Enhanced Search Engine using Proposed Framework and Ranking Algorithm based on Semantic Relations

M. M. El-Gayar ¹, N. E. Mekky ¹, A. Atwan ², and H. Soliman ¹

¹Faculty of Computer and Information Science, Mansoura University, Egypt
²Faculty of Computer Sciences and Information Technology, Northern Boarder University, KSA
Corresponding author: M.M.El-Gayar (e-mail: Mostafa_elgayar@mans.edu.eg).

ABSTRACT Today, most users need search engines to facilitate search and information retrieval processes. Unfortunately, traditional search engines have a significant challenge that they should retrieve high-precision results for a specific unclear query at a minimum response time. Also, a traditional search engine cannot expand a small, ambiguous query based on the meaning of each keyword and their semantic relationship. Therefore, this paper proposes a comprehensive search engine framework that combines the benefits of both a keyword-based and a semantic ontology-based search engine. The main contributions of this work are developing an algorithm for ranking results based on fuzzy membership value and a mathematical model of exploring a semantic relationship between different keywords. In the conducting experiments, eight different test cases were implemented to evaluate the proposed system. Executed test cases have achieved a precision rate of 97% with appropriate response time compared to the relevant systems.


I. INTRODUCTION

The proliferation of the Internet and social media have added new challenges to the traditional search engines that must be addressed. Information retrieval system may be affected by the user input query, and the retrieved topics may be different and not related to the meaning of the search subject [1]. Also, the retrieved results may be affected by the ranking process [2] based on their relevance and semantic relation to the subject of the search [3]–[5]. So, there are several significant challenges in this field that can be summarized as follows [6]–[8]:

i) Most current search engines rely on indexing and retrieving different pages on keywords only that are often small, unclear, and does not reflect the meaning of the topic.

ii) Different search engines also rely on the user's query itself to search in the databases without increasing the keywords in this query to expand the scope of the search and improve the accuracy of the results.

iii) The ranking procedure is a significant issue and not a concern for some search engines. If a page is relevant to a query but is ranked very low (e.g., below top 20), then the user is unlikely to look at the page.

The idea of semantic search aims to solve the limitations and problems associated with traditional search based on keywords. Semantic search relies on the purpose of retrieving information using tags with adding the advantage of linking these tags and know the meaning behind and add more tags that improve the search results [1]. An ontology-based approach used to represent the vocabularies and relationships between semantic entities. Ontology describes the elements that exist in any field or area to represent semantic relation [2]–[6]. The contributions in this paper are divided into four main categories and can be summarized as follows:

i) Semantic web retrieval framework is proposed which improves the input unclear query and retrieves the relevant data with high precision in a fair time using the techniques such as MapReduce, Latent Dirichlet Allocation (LDA), Resource Description Framework (RDF) and Not Only SQL (NoSQL) document-oriented Model.
ii) A novel of preprocessing algorithm is developed to extract useful keywords from crawled pages.

iii) A novel of ranking algorithm and mathematical model of calculating semantic score is developed to order and classify the relevant results of unclear query based on semantic relations.

iv) Enhancing the query engine using the ontology analyzer and Wordnet to increase the input keywords.

The remaining of this article is divided as follows: Section II reviews previous work of traditional, semantic information retrieval schemes and focused on ranking methods. Section III describes the global challenges that meet search systems. Section IV illustrated the various elements of the suggested framework and suggested ranking method. Section V describes test cases and experiment outcomes, and evaluation will be performed. Finally, the last section will provide conclusions and references.

II. Previous Work

In this section, some of the researches that are interested in information retrieval with the standard and modern ways will be reviewed. After that, some systems related to semantic retrieval will be discussed and focused on its weaknesses. Finally, we will focus on research that is related to the semantic ranking of results.

Most of the current search mechanisms are classified into several similar items in their components such as crawling and indexing but vary in the way they work. Search systems are organized into categories such as search engines such as Google, and directories such as Yahoo, and Meta Search systems. Most of the standard search mechanisms are very popular, but their results are sometimes inaccurate which have a lower precision and high recall. They lack to find the meanings of terms and expressions used in web pages and their relationships. The problem lies in the existence of words that have many meanings in natural languages [7]–[9].

Modern and intelligent search systems, like Swoogle, SWSE, Falcons Object Search, … etc., are designed based on the semantic approach to overcome the traditional problems. Swoogle is a system that relies on semantic crawling and indexing of web documents. It is divided into four main components: data discovery, the creation of metadata, data analysis and retrieval [10]. The most significant drawback of this system is that it is not a general-purpose search system and is limited to predefined ontologies files [7]. Also, it has some weaknesses such as weak indexing of massively large data and time-consuming of query response as discussed in section 5.

