Implementation and Evaluation of Personalized Semantic Search Engine (PSSE)

A.M. Riad, Hamdy K. Elminir, Mohamed Abu ElSoud, Sahar. F. Sabbeh

Abstract

This paper presents our implementation techniques for a personalized semantic search engine. The system includes several components such as a crawler, a preprocessor, searcher and ranking module. PSSE uses multi-crawlers to traverse web to gather resources. The preprocessor is used to identify crawled page importance based on link analysis techniques, annotate resources using agents that also mine document content and determine term importance. In this process, natural language processing techniques (NLP) i.e. stop-word removing and word stemming are applied to the raw resources. Searcher is responsible in turn for query completion activities making use on ontology as well as maintaining a log from users’ search activities. Finally, the query engine delivers search results ranked based on a final score calculated based on traditional link analysis, content analysis and a weighted user profile. In this paper we evaluate an implementation of PSSE using traditional information retrieval performance measures namely, precision, recall and F-measure. Results of this implementation have shown that PSSE worked efficiently.

Keywords: Search Engine, Semantic Web, Information Retrieval, Crawler, Personalization.

1. Introduction

If we took a look at the structure of the web, it’s composed of an enormous amount of documents and links between them. However, current web documents present human readable contents targeted at humans. Yet, the web is not used only by humans, as software agents are becoming users of the web too. This has led to the development of the semantic web [1, 2].

Information retrieval technology can draw massive benefits from using semantic web vision. Standard retrieval systems usually regard similarity between query terms and document metadata. However, these systems don’t take into consideration the semantic relationships between query terms and other concepts that might be significant to user. That necessitated the augmentation of semantic web vision into traditional retrieval systems resulting in the notion of semantic search. As search engines are seen as key application that can benefit from semantic web vision to provide improvements to recall and precision over traditional information retrieval (IR) techniques.

Traditional keyword-based search as it offers high recall but low precision, causing user to face too many irrelevant results. This is due to the deal with documents as a set of words disregarding web semantics as no semantic analysis is carried out. Semantic search provides enhancement to traditional search as it allows for retrieval that incorporates the underlying terms semantics [3, 4]. Whereas, users usually don't articulate well the terms they want to search providing only one or two terms for each search engine. In this context, using ontologies to represent relationships between concepts can improve search results [5-7].

That's why finding and ranking ontologies on the semantic web has been put forward as one of the motivations of the Semantic web vision and has been subject to many researches [8-10]. Found ontologies can be used to enhance search process either statically or dynamically [11].

Thereupon, researchers sake after developing a full featured semantic search engine. Swoogle was the first developed semantic search engine. Swoogle employs crawlers to discover RDF documents and
HTML documents with embedded RDF content. Swoogle exploits these RDF triples to record meaningful metadata about them in its database [12,13].

However, the birth of other markup languages such as RDFs, OWL ...etc and being widely used [14] made it mandatory to develop search engines that can deal with these languages.

After extensive study of the abovementioned systems' defects and limitation, we proposed PSSE [15], architecture of a personalized semantic search engine. PSSE uses multi-crawlers to provide crawling services for both semantic as well as traditional web. PSSE uses multi agents to provide annotation of web resources making use of ontology. Agents employ natural language processing (NLP) and information extraction techniques in order to crawl, analyze and extract useful information from resource content. Ontology is used as well to expand user query at search time to provide more enhanced search results. Finally, results are ranked based on a final score that takes into consideration link-based analysis, content-based analysis and a personalization factor (hereafter is referred to as PF) for more personalized results.

We shall begin this paper by a description of PSSE's main components. Then a detailed description of system implementation is presented in section 3. Evaluating PSSE using well known IR performance measures can be found in section 4. Our conclusion and future work is presented in section 5.

2. PSSE Architecture

Fig 1 shows the fundamental working principles of our system. Firstly, web pages are collected by multi-crawlers that traverse World Wide Web, collect web resources and store them in database. Crawlers work with the aid of information extraction techniques to find link information in the retrieved pages. Retrieved pages are then delivered to the preprocessor. Indexer and link analyzer works on the crawled pages to identify the global importance of the crawled page. This task is performed using PageRank (PR) is a more intelligent connectivity-based page quality metric with an algorithm that recursively defines the importance of a page to be the weighted sum of its back-links importance values [16-19]. Afterwards, come the role of annotation agents to annotate the crawled resources. Resources' content is then crawled making use of information extraction and natural language processing techniques. Extracted content is then weighted so as to determine their relevancy to web resource using term relevancy evaluator. Relevancy is measured using traditional vector space model TF-IDF [20]. The indexer sorts all the postings, then, creates an index to support fast accesses to the resources.

