TRIBHUWAN UNIVERSTIY

INSTITUTE OF ENGINEERING

Pulchowk, Lalitpur.

Project Report on

Knowledge Graph, A search engine

Submitted by:

Kishan K.C.

Roll No : 067BCT519

Submitted To:

Department of Electronics and Computer Engineering

Acknowledgement

I would like to thank our respected teacher, Dr. Shashidhar Ram Joshi for providing us guidelines for AI project. I would like to thank our teachers: Dr. Sanjeev Prasad Pandey and Anil Verma for providing us the opportunity to do the AI projects. Similarly I would also like to put my vote of thanks to my friends and my seniors.

Abstract

The main target of this project is to develop search engine to search the details of the keywords given by the users. Anyone can use this system to search about the data. The system is provided with the number of data as database in form of files. The system will then ask for the two inputs. The system analyses the relation between the two words matching their keywords with that of database. The system takes two input, then searches the keyword in input into the database, then determines the relation between the inputs on the basis of matching keywords.

Contents

1. Introduction

1.1 Currently Existing Technologies

Currently knowledge graph is implemented by giant search engine Google and the most popular social networking site Facebook.

1.1.1 Google Knowledge Graph

For many years search has been about matching keywords to queries. Search engines then were not able to understand real world entities and their relationships. The query "Darjeeling Tea" might return results about the tea brand even if the user was in fact interested about the place Darjeeling and its popularity for tea. Now google tends to solve the problem of this ambiguity by introducing knowledge graph. The knowledge graph enables users to search for things, that it knows about eg. landmarks, celebrities, movies, sport teams ,countries etc and many more. This is the first step in making search engines work as our human minds do. Google has 500 million objects and 3.5 billion facts about and relationships between them.

Knowledge graph enhances search by these ways:

- Finding the right thing
- Getting best summary

Search queries can be ambiguous. It would be best if the search engine itself can understand what the user meant and provide result only for those queries instead of providing all the results which contains the words in user queries. Now google does this by implementing knowledge graph. The search result for query "Taj Mahal" is now narrowed by differentiating whether the user meant the monument or the musician.

Usually user is not interested in the number of links search engines provide but instead he is not interested in the information clustered from all the sources. Google now provides user with summary about the object he is interested in along with the links of pages containing the information. For eg for a query "Sir Issac Newton" google will display all the major information about him like date of birth , his discoveries , where he was born , his achievements, date of his death etc . So the user does not need to open all the pages

and search for the information he requires by himself. The search engine itself is smarter on doing this.