Hogan et al. [11] have designed a model of the semantic search system called Semantic Web Search Engine (SWSE).

It is a comprehensive search system that provides services similar to traditional search engines based on RDF and link related data. It consists of the crawling process, indexing, Reasoning, and retrieval phase. But, SWSE has some weaknesses such as weak ranking of records because the ranking step is proceeding before the indexing step.

Cheng et al. [12] have built a retrieval model based on semantic relation called Falcons Object Search. For every detected object, the system produces an extensive implicit record containing the textual summaries obtained from its RDF representation. Unfortunately, this model is not carried by a ranking process to rate these objects that related to the submitted query.

Hakia [7] is another system that acts as a comprehensive semantic engine that works for general purposes. It is called a search engine based on meaning rather than search terms [7]. It consists of many components like query processor, ontology analyzer, QDex storage, and ranking approach. Unfortunately, this model has a scalability and indexing weakness because of virtual contents and dynamic contents.

A semantic search engine called Semantic Illegal Content Hunter (SICH) has been developed to detect illegal content and with the financial support of the European Crime Prevention and Control Organization [13]. The specific purpose of this engine is to analyze the semantic text and identify the illegal content. This engine consists of three stages, and the first stage includes the collecting and indexing of information. The second phase is the data analysis and is not sufficient in massive large data. Finally, a third phase that interacts with the user to retrieve the data but is not a way to rank outcomes [13].

Al-Yahya et al. [14] had created a model based on the ontology of Arabic dictionaries to search the texts in the Holy Quran. This proposed model is based on semantics, but in the field of Arabic, especially in the Quran, and although it achieves accurate results, it does not rank or index the results, but only depends on the similarity match.

Al-Safadi et al. [15] had proposed a systematic system for an Arab search engine based on the field of ontology, but only within Arab blogs. It categorized the Arabic language into a series of classes, characteristics, and relationships. But it works on Arabic only and in specific areas as it takes a lot of time to enforce the task.

Medhat et al. [16] had developed a proposed an Arabic semantic search system that relies on four main components, such as data acquisition, an indexer, classifier and retrieval engine. But it depends on acquired data from structured governmental data, which is not large.

Laddha et al. [17] had proposed a tourism-based search system based on understanding the user's query and
providing relative results for this query. However, this system is different from our proposed method. It is a specific system in the field of tourism only which it is not a comprehensive model and did not rank the results or classified the topics to facilitate the indexing and retrieval of large data.

Abatal et al. [18] had proposed an intelligent system based on semantic research and the integration between the ontology and cloud computing of health services. In this research, authors have been able to publish and share medical reports quickly and accurately. But despite the excellent work done by the researchers, it lacks some important points, such as dealing with large data, splitting reports before indexing, and the lack of system to arrange results in a way that makes it easier for the user to get information quickly and accurately.

Both Surgy Brin and Larry Page [19], [20] developed a method for ranking pages indexed through the Internet. Google works in this way. This algorithm state that if a page that contains important links and topics then its links to other pages become important also. But this method has become impractical at present and because of the fabrication pages with fabricated links and not related to the query subject. In [21] authors proposed a weighted page rank mechanism based on visited links.

In [22], authors were focused on the study area that intended to avoid confusing results by ranking query outcomes of a sparkle Protocol and RDF Query Language (SPARQL) query. Inappropriate, they produced a novel method that supports both keyword exploration and extending ranking measures.

Finally, authors [23] suggested a novel procedure of ranking process. The characteristics needed for ranking the relevance of entities are the number of subjects, the number of objects, average frequencies and number of literals. However, this approach is not fitting for semantic relations.

### III. Main Challenges in Semantic Search Retrieval Systems

As mentioned previously, most of the traditional or modern search engines face many problems. Some of these challenges or issues will be addressed in this section and will be solved using the proposed framework in the next section.

#### A. Efficiency

Search system’s accuracy depends upon the volume of documents to resemble, related outcomes, matching records, and response time. The main benefit of smart semantic search systems is retrieving the most significant results with high precision and lower recall. The returned results from Google had a higher recall than precision rate [8], [15], [22].

#### B. Ranking Procedure

The focal concept of the semantic web retrieval system is to recover the various important (most exactness) and actual outcomes as a reply to a query. A ranking approach is a challenging task given that there are “more than 12.3 billion sources on the internet” at the moment of typing [8], [27], and an appropriate query on a system may declare thousands of outcomes. It is crucial to classify and rate the recovered documents effectively [2].

