International Journal of Web Engineering 2013, 3(1): 1-10
DOI: 10.5923/j.web.20130301.01

Enhancing Semantic Search Engine by Using Fuzzy Logic in Web Mining

Salah Sleibi Al-Rawi[1,*], Rabah N. Farhan[2], Wesam I. Hajim[2]

[1]Information Systems Department, College of Computer, Anbar University, Ramadi, Anbar, Iraq
[2]Computer Science Department, College of Computer, Anbar University, Ramadi, Anbar, Iraq

Abstract The present work describes system architecture of a collaborative approach for semantic search engine mining. The goal is to enhance the design, evaluation and refinement of web mining tasks using fuzzy logic technique for improving semantic search engine technology. It involves the design and implementation of the web crawler for automatizing the process of search, and examining how these techniques can aid in improving the efficiency of already existing Information Retrieval (IR) technologies. The work is based on term frequency inverse document frequency (tf*idf) which depends on Vector Space Model (VSM). The time average consumed for the system to respond in retrieving a number of pages ranging between 20-120 pages for a number of keywords was calculated to be 4.417Sec. The increase percentage (updating rate) was calculated on databases for five weeks during the experiment and for tens of queries to be 5.8 pages / week. The results are more accurate depending on the recall and precision measurements reaching 95% - 100% for some queries within acceptable retrieved time and a spellcheck technique.

Keywords Semantic Search Engine, Information Retrieval, Web Mining, Fuzzy Logic

1. Introduction

Due to the complexity of human language, the computer cannot understand well and interpret users' queries and it is difficult to determine the information on a specific website effectively and efficiently because of the large amount of information[1]. There are two main problems in this area. First, when people use natural language to search, the computer cannot understand and interpret the query correctly and precisely. Second, the large amount of information makes it difficult to search effectively and efficiently[2].

Berners-Lee et al.[3] indicated that the Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.

Al-Rawi Salah et al.[4] Suggested building a Semantic Guided Internet Search Engine to present an efficient search engine - crawl, index and rank the web pages by applying two approaches. The first was implementing Semantic principles through the searching stage, which depended on morphology concept - applying stemming concept - and synonyms dictionary. The second was implementing guided concept during input the query stage which assisted the user to find the suitable and corrected words. The advantage of guided concept is to reduce the probability of inserting wrong words in a query.

As for feature-rich semantic search engine, searches are divided into a classic search and information search. If you search for a term that has more than one meaning, it will give you the chance to choose what you were originally looking for, with its disambiguation results.

Leelanupab, T.[5], explored three aspects of diversity-based document retrieval: 1) recommender systems, 2) retrieval algorithms, and 3) evaluation measures, and provided an understanding of the need for diversity in search results from the users' perspective. He was developing an interactive recommender system for the purpose of a user study. Designed to facilitate users engaged in exploratory search, the system is featured with content-based browsing, aspectual interfaces, and diverse recommendations. While the diverse recommendations allow users to discover more and different aspects of a search topic, the aspectual interfaces allow users to manage and structure their own search process and results regarding aspects found during browsing. The recommendation feature mines implicit relevance feedback information extracted from a user's browsing trails and diversifies recommended results with respect to document contents.

One way to improve the efficiency of semantic search engine is Web Mining (WM). WM is the Data Mining technique that automatically discovers or extracts the information from web documents. It consists of the following tasks[6]:

* Corresponding author:
dr_salah_rawi@yahoo.com (Salah Sleibi Al-Rawi)
Published online at http://journal.sapub.org/web

Resource finding, information selection and pre-processing, generalization and analysis.

The present work describes system architecture of a collaborative approach for semantic search engine mining. The aim of this study is to enhance the design, evaluation and refinement of web mining tasks using fuzzy logic technique for improving semantic search engine technology. All main parameters in this proposal is based on the term frequency inverse document frequency (tf*idf) which depends on VSM.

2. Theory and Methods

2.1 Vector Space Model (VSM)

The VSM suggests a framework in which the matching between query and document can be represented as a real number. The framework not only enables partial matching but also provides an ordering or ranking of retrieved documents. The model views documents as "bag of words" and uses weight vectors representation of the documents in a collection. The model provides the notions "term frequency" (*tf*) and "inverse document frequency" (*idf*) which are used to compute the weights of index terms with regarding to a document. The notion "*tf*" is computed as the number of occurrence of that term, normally weighted by the largest number of occurrence. Mathematically this issue may be treated as follows:

Let N be the number of documents in the system and n_i be the number of documents in which the index term t_i appears. Let $freq_{i,j}$ be the raw frequency of term k_i in the document d_j. Then the normalized term frequency is given by: [7]

$$tf_{t,d} = \frac{freq_{i,j}}{\max freq_{1,j}} \quad (1)$$

where the max $freq_{l,j}$ is the maximum number of occurrence of term t_l in document d_j .

The *idf*, inverse document frequency is given by:

$$idf_i = \log \frac{N}{n_i} \quad (2)$$

The classical term weighting scheme is based on the following equation :

$$w_{i,j} = tf_{t,d} \times idf_i \quad (3)$$

There are also some variations of equation 3, such as the one proposed by Salton et al[7]:

$$w_{i,q} = \left(0.5 + \frac{0.5\, freq_{i,q}}{\max freq_{1,q}}\right) \times \log \frac{N}{n_i} \quad (4)$$

The retrieval process is based on computation of a similarity function, also known as the cosine function, which measures the similarity of the query and document vectors.[8]

$$sim\,(d_j, q) = \frac{\bar{d}_j \cdot \bar{q}}{|\bar{d}_j| \times |\bar{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w^2{}_{i,j}} \times \sqrt{\sum_{i=1}^{t} w^2{}_{i,q}}} \quad (5)$$

where $\bar{d}_j$ and $\bar{q}$ are vector representation of the document j and query q.

