METHODOLOGY

During the course of exploring this area of research, we came across certain things that we have to know for research. These things are discussed below_

KEYWORD SEARCH

Keyword search[14] has been extensively used for extracting information from structured data such as relational databases, which can be modelled as graphs. Users of such databases usually do not have sufficient knowledge about the structure of data, and are often not familiar with query languages such as SQL. Thus, they need a simple system that receives some keywords and returns a set of nodes or tuples that together cover all or part of the input keywords. A node that contains one or more keywords is called a contentnode. Most of the existing methods for keyword search in graphs output either minimum connected trees or sub graphs that cover all or part of the input keywords. Both types of results show not only the content nodes that contain the input keywords but also some other nodes that connect the retrieved content nodes, which reveal some relationships among the content nodes. However, the tree-based methods have been criticized for producing too many trees with the same set of content nodes (albeit with different connections). Often, a user prefers to see different sets of content nodes that together cover the keywords and may not want to see the different relationships among the same set of content nodes. A graph based method can reveal multiple relationships within a single search result, and thus reduces the duplicated results. However, as we will show in this demo, the graph-based method may produce results that contain more irrelevant nodes than tree based results. In addition, in the current tree or graph based results, while some content nodes are close to each other, others may be far way from each other, meaning that weak relationships among content nodes may exist in the results. More importantly, current tree or graph based methods search through both content and non-content nodes in the graph for answers, which is very time-consuming when the input graph is large e.g., containing millions of nodes[14].

➢ PROPOSED SYSTEM

In our proposed system, we used the two algorithms i.e. Top-K algorithm and Keyword search algorithm. Our system work by following way_

Store The Files

For our minimal keyword search project we are taking keywords as an input.
First we store the files into database of various extensions, i.e. .pdf, .doc, .docx, .html, .xml etc. Different files are stored in a folder name as files as a database for choosing input for keyword search using graph. Here we can take any keywords as an input for file searching.

View all the files

After storing the files system checks how many files are stored in the database. Then in this each file is shown with their extension, it also shows the file format and also the size of that file. The next stage is keyword search.

Keyword Searching

In keyword searching, system first search the input keywords in database and the show the top –k files retrieved from database, i.e top-k file means the file which contained all the input keywords, then the remaining files are display. In this we also find out some other parameter for each file i.e total time and throughput time for individual file and it also shows the total lines in each file.

Algorithm For Keyword Search

In this project, we propose a new approach to keyword search over graph data which can produce not only relevant but also diverse results by searching top-k answers consisting of only reduced and duplicate-free answer trees[6].

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

Algorithm1 Keyword Search
Input: a keyword query q = {k1, k2, …, kl}, k€Z+
Output: a set of top-k answer trees for q
1: a priority queue Qtand a set C of nodeID'sbyφ
2: curRel[i] ← 0.0 and V[i] ← null for all I €[1,l]
3: Let L(q) = {KNList(ki) | ki€q (1<I <l)}.
4: **while** an entry exists in a list in L(q) **do**
5: Select a list Li in L(q) in a round-robin manner.
6: Read an entry (n, v, f, r) at the current position in Li.
7: curRel[i] ← r
8: if n € C then
9: V[i] ← (v, f, r)
10: **for-each** kj€ q such that j ≠ i**do**
11: Look up the first entry (vj, fj, rj) with key (n, kj) in NKMap.
12: **if** the entry was found **then** V[j] □ (vj, fj, rj)
13: **else** goto line #21
14: **if** T(n, V) is a non-reduced answer **then**
15: V ← findReducedAnswer(n, V, q)
16: **if** V ≠ □ and T(n, V) is a duplicate answer **then**
17: V ← findUniqueAnswer(n, V, q)
18: if V ≠ φ then Qt ← Qt ∩ {(n, V)}
19: C ← C ∩ {n}
20: **if** |Qt| = k and **then break**
21: Derive top-k answer trees from the top-k entries in Qt.

