Visualization for Information Retrieval in Regional Distributed Environment

Mamoon H. Mamoon, Hazem M. El-Bakry, Amany A. Slamaa

Faculty of computer science & information system
Mansoura University, EGYPT

Abstract
Information retrieval (IR) is the task of representing, storing, organizing, and offering access to information items. The problem for search engines is not only to find topic relevant results, but results consistent with the user's information need. How to retrieve desired information from the Internet with high efficiency and good effectiveness is become the main concern of internet user-based. The interface of the systems does not help them to perceive the precision of these results. Speed, resources consuming, searching and retrieving process also aren't optimal. The search engine's aim is developing and improving the performance of information retrieval system and gifting the user whatever his culture' level. The proposed system is using information visualization for interface problems, and for improving other side of web IR system's problems, it uses the regional crawler on distributed search environment with conceptual query processing and enhanced vector space information retrieval model (VSM). It is an effective attempt to match renewal user's needs and get a better performance than ordinary system.

Keywords:
Regional distributed crawler, VSM, conceptual weighting, visualization, WordNet, information visualization, web information retrieval.

Introduction

This paper tries to aggregate an optimal or at least semi-optimal information retrieval system by present visualized results supported by more efficient search engine than the standard. A search engine operates in the following order: Web crawling, Indexing, and Searching. The development include them as distributed regional web crawler, conceptual searching. The refinement on proposed system not stopped only at searching and results but also accommodate to involve personalization benefits.

The goal of an information retrieval system is to maximize the number of relevant documents returned for each query. Keyword information retrieval systems often return a proportion of irrelevant documents because matching keywords is imprecise: words can have different meanings when used in different contexts, and a single idea can often be expressed by several different words or synonyms. Information retrieval systems can be made more precise by matching concepts, keywords for which the intended meaning has been identified, either with information from a lexicographic database in the case of documents, or by asking the user to choose one meaning from several possible meanings in the case of queries.

The matching algorithms used by keyword IR systems are imprecise and retrieve irrelevant as well as relevant results. Two causes of this imprecision are terminology and semantics, both aspects of natural language. Terminology affects retrieval because different people use different words for the same concept. Terminology is often cultural; a pavement in the UK is a sidewalk in the US, for example. Semantics affects retrieval because the text of a document may not contain the exact keywords in the query but may nevertheless be about the topic of interest. This problem is exacerbated by polysemy. Polysemous words have different meanings in different contexts. For example, Java can refer to the Island in Indonesia, a type of coffee, coffee itself, or the object-oriented programming language. Matching a word does not identify the context in which it is used. The polysemous meanings, or senses, of words can lead to keyword queries that are ambiguous. Identifying the intended meaning of keywords can improve the precision of IR systems.

The concept IR model proceeds in three stages: the concepts in each document in the collection are identified, the concepts in a query are identified, and the query concepts are matched with the document concepts. Using the concept declared in details in sections 1, 2. And see how the concept also enhances one of information retrieval model – vector space Model - not only intended meaning problem in section 2.

As declared before and according to web information retrieval problems [18], there is attempt to solve some of them. In proposed system, there are some visualization forms, user select which one he prefer. The suitable form will display as default according to information that conclude from his profile and his set cabbalists – software and hardware -. This option is different on other retrieval – visualization system that it display only one form don't care that it easy and suitable all culture (inchoate or experienced) users. The system use personalization not only for customize results but also in improving searching process and increase time response. Figure 1 shows main components of the proposed system (VIZIRRD).

Manuscript received September 5, 2014
Manuscript revised September 20, 2014

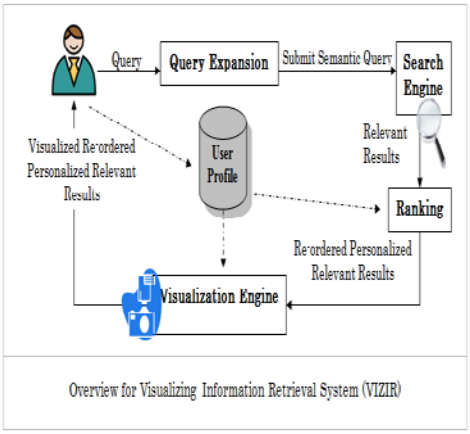


Figure 1: proposed system

The Visualizing Information Retrieval system (VIZIR) or visual information retrieval for the web has two main engines; search engine and visualization engine. Each one of them has own input and output and component that declared in next figure 2. This system also has a personalized feature. Combining these three will increase: performance, efficiency and each user get own system which declare how that achieve in the following sections.

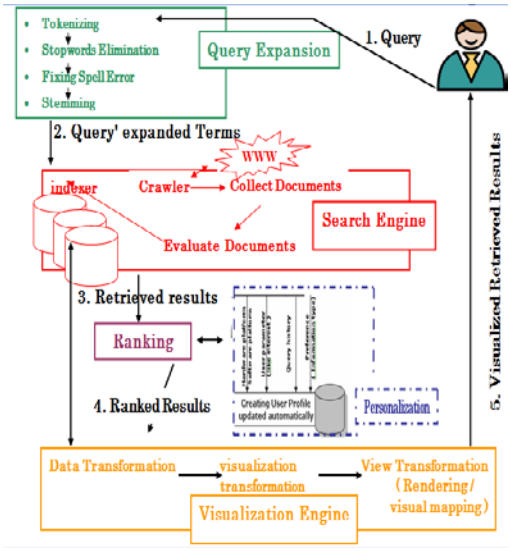


Figure 2: architecture of VIZIRRD system

1- Query Expansion

Whenever a user wants to retrieve a set of documents, he starts to construct a concept about the topic of interest; such a conceptualization is called the "information need". Given an "information need", the user must formulate a query that is adequate for the information retrieval system.

Usually, the query is a collection of index terms, which might be erroneous and improper initially. In this case, a reformulation of the query should be done to obtain the desired result. The reformulation process is called query expansion [4]. So, Query expansion (QE) is the process of reformulating a seed query to improve retrieval performance in information retrieval operations by expanding search query to match additional documents [9 and 41]. In the context of web search engines, query expansion involves evaluating a user's input (what words were typed into the search query area and sometimes other types of data) and expanding the search query to match additional documents. Query expansion involves techniques such as:

1. Finding synonyms of words, and searching for the synonyms as well.
2. Finding all the various morphological forms of words by stemming each word in the search query.
3. Fixing spelling errors and automatically searching for the corrected form or suggesting it in the results.
4. Re-weighting the terms in the original query [9].

1.2- Query expansion and WordNet:

Keyword IR systems retrieve documents by matching the keywords in the query with the keywords in the documents. A simple data structure that maps keywords to documents is an inverted index. Each keyword (Ki) is listed in an index and points to a list of the documents (Di) that contain the keyword:

K1 ⇒ D1, D2, D3, D4
K2 ⇒ D1, D2
K3 ⇒ D1, D2, D3
K4 ⇒ D1

Query expansion is one automated technique that has been used to address the imprecision of text retrieval techniques (Spink 1994). Query expansion adds keywords to a query that are related to the keywords supplied by the user, such as the synonyms of the keyword. For example, if the original query contains a single keyword, K, the synonyms of the keyword are added as disjunctions. The query, Q = K is expanded to incorporate the synonyms of K:

$QE = K \vee S1 \vee S2$

Adding the synonyms helps to overcome the problem of different words being used for the same concept also when user's query is only one, two, or maximum three words. Automatic methods of choosing synonyms are required because users find it difficult to come up with alternative search terms.