#### C. Expansibility

Scalability or expansibility is the ability of a system to manipulate a quickly expanding the volume of data. Relational databases are structured but not scale well. But, expansibility for records in a semantic web offers extra difficulties because of the massive volume of unstructured related data [8], [24], [25].

#### D. Unbalanced Big Data

One of the challenges of search engines is unbalanced big data (one of the most challenge of big data). The big challenge that many researchers have been working on lately is dealing with large unbalanced and unregulated data. Search result from any search directory or search engine may lead to many randomized results which are not well categorized [25], [26].

### IV. Proposed Framework and Algorithms

The recommended framework is introduced in a modular way and reasonably formed of two distinct stages. Firstly, the backend phase is comprising crawling, indexing, and ranking stages where the servers are operated separately from users. Secondly, the frontend or retrieval phase is containing browsing and retrieving stages where a user can operate directly with the backend servers. As shown in Fig. 1, Each blue box of this framework focuses on one contribution of this paper. The next subsections describe each part of the proposed framework in details.

#### A. Crawling Part in Backend Phase

Backend phase is divided into four main blocks, which are crawling, data mining, indexing, and ranking processes. These blocks have been performed away from users (not in a real-time) because they consume a long time. The first critical process in that phase is the crawling process. In this process, we use MapReduce [35] (a tool used to handle a large amount of data) as a multithreaded programming model to support parallelization and divide and conquer style over the input data (page file).
The first sub-step of crawling procedure is extracted useful tags as defined in Algorithm 1. The input parameter is the entry page, and the output parameters are some collection of arrays. The primary step of this algorithm is focused on traverses TREE objects (Nodes) of entry page (index or threaded page) to an array of objects and data nodes using DOM-TREE. After that, the looping process (through an array of DOM objects) is used to filter unused tags with depth less than 3 (such as HTML, BODY, HEADER, … etc.). Useful data nodes that pass-through filtration process are stored in an array of extracted nodes. But, if the tag is a link tag, then it is used to recrawling as a new thread. The pre-processing step is discussed in the next paragraph.

The second sub-steps of crawling procedure are parsing and cleaning steps (called pre-processing steps). The pre-processing procedure is used to parse data regions from useful DOM objects (nodes), cleaning data records and obtain valuable keywords as defined in Algorithm 2. This module is a blue box as in Fig.1. We can simplify parsing and cleaning processing method into three main processes. Firstly, the map function divides the input, for example, a DIV or TITLE object containing many texts, into chunks of independent data. For each separate section of this data, a pair of key and value is calculated. Secondly, these pairs send the key and value to parallel maps whose task is to search for an event, word, or several synonyms. The Mapper outputs one or more intermediate key-value pairs. Thirdly, the reduce function is used to process the intermediate output data (i.e., filtering, summarizing, sorting, caching, taking an average, or finding the maximum). In this part as mentioned before, crawling procedure with all sub-steps are implemented using Hadoop and MAP-Reduce tools to cover the time consumption (Efficiency challenge) that noted in section 3.

The first input parameter of pre-processing algorithm (named Algorithm 2) is node which extracted from the previous crawling step. The second input parameter is M. M is the maximum number of specialized nodes. \( \tau \) is the similarity threshold. The object comparison can be done using string distance. When the data regions are defined, then the stop words removal and stemming processes (called cleaning methods) are performed. Stemming is the process of removing inflection of words so that we have to convert different words in the same root or stem word which contains the similar meaning.
Fig. 2 illustrates an example of DOM TREE specialized nodes and data regions. Specialized nodes are nodes that have the same parent and they are adjacent in the DOM tree. Data region is a collection of specialized nodes or objects with following properties:

1) The specialized nodes or objects all have the same parent node or object.
2) The specialized nodes or objects are all adjacent.
3) Similarity between adjacent specialized nodes or objects is greater than the threshold \( \tau \)

For example, in Fig. 2, node 1, node 2, node 3 and node 4 are represented BODY, TABLE, IMG and DIV as parent nodes respectively. Node 5, node 6 and node 7 are represented TR tags as child nodes of TABLE nodes and considered as first data regions. Also, Node 8 and node 9 are represented as P (paragraph) tags and considered as second data regions. So, all gray nodes in Fig. 2 are considered as specialized node with different data regions.