Given a query q = {k1, k2, …,kl}, let L(q) be the set of keyword-node lists KNList(ki) for all keywords kiin q. The algorithm performs sequential scan on the lists in L(q) in parallel (line 5~6). Whenever a new entry is read from a list, its relevance value is stored in an array curRel(line 7). If an entry (n, v, f, r) regarding an optimal path from a node n is first retrieved from L(q), entries of the optimal paths pm(n, kj) for all the other keywords kjin q are looked up in NKMapand aggregated into an array V (line 8~13). If all the optimal paths are found, an optimal answer tree rooted at n can be derived. We examine whether it has a reduced form and has a unique set of content nodes compared to the other candidates in top-k queue. If it does not, we seek an alternative reduced and unique answer tree using algorithms which will be detailed later (line 14~17). The result tree is stored in Qtif it is one of the k most relevant found yet (line 18). Since the entries in each list are sorted in a decreasing order of relevance, the sum of the values in curRelcan serve as an upper bound of relevance of the answer trees which have not been found yet. Thus, the algorithm can terminate safely with the correct top-k answers in Qtif the condition in line 20 is met, where relkis the relevance of the k-th answer tree in Qt.

4.2.5. Finding Top-K minimal answers

Assume that the maximum number of nodes containing a query keyword in the input graph is m. Based on the definition of Answer, the total number of Answers might be up to ml, where l is the number of query keywords. Apparently, producing all of the Answersmay overwhelm the user since m and/or l can be large. Thus, it is important to produce top-k Answers(or all the answers if fewer than k answers exist) in a ranked order. The efficiency of a search engine is commonly measured based on the delay between producing two consecutive answers. If this delay is polynomial based on the input data, the algorithm is called a polynomial delay algorithm. Our algorithm for producing top-k duplication-free answers is an adaption of Lawler's procedure for finding top-k answers to discrete optimization problems. In Lawler's procedure, the search space is divided into disjoined sub-spaces. The best answer in each subspace is found and used to produce the current best global answer. The sub-space that produces the best global answer is further divided into sub subspaces and the best answer among its sub-subspaces is used to compete with the best answers in other sub-spaces in the previous level to find the next best global answer. Two main issues in this procedure are how to divide a space into subspaces and how to find the best answer within a (sub)space. To have duplication free answers, the procedure for dividing the search space into sub-spaces must produce disjointsub-spaces so that the same answer cannot be generated from different sub-spaces[1].

Algorithm 2 Generate Duplication Free Top-k Answers
Input: the input graph G; the query Q = {k1, k2, . . . , kl}; k
Output: the set of top-k ordered Answers printed with polynomial delay
1: C ← an empty set for storing content nodes

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

2: **for** i← 1 **to** l **do**
3: add the nodes in G containing ki to C
4: Queue ← an empty priority queue
5: A ← **FindBestAnswer**(G, Q, C, ∅, ∅)
6: **if** A _= NULL **then**
7: insert _A, ∅, ∅_ into Queue
8: **while** Queue _= ∅**do**
9: _A, Inc, Exc_ ← top element of Queue
10: **print**(A)
11: k ← k − 1
12: **if** k = 0 **then**
13: **return**
14: {n1, n2, . . . ,np} ← content nodes of A
15: **for** i← 1 **to** p**do**
16: Inci← Inc∪ {n1, . . . ,np−i}
17: Exci← Exc ∪ {np−i+1}
18: **if** Inci∩ Exci = ∅**then**
19: Ai ← **FindBestAnswer**(G, Q, C, Inci, Exci)
20: **if** Ai _= NULL **then**
21: insert _Ai, Inci, Exci_ into the right place of Queue according to Ai's weight

Algorithm calls the FindBestAnswer procedure to find the best answer in a search space specified by a set of content nodes and the constraints (i.e., the inclusion and exclusion sets). The best answer must contain the nodes in the inclusion set, exclude the nodes in the exclusion set and also have an optimal weight. Depending on the weight function used, FindBestAnswer can be designed differently.

V. SYSTEM MODULE

- **Data flow diagram:**

A Data Flow Diagram (DFD) is a graphical tool that allows system analyst (and system users) to depict the flow of data in an information system. The DFD is one of the method that system analyst used to collect information necessary to determine information system requirements. Data flow diagram (DFD) is used to show how data flows through the system and the processes that transform the input data into output. Data Flow diagram is a way of expressing system requirements in a graphical manner. DFD represents one of the most ingenious tools used for structured analysis.