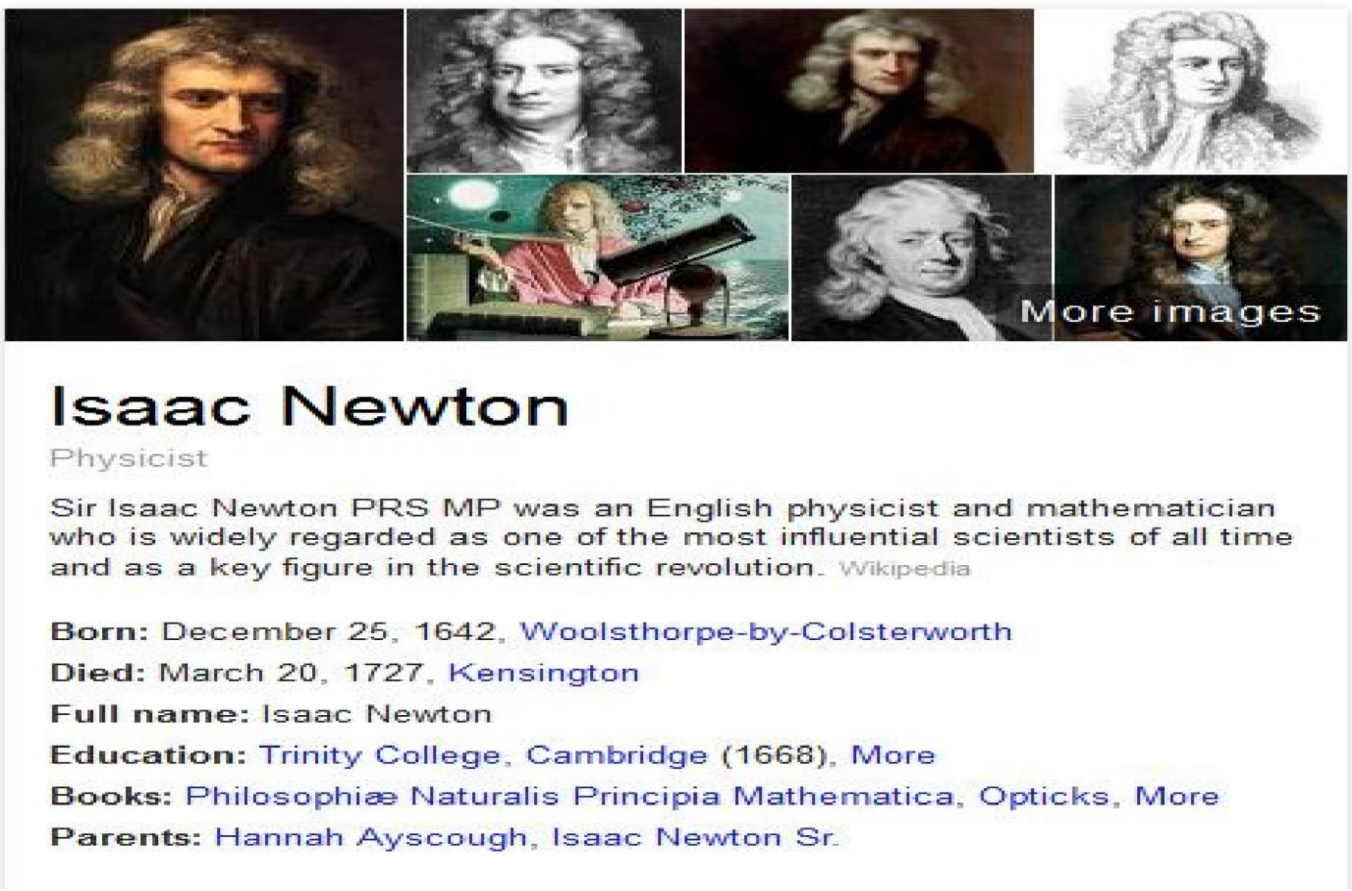

Fig1: Search result for "Sir Isaac Newton"

1.1.2 Facebook Graph Search

The Graph Search Feature was launched in January 2013. Rather than returning results based on matching keywords, the search engine is designed to match phrases, as well as objects on the site.

Search results are based on both the content of the user and their friends‟ profiles and the relationships between the user and their friends. Users can now search for the nearby restaurants, malls etc . Users can filter their search like friends from Delhi who are single and the friends who like specific movies etc an much more.

1.2 Analysis of Previous Research in This Area

With the introduction of knowledge graph ,web search is not limited to key words matching. Google is trying to make web search more and more closer to the way human mind works. With the implementation of knowledge graph, instead of simply matching keywords, Google‟s technology now try to understand user‟s query, and bring back the information he is looking for. This is a critical first step towards building the next generation of search, which and understands the world a bit more like people do. Hence knowledge graph implements "things not strings" which means user queries are the real world objects not just the combination of characters.

The knowledge graph is only in the beginning stages, slowly being released onto the world, and therefore is very limited. The knowledge graph appears for well-known topics like landmarks, celebrities , movies , actors etc however, it often cites Wikipedia, which can easily be edited by anyone.

Facebook graph search is also limited within their internal database , it works on the basis of likes, shares, tags etc. Friends have a relationship to friends, pages and photos via likes and comments. And these pages, photos and groups have a relationship with each other. Hence a huge graph is formed.