Query expansion techniques can be enhanced with concepts to make the expanded queries more specific. Once the keywords in a query have been disambiguated into concepts, the keywords relating to the generalizations, specializations, and the parts of the concepts can be added

to the query [7]. The importance of concept will be duplicated if we use it

1.3- Query expansion role in precision and recall

Search engines invoke query expansion to increase the quality of user search results. It is assumed that users do not always formulate search queries using the best terms. Best in this case may be because the database does not contain the user entered terms.
By stemming a user-entered term, more documents are matched, as the alternate word forms for a user entered term are matched as well, increasing the total recall. This comes at the expense of reducing the precision. By expanding a search query to search for the synonyms of a user entered term, the recall is also increased at the expense of precision. This is due to the nature of the equation of how precision is calculated, in that a larger recall implicitly causes a decrease in precision, given that factors of recall are part of the denominator. It is also inferred that a larger recall negatively impacts overall search result quality, given that many users do not want more results to comb through, regardless of the precision.
The goal of query expansion in this regard is by increasing recall, precision can potentially increase (rather than decrease as mathematically equated), by including in the result set pages which are more relevant (of higher quality), or at least equally relevant. Pages which would not be included in the result set, which have the potential to be more relevant to the user's desired query, are included, and without query expansion would not have, regardless of relevance. By ranking the occurrences of both the user entered words and synonyms and alternate morphological forms, documents with a higher density (high frequency and close proximity) tend to migrate higher up in the search results, leading to a higher quality of the search results near the top of the results, despite the larger recall [9].

2-A proposed Information Retrieval Model to use

A fundamental weakness of current information retrieval method is that the vocabulary that searchers use in formulating their queries is often not the same as the one by which the information has been indexed. In an attempt to resolve this drawback has been to combine Vector Space Model (VSM) and WordNet [4, 5 and 10] ontology after replaceing the Term frequency- Inverse Document Frequency TF-IDF term weighting with Concept-based Term Weighting (CBW) to Compatible with WordNet. WordNet is utilized to get conceptual information of each word in the given query context.

2.1- a new Query term weighting for Vector Space Model

2.1.1 Problem definition and suggestion

The main disadvantage of the vector space model is that it does not in any way de ne what the values of the vector components should be. The problem of assigning appropriate values to the vector components is known as term weighting. Early experiments by Salton (1971) and Salton and Yang (1973) showed that term weighting is not a trivial problem at all. They suggested so-called tf :idf weights, a combination of term frequency tf , which is the number of occurrences of a term in a document, and idf , the inverse document frequency, which is a value inversely related to the document frequency df , which is the number of documents that contain the term. Many modern weighting algorithms are versions of the family of tf :idf weighting algorithms. Salton's original tf :idf weights perform relatively poorly, in some cases worse than simple idf weighting [5].
As calculating query term importance was a fundamental issue of the retrieval process. The traditional term weighting scheme TF-IDF approach has following drawbacks:
Rare terms are no less important than frequent terms – IDF assumption
Multiple appearances of a term in a document are no less important than single appearance – TF assumption
Because of IDF assumption, the TF-IDF term weighting scheme assigns higher weights to the rare terms frequently. Thus, it will influence the performance of classification. Concept-based Term Weighting (CBW) calculates term importance by utilizing conceptual information found in the WordNet ontology. CBW was fundamentally different than IDF in that it was independent of document collection. The significance of CBW over IDF is that:
CBW introduced an additional source of term weighting using the WordNet ontology.
CBW was independent of document collection statistics, which is a feature that affects performance [5].

2.2.2 Vector Space Model (VSM) using WorldNet

Term significance can be effectively captured using CBW and then be used as a substitute or possible co-contributor to IDF. CBW presents a new way of interpreting ontologies for retrieval, and introduces an additional source of term importance information that can be used for term weighting. In proposed method, Concept-based Term Weighting (CBW) technique is used to calculate term importance by finding the conceptual information of each term using WordNet ontology. The significance of this technique is that:

1. it is independent of document collection statistics,
2. it presents a new way of interpreting ontologies for retrieval,
3. It introduces an additional source of term importance information that can be used for term weighting.

In this research project WordNet is the chosen ontology used by CBW. To determine generality or specificity for a term, conceptual weighting employs four types of conceptual information in WordNet:
1. Number of Senses.
2. Number of Synonyms.
3. Level Number (Hypernyms).
4. Number of Children (Hyponyms/Troponyms).
Overview of Concept based term weighting to calculate CBW value of a query term is shown in figure 3. As shown, there are three main steps involved to find the weight of a query. Extraction step extracts conceptual information of each word based on each POS (Noun, Verb, Adjectives) from WordNet. Weighting step find the weight of each extracted integer values for each POS based on weighting functions. After weighting fusion is applied to get a single CBW value for a query term. Any terms used in the query that are non-WordNet terms were given a default high CBW value. This is based on the assumption that the term does not appear in WordNet, is most likely a specific term, and thus it is highly weighted.

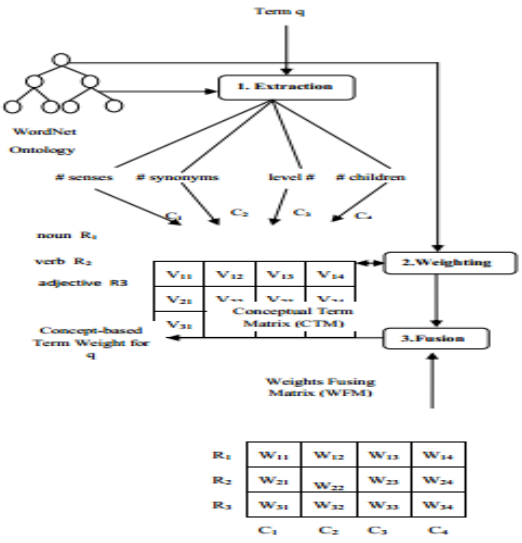


Figure 3: overview of concept-based term weighting (CBW) [5]

The block diagram shown in figure below consists of three main steps:
1. Extraction
2. Weighting
3. Fusion

Extraction:

This step works on a query given by user and extracts the conceptual value for each input query term from WordNet which includes number of senses, number of synonyms, level number (Hypernyms) and number of Children (Hyponyms/Troponyms). Extraction is done by using extraction algorithm [2] as shown below. Initially all values in conceptual term matrix (CTM) are set to -1. Then senses for each POS are counted from WordNet and listed in the first column of CTM. Similarly synonyms for each POS are found by selecting maximum synonyms for senses given by WordNet for a query term. Levels for each POS are found by selecting minimum hypernyms for senses given by WordNet for a input query term and listed in third column of CTM. And finally children for each POS are found by selecting maximum hyponyms/troponyms for senses given by WordNet for a query term. These extracted integer values are stored in Conceptual Term Matrix (CTM) [5].

Extraction Algorithm

Stemming the terms before building the inverted index has the advantage that it reduces the size of the index, and allows for retrieval of web pages with various inflected forms of a word (for example, when searching for web pages with the word computation, the results will include web pages with computations and computing). Stemming is easier to do than computing base forms, because stemmers remove suffixes, without needing a full dictionary of words in a language [5].

