Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication Online Open Access
Vol: 41 Issue: 11-2022
DOI 10.17605/OSF.IO/DA27F

DEVELOPMENT OF A NOVEL RANKING MECHANISM AND SEARCH ENGINE IN WEB OF THINGS

MUHAMMAD REHAN FAHEEM*
School of Systems & Technology, Department of Computer Science, University of Management & Technology, Lahore, Pakistan.

TAYYABA ANEES
School of Systems & Technology, Department of Computer Science, University of Management & Technology, Lahore, Pakistan.

MUZAMMIL HUSSAIN
School of Systems & Technology, Department of Computer Science, University of Management & Technology, Lahore, Pakistan.

ABSTRACT:

Nowadays, we live in a global village where many physical objects (Things) are connected to the Internet and are accessible through a web interface. Recent researches on the subject show that there has been a significant increase in the number of IoT devices during the last decade, which has attracted many researchers. Efficient searching of the Things connected to the Internet requires indexing and ranking, which are the two vital factors in the performance of search engines. Ranking is based on parameters such as the types of Things, services, current and historical locations, descriptions and features. The existing ranking techniques use either services, features or current locations as parameters for ranking, but other parameters such as historical location, type, and description are not profoundly analyzed in terms of performance. In this paper, a novel ranking mechanism for the Web of Things Search Engine (RMoWoTSE) is proposed, which ranks the indexed Things efficiently by using a combination of parameters and individual parameters. Results indicate that the accuracy, precision, recall and f-measure of the proposed ranking approach (RMoWoTSE) is better when a combination of ranking parameters is used compared to using individual ranking parameters.

KEYWORDS: Discovery of Things, Indexing, Internet of Things, Query, Ranking, Selection, Web of Things

1 Introduction

Over the last few decades, the World Wide Web (www) has become the world's most extensive information base. Over time, this information base has increased. It is becoming more populated with knowledge. Many people use the World Wide Web to search for information (text, image, video, and audio) [1]. The user interface used by the World Wide Web for searching anything is known as a search engine. The search engine is a system which finds the data from the information base against the user's query. The idea of a search engine is based on Archie, the first tool available on the internet for searching the data in the 1990s. Then in 1991, the "Rise of Gopher" was created by McCahill [2], which searches for the names of files and titles which were saved in the index of Gopher. After the invention of the Gopher, many researchers started working on search engines. In 1993, the first web robot was invented by Matthew Gray. The purpose of the web robot was to measure the web page's size. In

ISSN：1671-5497
E-Publication Online Open Access
Vol: 41 Issue: 11-2022
DOI 10.17605/OSF.IO/DA27F

1994, the first "text-based crawler" search engine was invented. Hence, much work has been done to make the efficient search engines we use today.

Due to the increasing importance of search engines nowadays, researchers are designing and developing a search engine that finds physical things [3] such as sensors, devices, actuators, etc. The search engine used to find physical things is known as the Internet of Things (IoT)/ Web of Things (WoT) based search engine. The author represents Fig. 1 to understand the similar steps between the IoT/WoT search engines and the Traditional Search engines.

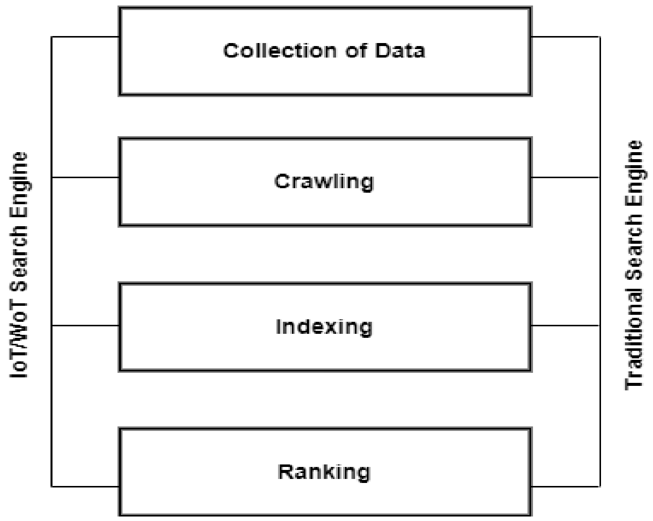

Figure 1: Similar Steps between IoT and Traditional Search Engine

As shown in Fig. 1, the similar stages between IoT/WoT and traditional search engines are collecting data, indexing, crawling, and ranking. Data collection is a step in which data is collected from different sources. Crawling is a process performed by a search engine to gather data and also check the changes which occur in the data. Indexing is a process in which the search engine saves and organizes the data found during the crawling process. Ranking is the process to display the results to the user at the top priority.

Fig. 1 represents that almost all the steps between IoT/WoT search engines and traditional search engines are similar; however, their working is different as shown in Fig. 2. Traditional search engines like Google, Yahoo, Ask and Bing store a vast number of documents as data that contain different kinds of information. Traditional search engines search a large number of documents, whereas the IoT/WoT search engine uses sensors' information as data. The information of sensors is dynamic as sensors change their physical positions while the information of documents is static in terms of physical location. In short, we can say that both search engines collect data, but the nature of data collection is different. The data is in text, audio, video, and images [1]. In IoT/WoT search engine, the data of Things may be numerical, symbolic, continuous, discrete, and static and collected through streaming [4].

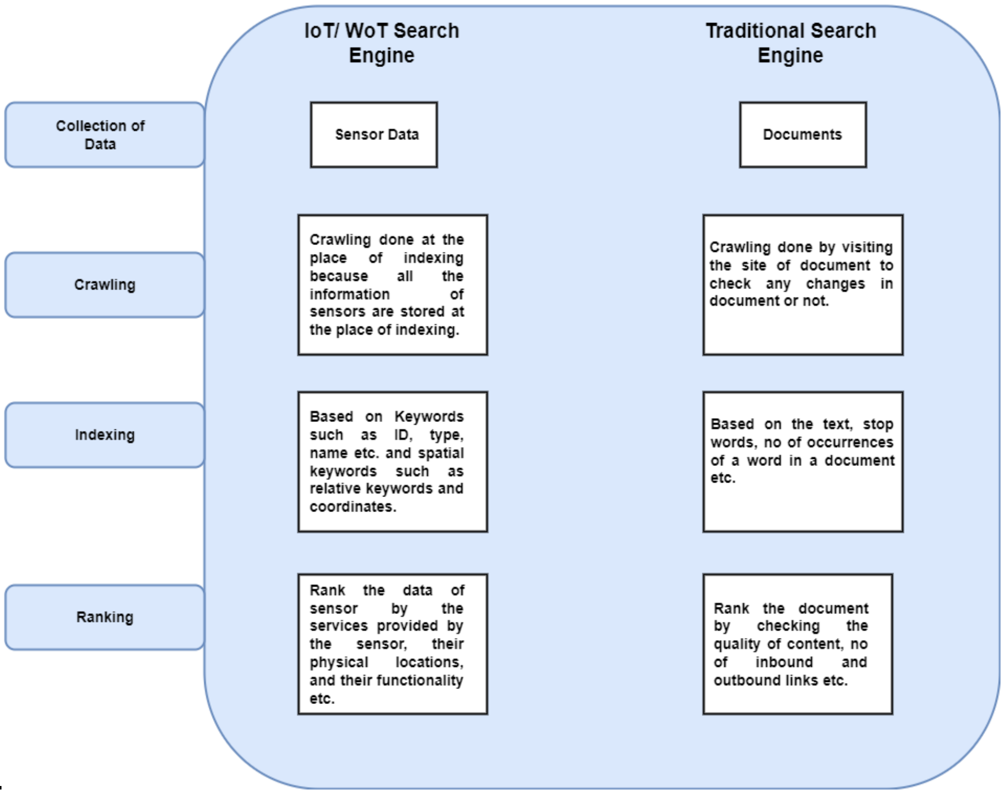
IoT/ WoT Search Engine
Traditional Search Engine
Collection of Data
Sensor Data
Documents
Crawling
Crawling done at the place of indexing because all the information of sensors are stored at the place of indexing.
Crawling done by visiting the site of document to check any changes in document or not.
Indexing
Based on Keywords such as ID, type, name etc. and spatial keywords such as relative keywords and coordinates.
Based on the text, stop words, no of occurrences of a word in a document etc.
Ranking
Rank the data of sensor by the services provided by the sensor, their physical locations, and their functionality etc.
Rank the document by checking the quality of content, no of inbound and outbound links etc.