The *tf*idf* algorithm is based on the well-known VSM, which typically uses the cosine of the angle between the document and the query vectors in a multi-dimensional space as the similarity measure. Vector length normalization can be applied when computing the relevance score, ***Ri,q***, of page ***Pi*** with respect to query ***q***:[9]

$$R(i,q) = \frac{\sum_{termj \in q}\left(0.5 + 0.5 \cdot \frac{tf\,(i,j) idf_j}{tf \max i}\right)}{\sqrt{\sum_{termj \in Pi}\left(\left(0.5 + 0.5 \cdot \frac{tf\,(i,j)}{tf \max i}\right) \cdot (idf_j)\right)}} \quad (6)$$

where $tf(i,j)$: the term frequncy of $\boldsymbol{Q_j}$ in $\boldsymbol{P_i}$ $tf \max i$: the maximum term frequency of a keyword in $\boldsymbol{P_i}$ idf_j :

$$\log\left(\frac{N}{\sum_{i=1}^{N} C(i,j)}\right) \quad (7)$$

The full VSM is very expensive to implement, because the normalization factor is very expensive to compute. In *tf*idf* algorithm, the normalization factor is not used. That is the relevance score is computed by:

$$R(i,q) = \sum_{termj \in q}\left(\left(0.5 + 0.5 \cdot \frac{tf\,(i,j)}{tf \max i}\right) \cdot \left(idf_j\right)\right) \quad (8)$$

The *tf*idf* weight (term frequency–inverse document frequency) is a numerical statistic which reflects how important a word is to a document in a collection or corpus. It is often used as a weighting factor in IR and text mining. The *tf*idf* value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others[10]. One of the simplest ranking function is computed by summing the *tf*idf* for each query term; many more sophisticated ranking functions are variants of this simple model.

It can be shown that *tf*idf* may be derived as;

$$tf * idf\ (t,d,D) = tf(t,d) \times idf(t,D) \quad (9)$$

where: *D* is the total number of documents in the corpus.

A high weight in *tf*idf* is reached by a high term frequency (in the given document) and a low document frequency of the term in the whole collection of documents; the weights hence tend to filter out common terms. Since the ratio inside the $idf's$ log function is always greater than 1, the value of *idf* (and *tf*idf*) is greater than 0. As a term appears in more documents then ratio inside the log approaches 1 and making idf and *tf*idf* approaching 0. If a 1 is added to the denominator, a term that appears in all documents will have negative idf, and a term that occurs in all but one document will have an idf equal to zero[11]. The above concepts were exploited to build an algorithm with which it can be used to reduce the frequency in the page. The proposed ranking algorithm will be implemented as given in Algorithm (1).

In the suggested system all or most of the priority parameters were taken into consideration as they are listed and defined in Table (1). Each is given a value according to its importance. These values are estimated in reasonable way.

Table 1. Parameters Importance Values

Tag	Value	Description
<head>	0.10	Defines information about the document
<title>	0.10	Defines the title of a document
<base>	0.05	Defines a default address or a default target for all links on a page
<link>	0.05	Defines the relationship between a document and an external resource
<meta>	0.05	Defines metadata about an HTML document
<style>	0.10	Defines style information for a document

The style value is extracted from a collection of elements as illustrated in Table (2). In our work we derived a new formula based on the values of the styles which we think useful in deciding the ranking values. The formula is given by the following equation.

$$SV = \sum_{i=1 \text{ to } n} Si \times Z \quad Z = \begin{cases} 0 \text{ if } Si = \text{F} \\ 1 \text{ if } Si = \text{T} \end{cases} \qquad (10)$$

where

SV : Style Value

$\sum_{i=1 \text{ to } n}$: Sum of Style Value Matrix

Si : Assumption value matrix

Z : Alternative Boolean Values (0, 1)

n : Number of parameters of Style

Table 2. Style Values

Style	Value
Bold	0.20
Italic	0.20
Underline	0.20
Font Size	0.10
Font Type	0.10
Color	0.05
Blue	0.05
Red	0.05
Green	0.05

Thus, the algorithm ranks pages according to term importance by observing the number of (HTML Tags) that pointed to each page. Additionally, some pages are given extra authority based upon the number and rank of the pages to which they point.