As in above figure nodes can be nouns like friends, pages, groups, posts and user‟s name. How these nouns are related to each other are represented by the arrows (which include relationship attributes like "Friend," "Tagged," "Photo," "Event," "Likes," etc called "Edges." So, what we see in this diagram is how user Kishan KC is related to the page titled Manchester United .In this case, the relationship is via a "like".

1.3 Problem definition and scope

Every search engine tries to provide user with the best information for his query. Previously search engines were concerned about how to provide users with the links to the pages containing information about user query. Later it was realized that user is not interested in the number of pages provided by the search engines for his query but he needs actual information clustered together from different pages. The information might be distributed in many pages. To facilitate user not to open all the pages and collect information by himself search engines should be made closer to the way our minds work. For this , search engines should understand that the user inputs are not just **strings** but they represent real world **things.** It can be done by implementing Knowledge graph in web search.

Let us see how differently search engines and our minds work. Let‟s say someone asks a question "Who is Mr. X?" E1 be the event where I was talking to Ram and he mentioned "**X is a boy. He studies at IOE and he's a good guy**." as one of his statement along with many other. Similarly Kishan mentioned that "**X belongs to Kathmandu and will be going to USA for further studies**." as one of his statements in his long conversation.

Now I match X with E1 and E2 and conclude that "**X is a boy who studies at IOE and he's a good guy. X belongs to Kathmandu and will be going to USA for further studies**" as the answer to my question instead of E1 or E2 as a whole or may be recommended in some order.

Let input query be "Who is Mr. X" and E1 and E2 (same as above) be different pages from various websites. Search engines match the frequency of the word X in these two pages E1 and E2 along with the consideration of their Page Rank and display the links to these pages in some order. This is not the desired result for my query.

To make search engines work more like human brains they should be made understand multiple inputs from users and find relationship between them which may be scattered over the web and take union or intersection of such information and display it in a single page.

The **scope** of knowledge graph based web search is huge. As the search engines become more close to human mind information about any topic can be retrieved from the web in few seconds as web is full of information. This makes research in any topic more easier and faster. Search engines can understand user query and provide user with the only information user is concerned of. Thus making web search efficient every domain can take advantage from it. From education to health domain from technology to research, every field can be fruitful of knowledge graph based web search.

1.4 Formulation of the present problem

Web search can be made work like human brains by implementing knowledge graph. For this search engines should be able to understand that the user inputs are not just set of characters called strings but are real world things. Search engines should also understand multiple inputs provided by users and find relationship between them instead of matching keywords only. Then search engines should provide users with the collected information.

Lets take an example, if I provide multiple inputs "Darjeeling" and "tea", instead of searching for the pages containing both the words or the pages containing at least one word, search engines should be smart enough to interpret that by Darjeeling Tea I want to extract information about the place Darjeeling and tea, a drink.

This can be done by implementing knowledge graph for multiple search inputs. For this, we documents should be **tokenized** on the basis of dictionary words like places, names, monuments etc . Then these objects should be **indexed.** Indexing refers to the process of tagging these tokens for retrieving their location. Then **Knowledge Graph** is made taking all the objects or tokens. The tokens act as nodes and the sentences where they belong act as vertices.

Whenever user provides multiple queries, all the information can be retrieved from the graph because the path linking these two objects is the information user is interested in.

2. Theoretical Tools: Analysis And Development

2.1 Tokenization of web documents

Tokens are the smallest individual unit in a c /c++ program .Tokens are also known as lexical unit. A token is a group of characters that logically belong together . C/C++ tokens can be classified as follows:

1. Keywords
2. Identifiers
3. Constants
4. Strings
5. Special Symbols
6. Operators

Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing .Tokenization of all the documents is necessary before making knowledge graph. Only the words which are required for further processing are taken into account in tokenization. For example words like „the‟ , „are‟ , „and‟ , „is‟ etc are not required for making graph hence they are neglected. For all the content of every documents it is checked whether the words are useful or not . Not useful words are just neglected and only the useful words are taken care of. For example in sentence : Kishan studies at IOE.