Figure 2: Working of IoT and Traditional Search Engine

The traditional search engines index the documents and rank the documents to make searching more efficient. The documents are indexed based on terms (such as database, transition), stop words (such as that, this, the, etc.), number of occurrences of a word in a document (such as how many times the word occurs in a document [5]. The IoT/WoT search engine performs the indexing based on different parameters [6] such as keywords-based indexing and spatial-based indexing. Both elements of search engines, indexing and ranking are controlled by crawling. Crawling is the crucial step responsible for checking the updates in the documents [7]. In IoT and WoT search engines, the crawling step is performed at indexing [6], [8]. Instead of visiting each Thing, it can visit the indexing place and check whether there is any change in the functionality, services, location, etc., of Things or not.

The traditional search engines perform ranking to make more relevant results appear at the top of the page. These search engines rank the document according to the quality of content, number of new words, number of inbound links (i.e., how many websites are linking your website), and the number of outbound links (your website links how many websites). The IoT/WoT search engine performs the ranking based on the services provided by the sensors, their physical locations and by their functionality etc.

Traditional search engines are best for searching documents but these search engines cannot find physical Things. For finding the physical Things, traditional search engines cannot use indexing and ranking mechanisms which is generally done by IOT/WOT search engines. Many authors have performed keyword-based and spatial-based indexing in searching the Things [6], [8]. As many physical Things provide the same functionalities and services and sometimes their location is also the same or nearby, in such cases, it becomes challenging to handle the ranking of such physical devices.

At present several different features are available for searching the Things that are essential for the proficient working of the search engines. One or more of these features is utilized by the search engines upon which the resulting performance of that search engine is based. These features are described below.

1.1 Discovery of Resource

The Discovery of resources is a step of finding the Things that refer to a need to describe the resources as a description and then discover the resources using some protocols like HTTP. In the web of Things, the resources are discovered in a centralized and decentralized manner [4]. Resources are discovered centralized using some APIs (sensor web APIs, Thing Directory, W3C Generic Sensor API). Resources are discovered decentralized by gathering the data of each Thing.

There are many advantages to discovering resources by using a centralized approach compared to a decentralized approach, such as efficiency, automation, massive data, and integration.

In WoT, the better approach to discover the resource is centralized because with the help of a centralized approach, a vast amount of data is gathered, and updated data is easily reached to the user.

1.2 Crawling

Crawling is a process in which a search engine checks out to look for any change in the content and find new content. In traditional search engines, crawling is done by sending a crawler to different websites to find any change in the website. In WoT, search engine crawling is done manually and automatically [12]. In manual crawling, the search engine visits the individual Thing site to check for the change in the Things information. Many search engines perform crawling fortnight. As a result, the user does not get the exact output in real-time. In automatic crawling, the search engine automatically detects the changes in Things data by using some autonomous agents or by registering the Things at the place of indexing. Automatic crawling performs well if the discovery of resources is centralized [12]. If the resource is discovered using a centralized approach, the search engine gets the updated content in real-time.

In WoT, the better approach to crawl is automatic because it is impossible to visit millions of devices individually.

1.3 Indexing

Indexing is how search engines store and organize the data found during the crawling process. The traditional search engines perform the indexing based on terms, stop

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication Online Open Access
Vol: 41 Issue: 11-2022
DOI 10.17605/OSF.IO/DA27F

words, and no occurrences of a word in a document. In WoT search engines, indexing is based on keywords (id, description, name of Thing) and spatial keywords (relative location and coordinates) [6]. In WoT, search engine indexing is centralized and distributed [27]. Centralized indexing means that the data found by the crawler are stored and organized in one place. Decentralized means that the data found by the crawler are stored and organized in different geographical places according to their town wise.

1.4 Ranking

The ranking is a process in which the search engine provides the output against the user's query at the top priority. In a traditional search engine, ranking is performed by checking the quality of content, no inbound and outbound links [5]. In WoT search engines, ranking of Things is performed according to the services provided by the Thing, their physical location, and their functionality.

1.5 Supporting Dynamic Nature

In traditional search engines, all documents are static, as documents are continuously changing their contents, not their physical locations. In WoT, all devices are dynamic as they are changing their data and physical locations. In WoT, the search engine is performing efficiently if search engine supports the Dynamic nature of devices. The proposed search engine supports the dynamic nature of devices by using the Historical Location attribute [6].

1.6 Query Type

The traditional Search engine performs different types of the query such as textual queries, numeric queries, and location-based queries. The WoT Search engine performs the two types of queries: the first is a keywords-based query, and the other is a spatial-based query [6]. The keywords-based query can be searched against the id, name, description, type, URL, and version. The keywords-based query works for individual keywords and the combination of multiple keywords. By combining the keywords, the user gets an accurate result. If the user wants to search the Things according to the location of a Thing, then the user uses the spatial-based query. The spatial-based query can be searched against the Relative location and coordinates. In the case of Relative location, the user searches the Thing of a specific district name or a specific street, country, and city. In the case of Coordinates, a user searches the Thing at its exact current location by using Latitude and Longitude.

The structure of the paper is as follows: Section 2 describes the state-of-the-art for ranking of Things and existing IoT search engines. Section 3 presents the research methodology of our study. Section 4 represents the proposed algorithm and section 5 entails the description of ranking parameters. Section 6 describes the evaluation and discussion. Section 7 describes the conclusion and future work.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication Online Open Access
Vol: 41 Issue: 11-2022
DOI 10.17605/OSF.IO/DA27F

2. State of the Art

In previous studies, the researchers have performed ranking based on services [9], ranking the Things based on previous searches [10], location (coordinates) [11] and features [12]. However, they do not include other ranking parameters such as historical location, functionalities, type, description, or combination. In this paper, we have worked on the hybrid ranking mechanism for searching the Things. The traditional search engines like Google, yahoo, ask, Bing, etc., use different ranking algorithms such as PageRank [13], Stochastic Approach for Link-Structure Analysis (SALSA) [14], DistanceRank and Hyperlink Induced Topic Search (HITS) [15].

Google uses the PageRank algorithm for ranking the documents. The PageRank algorithm ranks the documents according to the quality of content in the document, the number of inbound and outbound links and the document's structure. HITS ranks the document based on the number of inbound and outbound links. DistanceRank algorithm ranks the document depending on the distance between two pages. SALSA algorithm combines the working of PageRank and HITS algorithm. The techniques described above are not suitable for IoT or WoT devices because of several factors such as the nature of data in IoT and dynamic physical locations of IoT devices etc.