Therefore, if several pages with high authority all refer to a particular page, it will be ranked higher than another page that has only low-ranking pages pointing to it. These parameters values it can be computed by using Algorithm (1)

Algorithm (1): Ranking Algorithm
Inputs: input webpage & word with it synonyms
Outputs: rank of webpage
Begin
Page_url ← web_page
Page_content ← web_page
Page_abstract ← web_page
Page_rank ← page
If word exist in content
Add 0.18*no_of_repeat words in content
End if
If syn1 exist in content
Add 0.08*no_of_repeat words in content
End if
If syn2 exist in content
Add 0.08*no_of_repeat words in content
End if
If sy3 exist in content
Add 0.08*no_of_repeat words in content
End if
If syn4 exist in content
Add 0.08*no_of_repeat words in content
End if
If word exist in HTML_Tag
Add 0.40*no_of_repeat words in content
End if
If word exist in keyword_abstract
Add 0.05*no_of_repeat words in content
Number of visit to the webpage
Add 0.05*no_ of visit to the webpage
End if
End of Algorithm

3. Machine Learning by Artificial Intelligence

Machine learning techniques have been used in IR and text mining applications. The various activities and efforts in this area are referred to as web mining. The term web mining may involve the use of data mining techniques to automatically discover web documents and services, extract information from web resources, and uncover general patterns on the web. In this proposal we used two ways for learning operation. First way using Self-Organizing Maps (SOM), and the second way by using Fuzzy Logic.

3.1. (SOM) Learning Rules

Kohonen's SOM is very well known as a clustering and dimension reduction tool. Clustering can be used for categorization of input vectors. Dimension reduction can be used for visualizing and for reducing information in order to ease search, storage or processing of another kind. In Kohonen's implementation of SOM was for categorizing Internet documents (WEBSOM)[12].

In the beginning of the functioning, all weights vectors of the second layer's neurons are set to random values. After that, some input-vector from the set of learning vectors is selected and set to the input of the Neural Network (NN). At this step, the differences between the input vector and all neurons vectors are calculated as follows:[13]

$$D_{ij} = |X^1 - W_{ij}| = \sqrt{(x_1 - w_{ij1})^2 + \cdots + (x_n - w_{ijn})^2} \quad (11)$$

where, *i* and *j* are the indices of neurons in the output layer. After that, the NN chooses the *winner-neuron*, i.e., the neuron whose weights vector is the most similar to the input vector:

$$D(k_1, k_2) = \min_{i,j} D_{i,j} \quad (12)$$

Here, k_1 and k_2 are indices of the winner-neuron. Now, we need to make a correction of the weights vectors of the winner and all the adjacent neurons. The neighborhood of a neuron is determined by a topological neighborhood function, as follows:

$$h(p,t) = \exp\left(\frac{p^2}{2\sigma^2(t)}\right) \quad (13)$$

Here, p is the distance to the winner-neuron:

$$p = \sqrt{(k_1 - i)^2 + (k_2 - j)^2}$$

At the next step, after calculating the topological neighborhood function for each neuron, the weights of all the neurons are updated as follows:

$$W_{ij}(t+1) = W_{ij}(t) + \alpha(t)\, h(p,t)\left(X^1(t) - W_{ij}(t)\right) \quad (14)$$

Here a *(t)* is a learning rate function that also decreases with time. If a neuron is a winner or adjacent to the winner, then its weight vector is updated, or remains unchanged otherwise. On each step, the NN determines the neuron whose weights vector is the most similar to the input vector, and corrects it and its neighbors' weights vectors to make them closer to the input vector.

Often, each input vector from the training set is presented to the NN, and learning continues either some fixed number of cycles or whiles the difference between an input and the weights vectors reach some epsilon value. The difference between adjacent neurons decreases with time, and hence they organize into groups (maps) which correspond to one of the classes from the learning set.

The text data from (Index) many webpages were used as our database. The objective of the model was to retrieve information with certain subjects.

The weights of documents are calculated and a program is written to preprocess and code the text document into numeric vectors. To determine the weight vector for each document, the keyword frequency in document is used. In other words, document weight is defined as term frequency (*tf*) in document. To calculate the weights, the following equation is used:[14]

$$w_{ik} = \frac{\left(tf_{ik} \cdot log\left(N/n_k\right)\right)}{\sqrt{\sum_{j=1}^{i}\left((tf_{ij})^2 \cdot log\left(N/n_j\right)\right)^2}} \quad (15)$$

In this equation: tf_{ik} is keyword frequency in document d" N is document number and N_k refers to the documents that include the word or term (*tf*). The above parameters for 140 sample documents will be determined.

The SOM learning process is considered as competition learning in Algorithm (2).

Algorithm (2): Self-organizing maps (SOM)
Start Initial weights w_{ij} Set topological neighborhood parameters Set learning rate parameter While stopping condition is false for each input vector x, do $\boldsymbol{d(j) = \sum_i (w_{ij} - x_i)^2}$ Find index j such that d(j) is a minimum For all units j within specified neighborhood of j and for all i $\boldsymbol{w_{ij}(new) = w_{ij}(old) + \alpha[x_i - w_{ij}(old)]}$ Update learning rate Reduce radius of topological neighborhood at specified time Test stopping condition End of Algorithm

3.2. Fuzzy Semantic Search Engine (FSSE) Architecture

In this proposal, we develop a fuzzy search engine, called Fuzzy Semantic Search Engine (FSSE). FSSE is constructed by using fuzzy logic to capture the similarities of terms in the web, which offer appropriate semantic distances between terms to accomplish the semantic search of keywords.

In this way advantages are obtained via the updating of each page continually, and hence the values will be more precise effective.

First, the FSSE can thus automatically retrieve web pages that contain synonyms or terms similar to keywords. Second, users can input multiple keywords with different degrees of importance based on their needs. The totally satisfactory degree of keywords can be aggregated based on their degrees of importance and degrees of satisfaction. Third, the domain classification of web pages offers users to select the appropriate domain for searching web pages, which excludes web pages in the inappropriate domains to reduce the search

space and to improve the search results. Figure (1) shows the FSSE architecture.