During search process, searcher together with ranking module performs the actual task. As searcher receives user query, query analyzer captures and updates user log. As PSSE considers users search activities as important resource to improve the search results. User query is then expanded with terms that are semantically relevant to query using ontology. Afterwards query terms are processed by removing stop words and performing stemming. From users’ query log, searcher mines out and assigns weights for the frequently used search terms in user query. Weights can be used by the ranking module for more personalized results. Finally, search agent retrieves results that match user query. Results are then passed to ranking module. Ranking module is responsible for ranking the retrieved results. Three factors contribute to the final ranking score. The first one is the page's global importance calculated in preprocessing phase using link analysis techniques. The second is the relevancy of resource content to query terms which depends on content analysis. And finally, the third factor is the personalization factor (PF). PF is used to support tailoring results according to user's interests and preferences. Personalization factor is calculated based on the analysis of user's log file. Analyzing user's search history can result in a value that represents user's interests in a particular query term. The final ranking score is the combination of these three factors.
3. PSSE Implementation

3.1. Crawler

Crawler is one of the important components in the system. Crawler undertakes the responsibility of gathering web pages from WWW and recursively repeat this process in each new page contained in the crawled page. In order for us to implement a crawler, URL seeds must be maintained. Crawler starts to crawl seeds to end then recursively repeat the process for all of the new gathered URLs added to the seed queue. Crawlers make use of traditional information extraction techniques in order to extract new URLs from crawled pages. Making use of multi-crawlers that work in parallel reduces processing time and increases system overall throughput.

Crawler works using Breadth First Search algorithm [21, 22] in order to recursively traverse WWW. The algorithm works as shown in Fig.2.
**Input:** List of URL seeds $S$

**Output:** Adjusted URL List $S$

```plaintext
Crawl($S$)    // takes seed list $S$ as an input
Start:
for each $s \in S$
{
    if (! visited[$s$]) {
        Extracturls($s$);
    }
}
Extracturls ($s$)
{
    $S$.add($s$)    // Add crawled page to List $S$
    for each $u \in URLs(s)$ {
        $S$.add($u$)   // Add Extracted URLs to List $S$
    }
    visited($s$)    // Mark page $s$ as visited.
}
Return $S$;
```

**Fig 2.** Crawling algorithm

### 3.2. Preprocessor

The Preprocessor contains three components: indexer and link analyzer, annotation agents and term importance evaluator. Indexer and link analyzer works on the crawled pages to identify the global importance of the page based on link analysis techniques, namely Google's PageRank.

Annotation agents then work on the crawled pages making use of information extraction techniques as well as natural language processing NLP techniques [23, 24] to process page content. Mining page content and extracting keywords requires recognizing the page's character encoding, strip away HTML tags (if any), scripts, and styles, decode HTML entities, and remove unwanted punctuation, symbols, numbers, stop words and get a list of keywords within a page. Term Importance Evaluator works on that list to weight each keyword within a page. The whole process goes into steps as follows and can be shown in Fig.3:

1. Get page text string. This can be achieved by retrieving the page directly from a web server to insure getting a complete page.
2. Strip HTML tags, scripts, and styles
3. Convert all HTML character references and entities into UTF-8 multibyte characters.
4. Strip punctuation characters including full stops (periods), commas, quotes, brackets, dashes, and so on.
5. Strip number characters
6. Split the text into a word list
7. Stem the words, shortens a word to its root. All of the variants of a word collapse into the same stem word, letting us focus on the keywords themselves instead of each minor difference in the way they are used. For English, we mainly used "Porter stemmer" algorithm [25].
8. Remove stop words, like "the", "and", "a", "by", and others are very common and make for poor search words. Search indexing can skip these so-called "stop words."
9. Remove unwanted words, extremely common words, like "web" or "html", make poor keywords. This can significantly shorten the keyword list and simplify further processing.
10. Use Vector space model (TF-IDF) [20] to weight each keyword both by its frequency and by its global importance across some large document set.
Implementation and Evaluation of Personalized Semantic Search Engine (PSSE)

A.M. Riad, Hamdy K. Elminir, Mohamed Abu ElSoud, Sahar. F. Sabbeh
International Journal of Intelligent Information Processing, Volume 2, Number 1, March 2011

Input: List of crawled URLs \( S \)
Output: List of \( R(u, k, w) \) // Pairs or URL \( u \), Keyword \( k \) and weight \( w \)

Process(\( S \)) // takes crawled list \( S \) as an input

Start:
For each \( s \in S \)