**FIGURE 2.** An example of specialized nodes and data regions.

**Algorithm#1** Crawling Algorithm

**INPUT** ← URL (entry page)

**OUTPUT** ← DOA (DOM Objects Array from Threaded Pages) and ENA (Extracted Nodes Array)

1. \( \text{DOA} = \varnothing \)
2. \( \text{ENA} = \varnothing \)
3. Type = URL Detection (URL)
4. IF Type == index page OR threaded Page
5. New Thread ← URL
6. \( \text{DOA} = \text{DOM Objects of entry page} \)
7. For i: \( \text{DOA}.\text{Length} \) do
8. IF Tree Depth (DOA[i]) >= 3 then
9. IF DOA[i] == Link Tag then
10. Crawling (DOA[i]) // recursion
11. Else
12. \( \text{ENA}[i] = \text{DOA}[i] \)
13. End IF
14. End IF
15. End For
16. End IF
17. Return TPA, ETA and Rx

**Algorithm#2** Pre-processing Algorithm

**INPUT** ← Objects (Array of Crawled Objects), \( M, \tau \)

**OUTPUT** ← DR (Data Regions), KA (Keywords Array)

1. For i ← Objects.length
2. Match (Object[i].Children, M) //get all similar child
3. DR ← Identical Data Regions (Object, M, \( \tau \))
4. IF coveredObjects ← (Object.Children ∈ DR)
5. Foreach coveredObjects as CO do
6. Remove duplicates (CO.Child.Text)
7. Remove Stop Words (CO.Child.Text)
8. (keys, values) = MAP(CO.Child.Text)
9. \( \text{KA} = \text{Reduce} (\text{Maximum TF} (\text{keys, values})) \)
10. \( \text{KA} = \text{Stemming} (\text{KA}) \)
11. End For
12. IF UncoveredObjects ← (Object.Children ∈ DR)
13. Pre-processing (UncoveredObjects, M, \( \tau \))
14. End IF
15. Return DR, KA
16. End For

**B. Topic Analyzer Part (Backend Phase)**

Topic Analyzer comes after the crawling which considered as one of our main contributions. Association rule method like Apriori algorithm was applied as the first step to find a most relative keywords with high support. After that, Latent Dirichlet Allocation [25], [27] (LDA) method was applied as second step to perform topic or subject modeling. After that, fuzzy C-mean algorithm is used as third step to group related keywords and assign central keyword for each subject with a specific membership. Finally, ontology graph was created according to each topic cluster. The main goal of this part is to automatically assign the web pages with topic distributions, where a page may contain several topics that were learned with the help of statistical inference. LDA adopts the bag-of-keywords assumption since it does not take in consideration the order of the keywords in a page.

The word ontology arises from Greek ONTO (being) + LOGY (word). The subject of ontology is the study of the categories of things that exist or may exist in some domain. So, we use the ontology graph to facilitate knowledge representation and reuse [28]. The ontology graph construction is divided into two main processes. The first process is the initial subject creation process. In this process, the graph is initially in the form of a tree structure. The first node is the main subject of the topic, according to LDA, then the related sources or objects are added in the given hierarchy, including the central keyword node as shown in Fig. 3. Then, to each object, the weighted keywords are added in the given hierarchy. If a keyword is a duplicate in a subject and its objects, it is removed from the subject. To be semantically more specific, we decided to keep it as a keyword of the object. But, if a keyword is duplicated in two or more object of the same level under the same subject, it is removed from both and added to the subject.
The second process is the validation method. At this round, we have a collection of candidate keywords that require to be modified to their proper location in the graph. The issue here is about their relevance to the subject. Some of the selected keywords could be somehow far from developed subject content. Therefore, the concerned candidate keywords need to be filtered to remove irrelevant terms. The previous process is done in the Relevance validation. We start by removing the keywords that do not belong to the same category as the current node.

![FIGURE 3. Structure of the Semantic Ontology Graph.](image)

### C. Indexing Part (Backend Phase)

The part comes after the ontology creation part. In this part, outcomes from crawling, parsing and analyzing steps are stored into a database. But with massive data cannot be classified into a relational database because of scalability, storage, sorting, semantic and retrieval issues [29]. So, Indexer block performs essential first step such as convert ontology graphs to resource description framework [30] (RDF based on XML) data model file. Then, the second step is storing RDF documents and keywords within (key, value) pairs matching to terms and documents sequentially as displayed in Fig. 4. The initial step shows specifically what is needed to "invert," i.e., to accumulate all values relating to the equivalent key, matching to the documents that include the equivalent term. to overcome time consuming challenge.