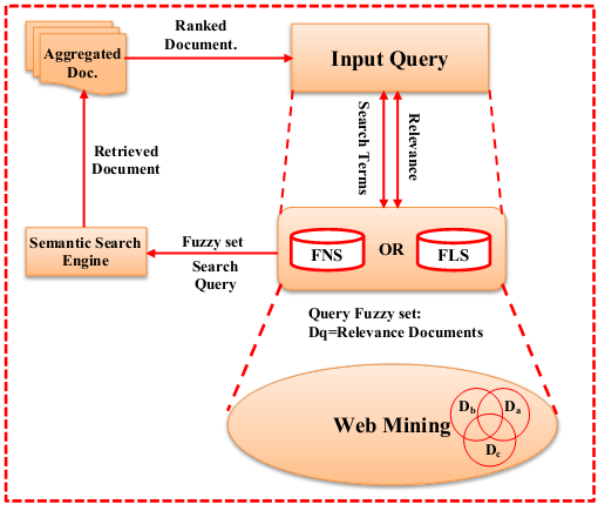

Figure 1. FSSE Architecture

The quality of the ranking function is an important factor that determines the quality of the IR system. Each document is assigned a score by the ranking function; the score indicates the likelihood of relevance of the document given a query.

In this proposal a fuzzy logic approaches is to define the ranking function. Fuzzy system provides a convenient way of converting knowledge expressed in a natural language in to fuzzy logic rules. The resulting ranking function could be easily viewed, extended, and verified as follows:

- if (*tf* is high) and (*idf* is high) ? (Relevance is high);
- if (overlap is high) ? (Relevance is high).

The relationship between IR system and Ranking Fuzzy Search System (R-FSS) is constructed use of Fuzzy Logic Toolbox. The first step in construction of an FSS is to define rules. Rules will be derived from a common knowledge about IR and the *tf***idf* weighting scheme. The order of the rules in fuzzy logic does not affect the output.

If many of the terms of the query are found in the document (overlap), the document is likely to be highly relevant as described below:

- if (overlap is high) ? (Relevance is high)

For each of the query terms the following rules are defined:

If a query term in a document has high *tf* and *idf* measures, the document is likely to be highly relevant:

- if (*tf* is high) and (*idf* is high) ? (Relevance is high)

We have found that the performance of the system is better if the rules that penalize low features are added. To achieve this we added the negated rules for each of the rules above:

- if (overlap is not high) ? (Relevance is not high)
- if (*tf* is not high) and (*idf* is not high) ? (Relevance is not high).

Approach to simply negate the rules is compact but it assumes that the opposing membership function is inversely symmetrical. Another approach to creating negated rules is by adding appropriate membership functions such as low and high.

Each rule has a weight associated with it. In case of R-FSS each *tf***idf* rule was assigned a weight of 1/t, where t is a number of terms in a query. Weight for the overlap rule was 1/6 of the weight of the *tf***idf* rule. Overlap rule weight is lower due to the fact that to some degree overlap rule is already represented in each of the *tf***idf* rules.

4. Web Mining Categories

The classification is based on two aspects: the purpose and the data sources. Retrieval research focuses on retrieving relevant, existing data or documents from a large database or document repository, while mining research focuses on discovering new information or knowledge in the data. On the basis of this, Web mining can be classified into three areas, Web Content Mining, Web Structure Mining and Web Usage Mining[15].

4.1. Web Content Mining

Web Content Mining is the process of extracting useful information from the contents of web documents. Content data corresponds to the collection of facts a web page that was designed to convey to the users. Web content mining is different from data mining and text mining. Nevertheless they are both related to web content mining. The latter is related to text mining because much of web contents are text based. Web content mining is different from text mining because of the semi-structure nature of the web, while text mining focuses on unstructured texts. Data mining is also different from web mining because web data are mainly semi-structured or unstructured. The technologies that are normally used in web content mining are Natural Language Processing (NLP) and IR[15].

4.2. Web Structure Mining

It is the process by which we discover the model of link structure of the web pages. We catalog the links; generate the information such as the similarity and relations among them by taking the advantage of hyperlink topology. PageRank and hyperlink analysis also fall in this category. The goal of web structure mining is to generate structured summary about the website and web page[16].

4.3. Web Usage Mining

It is the process by which we identify the browsing patterns by analyzing the navigational behavior of user. It focuses on techniques that can be used to predict the user behavior while the user interacts with the web. It uses the secondary data on the web. This activity involves the automatic discovery of user access patterns from one or more web servers. Through this mining technique we can ascertain what users are looking for on Internet[17], as shown in Figure. (2).

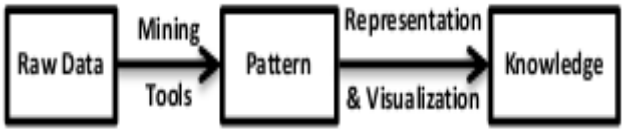

Figure 2. Web Mining Process

5. Proposed System

The system proposed in our work is schematically depicted in Figure (3) and Figure (4). It consists of two parts namely Client part and Server part. Each performs certain functions and they are interrelated.