Only Kishan and IOE are tokenized and „at‟ is ignored.

2.2 Graph

Using these tokens a network can be formed called graph. A graph is a collection of nodes called vertices, and the connections between them, called edges. When the edges in a graph have a direction, the graph is called a directed graph or digraph, and the edges are called directed edges or arcs. An undirected graph can easily be implemented as a directed graph by adding edges between connected vertices in both directions.

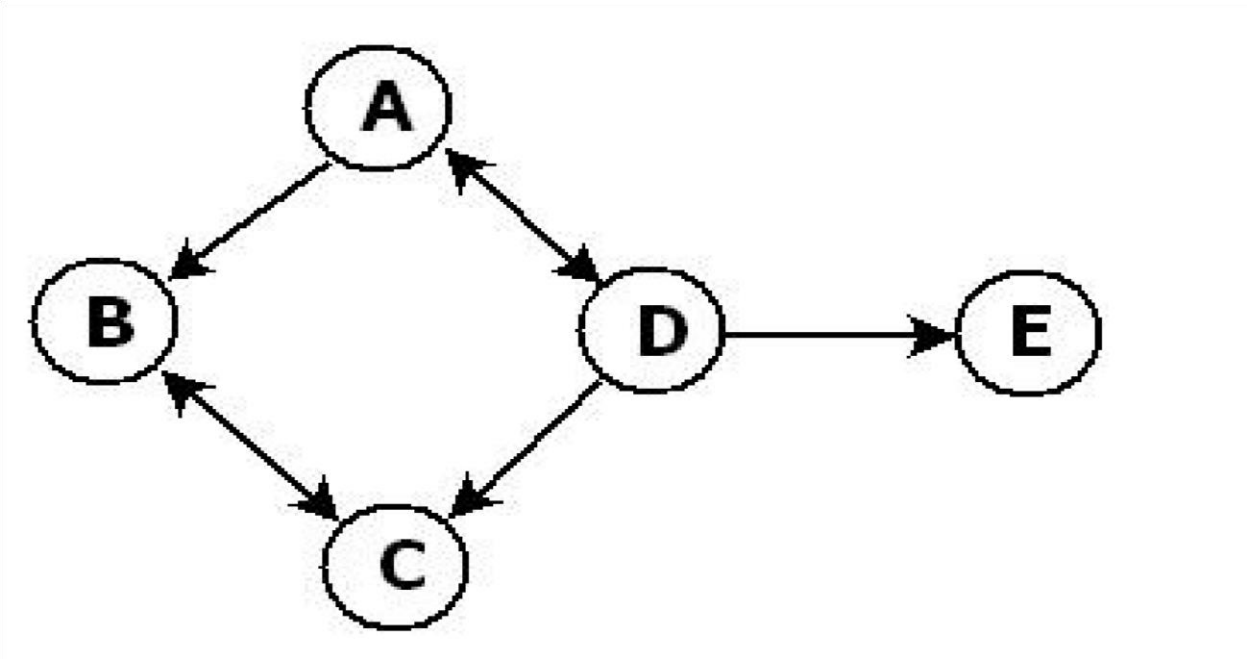

Fig2: A graph with vertices A,B,C,D and E[5]

Formally, a graph is an ordered pair, G = <V, E>, where V is the set of vertices, and E, the set of edges, is itself a set of ordered pairs of vertices.

For example, the following expressions describe the graph shown above in set-theoretic language:

V = {A, B, C, D, E}

E = {<A, B>, <A, D>, <B, C>, <C, B>, <D, A>, <D, C>, <D, E>}

Above is an example of graph with five vertices and seven edges. A graph implementation needs a basic set of functions to assemble and modify graphs, and to enumerate vertices and edges.

2.3 Path Finding

Path finding in graph refers to starting at one vertex and exploring adjacent nodes until destination node is reached. Whenever user puts multiple queries there exists one or more paths between them. Sometimes the shortest path is the desired one. There are several algorithms that can be used in finding paths between two nodes. For example:

- Depth First search
- Breadth First Search
- A* Algorithm
- Dijkstra‟s Algorithm and many more.