A previous study developed an offline data-driven algorithm which performs the ranking on the structured data for traditional search engines [16]. The proposed algorithm gets the web pages which have variety of keywords. It means that the proposed algorithm ranks the pages according to various keywords. Page gets the top priority which has more variety of keywords. Such a type of algorithm is not suitable for IoT data because IoT data does not have a specific structure and IoT devices are dynamic in nature.

The web page search is done by making the clusters of the web pages by Kalashnikov et al. [17]. The authors developed a system based on the Graph-based cluster and Graph-based Disambiguation algorithm. The algorithm is used to perform disambiguation among the web pages which have the same name. The graph-based cluster algorithm provides a collection of Web page clusters to individual users who search the Web page. Each web page cluster is the collection of all Web pages relevant to the user who is searching the content. The cluster also allows users to rank the web pages based on their interests. By doing this, the Web pages ignored by the traditional search engines are available to the users.

According to previous research, information should be retrieved according to the top priority depending on the user's demand [18], [19]. It means ranking the document based on the context. Similarly, other researchers suggested ranking the documents depending on the quality of the topic [20], [21]. It means that the search engine should use the PageRank algorithm to rank the documents. Haveliwala et al., proposed that the ranking of documents depends upon the precision of web crawlers [21]. It means how accurately the web crawlers are crawling the web page to identify the changes in the documents. At the same time, a study by Hoopman [22] suggested another solution to rank the documents using the re-rank technique. The Re-rank technique is a technique to get the results from the server and display them on the client-side. Then according to the client's demand, re-rank the results again, i.e., it works according to the user's choice.

A survey study on collecting data for IoT search engines described that data is collected using Nmap, Advanced port scanner, Angry IP scanner, Portscan, and stuff [23]. Network map (Nmap) is an open-source scanner that uses APIs to integrate the data of internet-connected objects into the user system. An advanced port scanner is user-friendly to connect your system to internet-connected objects by quickly finding the open ports. An angry IP scanner is a tool which allows the user to access the Things by providing the IP address of a Thing. Portscan and stuff is a networking tool which allows the user to find the Thing by providing the MAC address of a Thing. According to the research, the angry Ip scanner and Nmap are the best tools [23]. The research proposed a model that performs the selection and ranking of fog computing based on IoT. The proposed model is used to monitor the health system [24]. Their model uses the analytical network process (ANP) to select and rank the things in the health system.

All the techniques and algorithms discussed above work for the static searching concerning the location and based on a limited number of documents. These techniques and algorithms are not workable for IoT or WoT devices as these devices are dynamic and heterogeneous in nature. Moreover, such devices do not have a specific structure. The Things are also huge in number. For the reasons mentioned above, Traditional Search Engines are not suitable for finding physical devices.

This section aims to highlight the research work published on the ranking of Things in IOT/WOT environment. This section is further divided into two parts which are: Ranking of Things and existing search engines.

2.1 Ranking of Things

Ranking of Things is one of the most challenging tasks in searching the Web of Things due to the dynamic nature of physical Things. The ranking is the process of displaying the output against the user's query at the top priority. For instance, if the user searches the coffee machine, the search engine can display millions of coffee machines. Which coffee machine has to be displayed at the top of the result depends on the ranking mechanism of the search engine. For instance, search results can be optimized by displaying the coffee machine which is close to the user by the ranking algorithm.

Guinard et al. [12] perform the ranking of Things based on the network latency. Their proposed search engine arranges the list of Sensors according to the Network Latency. The proposed search engine gets the user's query and finds the sensors against the query. When the sensors are found, their search engine checks low latency in the network. Network Latency is the time taken to move data from source to destination. The sensor that has low Network Latency performs better. In this way, the user will get the Sensors with low network latency at the top.

CASSARAM stands for context-aware sensor search, selection and ranking model developed by Perera et al. [9]. This model aims to search the Things more efficiently by indexing and ranking the Things. It performs the ranking of Things depending on the user priorities. The author developed a CASSARA tool to identify the priorities of the user. CASSARA tool gets the characteristics of Things from users in a user query interface and then ranks the results according to the characteristics demanded by the user. It can work for a small number of Things and sometimes the result is not very beneficial. For instance, if a user searches for a coffee machine and enters its characteristics in a search query, the user can go to that coffee shop and drink the coffee of his/her taste, while that user's location is in California. But the search engine finds the coffee machine which fulfills the user requirements, which is in New York and displays the results of such coffee machine shops at the top of the search result. So, in this case, the result is not very beneficial for the user.

A ranking mechanism to rank the Things based on previous searches was proposed in [10]. Some researchers rank the Things according to the coordinates of Things [11]. One research ranked the Things on top, which changes their positions less frequently [25]. Researchers describe the ranking mechanism of "Thingful.net" to rank the Things based on the relative location of the Things [4]. In this paper, we have proposed a Ranking Mechanism for Web of Things Search Engines named RMoWoTSE, which ranks the indexed Things efficiently by using a combination of parameters and individual parameters.

2.2 Existing Search Engines

We have analyzed the existing search engines on the basis of some important features like discovery of resources, crawling, indexing, ranking, support dynamic nature, query type and results of the query. These features are discussed in section 1. The search engines analyzed in this paper are described as follows.

Dyser is a search engine proposed by Ostermaier et al. [10]. Dyser performs the searching based on “name” keyword. In Dyser discovery of resource is decentralized and it performs manual crawling and uses centralized indexing. Dyser ranks the Things according to the few previous searches. For instance, previous searches include five searches on gas sensor, three searches on temperature sensor and two searches on proximity sensor. Dyser ranks the results according to the highest number of previous searches on each individual sensor. Dyser does not support dynamic nature of devices and performs the Keyword based query (name). For example, Dyser works for the bicycle rental station. It finds the currently available bikes at the bicycle rental station. Dyser gives the result which exactly matches the query. If results do not match with the query, it does not return the result. For instance, if a user writes a query “bicycle station in Lahore” Dyser returns the results if this query exactly matches with the Resource gathered by Dyser. If a single word does not match with the Resource gathered by Dyser it does not return the results.

DiscoIoT is a search engine proposed by Mayer and Guinard [28]. DiscoIoT works against the “URL” of the Thing. In DiscoIoT discovery of resource is centralized. It uses the sensorThings API of temperature sensor to gather data as a resource. It performs the manual crawling and does not provide any indexing mechanism. It does not provide any ranking mechanism as it searches the specific Thing by using its URL. It displays the only one result at a time that’s why it cannot provide any ranking mechanism. It does not support dynamic nature of devices. DisoIoT performs the Keyword based query (URL). For instance, if a user enters the keywords or any description other than the URL, it cannot work. DiscoIoT returns the only one result at a time.

Thingful is a search engine described by Fathy et al. [4]. Thingful works against the “Description” and “Relative Location” of a Thing. In Thingful discovery of resource is decentralized. It works on the manual crawling. Thingful provide the Distributed Indexing mechanism and performs the ranking of a result according to the physical location (Relative Location) of a Thing. It does not support dynamic nature of devices. It performs the Keyword (Description) and Spatial (Relative Location) based query. Thingful works against the description and relative location of a Thing. For instance, if user enter a query “Temperature Sensor in Lahore”. Then, the Thingful will display all the results to the user which include the word “Temperature” or “Lahore” in database. In this case, many of the results are irrelevant i.e. Thingful will also display the result of “Proximity sensor in Lahore” because “Proximity sensor in Lahore” contains the word “Lahore”.