Weighting

Weighting is the next step after extraction. Weighting functions convert extracted integer values into weighted values in the range [0, 1]. These weighted values are stored in weighted conceptual term matrix. Based on min, max and avg values for each POS (noun, verb and adjectives) weighting functions are designed as shown in equation (1) and (2). The level number and the number of children are both set to zero for adjectives because adjectives are not organized in a conceptual hierarchy since they are only descriptors of nouns. Therefore, it is not possible to extract the level number and the number of children from WordNet for adjectives. Therefore weighting functions are not created for level number and number of children of adjectives.
a) General Weighting Function for
i. Nouns, Verbs Senses, Synonyms and Children
ii. Adjectives Senses and Synonyms

$$f(x)=\begin{cases}0 & ,x\geq Max\\ 0.5 & ,x=Avg\\ 1 & ,x=Min\\ f(x-\Delta x)-\dfrac{0.5\times\Delta x}{Avg-Min}, & Min<x<Avg\\ f(x-\Delta x)-\dfrac{0.5\times\Delta x}{Avg-Min}, & Max>x>Avg\end{cases}$$

Eq. (1)

b) General Weighting Function for Nouns, Verbs Levels

$$f(x)=\begin{cases}0 & ,x=Min\\ 0.5 & ,x=Avg\\ 1 & ,x\geq Max\\ f(x-\Delta x)+\dfrac{0.5\times\Delta x}{Avg-Min}, & Min<x<Avg\\ f(x-\Delta x)+\dfrac{0.5\times\Delta x}{Avg-Min}, & Max>x>Avg\end{cases}$$

Eq. (2)

In above functions is taken as an error factor. These all functions are based on Min, Max and Avg values of each POS. For noun, verb and adjective's senses, weight 0 is assigned for an integer value greater than or equal to Max, weight 0.5 is assigned for an integer value equal to Avg and weight 1 is assigned for an integer value equal to Min. For an integer value in the range [Min, Avg] is given a weight in the range [0.5, 1] and an integer value in the range [Max, Avg] is given a weight in the range [0, 0.5]. Same rules are applied for noun, verb and adjective's synonyms and children. For noun, verb and adjective's level, weight 0 is assigned for an integer value equal to Min, weight 0.5 is assigned for an integer value equal to Avg and weight 1 is assigned for an integer value greater than or equal to Max. For an integer value in the range [Min, Avg] is given a weight in the range [0, 0.5] and an integer value in the range [Avg, Max] is given a weight in therange [0.5, 1].

Fusion

Fusion is the last step to get single CBW value of a query that determines the importance of a term. Fusion is performed on weighted conceptual term matrix which is the result obtained by weighting. Fusion considers a new matrix named as Weights Fusing Matrix (WFM) of size 3*4 with all values set to 0.5 to give an average effect. WFM is shown below.

R_1	0.5	0.5	0.5	0.5
R_2	0.5	0.5	0.5	0.5
R_3	0.5	0.5	0.5	0.5
	C_1	C_2	C_3	C_4

Fusing steps:

1. Fuse each column of the weighting CTM with the columns of WFM using column weighted average function.

$$C_n=\frac{\sum_m Vm\times W_{mn}}{\sum W_m},n=1,2,3,4$$

Eq. (3)

2. Fuse the row R generated in step (1), as shown in previous using row weighted average to give the CBW term importance.

$$\mathrm{CBW_q}=\frac{\sum_n C_{n\times W_n}}{\sum_n W_n}$$

Eq. (4)

Where W is a set of weights with each element being a value in the range [0, 1], and set to 0.5 by default.

Note: weighting (CBW) = Weighted Conceptual Term Matri Weighted (CTM) X Weight Fusing Matrix (WFM)

3- Implementation of the concept in three phases

The first stage in the concept IR model is to identify the concepts in the documents in the collection. An index must be built that maps concepts to documents to enable fast retrieval. This process need only be done once for each document added to the collection. The index can be updated incrementally as each new document is added to the collection.

The concepts in a document are identified by first extracting the keywords and removing the duplicates and stop words. Each keyword is then added to a list of concepts. A keyword with more than one sense must be disambiguated before being added to the list of concepts. Five tests are performed to identify which sense of a keyword is present in a document. A point is awarded if any of the following conditions are met:

1. one or more of the synonyms of the sense occur in the document;
2. the sense is a part of a concept that occurs in the document;
3. a concept that occurs in the document is part of the sense;
4. the sense is a generalization of a concept that occurs in the document;
5. The sense is a specialization of a concept that occurs in the document.

The application of each test produces a matching score that indicates the algorithm's level of confidence that the concept is present in the document. Tied scores can be presented to a domain expert for final classification. Each concept (Ci) is stored in an index and points to a list of the documents that contain the concept. For example:

C1 ⇒ D1, D2, D3, D4
C2 ⇒ D1, D2
C3 ⇒ D1, D2, D3
C4 ⇒ D1

The list of documents for each concept in the index is augmented with the score Mi of matching concept C with document Di:

C ⇒ {D1, M1}, {D2, M2}, {D3, M3}

The same Boolean operations can be applied to an index of concepts as for an index of keywords.

For example, the simple query Q = java, must be disambiguated by asking the user which of the three senses of Java is intended. The keywords in a query can be disambiguated by presenting a list of the senses of each keyword and enabling the user to select the intended senses. Disambiguating a keyword by selecting one sense over the other senses indicates that documents containing the other senses should not be retrieved. If query Q is disambiguated into sense 3, the object-oriented programming language, then documents about the island or the beverage should not be retrieved. This requirement can be met by ensuring that documents containing the synonyms, specializations, generalizations, etc. of the other senses are not retrieved. This translates into a query such as:

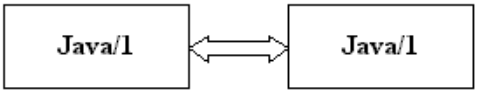
QE = java ∧ ¬(jakarta ∨ indonesia ∨ bali) ∧ ¬(espresso ∨ caffeine ∨ tea)

This query requests documents that contain the keyword Java but not the keywords that relate to the two senses of Java that are not required: the island and the beverage. The keywords that represent the island are one of its parts, Jakarta, the whole of which Java is a part, Indonesia, and another part of Indonesia, Bali. These are a selection of the keywords that might be present in a document about Java the island. The keywords that represent the beverage are a type of coffee, espresso, a substance that is part of coffee, caffeine, and an alternative to coffee, tea. These are a selection of the keywords that might be present in a document about Java the beverage.

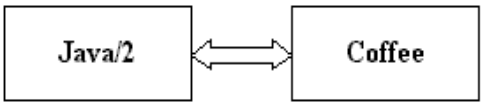
The final stage of the concept IR model is to match the concepts in a query with the concepts in the documents. Documents are matched with queries using the concept ⇒ document index. The degree to which the concepts in a query match the concepts in the documents is represented by a numerical matching score that is used to rank the results.

The concept IR model is more flexible than the strict matching performed by the Boolean keyword model. The Boolean model partitions documents into two sets: those documents that contain the query keywords and those those do not. This strict partitioning does not fit well with natural language. The concept IR model enables documents to be retrieved that match queries in varying degrees.

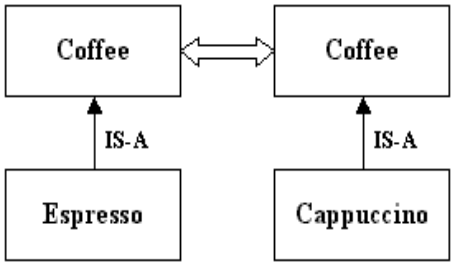
Five matching rules—based on the relations described in section 3—are used to generate a matching score. The base rule is that the same sense of a keyword always matches itself.