Level 0

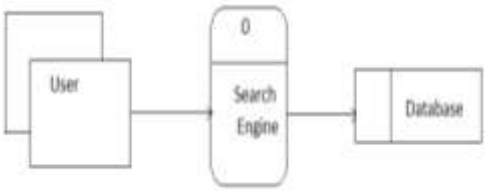

Figure 2: Level 0 DFD for Proposed System

This above Level 0 DFD represents shortly how our system will flow and user will interact with system for searching keywords, so system containing Search Engine for process of find files from huge data collection means database.

Level 1

In level 1 Data Flow Diagram Is intended to serve as a communication tool among

- System analysts
- End users
- Database designers
- System programmers
- Other member of the project team

In this DFD user representing by external entity is the origin of destination of data. Entities are external to the system. In this DFD shows number of processes doing in our system, with their sequence flow, process-performs some action

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

on data, such as extract the keywords, retrieve files, re-ranking the sequence of files etc. and Data store is shows the data collection in database.

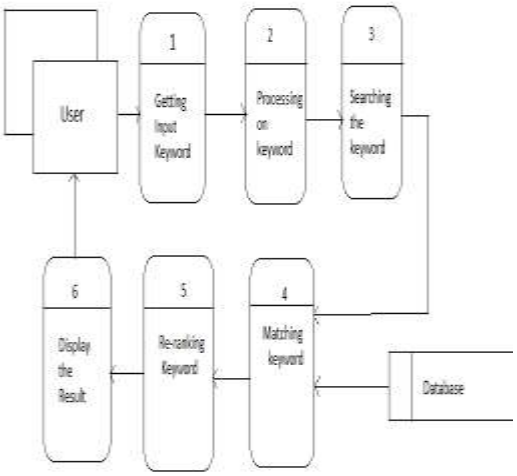

Figure 3: Data Flow Diagram

Figure 3 shows the data flow diagram of our system. In the above Figure 3 when user get the query it passes to next level i.e. Getting Input Keyword the that keyword are process and the system search for that keyword. Now, system match the keywords into database the re-ranking that keywords. Finally display the actual files that we want.

- **Unified Modelling Language**

Unified Modelling Language (UML) is a graphical language for visualizing, specifying, constructing and documenting the artifacts of software . UML is expressive language, addressing all the views needed to develop and then deploy such system.

Use case Diagram

Use case diagram is one of the UML diagram which specify actors and their roles in the system.Following figure shows actors as user and system and their roles are also specified. Roles of the actor are Entering input keyword, Pick relevant files. Roles of the system are giving extract the relevant keyword, match the keywords with files in database and store in database, ranking the files base on input keyword and displaying relevant result.

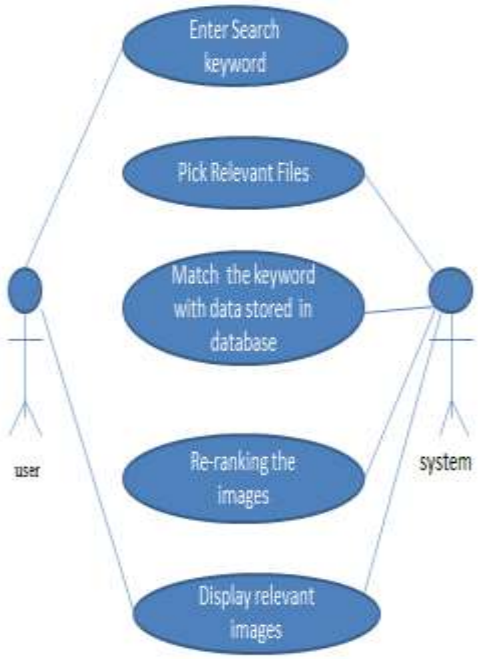

Figure 4: Use Case Diagram

DOI10.17148/IJARCCE.2017.6525

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

VI. EXPERIMENTAL RESULT AND ANALYSIS

Using the application we have the two important modules in our system i.e. first one is system shows all the files which are stored in our database and second is our actual keyword searching module. In keyword searching according to the keyword input graph will be generated and also shows the time required for searching.