\{ If(! processed[\( s \)]) {
  Processcontent(\( s \));
}

\}

Processcontent(\( s \)) \{ 
  \( T=\text{Gettext}(s) \); //Return page text \( T \)
  StripHTML(\( T \)); //Convert \( T \) to unicode.
  \( T=\text{CUTF}(T) \);
  \( T=\text{Strippunctuation}(T) \);
  \( T=\text{Stripnumber}(T) \);
  \( A=T.\text{split}("\ \") \) //Split string \( T \) and get array of keywords \( A \)
  for each \( a \in A \)
  \{
    a=\text{Stem}(a); // Stem term \( a \)
  \}
  for each \( a \in A \)
  \{
    If(stopword(\( a \))) {
      Remove(\( a \));
    }
  \}
  for each \( a \in A \)
  \{
    \( w=\text{TF}(a) \);
    \text{R.add}(s, a, w);
  \}
  processed(\( s \)); // Mark page \( s \) as processed.
\}

Return \( R \);

Fig 3. preprocessing algorithm

The former components work in the offline phase of the system while the actual searching process takes place online. As system receives user query, process it, retrieves results, ranks results and display these results to user. This phase contains the following components:

3.3. Searcher

This component is responsible for the actual searching process. It starts working on receiving user query. First query analyzer performs text mapping using traditional text processing and natural language processing (NLP) techniques. Additionally, system updates and maintains a user log so that its information can be used during ranking phase to provide personalized search results.

As ontology enable contextualizing user's search terms by making it possible associating concepts and properties around a specific domain. Ontology can be used manually [26] or in the form of ontological analysis that results in suggestions that user can use to refine his query [27, 28] Domain ontology as well can help in filtering results that match user queries [35,36]. Automating the whole process without any user intervention and making it all transparent is a goal for many researches [29-31]. Taxonomies too can be used in correlation with ontology so that to provide better document annotation [32].
In PSSE, query is expanded with terms that might be of relevance to user query making use of ontology. In this step, term Synonyms, sub-concepts and super concepts are added to improve retrieval performance. In our system, WordNet Ontology is used for this step. Afterwards, search agent retrieves unranked relevant resources and filters them in order to be ranked. The whole process is shown in Fig.4.

Input: \( Q, R \)  // Query \( Q \) and Resource set \( R \)
Output: List of results \( L \)  // takes user query as an input

search(\( Q \))

Start:
for each \( q \in Q \)
{
    \( Q \) += getsynonymus(\( q \));
}
for each \( q \in Q \)
{
    If (stopword(\( q \))) { 
        Remove(\( q \));
    } Else{
        \( q \)=stem(\( q \));
    }
}
for each \( q \in Q \)
{
    \( L \).add(Getresult(\( q \)));
}
Filter(\( L \));
Return \( L \);

Fig 4. Searching algorithm

3.4. Ranking Module

Ranking is considered a key function of any search engine. Usually, current search engines take into account Link analysis techniques to classify result relevance, such as PagRank [17 -19]. However, another point of view censured link analysis techniques for not paying attention to document's content, that's why many researches used using vector space model, regarding content-based ranking to be a better ranking methodology [33, 34]. At the level of semantic ranking, XSearch [35], XRank [36] were essays to rank XML elements. In PSSE, results are ranked according to a final score that represents a combination of three different factors:

The first factor is Page authoritativeness which is calculated using link analysis techniques, namely PageRank algorithm. Authoritativeness value is calculated during the preprocessing phase. The second factor is content relevancy. Query terms in correlation with the weighted annotations are used to calculate query relevancy to each document individually.

The third factor, user interests and search history are used to provide more personalized search results. To provide personalized results, a user profile must be created and maintained using information that can be collected implicitly by monitoring user behavior or by explicit user input or feedback [37-39].

PSSE maintains a user log that contains user's usage data and search history. Terms from user's search history. Data in user's log are then used during ranking to calculate personalization factor which in addition to the pervious factors form the final score according to which results are ranked.

During ranking stage, weights are assigned to terms by analyzing user log and usage data against query terms. The frequencies assigned to profile keywords are significant since they express the rate of user interests. The weighting step starts from these frequencies to calculate profile query term weights. Calculating the weights of the initial query terms is performed by pointing out the highest frequency
number and dividing each frequency number by this highest number. Personalization factor (PF) determines the degree of user's interest in a certain query term. Personalization Factor (PF) is calculated according to equation 3.

$$\text{PF}_{(j,u)} = \frac{\sum s_j(u)}{\sum s_k(u)} \quad \rightarrow 3$$

Where
- $s_j(u)$ is the frequency of term $j$ in user search history.
- $s_k(u)$ is the entire number of terms appeared in user search history.