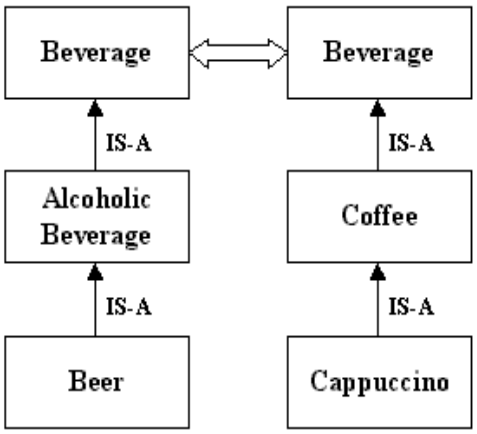
Synonyms of the same sense of a keyword always match.



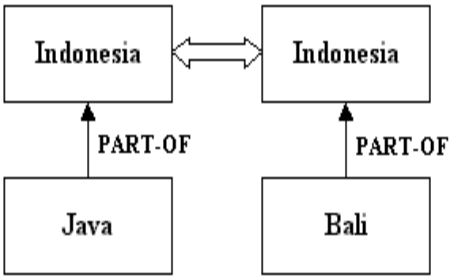
Concepts can be matched by the hyponym (generalization) relation. For example, espresso and cappuccino are both types of coffee, i.e. coffee is a generalization of both espresso and cappuccino. Concepts can be matched by applying a relation more than once.



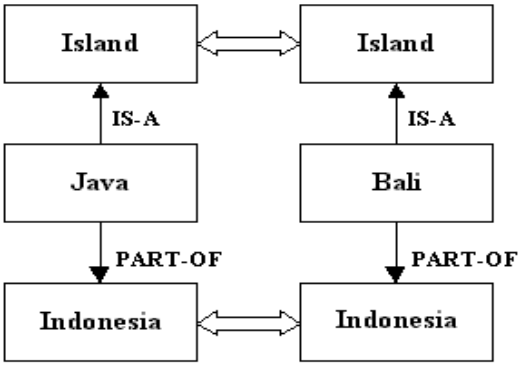
Similarly, beer is a type of alcoholic beverage and that cappuccino is a type of coffee. Alcoholic beverage and coffee are both types of beverage: beverage is a generalization of a generalization of beer and coffee.



Concepts can be matched by the meronym (part-of) relation. For example, Java and Bali are parts of Indonesia. Concepts can also be matched by more than one relation at once.

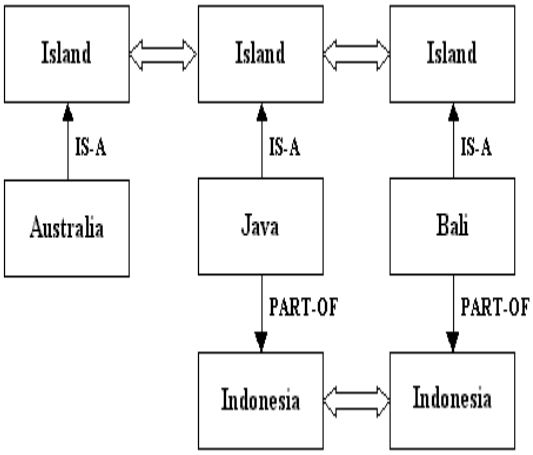


Java and Bali match Island with the hyponym relation, and match Indonesia with the meronym relation.



Two concepts, A and B, are matched in the following order:
1. check if A = B;
2. check if A is a synonym of B;
3. check if A and B are part of the same concept;
4. check if A and B have a common generalization;
5. check if A is a generalization of B.

The relation that matches two concepts determines the matching score. For example, the meronym relation is stronger than the hyponym relation. Concepts that are part of a whole are more closely related that generalizations of those parts. Java is more closely related to Bali than to Australia, for example, because Java and Bali are part of Indonesia, even though they are all islands.



Matching scores are weighted by the relation used to match the concepts. The relations have different weights and are weighted in the following order, from highest to lowest:
1. exact match;
2. synonyms;
3. parts;
4. specializations;
5. Generalizations.

Matching scores are also weighted by the number of relations used to match the concepts; the larger the number of relations used, the lower the score. For example, espresso and cappuccino would have a higher matching score than beer and cappuccino even though the three concepts have a common generalization, beverage. Espresso and cappuccino are matched with one application of the hyponym relation; beer and cappuccino are matched with two applications [7].

3- Feedback

To overcome query formulation and the inherent word ambiguity in natural language problems, researchers have focused on automatic query expansion to help the user formulate what information is really needed, declared before. Another widely used method of query expansion is the use of relevance feedback from the user which gives the relevance of documents to clarify the ambiguity. In fact, these two techniques complement each other. However, the mechanisms of relevance feedback based on words or documents in the past research both have their own deficiencies [8]. This involves the user performing a preliminary search, then examining the documents returned and deciding which are relevant. Finally, terms from these documents are added to the query and the search is repeated. This obviously requires human intervention and, as a result, is inappropriate in many situations. However, there is a similar approach, sometimes called pseudo-relevance feedback, in which the top few documents from an initial query are assumed relevant and are used for automatic feedback [4].

Spink et al. (2000) present results from the use of relevance feedback in the Excite search engine. Only about 4% of user query sessions used the relevance feedback option, and these were usually exploiting the ``More like this" link next to each result. About 70% of users only looked at the first page of results and did not pursue things any further. For people who used relevance feedback, results were improved about two thirds of the time [11].

4- User interface and information visualization

User interfaces are a communication between human information seekers and information retrieval systems. Information seeking is an imprecise process. When users

approach an information access system they often have only a fuzzy understanding of how they can achieve their goals. Thus the user interface should aid in the understanding and expression of information needs. It should also help users formulate their queries, select among available information sources, understand search results, and keep track of the progress of their search [1].
The roles of user interface are:
1. Aiding in the understanding and expression of information needs.
2. helping users formulate their queries, select among available information sources, understand search results, and keep track of the progress of their search.(formulate/ select/ understand/ keep track)
What makes an effective human- computer interface?
"Well designed, effective computer systems generate positive feelings of success, competence, mastery, and clarity in the user community. When an interactive system is well-designed, the interface almost disappears, enabling users to concentrate on their work, exploration, or pleasure [1] ."…….Ben shneiderman
Below we discuss those principles that are of special interest to information access systems.
1. Design principles
a. Offer informative feedback: users with feedback about the relationship between their query specification and documents retrieved, about relationships among retrieved documents, and about relationships between retrieved documents and metadata describing collections. If the user has control of how and when feedback is provided, then the system provides an internal locus of control.
b. Reduce working memory load: information access is an iterative process, the goals of which shift and change as information is encountered. one key way information access interfaces can help with memory load is to provide mechanisms for keeping track of choices made during the search process, allowing users to return to temporarily abandoned strategies, jump from one strategy to the next, and retain information and context across search sessions. Another memory-aiding device is to provide browsable information that is relevant to the current stage of the information access process. This includes suggestions of related terms or metadata, and search starting points including lists of sources and topic lists.
c. Provide alternative interfaces for novice and expert users: an important tradeoff in all user interface design is that of simplicity versus power. Simple interfaces are easier to learn, at the expense of less flexibility and sometimes less efficient use. Powerful interfaces allow a knowledgeable user to do more and have more control over the operation of the interface, but can be time-consuming to learn and impose a memory burden on people who use the system only intermittently. A common solution is to use a "scaffolding" technique. The novice user is presented with a simple interface that can be learned quickly and that provides the basic functionality of the application, but is restricted in power and flexibility. Alternative interfaces are offered for more experienced users, giving them more control, more options, and more features, or potentially even entirely different interaction models. Good user interface design provides intuitive bridges between the simple and the advanced interfaces [1].
From the viewpoint of user interface design, people have widely differing abilities, preferences, and predilections. Important differences for information access interfaces include relative spatial ability and memory, reasoning abilities, verbal aptitude, and (potentially) personality differences. Age and cultural differences can contribute to acceptance or rejection of interface techniques. An interface innovation can be useful and pleasing for some users, and foreign and cumbersome for others. Thus software design should allow for flexibility in interaction style, and new features should not be expected to be equally helpful for all users [1].
An important aspect of human-computer interaction is the methodology for evaluation of user interface techniques. Users require only a few relevant documents and do not care about high recall to evaluate highly interactive information access systems, useful metrics beyond precision and recall include: time required to learn the system, time required to achieve goals on benchmark tasks, error rates, and retention of the use of the interface over time [1].