Result-1 for efficient duplication free

Figure 5: View all files module

Figure 5 shows all the file from the database that we are uploaded in attaché file module. In this first it shows the total uploaded files in the system, each file having its particular id which we can given at the time of uploading. In this name of the file will be shown with their extension. It also shows the size of file in bytes and total lines in that file.

Result-2 for searching 1-Keyword

After the viewing of all the files, we can move to keyword searching module. In this we have given a keywords as an input, on the basis of input keywords graph will be generated and the also shows the each file in the system which contain the input keywords.

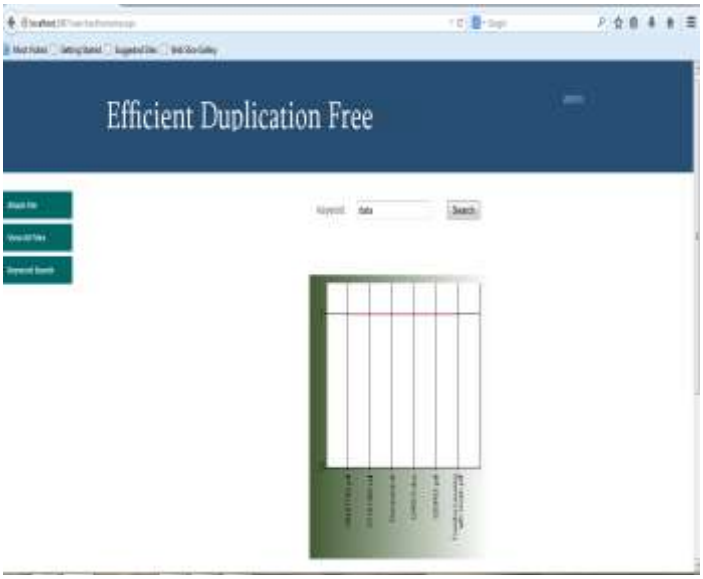

Figure 6: Graph for one keyword

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

Figure 6 shows the graph for only one input keyword. In the graph x-axis shows the name of the file containing the input keyword and y-axis shows the number of keywords. For only one input keyword following factors are also calculated by our system. In the above line graph, it will show straight line because we will search for only one input keyword.

Table 1:Time Calculation for searching one keyword

Sr. No	Name of file	Extension	File Size	Throughput Time (sec)	Total Time (sec)	Total lines
1	06427742	.pdf	2582711bytes	0.268015	2.161222	14580
2	0716160	.pdf	967739bytes	0.161009	2.428138	8435
3	Document	.rtf	1740bytes	0.020001	2.501143	11
4	IJARCS	.doc	473600 bytes	0.397022	3.116182	3030

Time required for searching one input keyword. On the basis of timing constraints the following graph will be generated.