MAX is a search engine proposed by Yap et al. [29]. It works against the specific description of a Thing. In MAX discovery of resource is decentralized. MAX gathers the data of Tag based Energy and Environment Things as a resource. It performs the manual crawling and there is no proper indexing mechanism. In MAX, there is no ranking mechanism as it searches the data against the location entered by the user and it displays the only single output at a time. It does not support dynamic nature of devices. It performs the Keyword based query (Specific Description of Things). MAX works against the specific description of Thing. For instance, temperature sensors are of four types: one is thermocouples, RTDs, thermistors, and semiconductor based integrated circuit. MAX work against the query which contains the word thermocouples or RTDs or thermistors etc. It cannot work against the query which contains the word “Temperature”. It will display the single result at a time.

Sense Web is a search engine proposed by Grosky et al. [11]. It works against the coordinates of a Thing i.e. Latitude and Longitude. In Sense Web discovery of resource is centralized. It collects the data of Soil Sensor from JHU, Parking Space data from US city, Traffic conditions from Washington State Department of Transportation. It performs manual crawling and centralized indexing. It ranks the results according to the physical location (coordinates) of Things. It does not support dynamic nature of devices. Sense Web performs the spatial based query (Coordinates). It works against the Latitude and Longitude of a Thing. For instance, if user enters a query to find the parking space available in KFC hotel at 40.7128, 74.0060 (New York). Then, the system displays all

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication Online Open Access
Vol: 41 Issue: 11-2022
DOI 10.17605/OSF.IO/DA27F

the results of parking spaces of KFC in New York. The system displays the result of parking space at top which is nearest to the location of the user location.

Wang et al. proposed a search engine named Snoogle [25]. It works against the description of Things without a specific location. In Snoogle, discovery of resource is centralized and it performs crawling manually. Indexing is distributed. It ranks the data of Things according to the specific description entered by the user in the query. It displays the data on top which is more stable. For instance, if the user enters a query "Coffee Machine in California" and the database of Snoogle contains 20 results which have "Coffee Machine in California", Snoogle displays the result at top which do not change its position frequently. In other words, it displays the result at top which has less frequency of changing position. It supports the dynamic nature of devices periodically not in real time. It works against the Keyword based query (Description). It displays multiple results at a time. It works against the description of the Thing. If a user enters "Toaster Machines in Hotels of California". Then, Snoogle displays the results of Toaster Machine at top which do not change its position frequently.

3. Research Methodology

In this paper, we have followed the approach we have proposed in our paper [6] as illustrated in Fig. 3. The proposed approach consists of four layers. The first layer is the admin layer, the second layer is the indexing layer, the third layer is the ranking layer and the last layer is the user layer. In this approach, we have used the Solr tool to save data and our model uses SPARQL queries to give the information requested by users. The brief detail of these layers is as follows:

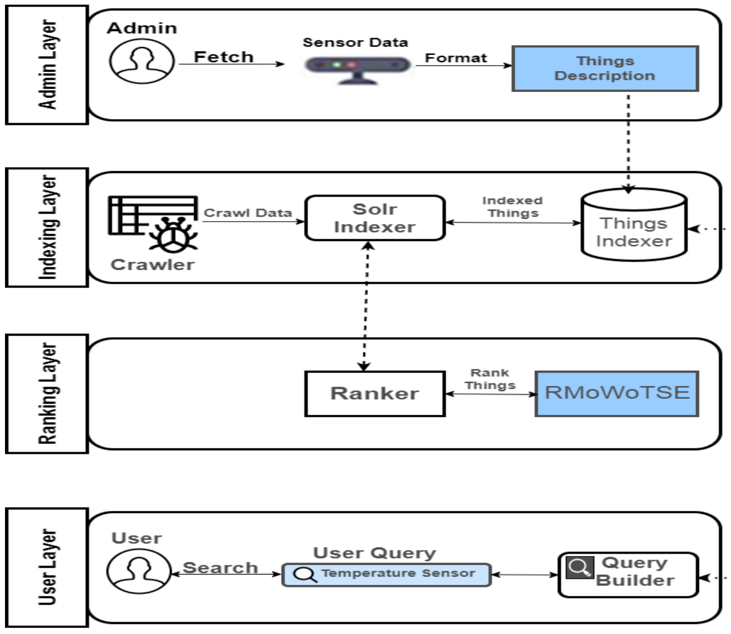

Figure 3: Working Model of WoT Search Engine

3.1 Admin Layer

The admin layer consists of two components: the first component is sensor data and the second is Things Description. The admin fetch the sensor data from W3C API Things Directory and then pass the sensor data to Things Description. The proposed model uses the "RESTFUL APIs" for the Sensor Web APIs. The RESTFUL API consists of HTTP and identifiers (URL). With the assistance of the get function, the proposed model retrieves the present data of the sensor. With the assistance of the post function, the proposed model gets the changes in data.

Once the changes in data are retrieved, the proposed model uses the put function to place the changes in data. For example, consider the coffee machine connected to the sensor. With the assistance of the get function, the present quantity of coffee within the machine or the other state of that machine is retrieved. The post function is employed to see if any change has occurred in the state or quantity of the coffee machine. If there is any change observed in the state of the coffee machine, then the information is updated again with the assistance of the put function. The information is fetched and then saved into the items indexer by passing through the items description. The fetched data is then formatted W3C JSON-LD format [6]. Once the item data is formatted, it is passed to the items indexer, the first component of the indexing layer. The aim of passing the sensor data to the items indexer is to save the sensor data.

3.2 Indexing Layer

The indexing layer consists of three components: the first one is the Things indexer, the second one is the Solr sensor, and the last component is the crawler. Once the data is formatted, it is passed to the Things indexer to validate the schema of Things in PHP with the schema of Things in Solr. If the schema matches, it is indexed in the Solr indexer. If the schema does not match, it is not passed on to the Solr indexer. In this layer, crawling is also performed. The crawler is connected to the Solr indexer to check for any changes in the sensor data. If it detects a change in the sensor data, it updates the sensor data at each place where it is indexed [6].

3.3 Ranking Layer

The ranking layer consists of two components: the first is the ranker and the second is the Ranking Mechanism of the Web of Things Search Engine (RMoWoTSE). The ranker is connected with the Solr indexer as well as the RMoWoTSE. The ranker gets the data of Things from the Solr indexer according to the user's query and ranks the Things according to RMoWoTSE. RMoWoTSE performs the ranking of sensor data, which we have discussed in the ranking parameters section 3.3.

3.4 User Layer

The last layer is the user layer which is responsible for searching and displaying the Things against the query entered by the user on the user interface. The proposed user layer consists of User Interface and query builder defined in [6].