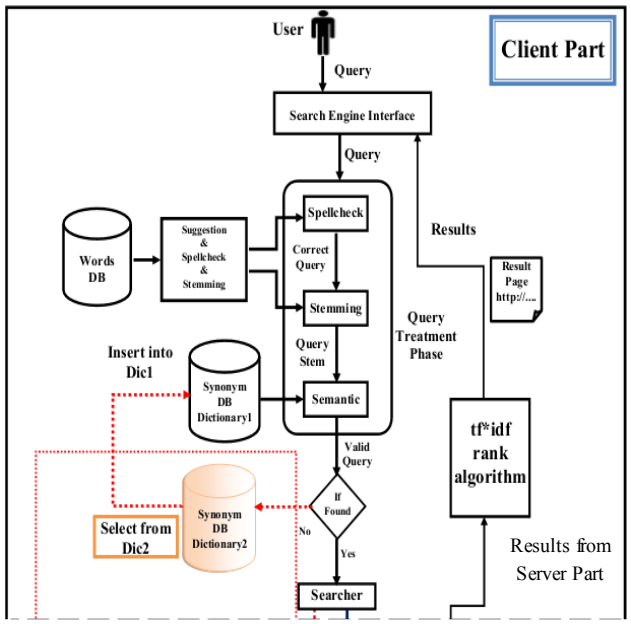

Figure 3. Client Part Architecture

The main goal of this project is the learning of the network that is the process of updating the database that contains the web pages automatically. That is through inquiry entered by the user, where the words entered are used in the search for pages required and are also used to search for alternative pages sites or other databases in the case of failure to obtain them from the main source. The query in the system passes through many stages for extracting best results for satisfying the user.

1). Removing stop words (Noise).
2). Spell check.
3). Extracting the root of the word (Stemming).
4). Bringing synonyms.
5). The searcher phase in the index.
6). PageRank derived through algorithm (*tf*idf*) used in this system.

● The first six steps represent the client part (forward part). The next steps represent server part (backward part).

7). Crawling: a so-called Spider.
8). Collection.
9). Indexing.
10). Inverted index.

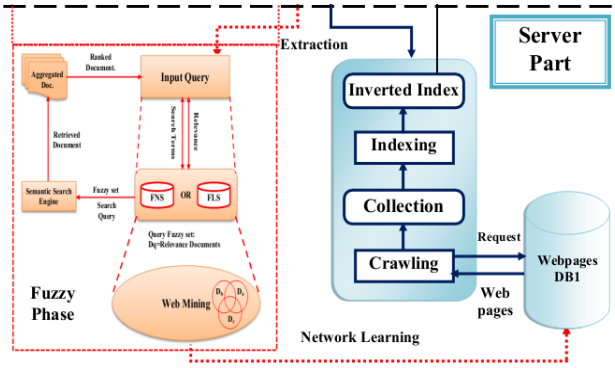

Figure 4. Server Part Architecture

● All the previous ten steps are traditional semantic engines steps. In this project new steps are considered and added for improving the work of these engines, as following:

1). Machine Learning: Fuzzy logic is used to train the network to update the master database.
2). Indexing new pages.
3). Updated thesaurus.

This proposal suggests a new algorithm that is based on some algorithms which considers semantic aspects and uses them to improving the semantic search engine, Algorithm (3) display the semantic aspects.

Algorithm (3): Semantic Search Algorithms

Inputs: terms of the query
Outputs: Ranked WebPages
Begin
Check the query
If empy then
Return empty webpage
Else
Correct_input ← Call check spell algorthim
List_index ← Call statement_searching_algorthim
If list_index is empty
Call word_searching_algorithm
End if
If list_index is empty
Call Updating Content
For i – sizeof(list_index)
SQL Procedure
Where list_index[i] = id of webpage in content table
Select * from content as selected_web_page
End SQL Procedure
Call Ranking algorthim
End for
Sorting the result_webpage depend on rank value of each webpage
Display the result
End of Algorithm

6. Results and Discussion

The Result Interface considered the main interface at which the retrieved data are displayed from the search operation, Figure (5) shows the results page.