Visualization

The human perceptual system is highly attuned to images, and visual representations can communicate some kinds of information more rapidly and effectively than text. For example, the familiar bar chart or line graph can be much more evocative of the underlying data than the corresponding table of numbers. The goal of information visualization is to translate abstract information into a visual form that provides new insight about that information. Visualization has been shown to be successful at providing insight about data for a wide range of tasks.
The field of information visualization is a vibrant one, with hundreds of innovative ideas burgeoning on the Web. However, applying visualization to textual information is quite challenging, especially when the goal is to improve search over text collections. As discussed, search is a means towards some other end, rather than a goal in itself. When reading text, one is focused on that task; it is not possible to read and visually perceive something else at the same time. Furthermore, the nature of text makes it difficult to convert it to a visual analogue.

Proposed visualization engine

The idea was to select existing visualizations for text documents and to combine them in a novel way. Our selection of existing visualizations was based on the assumption to find expressive visualizations keeping in mind the target users, their tasks, their technical environment (typical desktop PC and not a high end workstation for extraordinary graphic representations) and the type of data to be visualized (text documents). The idea was to visualize additional information about the retrieval documents to the user in a way that is intuitive, fast to interpret and can scale to large document sets.

Another important difference of our VIZIR system with existing retrieval systems for the Web is the comprehensive visual support of different steps of the information seeking process. The visual views used in VIZIR supports the interaction of the user with the system during the formulation of the query (e.g. visualization of related terms of the query terms with a graph), during the review of the search results (e.g. visualization of different document attributes like date, size, relevance of the document set with a scatter plot or visualization of the distribution of the relevance of the query terms inside a document with a TileBar), and during the refinement of the query (e.g. visualization of new query terms based on a relevance feedback inside the graph representing the query terms).

Visualization engine component

Systems combing the functionality of retrieval systems with the possibilities of information visualization systems are called visual information seeking systems. The next design decision after retrieving was to transform and save all search results and their characteristics in a local repository (RDBMS) with a specific data schema. The data schema for each document is described in data tables and represents additional information about the retrieved documents. There are two categories of additional information that could be visualized: visualization of document attributes, and visualization of inter-document similarities. It use predefined document attributes (e.g. title, relevance, date, size, document type, server type), and visualizations that show how the retrieved documents relate to each of the terms used in the query (query terms` distribution).

The next step in the development process was to find visual mappings of the data tables to good visual structures. All available attributes of each document are shown in different columns of the table. Each row shows one document. The user has the possibility to sort each document attribute in an increasing or decreasing order or to customize the table to his personal preferences (e.g. to show only the attributes he is interested in or to rearrange the order of the columns).the important design decision was to use a multiple view approach offering the user the possibility to choose the most appropriate visualization view for his current demand or individual preferences.

In all different views we have made extensive use of different interaction techniques (e.g. direct manipulation, details- on - demand, zooming, dynamic queries, sorting) to give the user control over the mapping of data to visual structures [12].

Briefly each technique breaks down into four data stages, three types of data transformation and four types of within stage operators. The four data stages are: value, analytical abstraction, visualization abstraction, and view, as seen in table 1. Transforming data from one stage to another requires one of the three types of data transformation operators: data transformation, visualization transformation, and visual mapping transformation, as seen in table 2 [13].

Table 1: data stages in the data state model

Stages	Description
Value	The raw data
Analytical abstraction	Data about data, or information, meta data
Visualization abstraction	Information that is visualizable on the screen using a visualization technique
view	The end-product of the visualization mapping, where the user sees and interprets the picture presented to her

Table 2: transformation operators

Processing step	Description
Data Transformation	Generates some forms of analytical abstraction from the value (usually by extraction)
Visualization Transformation	Takes an analytical abstraction and further reduces it into some form of visualization abstraction, which is visualizable content.
Visual Mapping Transformation	Takes information that is in a visualizable format and presents a graphical view.

Distributed Crawler, distributed search engine, personalization

Distributed search is a search engine model in which the tasks of Web crawling, indexing and query processing are distributed among multiple computers and networks. Originally, most search engines were supported by a single supercomputer. In recent years, however, most have moved to a distributed model. Google search, for example, relies upon thousands of computers are crawling the Web from multiple locations all over the world. Our proposed distributed crawler is in detail in the next section.

In Google's distributed search system, each computer involved in indexing crawls and reviews a portion of the Web, taking a URL and following every link available from it (minus those marked for exclusion). The computer gathers the crawled results from the URLs and sends that information back to a centralized server in compressed format. The centralized server then coordinates that information in a database, along with information from other computers involved in indexing.

When a user types a query into the search field, Google's domain name server (DNS) software relays the query to the most logical cluster of computers, based on factors such as its proximity to the user or how busy it is. At the recipient cluster, the Web server software distributes the query to hundreds or thousands of computers to search simultaneously. Hundreds of computers scan the database index to find all relevant records. The index server compiles the results, the document server pulls together the titles and summaries and the page builder creates the search result pages.

Some projects, such as Wikia Search (formerly Grub) are moving towards an even more decentralized search model. Similarly to distributed computing projects such as SETI@home , many distributed search projects are supported by a network of voluntary users whose computers run client software in the background [17].

5- Distributed crawler on client machine

The challenging task of indexing the web (usually referred as web-crawling) has been heavily addressed in research literature. However, due to the current size, increasing rate, and high change frequency of the web, no web crawling schema is able to pace with the web. While current web crawlers managed to index more than 3 billion documents, it is estimated that the maximum web coverage of each search engine is around 16% of the estimated web size [14]. Distributed crawling was proposed to improve this situation [19]. This has following benefits: (1) increased resource utilization, (2) effective distribution of crawling tasks with no bottle necks, (3) Configurability of the crawling tasks [14].

The paper describes the design and implementation of a crawler on client machine and delivery of the information from a web browser to search engine's central database, and preprocessing, storage and retrieval of the information at the central location. The crawler scales to (at least) several hundred pages per second, is resilient against system crashes and other events, and can be adapted to various crawling applications. We present a new model and architecture of the Web Crawler using multiple HTTP connections to WWW. The multiple HTTP connection is implemented using multiple threads and asynchronous downloader module so that the overall downloading process is optimized. Unloads search engine's crawling task to the millions of client machines that continuously scour the web, allows using processing power of these remote machines to extract the information from a web site that is being currently visited by a web browser. Since the extraction of information from visited pages is occurring in the web browser, there is no need to store these pages on the central location computers of the search engine. Thus, the proposed approach may significantly alleviate three difficult problems of retrieval of information from the web – insufficient efficiency to harvest information from the web by crawlers, enormous requirements for storage of harvested pages, and requirements for processing power to extract the information from the pages.