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication Online Open Access
Vol: 41 Issue: 11-2022
DOI 10.17605/OSF.IO/DA27F

4. Algorithm

The algorithm of the proposed search engine is described below:

```
1.  Schema→Schema defined in Solr
2.  Schema→Schema defined in Solr
3.  For→Format according to JSON-LD, XML, JSON
4.  D→Description of things
5.  If (Schema== D) then I→Indexed data
    Else
    Do→Data is not indexed
    End If
6.  Q→Query of user
7.  If (I==Q) then
    R→Rank the data according to Ranking Mechanism
    {
    T→Type of Sensors, S→Services of Sensors, L→Location of Sensors, Fe→Feature
    of Sensors, N→Network Delay, B→Bandwidth, Ba→Battery Power, Av→Availability,
    R→Reliability.  D→Description of Sensor, H→Historical Background of Sensors,
    C→Combination of Sensors
8.  Q→Query of User
9.  If (I==Q) then
    F→Filter the query for ranking the result
    If (F==T || F==S || F===L || F==D|| F==H)
    Display the result.
    else If (F===Fe)
    If (F==T || F==S || F===L || F=== N || F==B || F===Ba|| F==Av|| F==R|| F==D||
    F==H)
    }
    Display the result.
10. Exit
```

In step 1, we define the schema in Solr which indicates what type of data we will store and manage in Solr. Without defining schema in Solr, the data cannot be stored, managed, and retrieved in Solr. If the Search engine does not follow the

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication Online Open Access
Vol: 41 Issue: 11-2022
DOI 10.17605/OSF.IO/DA27F

schema, then the search engine cannot store, manage and retrieve data in Solr. In step 2, proposed search engine (PSE) gets the data of Sensors/Things. The PSE retrieves the data from different sources, so the data format is also different. It contains much-unneeded information. In step 3, PSE formats the data according to JSON-LD, JSON and XML formats. After we get the formatted data named "Description of Things", in step 4, we match the description of things with the schema defined in Solr. In step 5, if it matches, we store and manage data in Solar. In step 6, users enter the query in the query interface. In step 7, the query is directed towards the Solr. If the query matches the data, then PSE checks what type of query is entered by the user. If the query is based on the type of sensor, then PSE performs the ranking based on the type of sensor. If the query is based on Services, then PSE performs the ranking based on services. If the query is based on Features, then PSE performs the ranking based on bandwidth, network delay, battery power, availability of sensor and reliability. Once the data is ranked, PSE displays the results to the user.

5. Ranking Parameters

In this paper, we have discussed two kinds of ranking mechanisms. The first is the ranking of textual data and the second is the ranking of sensor data. The traditional search engine uses textual data for searching and the Web of Things search engine uses sensor data. The parameters required for the ranking of textual data are quality of content used in the document, inbound or outbound links, quality and design of the webpage and engagement metrics.

The brief description of the ranking parameters of textual data is as follows:

5.1 Quality of Content

Quality of Content includes the answer to the question asked by the user in the search query. The content is considered more valuable and informative if the content answers the user's question. It is a crucial parameter for ranking the website when searching for textual data.

5.2 Inbound or Outbound Links

Inbound links include the links to other websites on your webpage. Outbound links include how many other websites are linked to your webpage. If many different websites are linked to your webpage, then it means that your webpage has good quality.

5.3 Quality and Design of Webpage

The layout of your web page plays a vital role in ranking textual data websites. If your web page has a pleasant or sophisticated and well-managed design, then your website has a better ranking.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication Online Open Access
Vol: 41 Issue: 11-2022
DOI 10.17605/OSF.IO/DA27F

5.4 Engagement Metrics

Engagement metrics include how many people click on your webpage. How much time the user spends on your webpage and how much time your webpage sessions maintain. All these elements are included in engagement metrics. The web page having a high number of clicks, maximum time spent by users and maximum time of maintaining sessions, has a high ranking compared to the websites with fewer engagement metrics.

The ranking model discussed by Perera et al. [9] performs the ranking of sensors based on the index. The proposed ranking model displays the results, which are inefficient because it selects n number of sensors from the query requested by the user. For instance, if a user enters a query to find the temperature sensor in California, then the proposed system shows the result of all temperature sensors in California. In order to improve the efficiency, the authors in this paper included different parameters for ranking, which are shown in Fig. 4.

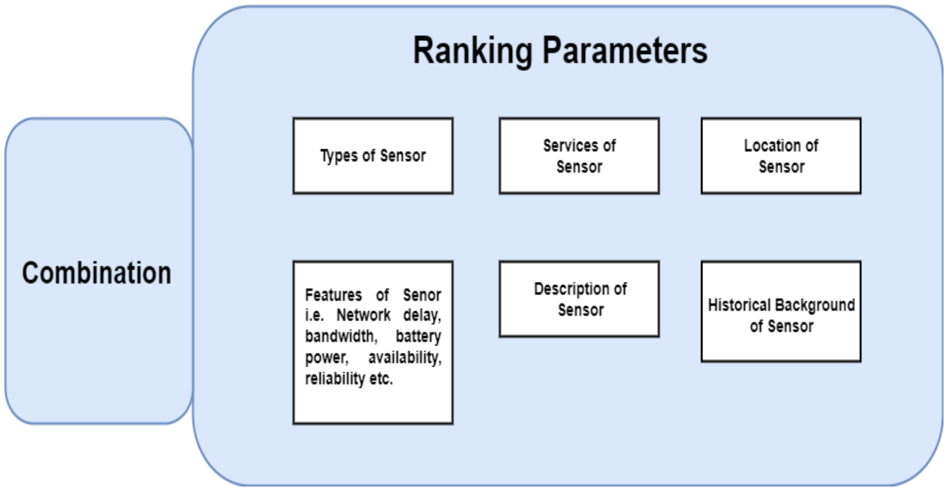
Ranking Parameters
Combination
Types of Sensor
Services of Sensor
Location of Sensor
Features of Senor i.e. Network delay, bandwidth, battery power, availability, reliability etc.
Description of Sensor
Historical Background of Sensor

Figure 4: Ranking Parameters to rank Things

A brief description of ranking parameters of sensor data is as follows:

5.5 Types of Sensor

There are different types of sensors, such as gas sensors, temperature sensors, proximity sensors etc. The proposed ranking model ranks the sensors based on the type of sensor entered by the user in the query instead of displaying all types of sensors.

5.6 Services/ Functionality of Sensor

There are different services/functionality of sensors, i.e., temperature sensors have different services and functions, such as temperature sensors being used to measure the thermal characteristics of liquids, solids, and gases [26]. The proposed ranking model ranks the sensors based on the services/ functionality to display the exact list of sensors.

5.7 Location of Sensor

The location of the sensor depends on the absolute location and relative location. The relative location includes the city, country, postal code, etc. Absolute location includes latitude and longitude. The proposed ranking model ranks the sensors based on the absolute and relative location of the sensors. For instance, if a user searches the gas sensor in California, the proposed model displays the result of sensors in California. If a user searches the gas sensor in Specific streets and the specific town of California, the model displays the result of sensors in specific streets and towns queried by the user instead of displaying the sensors all over California.

5.8 Features of Sensor

There are different features of sensors, such as network delay, bandwidth, and battery power. The ranking algorithm displays the sensor which has better bandwidth and high battery power instead of displaying the sensors which have a low battery.

5.9 Description of Sensor

It describes the brief description of sensors, i.e., what the sensor is. The ranking algorithm also ranks the sensor based on the description of the sensor.

5.10 Historical Background of Sensor

The historical background of the sensor represents how often the sensor changes its location. The sensor that changes its location more frequently means that such a sensor is not reliable and may not be available. Reliable sensors mean that the sensor performance is excellent and consistent. By focusing on the reliability and availability of sensors, the proposed ranking model ranks the sensors based on the history of their changing locations and displays the result of sensors that change locations less frequently.

6. Evaluation

This section explains the study questions, data set, metrics, and evaluation procedure for the suggested approach.

6.1 Research Questions

The proposed strategy is assessed by examining the following questions of interest:

Q1: Investigation and comparison of proposed approach with existing ranking approaches for the web of things.

Q2: Which ranking approach optimizes search results in terms of performance and accuracy?

Q3: Comparison of the proposed search engine with the existing search engines.

The first research question investigates the previous approaches which have performed ranking, as mentioned in the state-of-the-art section.