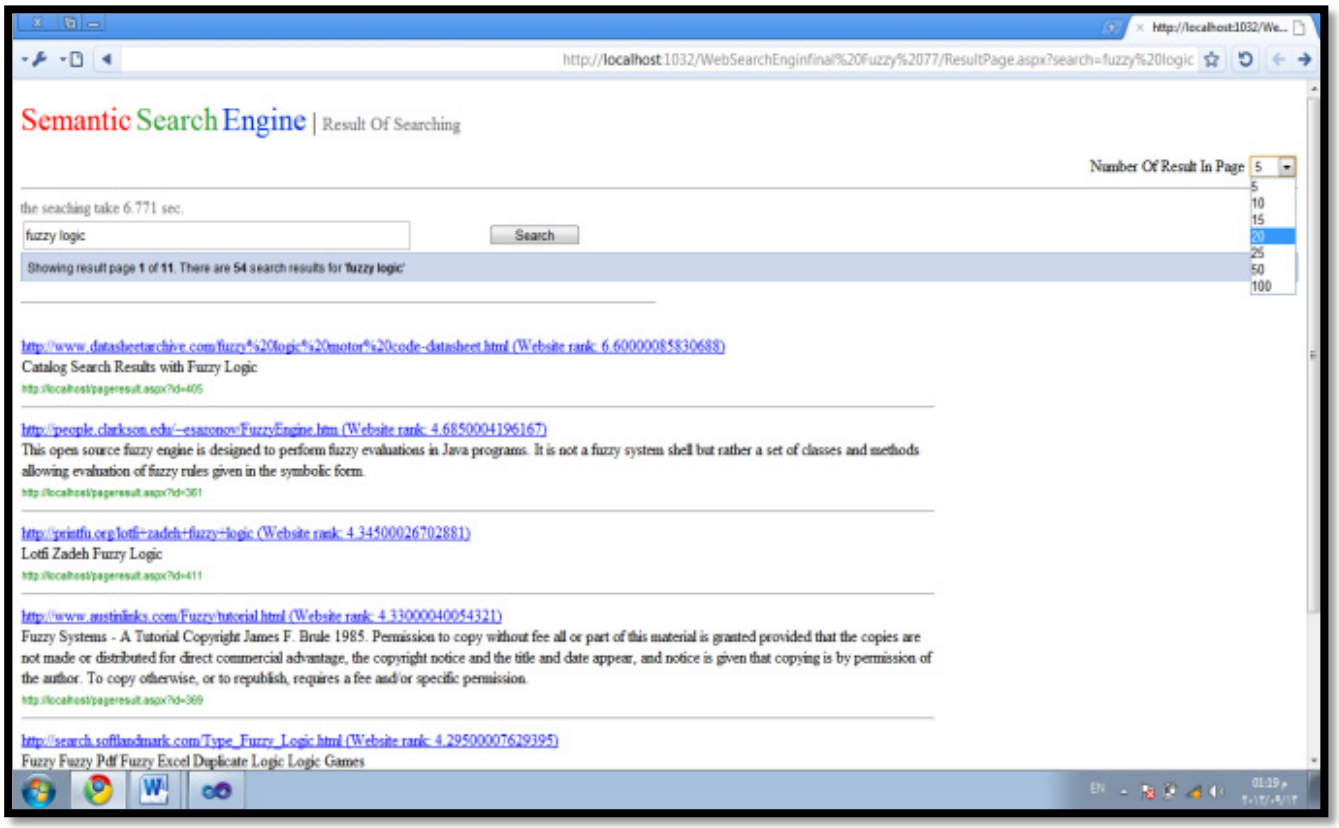

Figure 5. Results Page

The result page displays the number of results and the URLs associated to each result. In addition a brief summary on the contents of each title is given. The features of the results page is ability and the possibility of controlling the number of result displayed and provides the time consumer to obtain the results.

In this proposal the system has been divided into two phases. In order to improve its performance via obtained better results. In each phase a specific technique of Artificial Intelligence was employed for achieving the machine learning and web mining process, as follows:

Phase one: Self-Organizing Maps algorithm was used in this phase which is an Artificial Neural Network (ANN) technique. The system requirements were prepared and the needed Databases (DBs) were constructed to execute the system where around 86000 keyword has been used in the keywords database. This keywords DB are usually used to extract the keyword query root. Also about 300 webpages have been used. They are distributed on two DBs; the first is considered the main one and the second for updating and testing. In addition two synonyms, Dictionaries are used.

The first involves 650 keywords with four synonyms for each word, and the second involve 520 words with their synonyms, plus stop word database.

The system has been executed by the employed technique and certain tests on the performance and the efficiency of the system have been done where specific results are obtained.

From the results, it was noticed that the system response time will be slower in the case of increasing of the database size. It is due to this reason we moved for an alternative technique in order to improve the system efficiency and to obtain better and more precise results.

Phase 2: Fuzzy logic technique has been used and updating on the system has performed, where a novel step (Spellcheck) was introduced in addition to a large increase in the DBs size.

Table (3) illustrates the results and differences between the two phases.

Table 3. Comparison Results between SOM and Fuzzy

Query	SOM Results		Fuzzy Logic	
	No. of Retrieved Pages	Time (Sec.)	No. of Retrieved Pages	Time (Sec.)
Network	32 – 50	11.372	48 – 50	2.866
Neural	11 – 20	5.460	15 – 20	1.311
University	16 – 20	5.822	18 – 20	3.67
Sciences	33 – 40	9.337	37 – 40	7.73
Image	49 – 60	10.12	54 – 60	6.51

6.1. Spellcheck

Here we would like to show how the property is dealt within this system. A specified rang is labeled to indicate error percentages between the query and the word database as previously mentioned it contains more than 230,000 words.

This size in database involves the roots, suffix and prefix of each word. This percentage is bounded between 0 and 1 to

be more precise it is bounded between 0.12 to unity.

The percentage may be varied by the designer. The four operators **Substitution, Deletions, Insertions and Transposition** in the algorithm are specified. Figure (6) indicate how this property is implemented in our system.