The new model for the process of information retrieval from the web has the process consisting of the three major conceptual stages: information harvesting by a web browser at a client's location, delivery of the information from a web browser to search engine's central database, and preprocessing, storage and retrieval of the information at the central location.

The user specifies the start URL from the GUI provided. It starts with a URL to visit. As the crawler visits the URL, it identifies all the hyperlinks in the web page and adds them to the list of URLs to visit, called the crawl frontier. URLs from the frontier are recursively visited and it stops when it reaches more than five levels from every home pages of the websites visited and it is concluded that it is not necessary to go deeper than five levels from the home page to capture most of the pages actually visited by the people while trying to retrieve information from the internet. The web crawler system is designed to be deployed on a client computer, rather than on mainframe servers which require a complex management of resources, still providing the same information data to a search engine as other crawlers do, discuss the performance bottlenecks, and describe efficient techniques for achieving high performance [14, 15, and16].

The proposed distributed crawler

Crawlers consume resources: network bandwidth to download pages, memory to maintain private data structures in support of their algorithms, CPU to evaluate and select URLs, and disk storage to store the text and links of fetched pages as well as other persistent data.

A crawler for a large search engine has two major components, see figure 4. First, it has to have a good crawling strategy i.e. a strategy to decide which pages to download next, it called crawling application. Second, crawling system, it needs to have a highly optimized system architecture that can download a large number of pages per second while being robust against crashes, manageable, and considerate of resources and web servers.

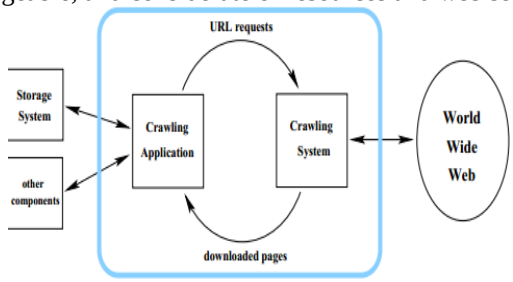


Figure 4: two basic component of crawler

A model of a crawler on the client side with a simple PC, which provides data to any search engines as other crawler provide. To retrieve all webpage contents, the HREF links from every page will result in retrieval of the entire web's content

1. Start from a set of URLs
2. Scan these URLs for links
3. Retrieve found links
4. Index content of pages
5. Iterate

The crawler designed has the capability of recursively visiting the pages. The web pages retrieved is checked for duplication i.e. a check is made to see if the web page is already indexed if so the duplicate copy is eliminated. This is done by creating a data digest of a page (a short, unique signature), then compared to the original signature for each successive visit as given in figure 5. From the root URL not more than five links are visited and multiple seed URLs are allowed. The indexer has been designed to support HTML and plain text formats only. It takes not more than three seconds to index a page. Unusable filename characters such as "?" and "&" are mapped to readable ASCII strings. The WWW being huge, the crawler retrieves only a small percentage of the web.

```
Input: Start URL.            ; say u
1. Q = {u}. { assign the start URL to visit}
2. while not empty Q do
3. Dequeue u ∈Q
4. Fetch the contents of the URL asynchronously.
5. I = I ∪ {u} {Assign an index to the page visited
and pages indexed are considered as visited}
    6. Parse the HTML web page downloaded for
text and other links present. {u1, u2, u3, ...}
    7. for each {u1, u2, u3, …} ϵ u do
        8. if u1 ∉I and u1 ∉Q then
        9. Q = Q ∪{u1}
        10. end if
    11. end for
12. end while
```

Figure 5: web crawler algorithm

We have considered two major components of a crawler - collecting agent, and searching agent. The collecting agent downloads web pages from the WWW and indexes the HTML documents and storing the information to a database, which can be used for later search. Collecting agent includes a simple HTML parser, which can read any HTML file and fetch useful information, such as title, pure text contents without HTML tag, and sub-link. The searching agent- searching agent is responsible for accepting the search request from user, searching the database and presenting the search results to user. When the user initiates a new search, database will be searched for any matching results, and the result is displayed to the user, it never searches over WWW but it searches the database only. A high level architecture of a web crawler has been analyzed as in figure 6 for building web crawler system on the client machine. Here, the multi-threaded downloader downloads the web pages from the WWW, and using some parsers the web pages are decomposed into URLs, contents, title etc. The URLs are queued and sent to the downloader using some scheduling algorithm. The downloaded data are stored in a database.

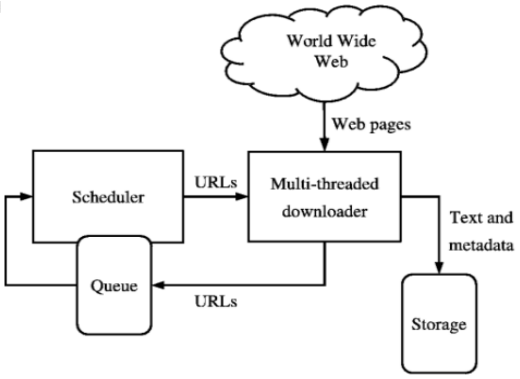


Figure 6: High-level architecture of a standard Web crawler.

Soft ware architecture

The architecture and model of our web crawling system is broadly decomposed into four stages. The figure 7 depicts the flow of data from the World Wide Web to the crawler system. The user gives a URL or set of URL to the scheduler, which requests the downloader to download the page of the particular URL. The downloader, having downloaded the page, sends the page contents to the HTML parser, which filters the contents and feeds the output to the scheduler. The scheduler stores the metadata in the database. The database maintains the list of URLs from the particular page in the queue. When the user request for search, by providing a keyword, it's fed to the searching agent, which uses the information in the storage to give the final output.

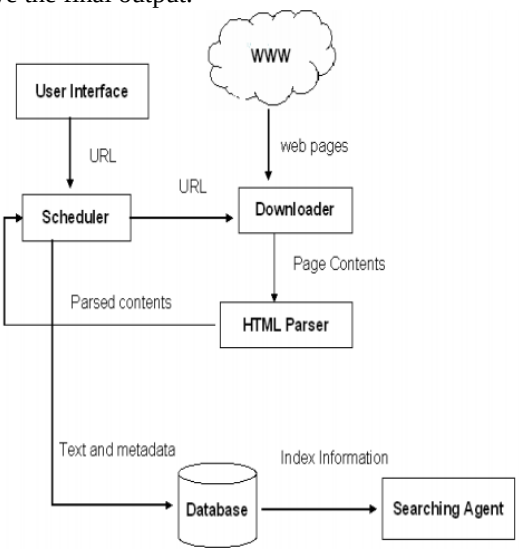


Figure 7: Software Architecture

1. HTML parser

We have designed a HTML parser that will scan the web pages and fetch interesting items such as title, content and link. Other functionalities such as discarding unnecessary items and restoring relative hyperlink (part name link) to absolute hyperlink (full path link) are also to be taken care of by the HTML parser. During parsing, URLs are detected and added to a list passed to the downloader program. At this point exact duplicates are detected based on page contents and links from pages found to be duplicates are ignored to preserve bandwidth. The parser does not remove all HTML tags. It cleans superfluous tags and leaves only document structure. Information about colors, backgrounds, fonts is discarded. The resulting file sizes are typically 30% of the original size and retain most of the information needed for indexing.