<table>
<thead>
<tr>
<th>Query Keyword</th>
<th>RDF ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>QKW1</td>
<td>ID1, ID2, ID3, ...</td>
</tr>
<tr>
<td>QKW2</td>
<td>ID3, ID4, ID6, ...</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>QKWn</td>
<td>IDn, IDn+1, IDn+2, ...</td>
</tr>
</tbody>
</table>

(a)

![FIGURE 4. Cache tables of URL resources (a) Cache table of query keywords as key and corresponding RDF Id as a value (b) Cache table of RDF Id as key and corresponding document source as a value.](image)

Due to scalability, we use NoSQL document-oriented module instead of a relational DB. Cache tables contain keywords and related RDF ID. Also, due to the semantic and unstructured problem, RDF based on XML scheme is used to support ontologies and relations.

### D. Ranking Part (Frontend)

The rank process is the main contribution of this paper and represented in Algorithm 3. All the above sections will be combined to find out the ranking score in this section. However, outcomes of crawling and indexing processes are keywords, positions, and RDF documents. We proposed a mathematical equation (3) that combines the outcomes of all the above sections that calculated in equation (1) and equation (2) respectively to find the ranking score for each document to retrieve the related document easily.

\[
LW_{ij}^{\text{in}} = \frac{L_i}{\sum_{k \in j} L_k}
\]  

(1)

Where \(LW_{ij}^{\text{in}}\) donates weight of link for pages i and j. \(L_i\) and \(L_k\) realize the number of in links of document i and document j, sequentially. \(R(j)\) donates the reference page list of page j.

\[
CW_{xi} = \left(\frac{\text{freq}_x}{N}\right) U_{xi}
\]  

(2)

Where \(CW_{xi}\) donates weight of a content for keyword x on page i. \(\text{freq}_x\) donates the number of frequencies of keyword x on page i. \(N\) donates the total number of strings in page i ignoring stop words. \(U_{xi}\) donates fuzzy membership of keyword x based on cluster center page i.

\[
SR_i = \sum_{x,j \in i} PR_i CW_{xi} LW_{ij}
\]  

(3)

Where \(SR(i)\) donates semantic rank for a document i. \(PR_i\) donates google page rank of document i.
CW_{xi} donates weight of a content for keyword x on page i calculated from equation (2). LW^in_{ij} donates weight of link for pages i and j calculated from equation (1).

**Algorithm #3: Create Ranking Score**

**INPUT**: Page (P), link Weights of P, Query (Q)

**OUTPUT**: Semantic Score (SS)

1: Begin Procedure
2: Initially enter a Q to search.
3: N = Expand Q with other keywords using wordnets
4: If \( \forall N \in P \)
5: Loop x \( \in N \)
6: Calculate CW in equation 2 for x on P
7: Find all links of P
8: Calculate LW in equation 1 for P
9: GET Page Rank (P)
10: End Loop
11: Return SS (P) in equation 3
12: End Procedure

**E. Retrieval Part (Frontend)**

In this stage as displayed in Fig. 1, three main blocks are utilized which directly connected to the user. The First block is the query engine which has a keyword generator manner that applied to divide the query call into separate words. The Second block is query optimizer which has two main sub-steps. Firstly, scanning the keywords claims by the end user and parsing it using algorithm#3. Secondly, optimizing the query and checks grammatical errors using the Levenshtein as character distance method.

Finally, the third block utilizing word-net to expand keywords of a given query [31]. After that, query engine compute similarity score between expanded tags and tags in caching table. Tags with highest semantic score used to retrieve RDF documents.

The similarity score measurement is also one of the main contributions of this paper. We propose the following normalized semantic score (NSS) equations for input links in ontology graph between resource i and resource j:

\[
NSS_{in}(R_i, R_j) = \begin{cases} 
0, & R_i \neq R_j \\
1, & R_i = R_j \\
\text{otherwise}, & R_i \text{ related to } R_j 
\end{cases}
\]  

(5)

Where:
- Ri is resource i of ontology graph with assigned keyword i
- Rj is resource j of ontology graph with assigned keyword j
- T is the total resources in the graph

Fig. 5 illustrates an example of frontend retrieval processing using search engine query. If we have a query (e.g. Search Engine), then the first step to split, clean and transform this query. After that, the query divided into two distinct keywords (search) and (engine). Then, every keyword was expanded using wordnet and the normalized semantic score in equation 8 was calculated. The final step, the intersection resources was summed to get the final related resources and retrieved these resources.

**FIGURE 5.** Example of Frontend Processing.
V. Experimental Results

A. Implementation Tools

The recommended system has been implemented using some tools and technologies as presented in Table 1, including Apache server version 2, RDF scheme based on XML (Structure format), PHP programming language (for Sending HTTP Request, perform crawling, indexing and ranking processes), and MapReduce technology (Hadoop).