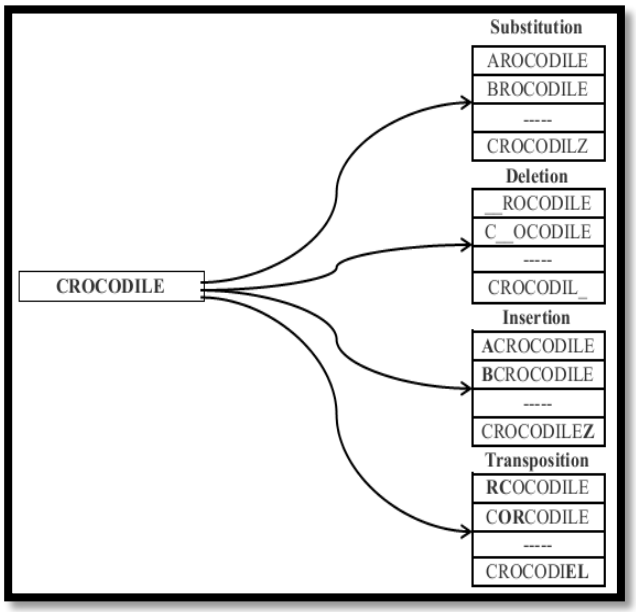

Figure 6. The four operations of Bitap

Here the query is represented in the binary system then a comparison with the existing word in the database is done by using operation a correct similarity is obtained. The correct operation it is illustrated in Algorithm (4).

Algorithm (4): Check spell algorithm

Inputs: user input query
Outputs: spell checked user input query
Begin
Spilt query into input_words_array
Retrieve correct_words list from words table
For I – sizeof(intput_words_array)
if words_array[i] not in dictionary
retrieve the suggestion words
take the most words as the_correct_word
replace wrong word with the the_correct_word in
input_words_array
end if
End for
Convert input_words_array into string
Return corrected_input_string
End of Algorithm

Nevertheless in some cases the retrieved information do not give the correct answers in particular when the input words are not typed correctly regarding spelling, Figure (7) illustrate Spellcheck feature, Table (4) displays an examples of correct words.

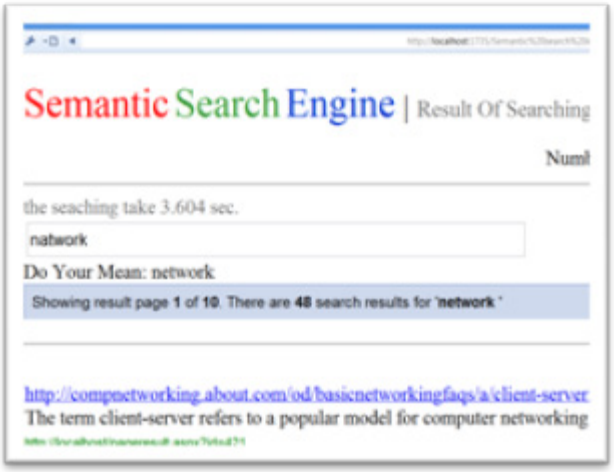

Figure 7. Spellcheck operation

Table 4. Examples of Correct (Spellcheck)

Spellcheck	
Error Query	Correct Query
Natwork	Network
Fuzzzy	Fuzzy
Boak	Book
Boook	Blood

6.2. Searcher

This component is responsible for searching and retrieving relevant results. First query analyzer performs mapping of query terms as well as query expansion using a semantic search. This component is responsible too for maintaining user log and keeping track of user search history.

This operation divides into two styles, search for sentence and search for word; it is illustrated in Algorithm (5) and Algorithm (6).

Algorithm (5): Statement_searching_algorithm

Inputs: correct input string
Outputs: index list
Begin
Spilt input_query into array_of_words
For I – sizeof(array_of_words)
Call SQL procedure
While Fuzzy_Compare(array_of_words[i] = word in indexing table) do
Add index_value in indexing table to list_of_indexing
End while
End for
Find the intersecting of list_of_index
Return list_of_index
End of Algorithm

Algorithm (6): Word_searching_algorithm

Inputs: correct input string
Outputs: index list
Begin
Spilt input_query into array_of_words
Array_of_syn_words ← Call find synonyms
For I – sizeof(Array_of_syn_words) of search words
Call SQL procedure
While Fuzzey_Compare(Array_of_syn_words[j][i] = word in indexing table)do
Add index_value in indexing table to list_of_indexing
End while
End for
Find the union of list_of_index
Return list_of_index
End of Algorithm

6.3. Updating the System Databases

The main source is updated by adding new pages that have been retrieved from other sites. This technology is called web mining; web mining is the application of machine learning (data mining) techniques to web-based data for the purpose of learning or extracting knowledge.

The DBs size was increased up to 234,000 words in addition to two synonyms dictionaries DBs. The first one content was more than 1900 keywords with their synonyms and the second content was around 1200 keywords, with their synonyms too.

This increase is about three times with respect to what it was at first phase i.e. same phase. The system at the second phase was tested and it was noticed that there is a significant difference regarding precision and the period of retrieving results in comparison with the first phase taking in to account the large differences in the DBs size and the system steps also. The Algorithm (7) refers to increase the main database.

Algorithm (7): Updating Content

Inputs: array of words with it synonyms
Outputs: updated content table
Begin
For I – sizeof(array of words with synonyms)
Call SQL procedure
While Fuzzey_Compare(Array_of_syn_words[j][i] exist in content in content2 table) do
Result_row = Select * from content2
Update content table by adding result_row
Indexing_new_row
End for
End of Algorithm

In addition to the updating the web pages database, the main thesaurus is updated via user queries.

In the case of the lack of a search word and synonyms in the main dictionary, they are looked up in dictionaries used in other engines or other sites. These operations take place without the user intervention since they are automatically obtained and the update is done by searching in (secondary) databases that have been created in order to experience the system. The proposed system was tested in different periods for many weeks, and then we had good results obtained for updating process. Table (5) shows the increase percentage and the updating on the dictionaries and webpages.