2.4 Depth First Search

DFS involves traversal of graph. Traversal of a graph means visited each node and visited only once . It is like pre-order traversal of tree. Traversal can start from any vertex , say Vi.

Vi is visited and then all the vertices adjacent to Vi are traversed recursively using DFS.

Since a graph can have have cycles , re-visiting anode should be avoided . This can be done by marking a vertex v as visited. This is done with the help of a global array visited[] which is initialized to false(0)

Algorithm

```
n← number of nodes (i) Initialize visited[] to false(0)
   For ( i = 0; i<n; i++)
Visited[i]=0;
(ii) void DFS (vertex i) (DFS starting from i)
{
  Visited[i] = 1;
  For each w adjacent to i
            If (!visited [w])
               DFS(w);
}
```

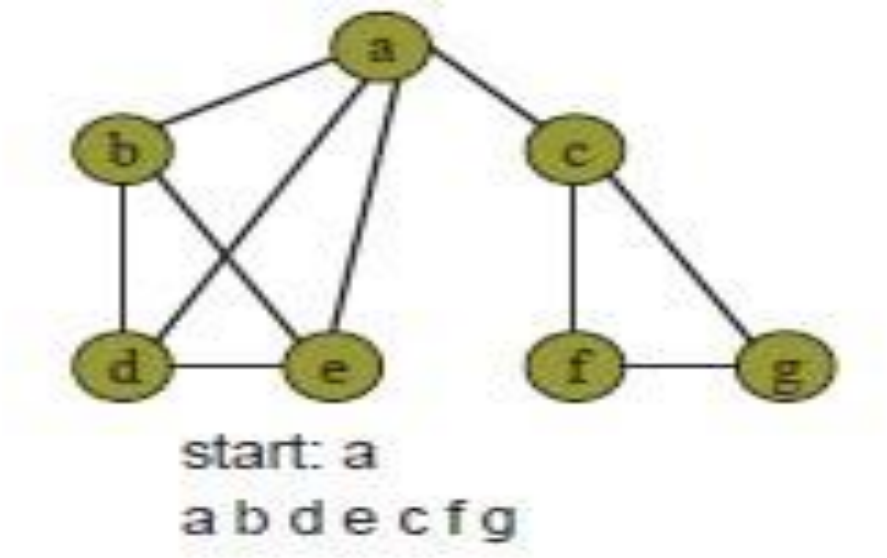

Fig3: Example of DFS starting from a traversing all the nodes

DFS on a graph with branching factor b and depth d has $O(_b\, d+1)$ worst time and O(bd) worst space complexity

3. Development of Software

The whole project is divided into four modules. These modules are as follows:

1. Tokenizing Web Documents
2. Indexing
3. Forming Knowledge Graph
4. Performing Search and Constructing Information.
5. Displaying Information

3.1 Tokenizing documents

A **token** is a string of characters, categorized according to the rules as a symbol. The process of forming tokens from an input stream of characters is called **tokenization.** A token can look like anything that is useful for processing an input text stream or text file. We need to break the document with respect to dictionary words eg. places, names etc contained in the document.

For example :

If there are four documents with many sentences

Document1 consist of sentence S1: “Kishan studies at IOE.”

Document2 consist of sentence S2: “IOE is in Lalitpur.”

Document3 consist of sentence S3: “Hari was born in Lalitpur.”

Document4 consist of sentence S4: “Bagmati is a holy river.”

Tokenizing a line of Document 1-“Kishan studies at IOE.” We need to break it such as S1 contains Kishan, IOE.