The proposed implementation runs on five personal computers which one master PC used to split tasks between slaves, and others are slaves used to perform different offline processes and PCs characteristics are Intel (R) Core (TM) i7-4702MQ CPU @ 2.00 GHz processor. PCs used to crawl datasets about 30GB of over 5 million web pages.

<table>
<thead>
<tr>
<th>Devices</th>
<th>5 PCs (1 master, 4 slaves).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Apache Server version 2.</td>
</tr>
<tr>
<td>Platform</td>
<td>Intel (R) Core (TM) i7-4702MQ CPU @ 2.00 GHz processor.</td>
</tr>
<tr>
<td>RAM</td>
<td>8 GB for each PCs.</td>
</tr>
<tr>
<td>Database scheme</td>
<td>RDF scheme with NoSQL document-oriented module.</td>
</tr>
<tr>
<td>Programming and Query languages</td>
<td>PHP and SPARQL.</td>
</tr>
<tr>
<td>Size of Crawled Data</td>
<td>25 GB of two million web pages of different fields.</td>
</tr>
</tbody>
</table>

B. Implemented Test Cases

In this subsection, we will present different test cases using various criteria. To see if the proposed methods improve the results, we should test different queries in length using different technologies. Table 2 represents the summary of different test cases using different criteria. The implemented eight different test cases will be conducted as follows:

1) Test Case I: In this test case, the system is triggered by a simple unclear single word as query (e.g. engine). Also, in this test case, we will apply all different proposed techniques explained in section 4 to validate system performance and response time.

2) Test Case II: In this test case, the system is triggered by the same query as in test case I (e.g. engine). But, in this test case, we will apply all different proposed techniques without applying proposed ranking process and caching system.

3) Test Case III: In this test case, the system is triggered by the same query as in test case I (e.g. engine). But, in this test case, we will apply all different proposed techniques without applying proposed ranking process and caching system.

4) Test Case IV: In this test case, the system is triggered by the same query as in test case I (e.g. engine). The basic parts of any search engine will be applied like crawling, indexing and retrieval based on query itself. But, in this test case, we will not apply all the proposed techniques.

5) Test Case V: In this test case, the system is triggered by complex multiple words as query (e.g. semantic search engine). Also, in this test case, we will apply all different proposed techniques.

6) Test Case VI: In this test case, the system is triggered by the same query as in test case V (e.g. semantic search engine). But, in this test case, we will apply all different proposed techniques without applying keywords expansion and similarity score based on semantic relations processes to a query.

7) Test Case VII: In this test case, the system is triggered by the same query as in test case V (e.g. semantic search engine). But, in this test case, we will apply all different proposed techniques without applying proposed ranking process and caching system.

8) Test Case VIII: In this test case, the system is triggered by the same query as in test case V (e.g. semantic search engine). The basic parts of any search engine will be applied like crawling, indexing and retrieval based on query itself. But, in this test case, we will not apply all the proposed techniques.

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Query Words</th>
<th>Support Ontology</th>
<th>Proposed Rank</th>
<th>Semantic Score</th>
<th>Keyword expansion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Case I</td>
<td>Single</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Test Case II</td>
<td>Single</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Test Case III</td>
<td>Single</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Test Case IV</td>
<td>Single</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Test Case V</td>
<td>Multi</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Test Case VI</td>
<td>Multi</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Test Case VII</td>
<td>Multi</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Test Case VIII</td>
<td>Multi</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
C. Experimental Results and Evaluations of Test Cases

In this sub-section, we will review the results of applying test cases mentioned in the previous subsection. It should be noted that all the results used to measure the performance of the system were calculated according to the following metrics and equations:

\[
R = \frac{TP}{TP + FN} \tag{6}
\]

\[
P = \frac{TP}{TP + FP} \tag{7}
\]

\[
F-Score = \frac{2PR}{P + R} \tag{8}
\]

Where, TP, FP, TN, and FN which refer to true positive, false positive, true negative and false negative respectively. Recall (R) refers to the percentage of returned records that were retrieved correctly and labeled as negative as not related to the query. Precision (P) refers to the percentage of returned records that were retrieved correctly and labelled as positive as most related records to the query.