2. Creating an efficient multiple HTTP connection

Multiple concurrent HTTP connection is considered to improve crawler performance. Each HTTP connection is independent of the other so that the connection can be used to download a page. A downloader is a high performance asynchronous HTTP client capable of downloading hundreds of web pages in parallel. We use multi-thread and asynchronous downloader. We use the asynchronous downloader when there is no congestion in the traffic and are used mainly in the Internet-enabled application and activeX controls to provide a responsive user-interface during file transfers. We have created multiple asynchronous downloaders, wherein each downloader works in parallel and downloads a page. The scheduler has been programmed to use multiple threads when the number of downloader object exceeds a count of 20.

3. Scheduling algorithm

As we are using multiple downloaders, we propose a scheduling algorithm to use them in an efficient way. The design of the downloader scheduler algorithm is crucial as too many objects will exhaust many resources and make the system slow, too small number of downloader will degrade the system performance. The scheduler algorithm is as follows:

System allocates a pre-defined number of downloader objects (20 in our experiment).

User input a new URL to start crawler.

If any downloader is busy and there are new URLs to be processed, then a check is made to see if any downloader object is free. If true assign new URL to it and set its status as busy; else go to 6.

After the downloader object downloads the contents of web pages set its status as free.

If any downloader object runs longer than an upper time limit, abort it. Set its status as free.

If there are more than predefined number of downloader (20 in our experiment) or if all the downloader objects are busy then allocate new threads and distribute the downloader to them.

Continue allocating the new threads and free threads to the downloader until the number of downloader becomes less than the threshold value, provided the number of threads being used be kept under a limit.

Go to 3.

This mode is using a least amount of resources on the client machine. When a browser-crawler is site oriented then it crawls in a background the site that the user pointed a browser to. The crawling is performed while the user is viewing already downloaded page, it stops when the user points to a different page, and resumes when the page requested by a user is downloaded and can be viewed by

the user. Hence, the crawling is transparent to a user and the only difference besides sending of the downloaded pages to the central location is that the client's CPU cycles are utilized that would be wasted with conventional browser. When the site is completely crawled then a browser-crawler continues breadth-first search (BFS) crawling of sites connected to the one that was completely harvested until a user points the browser to a new web site.
The immediate benefit to the user would be a cache of the crawled pages. Since it is likely that a user will want to view more than one page of a visited web site then his/her surfing experience will be enhanced by loading pages faster from the cache.

4. Storing the web page information in a database

After the downloader retrieves the web page information from the internet, the information is stored in a database. The information harvesting that is performed by a web browser of a user surfing the web can also involve information extraction. This means that text of the downloaded page is processed to extract words that will be part of the index of the search index at the central location. The only thing that is different from conventional browser is that the downloaded pages are sent to the central location.
The delivery of the information from a web browser to search engine's central database can either be mandatory one-way communications or it could be regulated delivery using two-way communications. When using mandatory delivery client-crawler sends information extracted from each page to the central location unless the page is revisited during the same session and is the same as the one in the cache. When using regulated delivery, client-crawler sends a single request per site to the central location with the URL of the site that the user points to. The central location checks in the database to determine whether the site needs to be crawled (if the site has never been indexed before or if the information has not been updated recently), and sends response to the browser/client-crawler. If the site needs to be crawled then the browser-crawler continues with the BFS crawl. If the site has already been indexed recently, then browser-crawler starts crawling sites that are connected to the current one (if the central location indicates that these sites need to be crawled) until the user points the browser to a different site. This scheme avoids sending duplicate information and generating associated unnecessary traffic that would be resultant of situation when many users visit the same popular sites.
Another opportunity for optimization may be capability of the central location to direct client-crawlers to crawl web sites of interest while user browses a web site that is already completely harvested by other browser-crawlers.

Evaluation

The current model for information retrieval from the web is found to be flawed and inefficient:
A significant portion of the web cannot be accessed by crawlers and, thus, is not available to retrieve information from.
Crawlers have relatively low harvesting rate that translates into low refresh rate for indexed pages that leads to 'stale' pages that are out of date.
Crawlers generate huge amount of traffic that impedes useful communications
Crawlers require significant resources such as enormous computer farms to harvest and store pages.
The new model for information retrieval from the web has been proposed where web browsers are used as main tool to harvest web pages and extract information from the pages. The extracted information is sent by the browser-crawlers to a central location where it is indexed, stored, and is accessible for retrieval by end users. Traditional crawlers are used as an auxiliary tool to harvest pages from portions of the web that are not currently being harvested by browser-crawlers. Given a large user base of browser-crawlers the new model can provide the following benefits:
Our crawling system which can be deployed on the client machine to browse the web concurrently and autonomously, it combines the simplicity of asynchronous downloader and the advantage of using multiple threads.
It reduces the consumption of resources as it is not implemented on the mainframe servers as other crawlers also reducing server management. The proposed architecture uses the available resources efficiently to make up the task done by high cost mainframe servers.
The coverage of the web can be significantly improved where browser-crawlers can harvest pages that could not be retrieved by traditional crawlers such as static HTML pages that are not reachable to traditional crawlers, dynamic pages that require user interaction, and pages that are prohibited to crawlers
The harvesting rate can be significantly improved given a large user base of browser-crawlers that may lead to decreased level of 'staleness' of the indexed pages.
The new model provides near real time dynamic view of the usage of the web providing wealth of information about web usage patterns and statistics.
The new model may significantly speed up discovery of new web sites since during deployment of a new web site its web pages are accessed with browser-crawlers to test the site.
The new model may allow reduction in resources required for the process of information retrieval since the tasks of harvesting pages from the web and extracting information from the pages is off loaded to computers hosting browser-crawlers

The use of browser as a crawler may noticeably improve its user's web-surfing experience by providing a large cache of harvested pages.

Personalization

In the modern Web, as the amount of information available causes information overloading, the demand for personalized approaches for information access increases. Personalized systems address the overload problem by building, managing, and representing information customized for individual users. This customization may take the form of filtering out irrelevant information and/or identifying additional information of likely interest for the user.

In the paper, the user can get the personalization benefits in customize his search or his profile, or in indirect way by displaying the suitable visualization technique according his resource's ability. In the following sections, the paper describes how user profile improves retrieval process and helps other users. Section 1 explains what user profile, discusses user profiles specifically designed for providing personalized information access. Section 2 handle regional crawler that not conflict with distributed crawler, and don't mean boundaries with regional word.

There are a wide variety of applications to which personalization can be applied and a wide variety of different devices available on which to deliver the personalized information. Early personalization research focused on personalized filtering and/or rating systems for e-mail, electronic newspapers, Use net newsgroups, and Web documents. More recently, personalization efforts have focused on improving navigation effectiveness by providing browsing assistants, and adaptive Web sites. Because search is one of the most common activities performed today, many projects are now focusing on personalized Web search.

Most personalization systems are based on some type of user profile, a data instance of a user model that is applied to adaptive interactive systems. User profiles may include demographic information, e.g., name, age, country, education level, etc, and may also represent the interests or preferences of either a group of users or a single person. Personalization of Web portals, for example, may focus on individual users, for example, displaying news about specifically chosen topics or the market summary of specifically selected stocks, or a groups of users for whom distinctive characteristics where identified, for example, displaying targeted advertising on e-commerce sites.

In order to construct an individual user's profile, information may be collected explicitly, through direct user intervention, or implicitly, through agents that monitor user activity. Although profiles are typically built only from topics of interest to the user, some projects have explored including information about non-relevant topics in the profile. In these approaches, the system is able to use both kinds of topics to identify relevant documents and discard non-relevant documents at the same time.