Similarly contents of all documents are tokenized as:

Document 1 S1- “Kishan studies at IOE.” tokenized as Kishan, IOE
Document 2 S2- “IOE is in Lalitpur.” tokenized as IOE, Lalitpur

Document 3 S3-“ Hari was born in Lalitpur.” tokenized as IOE, Lalitpur

Document 4 S4 - “Bagmati is a holy river.” tokenized as Bagmati, Holy,river

3.2 Indexing

All search engines keep index of the key words in web pages so that the pages can be retrieved whenever user provides the search queries . Hence it is necessary to index the documents . After forming tokens , the tokens are indexed .In above example tokens Kishan, IOE, Lalitpur, Hari are indexed so that the sentences they come from and the document can be retrieved according to the user’s query .

3.3 Forming Knowledge Graph

After tokenizing all the documents and indexing the tokens, a graph can be formed using the tokens as the nodes and the sentences they belong to as the vertices. Hence the content of the documents can be now represented as a network graph known as knowledge graph.

Taking the above example:

S1: “Kishan studies at IOE.”

S2: “Sudha studies at IOE.”

S3: “IOE is in Lalitpur.”

S4: “Sudha was born in Lalitpur.

Tokens are Kishan, IOE, Sudha, Lalitpur.

Now taking these tokens as nodes and the sentences where they occur as edges a graph can be formed as below:

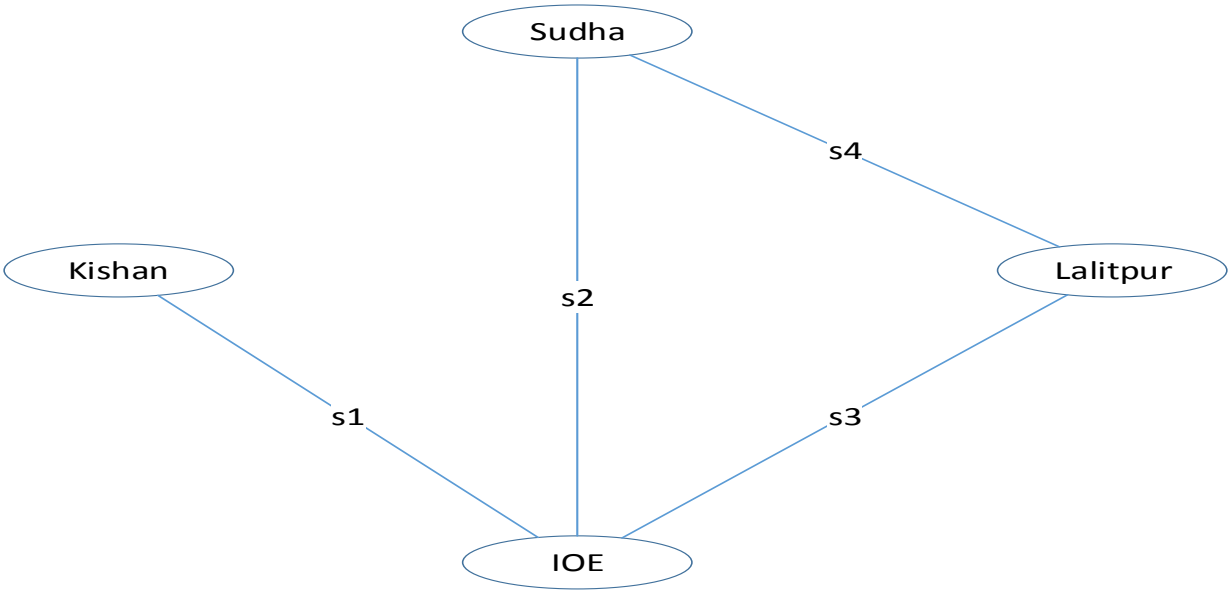

Fig 4: Knowledge Graph

3.4 Performing Search

Now let"s say the user inputs the query "Kishan and Sudha". Instead of considering this as one query and trying to find a single document containing both these terms we would like to find the information linking kishan and sudha and then summing up information to make it meaningful to the user.

As we look at the knowledge graph,
There are two ways we can reach from kishan to sudha i.e

1. S1 S2

2. S1 S3 S4

The first route actually means that kishan and sudha both study at IOE. The second route means Kishan studies at IOE which is in Lalitpur where Sudha was born.