**TABLE 3.** Results of Different Test Cases using Single Keyword in the query.

<table>
<thead>
<tr>
<th>Input Keywords</th>
<th>Test Case I</th>
<th>Test Case II</th>
<th>Test Case III</th>
<th>Test Case IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extended Keywords</td>
<td>8</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>True Positive</td>
<td>8690</td>
<td>6481</td>
<td>8630</td>
<td>4991</td>
</tr>
<tr>
<td>False Positive</td>
<td>302</td>
<td>430</td>
<td>362</td>
<td>1120</td>
</tr>
<tr>
<td>Precision</td>
<td>0.966</td>
<td>0.937</td>
<td>0.959</td>
<td>0.889</td>
</tr>
<tr>
<td>Recall</td>
<td>0.938</td>
<td>0.943</td>
<td>0.921</td>
<td>0.957</td>
</tr>
<tr>
<td>F-score</td>
<td>0.952</td>
<td>0.935</td>
<td>0.94</td>
<td>0.921</td>
</tr>
<tr>
<td>Response Time at online Phase</td>
<td>2.57 second</td>
<td>1.22 second</td>
<td>3.49 Second</td>
<td>5.4 Second</td>
</tr>
<tr>
<td>Number of Fetched Document</td>
<td>5300 RDF Files</td>
<td>10980 RDF Files</td>
<td>6003 RDF Files</td>
<td>22033 RDF Files</td>
</tr>
<tr>
<td>Methods used</td>
<td>Basics + All proposed methods</td>
<td>All Except Keyword expansion &amp; Semantic Score</td>
<td>All Except Proposed Raking &amp; Caching system</td>
<td>Basics only</td>
</tr>
</tbody>
</table>

**TABLE 4.** Results of Different Test Cases using Multiple Keywords in the query.

<table>
<thead>
<tr>
<th>Test Case V</th>
<th>Test Case VI</th>
<th>Test Case VII</th>
<th>Test Case VIII</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Keywords</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Extended Keywords</td>
<td>12</td>
<td>0</td>
<td>12</td>
</tr>
<tr>
<td>True Positive</td>
<td>4871</td>
<td>3982</td>
<td>4815</td>
</tr>
<tr>
<td>False Positive</td>
<td>93</td>
<td>242</td>
<td>160</td>
</tr>
<tr>
<td>Precision</td>
<td>0.98</td>
<td>0.942</td>
<td>0.967</td>
</tr>
<tr>
<td>Recall</td>
<td>0.965</td>
<td>0.973</td>
<td>0.968</td>
</tr>
<tr>
<td>F-score</td>
<td>0.973</td>
<td>0.957</td>
<td>0.967</td>
</tr>
<tr>
<td>Response Time at Online Phase</td>
<td>1.78 second</td>
<td>0.88 second</td>
<td>2.18 Second</td>
</tr>
<tr>
<td>Number of Fetched Document</td>
<td>2309 RDF Files</td>
<td>6400 RDF Files</td>
<td>4055 RDF Files</td>
</tr>
<tr>
<td>Methods used</td>
<td>Basics + All proposed methods</td>
<td>All Except Keyword expansion &amp; Semantic Score</td>
<td>All Except Proposed Raking &amp; Caching system</td>
</tr>
</tbody>
</table>

**FIGURE 6.** System performance chart for test cases from I to IV that applied a single word in the query.
FIGURE 7. Response time chart for test cases from I to IV that applied a single word in the query.

FIGURE 8. Number of fetched documents chart for test cases from I to IV that applied a single word in the query.

FIGURE 9. System performance chart for test cases from V to VIII that applied multi words in the query.

FIGURE 10. Response time chart for test cases from V to VIII that applied multi words in the query.
Test case III, test case IV, test case VII and test case VIII achieved a higher time consumption than other first four test cases because the caching system and ranking process were not applied in this test case.

Test case IV and test case VIII fetched the highest number of RDF documents because the ranking, ontology, semantic score and keyword expansion processes were not applied.

The more keywords user entered in the query, the more keywords can be extracted and more relationships can be created.

The more keywords user entered in the query, the fewer documents that have been fetched.

E. Comparative study between proposed ranking algorithm and other ranking methods

In this sub-section, we conduct a comparison between proposed ranking procedure and other related ranking procedures of web pages. Each procedure has its benefits and drawbacks. The procedure which comes under the diverse types is compared under some factors like type of search, parameters, mining procedure, limitation, and performance. Table 5 shows the assessment of diverse procedures against the proposed procedure based on different criteria.