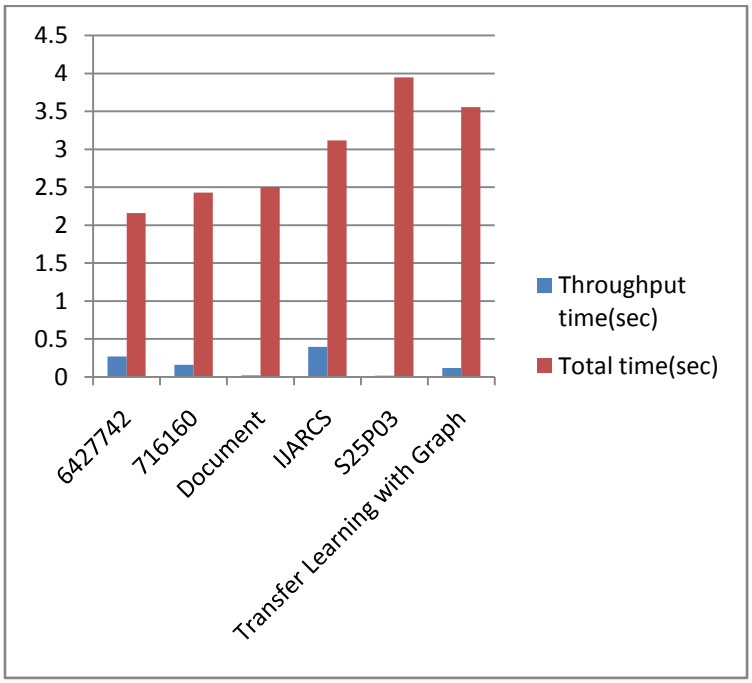

Figure 7: Comparison of Throughput and Total Time for one keyword

Figure 7 shows the camparison of throughput and total time for one keyword search. The graph shows the throughtput and total time in second. Throughput time means the time required for actual searching and total time means the time required for overall searching.

In the graph blue bar shows the throughtput time and red bar shows the total time. Throughput time is more than total time. Throughput time and total time will be depend on size of the file and total line. Y-axis shows the time of file required and X-axis shows the name of file.

Result-3 for searching of 2-Keyword

Now we have two keywords as a input, on the basis of that graph will be generated.

Figure 8 shows the graph for only two input keyword. In the graph x-axis shows the name of the file containing the input keyword and y-axis shows the number of keywords. For two input keyword following factors are also calculated by our system. In the above graph first file 07161360 contains two input keyword and second file Document.doc also contain two input keyword.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

Figure 8: Graph for Two keywords

Table 2: Time Calculation for searching two keyword

Sr. No.	Name of file	Extension	File Size	Throughput Time (sec)	Total Time (sec)	Total lines
1	07161360	.pdf	967739bytes	0.624001	2.161222	8435
2	Document	.rtf	1740bytes	0.0468000	2.428138	11
3	IJARCS	.doc	473600 bytes	0.0780001	2.501143	3030
4	S25P03	.pdf	7097721bytes	0.09360009	3.116182	6920

Table 2 shows the total and throughput time required for two input keyword. On the basis of timing constraints the following graph will be generated. The time will be depend on the size of file.

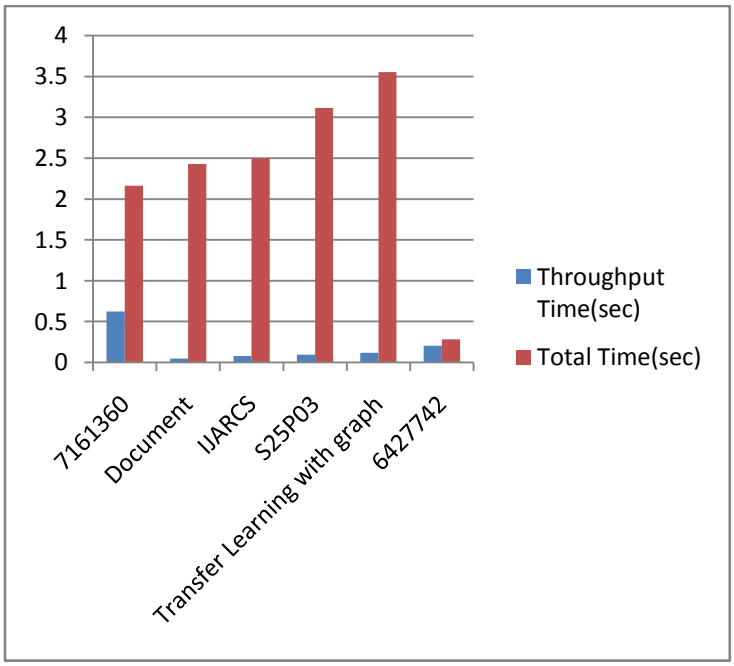

Figure 9: Comparison of Throughput and Total Time for searching two keyword

DOI10.17148/IJARCCE.2017.6525

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

Figure 9 shows the camparison of throughput and total time for one keyword search. The graph shows the throughtput and total time in second. The file which contain the all input keyword that file requires more throughput time. Throughput time means the time required for actual searching and total time means the time required for overall searching.
In the graph blue bar shows the throughtput time and red bar shows the total time. Throughput time is more than total time. Throughput time and total time will be depend on size of the file and total line. Y-axis shows the time of file required and X-axis shows the name of file.