The second research question analyzes the evaluation of different proposed ranking parameters used by authors in order to figure out which ranking parameters, when used with the proposed approach and gives the most accurate results.

The third research question compares the proposed search engine's working in terms of indexing, crawling, ranking, queries and what results it provides against the queries. All of these working elements can be used to compare your search engine's performance with that of existing search engines.

6.2 Dataset

The dataset is compiled using Energy Sensor data and Environment sensor data from Sensor Web API's, Home sensors data from W3C API's and Bicycle rental station Sensor API's.

6.3 Metrics

The accuracy, precision, recall, and f-measure, which are the most common and well-known metrics, are used to measure the performance of the proposed method. The equations for these metrics are shown below [31].

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

The authors calculated the accuracy by using the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values. The output consists of positive and negative elements, which depend on TP, TN, FP, and FN.

1. True Positive: The output is true positive if the search engine displays the results exactly according to the query.

2. True Negative: The output is true negative if the search engine displays the results other than the exact query.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication Online Open Access
Vol: 41 Issue: 11-2022
DOI 10.17605/OSF.IO/DA27F

3. False Positive: The output is false positive if the search engine incorrectly identifies the accurate results.

4. False Negative: The output is false negative if the search engine incorrectly identifies the inaccurate results.

6.4 Process

We have uploaded the description of 750 sensors in Windows 7, using a 32-bit operating system. After 750 requests, the system was not scalable anymore. The browser was unable to process requests, and requests were timed out. Results were taken with respect to accuracy, precision, recall, and f-measure. The results of TP, TN, FP, and FN based on sensor types, services, location, features, description, historical background, and combination of all of these is shown in Table 1.

Table 1: Evaluation of TP, TN, FP, and FN Values of different ranking parameters

Ranking Parameters	**TP**	**TN**	**FP**	**FN**
Types of Sensors	The system displays the result of 307 sensors from 480 whose type is temperature.	System displays the result of 184 sensors whose type is not temperature out of 270 remaining sensors.	System does not display the results of 86 sensors whose type is not temperature out of remaining 270 sensors.	System does not display the results of 173 sensors from 480 sensors whose type is temperature.
Services/ Functionality of Sensors	The system displays the result of 93 sensors from 120 temperature sensors whose thermal characteristics are gases.	System does not display the result of 214 sensors whose thermal characteristics are not gases.	System displays 16 sensors whose thermal characteristics are not gases.	System does not display the 27 sensors whose thermal characteristics are gases.
Location of Sensors	Out of 750 sensors, the system displays the result of 592 sensors accurately by using spatial approach.	System does not display the results of 51 sensors out of remaining 158 sensors who do not match with the query.	System displays the results of 56 sensors, which do not match with the query out of remaining 158 sensors.	System does not display the results of 51 sensors, which are matched with the query out of remaining 158 sensors.
Features of Sensors	Out of 750 sensors, the system displays the result of 492 sensors, which have better bandwidth and high battery power.	System does not display the results of 133 sensors out of remaining 258 sensors, which do not have better bandwidth and battery power.	System displays the results of 46 sensors, which has low battery power and low bandwidth.	System does not display the results of 79 sensors, which have better bandwidth and battery power.

6.5 Results and Discussion

Research Question 1: Investigation and comparison of proposed approach with existing ranking approaches for the web of things.

To investigate research question 1, we have explored the existing ranking approaches. In this paper, we have proposed a ranking model, "RMoWoTSE," for searching the things. With this model, the search engine performs good because in RMoWoTSE searching is done with more ranking parameters as compared to [4], [9], [10], [11], [12], and [25]. Exact performance comparison was not possible as we were unable to locate empirical or statistical results. We compared our findings with the theoretical data and approaches used which are available in prior literature. These existing studies have used a single parameter for ranking the search results. One of the studies uses the service parameter for ranking the Thing [9] and another study uses network latency for ranking the Things [12]. In addition to this, a ranking mechanism to rank the Things based on previous searches was proposed in [10]. Some researchers ranked the Things according to their coordinates [11]. One study ranked the Things that change their position the least [25]. Researchers describe the ranking mechanism of "Thingful.net" to rank the Things based on their relative location [4]. We have used six parameters to rank the search results and to make the search process faster and more efficient.

Research Question 2: Which ranking approach optimizes search results in terms of performance and accuracy?

To investigate research question 2, we compared the performance of the proposed approach with different ranking parameters. As statistical/empirical results were not available we performed several experiments to analyze how better searching can be performed. We have performed a number of experiments to evaluate our algorithm, such as the search engine returning accurate results when a user searches for Things by Type query. Search engine accuracy needs to be checked since there can be cases where error-free results are not returned. For example, the search engine results might cite "thermal temperature" when the user requested a "dock sensor." It is also possible that when the user searches for "proximity sensors," the search engine provides only ten results, even though 200 proximity sensors are available. The search engine is therefore performing an inaccurate search. We conducted many experiments to address such misleading scenarios and evaluate our algorithm's performance. When ranking is improved, searching is also improved. When users conduct a search for Things, their primary concern is the accuracy of the results. In this work, we have developed a method for dynamically searching for objects in the WoT environment. To test our search engine, we conducted an analysis of its accuracy. We also looked at standard metrics like precision, recall, and f-measure to evaluate how well our new ranking system work.

Table 2 and Fig. 5 show the accuracy of results with TP, TN, FP, and FN. Table 3 and Fig. 6 describe the results according to precision, recall, and f-measure. From Table 3, we have obtained the best results with a combined approach, which has 85% accuracy and we obtained the worst results for ranking the "Things" using the type approach, which has 47% accuracy. Based on our results, we recommend using the combined approach for ranking.

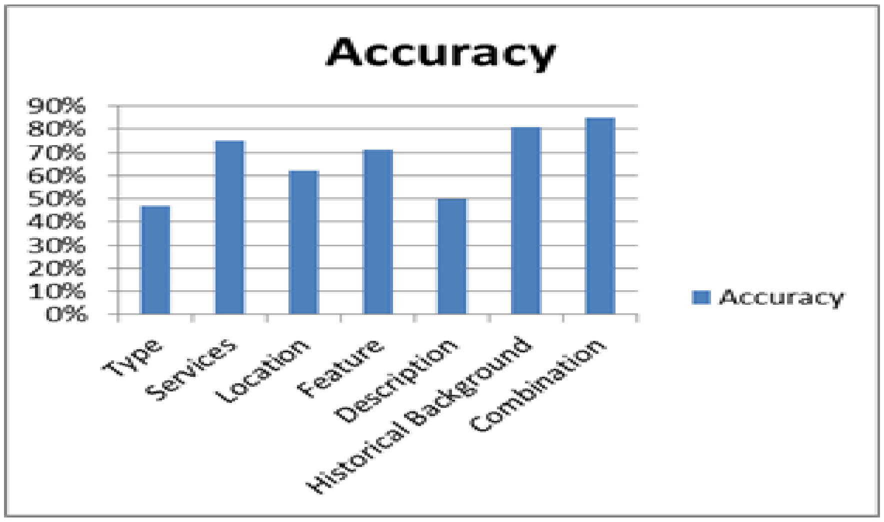

Figure 5: Accuracy of Results with TP, FP, FN, and TN

Table 2: Accuracy of Results with TP, FP, FN, TN

Case	**Positive**		**Negative**		**Accuracy**
	TP	**FP**	**TN**	**FN**	
type	307	184	173	86	47%
services	93	16	27	214	75%
location	592	56	51	51	62%
feature	492	46	79	133	71%
description	435	149	99	67	50%
historical background	590	42	38	80	81%
combination	698	22	10	20	85%