TABLE 5. Comparison between the proposed ranking method and other ranking methods in different systems

<table>
<thead>
<tr>
<th>Ranking method</th>
<th>Type</th>
<th>Mining method</th>
<th>Cons</th>
<th>Factors</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weighted Page Rank</td>
<td>Text Ranker</td>
<td>Web Structure and content</td>
<td>Not Efficient</td>
<td>visited links</td>
<td>Fast</td>
</tr>
<tr>
<td>Distance Ranking Method</td>
<td>Text Ranker</td>
<td>Web Structure</td>
<td>Weak with Big data</td>
<td>Back links only</td>
<td>Medium</td>
</tr>
<tr>
<td>Hits Ranking method</td>
<td>Image Ranker</td>
<td>Web content</td>
<td>Speed Problem</td>
<td>Image Content</td>
<td>Slow</td>
</tr>
<tr>
<td>Proposed Ranking method</td>
<td>Text Ranker</td>
<td>Web Structure and some Content</td>
<td>Work with Page Rank</td>
<td>Backlink, title and page meta data</td>
<td>Medium</td>
</tr>
</tbody>
</table>
F. Comparative study between proposed system and other related systems

Also, we conduct a comparison between the proposed system and other related systems in the same field. Every system has its pros and cons. The comparison comes under the different criteria like supporting semantic, handling big data, support ontologies, performance and time-consuming. The test cases I and V were conducted famous search system such as Google, Swoogle, falcon and proposed system. Test cases I and V were used in the comparison with other related systems because they use all proposed techniques and they achieved the highest accuracy among test cases in previous sub-section. Experimental outcomes with averages were conducted and represented in table 6 and table 7 for test cases I and V respectively. We can infer from these outcomes that the proposed system achieved high precision than other related systems. Table 8 shows the assessment of diverse related systems against the proposed system of web search based on different criteria.

**TABLE 6.** Comparison between different systems using test case I (simple unclear query)

<table>
<thead>
<tr>
<th></th>
<th>Google System</th>
<th>Swoogle System</th>
<th>Falcon System</th>
<th>Proposed System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>0.93</td>
<td>0.95</td>
<td>0.94</td>
<td>0.97</td>
</tr>
<tr>
<td>Recall</td>
<td>0.98</td>
<td>0.90</td>
<td>0.93</td>
<td>0.93</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.955</td>
<td>0.925</td>
<td>0.935</td>
<td>0.95</td>
</tr>
<tr>
<td>Retrieved Data</td>
<td>20M</td>
<td>20K</td>
<td>5K</td>
<td>10K</td>
</tr>
<tr>
<td>Avg. Response Time</td>
<td>1.53</td>
<td>3.11</td>
<td>2.92</td>
<td>2.57</td>
</tr>
</tbody>
</table>

**TABLE 7.** Comparison between different systems using test case V (Complex query)

<table>
<thead>
<tr>
<th></th>
<th>Google System</th>
<th>Swoogle System</th>
<th>Falcon System</th>
<th>Proposed System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>0.95</td>
<td>0.96</td>
<td>0.95</td>
<td>0.98</td>
</tr>
<tr>
<td>Recall</td>
<td>0.98</td>
<td>0.94</td>
<td>0.95</td>
<td>0.96</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.96</td>
<td>0.95</td>
<td>0.95</td>
<td>0.97</td>
</tr>
<tr>
<td>Retrieved Data</td>
<td>2M</td>
<td>13K</td>
<td>Less 1K</td>
<td>8K</td>
</tr>
<tr>
<td>Avg. Response Time</td>
<td>0.65</td>
<td>2.44</td>
<td>2.15</td>
<td>1.78</td>
</tr>
</tbody>
</table>

**TABLE 8.** Comparison between the proposed system and other related systems

<table>
<thead>
<tr>
<th></th>
<th>Google</th>
<th>Swoogle</th>
<th>Falcon</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Ontology</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Semantic Relations</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Support Ranking</td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Handle Big Data</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>High Precision</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>High Recall</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Response Time</td>
<td>Fast</td>
<td>Medium</td>
<td>Slow</td>
<td>Medium</td>
</tr>
</tbody>
</table>

**FIGURE 12.** Performance chart of Different Systems using Test Case I
FIGURE 13. Performance chart of Different Systems for Test Case V

FIGURE 14. Average Elapsed Time of Different Systems using Different Test Cases

IV. Conclusion

This paper proposes an information retrieval framework for extracting information from the recorded data based on ontology annotations and a mathematical ranking model based on semantic associations. Experimental results showed high precision and accuracy of our test cases in minimum time comparing to related systems. Experiments with eight different test cases are conducted to evaluate the proposed system. These test cases show that our model obtains comparable accuracy with high precision 97% in good response time comparing to related systems.

References


This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2941937, IEEE Access