Result-4 for searching of 3-Keyword
Now we have three keywords as an input, on the basis of that graph will be generated.

Figure 10: Graph for Three keywords

Figure 10 shows the graph for only three input keyword. In the graph x-axis shows the name of the file containing the input keyword and y-axis shows the number of keywords For three input keyword following factors are also calculated by our system. In the above graph first file Document.doc contains all three keyword that we are given second file also contain all three keywords. Then third file P60_kargar.pdf contains two keywords.

Table 3: Time Calculation for searching three keyword

Sr. No	Name of file	Extension	File Size	Throughput Time (sec)	Total Time (sec)	Total lines
1	Document	.rtf	1740bytes	0.0780001	0.4212006	11
2	IJARCS	.doc	473600 bytes	0.109202	0.5616009	3030
3	P681.krgar	.pdf	597760 bytes	0.0156000	0.9048015	5143
4	S25P03	.pdf	7097721bytes	0.046800	0.967201	6920

Table 3 shows the total and throughput time required for three input keyword. Time will be vary according to file content and size of file. On the basis of timing constraints the following graph will be generated.

Figure 11 shows the camparison of throughput and total time for one keyword search. The graph shows the throughtput and total time in second. Throughput time means the time required for actual searching and total time means the time required for overall searching.
In the graph blue bar shows the throughtput time and red bar shows the total time. Throughput time is more than total time. Throughput time and total time will be depend on size of the file and total line. Y-axis shows the time of file required and X-axis shows the name of file.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

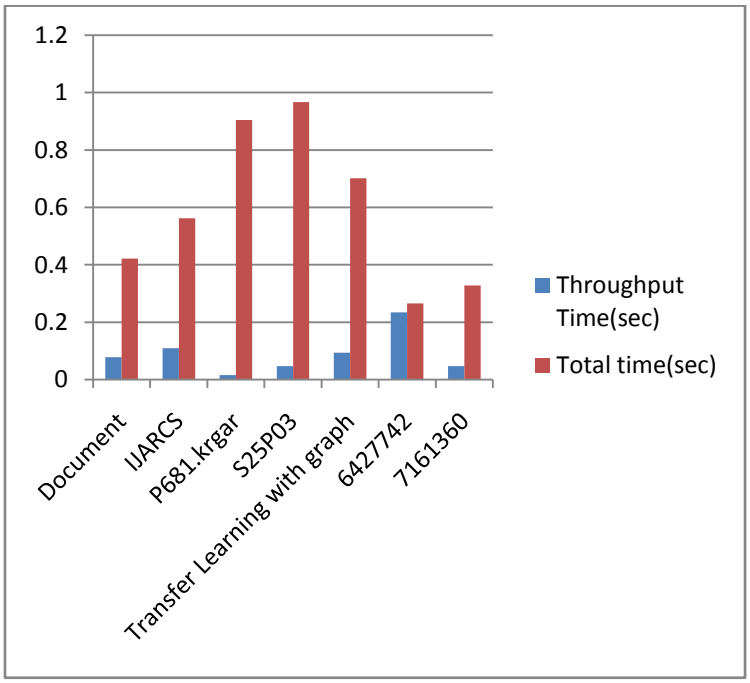

Figure 11: Comparison of Throughput and Total Time for three keyword

Camparison with Existing System

In this we will discuss about camparison between existing system and proposed system. New method that we proposed in our system is more efficient than existing system. Following are the few points of camparison between existing and proposed system.