Table 5. Databases Updating for Webpages and Synonyms Dictionaries

Empiric period	Synonyms Dictionary		Webpages	
	From	To	From	To
First Week	1400	1403	142	145
Second Week	1403	1409	145	149
Third Week	1409	1417	149	155
Fourth Week	1417	1427	155	163
Fifth Week	1427	1436	163	171

7. System Evaluation

Many tests have been conducted on the suggested system using number of queries, where good results were obtained regarding precision, recall and speed in comparison with another SSE.

The search engine used is compared with (Lexxe) internet search engine (http://www.lexxe.com).

The comparison showed that for example using the word (Cryptography), the suggested system recall pages that contain (Cryptology, Steganography and Secret write). While with the (Lexxe SSE), the retrieved pages contain the word Cryptography only.

Also in another query such as (Network) in our proposed system pages titled (Meshwork, Mesh, and Web) are retrieved.

According to Table (6) it was noticed that the time response in searching for the keyword (Cryptography) was longer than in the case of other queries, while the number of pages is less than that in the case of other queries. This is because of the retrieved pages which contain the synonyms of that word which are not indexed. So pages will be transcribed then indexed before being displayed, because the search processes in the index table are not in the page contents. The same thing occurs with the word (Computer Vision) keyword. There were 34 retrieved pages out of 35 within 5.524 Sec. time period, while the number of pages in the (Search Engine) keyword was 115 out of 120 pages within almost slightly longer time period.

Table 6. Searching Results Matrices Calculate

Query	Retrieve Results	Recall	Precision	Time (Sec)
Neural	15 – 20	0.75	100 %	1.311
Network	46 – 50	0.92	100 %	1.966
Fuzzy logic	54 – 57	0.90	100 %	1.997
Search Engine	115 – 120	0.95	100 %	6.021
Cryptography	34 – 35	0.97	100 %	5.524
Computer Vision	76 – 80	0.95	100 %	8.338

8. Conclusions

The concluding remarks drawn from the results of the present work are as follows

1). Web Mining can improve the search results by exploiting the new semantic structures in the Web, such as extracting and utilizing semantics.

2). Web Mining technique, can be able increasingly treat the web content, web structure, and web usage.

3). The Fuzzy Logic is able to solving queries, because it not necessarily matching the query exactly.

4). The average of the time for the five tests by using fuzzy logic its 4.417Sec. While by SOM its 8.422 Sec although the large differences in the databases size and the system steps also.

5). The increase percentage (updating rate) on databases for five weeks during the experiment and for tens of queries to be 5.8 / week, this result was considered very good.

REFERENCES

[1] SUN, LEILEI, "Smart Search Engine for Information Retrieval". Master's thesis, Durham University (2009).

[2] Blachman, N. & Peek, J, Google Guide: How Google Works, (2003). Available at:http://www.googleguide.com/google_works.html .

[3] Lee T. B., "Conceptual Graphs and Semantic Web", URL: http://www.W3C.org/DesignIssues/Cg.html (2001).

[4] Al-Rawi Salah., Ahmed T. S., and Sumaya Abd. H., "Design and Evaluation of Semantic Guided Search Engine". International Journal of Web Engineering. 1(3): DOI: 10.5923/j. web. pp. 15-23, (2012).

[5] Leelanupab, T. A ranking framework and evaluation for diversity-based retrieval. PhD thesis (2012).

[6] T. Bhatia., Link Analysis Algorithms for Web Mining. International Journal of Computer Science and Technology, IJCST. Vol. 2, Issue 2, pp: 243-246, (2011).

[7] G. Salton and C.Buckley, "Term-Weighting Approached in Automatic Retrieval", Information Processing and Management, Vol. 24, No. 5 pp.513-523, (1988).

[8] Sergey Brin and Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search engine", Proceedings of the seventh international conference on World Wide Web 7, pp 107—117, Elsevier Science Publishers, (1998).

[9] Yuwono, B., Dik L. Lee. Search and Ranking Algorithms For Locating resource on the World Wide Web. In Proceeding IEEE .1996. Available at: http://citeseer.ist.psu.edu/yuwono 96search.html.

[10] Isra'a T. Ali, "Enhancement of Ranking and Query Optimizer in Internet Search Engine ", A thesis for M.S.C. Computer Sciences Department of University of Technology (2005).

[11] Pooja Sharma, Pawan Bhadana, "Weighted Page Content Rank For Ordering Web Search Result", International Journal of Engineering Science and Technology, Vol. 2, (2010).

[12] Kohonen, T., Self-Organizing Maps. Springer-Verlag, third edition, (2001).

[13] Kohonen, T. Self-Organizing Maps. Series in Information Sciences, Vol. 30. Springer, Heidelberg. Second ed. (1997)

[14] Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J. SOM_PAK: The self-organizing map program package. Report A31. Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland (1996).

[15] T. Bhatia., Link Analysis Algorithms for Web Mining. International Journal of Computer Science and Technology, IJCST. Vol. 2, Issue 2, pp: 243-246, (2011).

[16] R. Kosala, and H. Blockeel, Web Mining Research: A Survey, SIGKDD Explorations, Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining Vol. 2, No. 1 pp 1-15, (2000).

[17] Rekha Jain, G. N. Purohit, "Page Ranking Algorithms for Web Mining", International Journal of Computer Applications (0975 – 8887) Volume 13– No.5, January (2011).