Both the routes make perfect sense and might be the exact what the user was looking for. Instead of providing user with the documents containing both the documents knowledge graph search can provide user the best information he is looking for clustering all the information in different documents. Hence adding human cognition to search knowledge graph search does not treat user inputs as mere strings but as real world things.

Discovery of route can be done with the help of Depth First Search (DFS).

3.5 Displaying Information

Once the Information has been fetched it has to be displayed in a manner such that the user is able to make the able to understand the content and conclude various things and features of difference and similarities. A web page is used for this purpose. The page is written in Php and hosted using wamp server which can display information according to user query from several documents stored as database.

4. Testing and Analysis

Software testing is any activity aimed at evaluating capability of a program or system and determining that it meets its required results. I have taken some test cases and tested the project. The test process of Knowledge Graph Search is explained as:

In database file there are some text documents containing some sentences.

Db1 consists of sentences: Kishan studies at IOE. Kishan is from Bangalore.

Db2 consists of sentences: IOE is in Lalitpur. IOE is one of the best colleges in Lalitpur.

Db3 consists of sentences: Sudha was born in Lalitpur. Lalitpur is in CDR.

Db4 consists of sentences: Sangam is a holy river. Sangam is in Lalitpur.

Db5 consists of sentences: There are many colleges in Lalitpur. ACEM is in Lalitpur.

If an user is interested in finding relationship between Kishan and Sudha, he inputs his queries in two fields of the search engine as Sudha and Kishan. From above sentences we know that Kishan studies at IOE (from Db1) which is in Lalitpur (Db2) where Sudha was born (Db3). Hence result should be three sentences.

By analyzing the result it can be concluded that the search engine is capable of understanding multiple queries provided by user and find relationship between them by searching several documents and providing user a single page full of information he requires.

5. Conclusions

Web Search optimization has always been an area of huge research and work. People are working to make search engines more efficient to provide user with the best information according to his query. Recently research is being done in order to make web search human

cognitive and make search engines work like our minds do .I have tried to implement knowledge graph in multiple inputs queries.

After implementing knowledge graph based search and seeing the results it can be concluded that the knowledge graph based search is more and more similar to human cognition which recognizes user inputs as real world things rather than mere set of strings. It is able to provide user with the information distributed in different pages clustering together .Hence the project gives a very good approach in making search engines more human cognitive. Further research and works are required to make the project more efficient and work in real time.

6. Recommendations and Future Work

This project tries to provide an approach to make the web search more similar as human mind works. It provides user with a single page which contains the information scattered in different documents. Hence user should not open all the pages and make decisions by himself. Instead he can easily get the desired information from the search engine itself.

Also the inputs are not treated as mere "strings" but are treated as real world "things" and differences and similarities between them are fetched using knowledge graph instead of searching for the pages containing both the words and sorting the pages on the basis of link popularity and words relevancy.

The project does not work in real time .Future works can be done to make it work for real time. Using Natural language processing, the project can be extended as a search engine which can work in real time .Also a browser extension can be made which can differentiate between single input and multiple inputs whenever a user makes search in search engines like Google.

For example if an user provides his query as Darjeeling and tea , present days search engines treat it as single input and provide user with the pages containing both Darjeeling and tea , as a result search engines might provide misleading results . Darjeeling tea can be treated as a brand. In reality user might be interested in relation between place called Darjeeling and drink tea. Knowledge graph based web search will solve this problem.

Hence a huge research and lots of work can be carried out in this field.

Appendix - Explanation of the Source Code

The project uses c++. There are two C++ programs , main.cpp and merge.cpp.

1. Merge.cpp

It merges all the content from different documents in a single text file named final_merged.txt. Then it splits the whole document into individual sentences so that there is single sentence in each line. It is necessary for indexing because sentences act as edges in the graph. So during indexing sentence number is necessary.