Table 4: Comparison for different parameter

Sr No.	Contents	Existing System	Proposed system
1	Algorithm used	Greedy Algorithm	Top-K Algorithm
2	Time Constraint	Only total time calculate	Throughput & total time calculate
3	Time Required for Searching	Greater	Less
4	For Probability computation	Separate Stack is use	Only database is use
5	Answer Produce	Produced diversified answer	Produced unique answer
6	Generating Graph	More time required for graph generation	Less time required for graph generation
7	Searching efficiency	Medium	High

Table 4 shows the camparison between existing system and proposed system. In the existing system Greedy algorithm is used for searching and in our proposed system we used Top-K algorithm. In the existing system only total time will be calculated that's why time required for searching will be more as compared to our proposed system, because in the proposed system we calculate both total and throughput time means the actual searching time. For calculating the probability of searching, existing system used separate stack and in our system directly database is used.

In the existing system diversified i.e. duplicate answers are produce and in our proposed system unique answers are produced. In the existing system more time required for graph generation and in the proposed system instantly graph generated. And efficiency of searching keywords in existing system will be medium and keyword searching efficiency of proposed system is more.

Now we compare the time required for keyword searching for in that greedy algorithm [1] and proposed system in that Top-K algorithm. For this we have some timing for one keyword, two keyword upto the five keyword, mention in below table.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

Table 5: Comparison of time with existing system

Number of keywords	Time for Existing system(sec) (Greedy Algorithm)	Time for Proposed system(sec) (Top-K Algorithm)
1	1.0001	0.92040
2	3.9240	1.163602
3	5.00923	1.000340
4	6.8304	1.2132020
5	7.09236	2.13612

Table 5 shows the time searching for different input keyword in existing and proposed system. In existing system time required will be greater than proposed system. So our system is more efficient than existing system. Graphical representation of this comparison is given below. The result for greedy algorithm [1] taken from the existing work of the system and next result for proposed system in this we used top-K algorithm

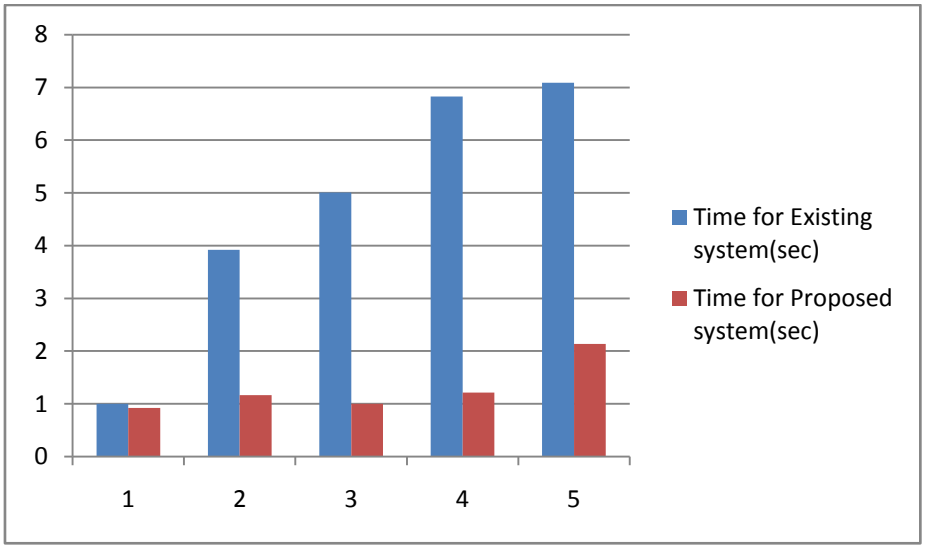

Figure 12: Comparison Graph for time with existing system

Figure 12 shows the comparisons graph for searching timing of keywords with existing system and proposed system. Blue bar indicates the time required for existing system and red bar indicates the time required for our proposed system. In the graph x-axis indicates the number of keywords and y-axis shows the time required in seconds. Existing system required more time than proposed system.

VI. CONCLUSION

This system proposed novel and efficient methods for keyword search in graphs. Our system also finds the top-k file with their ranking for input keywords. We also compare the throughput time and total time for different keyword input i.e. for only one keyword as a input the two keyword. Then we increase the keyword as input, from that we have different graphs for different number of input keyword. The time for searching the keyword in every file depend on their size and their number of lines in the file. For this we used the top-k algorithm for fast searching. Our approach is faster than previous because previous approach first search for unique answers the actual searching will be start that's they required more time.Finally, we show that the minimal answers have higher quality and also finds the exact file on the basis of input keyword.