Finally, ranking module calculates the final score using weights calculated from link analysis, weighted annotation and PF as in equation 4,5:

$$\text{Score}_{(i,j,q,u)} = \sum_{j\in q} \text{sem}_{(i,j,u)} + PR(A_i) \quad \rightarrow 5$$

where:
- $\text{sem}(i,j,u)$: the similarity between document $i$ and query term $j$ for user $u$.
- $\text{score}(i,q,u)$ is the final weight assigned to document $i$ against query $q$ for user $u$.

Ranking module then passes results back to search agent which in turn passes them to user interface.

4. Evaluation and Analysis

In this section we describe the evaluation criteria as well as the data sets used for our experiments. Our goal is to build a personalized semantic search engine that provides a set of results that match to user query, in a reasonable amount of time, by making use of ontology to expand user query. In order for results to be as relevant as possible, PSSE has to expand user query with relevant terms making use of ontology.

PSSE gives better results when using domain ontology rather than a general purpose one. Thus, it's not considered as PSSE failure if the used ontology does not cover the information of the domain to expand user query.

We tested our prototype on a set of web pages that are collected, analyzed, annotated and saved in our local database using our crawler starting initially from www.en.wikipedia.com as the main URL seed. Query expansion is performed mainly using WORDNET ontology with around 147000 entities and 89489 relations. In total, we collected around 40,000 pages.

In order to evaluate the quality of PSSE, we used recall, precision and F-measure rates of the retrieved results against manual human opinions. Recall is a measure of how well PSSE performs in finding relevant items, while precision indicates how well it performs in not returning irrelevant items and F-measure is an average of the formers. PSSE tend to examine only the first $k$ results of a search. It measures what fraction of these $k$ results is relevant to the query on average. The value $k$ is commonly called the document cut-off value. Recall, Precision and F-measure are shown in formulas (1), (2) and (3) as defined in [40].

$$R@k(V) = \frac{\sum_{i=0}^{k} \text{rel}(v_i)}{\sum_{i=0}^{n} \text{rel}(v_i)} \quad (1)$$
Where: K : Document Cut-off value
n : The number of all retrieved documents.
R@K(V) = Recall value at top K documents.
Rel(Vi) : Relevancy of document vi against query

\[
P@K(V) = \frac{1}{k} \sum_{i=0}^{k} rel(v_i)
\]

Where: K : Document Cut-off value
P@K(V) = Precision value at top K documents.
Rel(Vi) : Relevancy of document vi against query

\[
F1@K(V) = \frac{2}{\frac{1}{P@K(V)} + \frac{1}{R@K(V)}} = \frac{2\sum_{i=1}^{n} rel(v_i)}{K + \sum_{i=1}^{n} rel(v_i)}
\]

The results of the evaluation, according to these quality measures, are shown in Fig.5. Results are evaluated when ranking results according to its global importance represented in page’s pagerank. Second, if content-based ranking is performed according to vector space model (TF-IDF). And finally after performing query expansion using domain ontology and ranking results using PSSE score.

Observing the obtained scores, we see that measures (Recall, Precision and F-measure) are high in case of using query expansion together with our final ranking score. While ranking based on content analysis comes next.

Although, this evaluation is preliminary and can be extended to a larger web page collections, the overall results suggest for appropriateness of using PSSE.

![Fig 5. Results](image_url)

5. Conclusion and Future work

In this paper, we provide a comprehensive survey on our implementation technologies of the text based retrieval system. This paper introduces our methodologies on each component of our system, such as Crawler, Preprocessor, Searcher and ranking module. Our system focuses on analyzing content as well as considering page's global importance based on analysis of link structure namely, Google’s Pagerank. In web environment, user query may lack relevant and significant terms. For this objective, PSSE makes use of ontology (WordNet TM) to expand user query. PSSE ranks results based on a weight that combines the analysis of link structure, content as well as a search activities. Evaluating PSSE performance proved that system worked efficiently.
In our future work, we will try to enhance content analysis by expanding the meta-data associated with each resource and try to find a way to weight these expanded data. Additionally, we believe that using multiple ontologies together with domain determination techniques can enhance the overall performance of PSSE. Other improvements can include weighting relationship between ontology concepts and include that in the final ranking of the results.

6. References

Implementation and Evaluation of Personalized Semantic Search Engine (PSSE)
A.M. Riad, Hamdy K. Elminir, Mohamed Abu ElSoud, Sahar. F. Sabbeh
International Journal of Intelligent Information Processing, Volume 2, Number 1, March 2011