Table 3: Results According To Precision, Recall, and F-Measure

Case	Precision	Recall	F-Measure
Type	0.625	0.639	0.6323
Services	0.85	0.77	0.81
Location	0.913	0.920	0.91
Feature	0.914	0.861	0.88
Description	0.744	0.814	0.778
Historical background	0.933	0.939	0.936
Combination	0.969	0.985	0.977

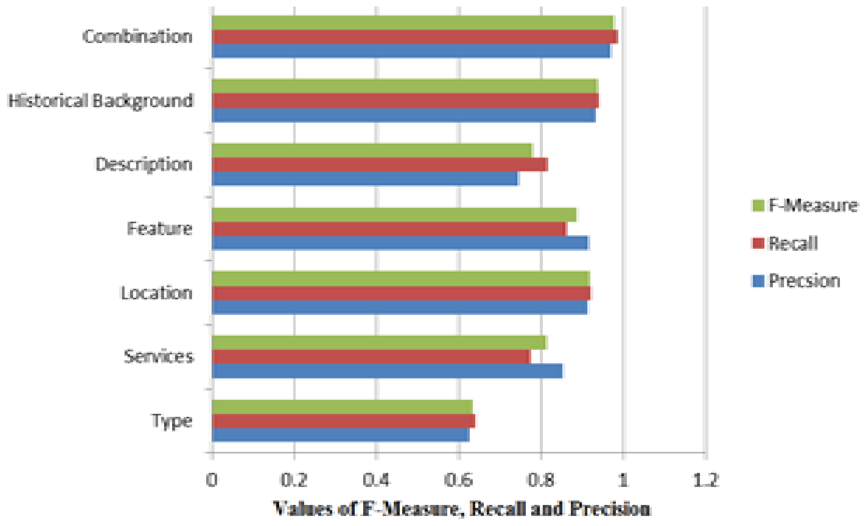

Figure 6: Results according to Precision, Recall and F-Measure

Research Question 3: Comparison of the proposed search engine with the existing search engines

To investigate research question 3, we compared the proposed and existing search engines based on different features such as discovery of resources, crawling, indexing, ranking, support dynamic nature, query type and results of the query [4]. The search engine cannot function properly without these elements.

Proposed search engine works against the Keywords (name, description, URL, version, type, and id) and Spatial (Relative Location [Postal Code, District name, Street Name, City and Country], Coordinates [Latitude, Longitude]).

In proposed search engine discovery of resource is centralized. Proposed Search Engine performs the automatic crawling and the centralized indexing with ranking of Things according to the types of sensors, services of sensors, location of sensors, features of sensors, description of sensors, historical background of sensors and combination of all of these. It supports the dynamic nature of devices. Proposed Search Engine Performs the Keyword (name, description, id, version, type and URL) and Spatial (Relative location [Postal code, district name, street name, city, country], Coordinates [Latitude, Longitude]) based Query. It works against the energy, environment, home and bicycle rental stations. For instance,

Query 1: “Temperature sensor”

Query 2: “Dock sensor in street no 2 in New York”.

Query 3: “Bicycle available in bicycle rental station”.

Query 4: “Bicycle available in Toronto bicycle rental station 43.6532, 79.3832”.

Query 5: “Water sensor in district Kasur city Kasur”.

ISSN：1671-5497
E-Publication Online Open Access
Vol: 41 Issue: 11-2022
DOI 10.17605/OSF.IO/DA27F

Proposed search engine displays the single result if user enters the query of specific location i.e. if user enters the query “Dock sensor at 53.4808, 2.2426” then it will display the exact Thing. It will display the multiple results if user enter “Acceleration sensor in Manchester”. Then, it will display the entire Acceleration sensors available in Manchester. For instance, if there 32 Acceleration Sensors available in Manchester, then the proposed search engine displays the Acceleration Sensor at top which is more reliable, which has better bandwidth, whose battery power is high and which provides the maximum number of services etc.

We have used Tableau a statistical tool [32] to analyze the performance of proposed search engine and compare with the existing search engines. To do this we assign the values of 0-2 to features of search engines. In discovery of resource a value of 0 shows decentralized process while 1 refers to centralized process. If the search engine performs manual crawling it is assigned 0; if the search engine performs automatic crawling it is assigned a value of 1. We assign 0 if the search engine performs distributed indexing and 1 if search engine performs centralized indexing. We assign 1 if search engine performs ranking and 0 if search engine does not perform ranking. We give 1 if search engine support dynamic nature and 0 if search engine does not support. A value of 2 is assigned if the search engine accepts both types of query i.e. keyword and spatial, while 1 is given when the search engine accepts single type of query. After analyzing this on Tableau, proposed engine performs better in comparison with other search engines as shown in Fig. 7**.**

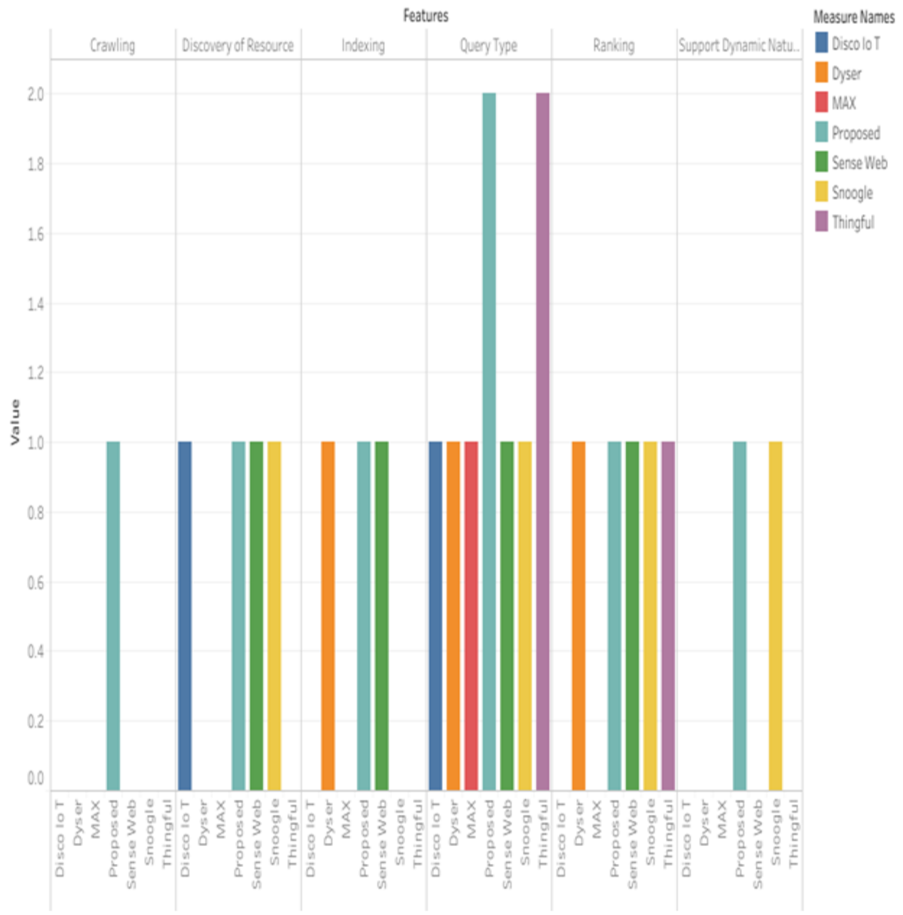

Figure 7: Comparative analysis of different search engines on Tableau

7. Conclusion

Due to a rapid increase in the usage of Sensors in objects in real-world life, the demand for search engines for searching the Things has also increased. The primary purpose of a search engine is to provide the sensors to the user so that the user gets the services of the sensor he wants. Search engines are of two types, one is used to search for general data, and the other is used to search for sensor data. There are two crucial components of both types of search engines: one is indexing, and the other is ranking. Without indexing, the search engine is not capable of storing data efficiently. And, without ranking, the search engine will not be able to display the most relevant results at the top.