VII. FUTURE SCOPE

Since we have implement our system from scratch thus there are many area where we can apply the keyword searching and we can also modified and extended our system. A keywordproximityqueryis a set of keywords and the results are trees of XML fragments that contain all the keywords and are ranked according to their size. XKeyword provides efficient keyword proximity queries on large XML graph databases. A query is simply a list of keywords and does not require any schema or query languageknowledge for itsformulation. XML and its labeled graph abstraction emerge as

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 5, May 2017

the data model of choice for representing semi structured self-describing data. Semi structured query languages provide features, such as flexible path expressions, that allow one to query semi structured data, i.e., graph data that are not characterized by rigid structure. However, one still needs sufficient knowledge of the structure, role of the requested objects and XQuery in order to formulate a meaningful query.

REFERENCES

[1] Mehdi Kargar,AijunandXiaohui Yu," Efficient Duplication Free and Minimal Keyword Search in Graphs", IEEE Transactions on knowledge and data engineering, vol. x, january 2013.
[2] H.He, H. Wang, J. Yang, and P. Yu, "Blinks: ranked keyword searches on graphs in Proc. ofSIGMOD'07, 2007.
[3] G. Kasneci, M. Ramanath, M. Sozio, F. M. Suchanek, and G. Weikum, "Star: Steiner-tree approximation in relationship graphs," in Proc. ofICDE'09, 2009.
[4] Chang-Sup Park "Reducing Redundancy in Keyword Query Processing on Graph", in JISE-32,2016.
[5] Mehdi Kargar and AijunAn,"Keyword Search in Graphs: Finding rcliques**",** ProceedingsoftheVLDBEndowment, Vol. 4, No. 10, 2011, pp .681-692.
[6] Chang-Sup Park "Keyword Search over Graph-structured Data for Finding Effective and Non-redundant Answers", IEEE Transaction vol X.
[7] K. Golenberg, B. Kimelfeld, and Y. Sagiv, "Keyword proximity search in complex data graphs," in Proceedings of ACMSIGMODConference on ManagementofData, 2008, pp. 927-940.
[8] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword searching and browsing in databases using BANKS," in ProceedingsofIEEE InternationalConferenceonDataEngineering, 2002, pp. 431-440.
[9] H. He, H. Wang, J. Yang, and P. S. Yu, "BLINKS: ranked keyword searches on graphs," in Proceedings of ACMSIGMOD Conference on Management of Data, 2007, pp. 305-316.
[10] Hang YuP, Zhihong Deng ,Yongqing Xiang, NingGaoP, Ming ZhangP, Shiwei Tang," Adaptive Top-k Algorithm in SLCA-Based XML Keyword Search", 12th International Asia-Pacific Web Conference,2010.
[11] B. Zhou and J. Pei."Answering aggregate keyword queries on relational databases using minimal group-bys". In Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, EDBT 2009, Saint Petersburg, Russia, March 24-26, 2009, pages 108{119. ACM, 2009.40
[12] Jianxin Li, Chengfei Liu, Rui Zhou, Wei Wang," Top-k Keyword Search over Probabilistic XML Data", Swinburne University of Technology, Australia
[13] Pei Cao, ZheWang ,"Efficient Top-K Query Calculation in Distributed Networks", Department of Computer Science Princeton University Princeton, NJ 08540
[14] Mehdi Kargar and AijunAn," Efficient Top-*k*Keyword Search in Graphs with Polynomial Delay", IEEE 28th International Conference on Data Engineering, 2012
[15] VagelisHristidis, YannisPapakonstantinou, AndreyBalmin," Keyword Proximity Search on XML Graphs", University of California, San Diego
[16] GauravBhalotia, ArvindHulgeri, CharutaNakhe, Soumen Chakrabarti," Keyword Searching and Browsing in Databases using BANKS", Computer Science and Engg. Dept., I.I.T. Bombay
[17] Syamily K R, G Naveen Sundar," Survey on Scalable Continual top-k Keyword search in Relational Databases"IJRET, 2013