System command “type *.txt >> merge.txt” is used to merge all the text files from folder Database. After merging all the text documents into merge.txt, each sentences are arranged in a single line. Whenever a full stop or question mark is found , the sentence is printed in new line.

2. Main.cpp

There are several functions in it. They are described briefly below:

- **Void read ();**

This function opens the text document final_merged.txt and reads individual strings from the file then it performs vector operation push_back to store the strings in vector of string named vector<string> input.

- **bool checkSW(string s)**

This function checks whether the strings from input file final_merged.txt are useful for tokenizing or not . The words that can be ignored are defined in an array called string stopwords[SW].

string stopwords[SW] = { "the", "of", "in", "and", "a", "an", "to", "with", "is", "was", "were", "by", "that", "for", "be", "from", "as", "are", "on", "or", "this", "an", "not", "it", "which", "with", "too", "at", "these" };

The function checks the strings are one of these stop words or not.

- **void tokenize()**

This function tokenizes the strings contained in final_merged.txt if they are not one of the stop words defined above.

- **void createIndex()**

After tokenization it is necessary to index the tokens , this function tags each token with the sentence number it belongs to.

- **void userInput(string arg1 , string arg2)**

This function takes input from the user. User is provided with two fields in search engines. He can provide his query which must be a string in these fields.

- **void dfs(string w, int level, string cs, string cd)**

Using tokens as nodes and sentences where they occur as edges a graph is made .

- **void execute()**

This function executes dfs and finds the path between two search words in the graph.

- **void output()**

This function provides the result in result.txt. The result is the path between two search words provided by the user.

- **int main(int argc, char* argv[])**

This function calls all the functions discussed above.

```
int main(int argc, char* argv[])

{      for (int i = 0; i<argc; i++)

  printf("argv[%d] = %s\n", i, argv[i]);

    read();

cout << endl;

tokenize();

cout << endl;

createIndex();

cout << endl;

userInput(argv[1],argv[2]);

cout << endl;

execute();
```

```
cout << endl;
output();
return 0;
}
```

References

1. Amit Singhal.2012 May 16, *Introducing Knowledge Graph: Things not strings.* Available: http://googleblog.blogspot.com/2012/05/introducing-knowledge-graphthings-not.html

2. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, "Tokens, Patterns, and Lexemes," in Compilers *Principles, Techniques, and Tools,* second edition.Pearsons Addisons Wesley. 1986 January1, pp.11.

3. Dilip Kumar Sultania ," Traversal of Graphs," in *Data Structures Using C.* Tech-max publications, Pune, August 2009, pp.8-19.

4. Thomas Steiner, Ruben Verborgh, Raphael Troncy, Joaquim Gabarro, and Rik Van de Walle ,"Adding Real time Coverage to the Google Knowledge Graph," Universitat Politècnica de Catalunya – Department lsi, Spain , 2012

5. Samir Khuller, "Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland," Balaji Raghavachari, "Department of Computer Science, University of Texas at Dallas," "Graph and Network Algorithms,"2012
6. Google. (2013). How Search works. In Google Inside Search. Retrieved June 28, 2013, from http://www.google.com/intl/en/insidesearch/howsearchworks/thestory/.
7. Harrison Weber. (June 6, 2013). How Facebook's Entity Graph evolved from plain text to the structured data that powers Graph Search. In The next web. Retrieved June 29, 2013, from http://thenextweb.com/facebook/2013/06/06/the-evolution-of-facebooks-entity-graph-the-structured-connections-behind-graph-search/.
8. SEW Staff. (March 13, 2007). How Search Engines Work. In Search engine watch. Retrieved June 25, 2013, from http://searchenginewatch.com/article/2065173/How-Search-Engines-Work.
9. Search engine details. Retrieved June 25, 2013, from http://ads.harvard.edu/pubs/A+AS/2000A+AS..143...61E/node4.html.
10. Linda Barlow. (November 05, 2004). How To Plan The Best Search Strategy. In How To Use Web Search Engines. Retrieved June 26, 2013, from http://www.monash.com/spidap1.html