In this paper, we have focused on ranking the search data of sensor. We have presented a ranking mechanism named "RMoWoTSE" in this study. We have employed six ranking parameters to get the best results as prior studies in this domain only provide theoretical details about working but no empirical data has been discussed.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN：1671-5497
E-Publication Online Open Access
Vol: 41 Issue: 11-2022
DOI 10.17605/OSF.IO/DA27F

Thus, when the mentioned findings from previous studies given in Table 4 and our results in Table 2 and Table 3 are analyzed, it is revealed that our proposed search engine performs better. We intend to continue working on correlation-based search techniques in the future.

Reference:

1. Kumar, S., Tiwari, P., & Zymbler, M. (2019). Internet of Things is a revolutionary approach for future technology enhancement: a review. J Big Data, 6(111).
2. McCahill, M. P., & Anklesaria, F. X. (1996). Evolution of internet Gopher. In The Journal of Universal Computer Science (pp. 235-246), Springer, Berlin, Heidelberg.
3. Faheem, M. R., Anees, T., & Hussain, M. (2019). The web of things: findability taxonomy and challenges. IEEE Access, 7, 185028-185041.
4. Fathy, Y., Barnaghi, P., & Tafazolli, R. (2018). Large-scale indexing, discovery, and ranking for the Internet of Things (IoT). ACM Computing Surveys (CSUR), 51(2), 1-53.
5. Balabantaray, R. K. (2017). Evaluation of web search engine based on ranking of results and its features. International Journal of Information and Communication Technology, 10(4), 392-405.
6. Faheem, M. R., Anees, T., & Hussain, M. (2022). Keywords and spatial based indexing for searching the Things on Web. KSII Transactions on Internet and Information Systems*,* 16(5).
7. Kumar, K., & Al-Besher, A. (2019). Critical Analysis of Major Search Engines. Journal of Computational and Theoretical Nanoscience, 16(9), 3712-3716.
8. Ding, Z., Chen, Z., & Yang, Q. (2014). IoT- SVKSearch: a real- time multimodal search engine mechanism for the internet of things. International Journal of Communication Systems, 27(6), 871-897.
9. Perera, C., Zaslavsky, A., Christen, P., Compton, M., & Georgakopoulos, D. (2013). Context-aware sensor search, selection and ranking model for internet of things middleware. In IEEE 14th Int. conference on mobile data management (pp. 314-322), Milan, Italy.
10. Ostermaier, B., Römer, K.., Mattern, F., Fahrmair, M., & Kellerer, W. (2010). A real-time search engine for the web of things. In Internet of Things (IOT) (pp.1-8), Tokyo, Japan.
11. Grosky, W. I., Kansal, A., Nath, S., Liu, J., & Zhao, F. (2007). Senseweb: An infrastructure for shared sensing. IEEE multimedia, 14(4), 8-13.
12. Guinard, D., Trifa, V., Mattern, F., & Wilde, E. (2011). From the internet of things to the web of things: Resource-oriented architecture and best practices. In Architecting the Internet of things(pp. 97-129), Springer, Berlin, Heidelberg.
13. Bianchini, M., Gori, M., & Scarselli, F. (2005). Inside pagerank. ACM Transactions on Internet Technology (TOIT), 5(1), 92-128.
14. Lempel R., & Moran, S. (2000). The stochastic approach for link-structure analysis (SALSA) and the TKC effect. Computer Networks, 33(1-6), 387-401.
15. Bidoki, A. M. Z., & Yazdani, N. (2008). DistanceRank: An intelligent ranking algorithm for web pages. Information processing & management, 44(2), 877-892.
16. Cheng, T., Lauw, H. W., & Paparizos, S. (2011). Entity synonyms for structured web search. IEEE transactions on knowledge and data engineering, 24(10), 1862-1875.

17. Kalashnikov, D. V., Chen, Z., Mehrotra, S. & Nuray-Turan, R. (2008). Web people search via connection analysis. IEEE Transactions on Knowledge and Data Engineering, 20(11), 1550-1565.
18. Marchionini, G. (1992). Interfaces for end- user information seeking. Journal of the American society for information science, 43(2), 156-163.
19. Ciortea, A., Mayer, S., Bienz, S., Gandon, F., & Corby, O. (2020). Autonomous search in a social and ubiquitous Web. Personal and Ubiquitous Computing, 1-14.
20. Bhara, K., & Henzinger, M. R. (1998). Improved algorithms for topic distillation in a hyperlinked environment. In Proc. of the 21st annual international ACM SIGIR conference on Research and development in information retrieval (pp. 104-111).
21. Haveliwala, T. H., Gionis, A., Klein, D., & Indyk, P. (2022). Evaluating strategies for similarity search on the web. In Proc. of the 11th international conference on World Wide Web (pp. 432-442).
22. Hoopman, E. (2012). A Comparison of Commercial Aeration Systems in Northern Wisconsin Aquaculture Ponds. Bayfiel High School Research.
23. Fagroud, F. Z., Ben Lahmar, E. H., Toumi, H., Achtaich, K., & El Filali, S. (2020). IOT search engines: study of data collection methods. In Advances on Smart and Soft Computing (261-272), Singapore, Springer.
24. Xue, D., Nazir, S., Peng, Z., & Khattak, H. (2021). Selection and Ranking of Fog Computing-Based IoT for Monitoring of Health Using the Analytic Network Approach. Complexity.
25. Wang, H., Tan, C. C., Li, Q. (2009). Snoogle: A search engine for pervasive environments. IEEE Transactions on Parallel and Distributed Systems, 21(8), 1188-1202.
26. Javaid, S., Afzal, H., Arif, F., Iltaf, N., Abbas, H., & Iqbal, W. (2019). CATSWoTS: Context aware trustworthy social Web of things system. Sensors, 19(14), 3076.
27. Alashti, A. H., Rezaei, A. A., Elahi, A., Sayyaran, S., & Ghodsi, M. (2020). Parsisanj: a semi-automatic component-based approach towards search engine evaluation. arXiv preprint, 12097.
28. Mayer, S., & Guinard, D. (2011). An extensible discovery service for smart things. In Proc. of the Second International Workshop on Web of Things(1-6), San Francisco, California, USA.
29. Yap, K. K., Srinivasan, V., & Motani, M. (2008). Max: Wide area human-centric search of the physical world. ACM Transactions on Sensor Networks (TOSN), 4(4), 1-34.
30. Desai, B. A., Divakaran, D. M., Nevat, I., Peter, G.W., & Gurusamy, M. (2019). A feature-ranking framework for IoT device classification. In 2019 11th International conference on communication systems & networks (COMSNETS), pp. 64-71.
31. Iftikhar, H. U., Rehman, A. U., Kalugina, O. A., Umer, Q., & Khan, H. A. (2021). Deep Learning-Based Correct Answer Prediction for Developer Forums. IEEE Access, 9, 128166-128177.
32. Hwang, J., & Yoon, Y. (2021). Data Analytics and Visualization in Quality Analysis Using Tableau. CRC Press.