Profiles that can be modified or augmented are considered dynamic, in contrast to static profiles that maintain the same information over time. Dynamic profiles that take time into consideration may differentiate between short-term and long-term interests. Short-term profiles represent the user's current interests whereas long-term profiles indicate interests that are not subject to frequent changes over time. For example, consider a musician who uses the Web for her daily research. One day, she decides to go on vacation, and she uses the Web to look for hotels, airplane tickets, etc. Her user profile should reflect her music interests as long-term interests, and the vacation-related interests as short-term ones. Once the user returns from her vacation, she will resume her music-related research, and the vacation information in her profile should eventually be forgotten. Because they can change quickly as users change tasks, and less information is collected, short-term user's interests are generally harder to identify and manage than long-term interests. In general, the goal of user profiling is to collect information about the subjects in which a user is interested, and the length of time over which they have exhibited this interest, in order to improve the quality of information access and infer user's intentions.

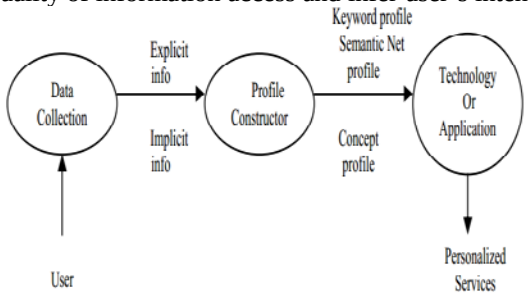


Figure 8: overview of user-profile – based personalization

As shown in Figure 8, the user profiling process generally consists of three main phases. First, an information collection process is used to gather raw information about the user. The second phase focuses on user profile construction from the user data. The final phase, in which a technology or application exploits information in the user profile in order to provide personalized services.

2- Regional distributed crawler

2.1- Regional Crawler Method

In this method, the crawling strategy is based on users' interests and needs in certain domains. These needs and interests are determined according to common characteristics of the users like geographical location, age,

membership and job. Regional Crawler uses these interests as basic data for crawling strategy. In the other words, people in the same region are more likely to search for similar subjects and ignore the other categories that may be important for people in other areas. For example, people in Iran are usually searching for information about soccer and Middle East news, but in the U.S users are more likely to search for baseball events. Even people in a CS department usually look for similar information, (computer science articles for example), so the region could even be defined as small as a LAN. The more a document contains common interests of different domains; the more is its chance for getting crawled.

2.2- Searching and user profile:

The Architecture of most Agent-Based search is based on a Three-Layer Model. The main idea of this Three-Layer model is to divide the internet structure into three layers and devote some particular activities to each layer.

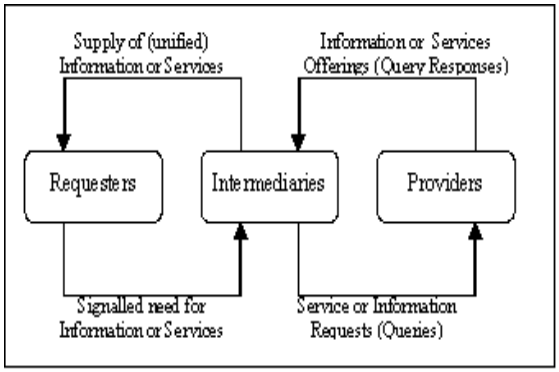


Figure 9: Three Layer Architecture

According to figure 9, the requesters are the users who enter the query into the system and have an individual unique user profile. User profile contains the user interests and the results of previous searches. Providers section also contains the services and information of the providers that are being searched for pages related to users' queries. Intermediaries are responsible for matching the users' requests with the information available from the providers or information which have become accessible by the other users according to their user profile.

2.3- Regional Crawler and personalization benefits

As we explained before, current distributed and Agent-Based search engines are usually constructed based on a Three-Layer structure. Generally the main structure of a Personal Agent in most of Agent-Based search engines is just like figure 10.

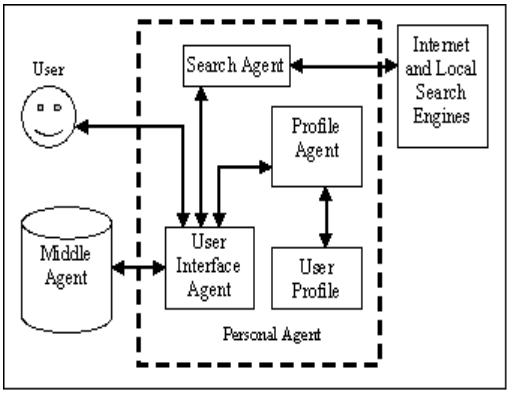


Figure 10: Personal Agent in Agent-Based Search Engines

Search Agent will search the internet and User Interface Agent acts as an interface between the user and the whole Personal Agent structure and enables the collaboration between the user and Agents for searching and entering the queries. Middle Agent plays the most important role in this architecture. It's a bridge between users and providers in the way that providers would announce the services that they provide and users will ask for their needs on the other hand and the Middle Agent would act as a Match Maker between those groups. The Advantage of this methodology is that some user profiles would be devoted to the users which show their needs, interests and past search results. When a query entered by a user and got ready for the search. Middle Agent would get the query and specify the subject of it, then it would search through the user profiles and User Agents to find a similar user according to the public profiles and get information from its past search results for the same or similar queries and send these results as an answer for the user who has entered the query. By adding the Regional Crawler as a Regional Agent to the above architecture we would have Figure 11:

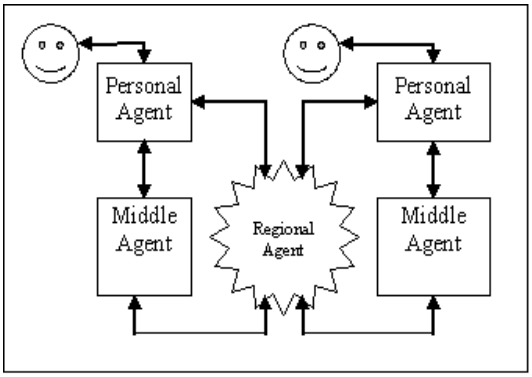


Figure 11: Regional Crawler in Agent-Based Architecture

Regional Agent is responsible for collecting the users' interests of a specific region. Users with similar User Profile will be gathered together by Reinforcement Learning methods or Supervised learning depending on the Middle-Agent architecture. Regional Agent will search for users with similar interests and gather them in a unique public agent. Then we devote a special crawler for each regional agent and ask it to crawl the web in the way that it can satisfy the users' interests. This means that the crawler should look for the pages related to region interests before the other topics available on the web. Since the crawlers are in cooperation with Search Agents, Regional Agent will ask Search Agents to update the important web pages (look for important new pages) by announcing the user interests and needs to them. The important point in this architecture is that by implementing the RL methods, regions domain will be unlimited and as an example two Fans of a particular soccer club in two different locations of the world would be in the same region. So by adding a regional Agent to the Architectures above, we expect the important pages from the users (user agents) perspective become updated more frequently [14].

Conclusion

Search engine and web information retrieval field still in developing circle not stopped at any station until user's need not curb and World Wide Web expand. Despite new and effective solutions for web information retrieval system are suitable and solved many problems in the past, now, they consider bad and generated problems. For example, WordNet tool that widely used in proposed system, it not suitable for proximity search. The proposed system handle some problems such as: low precision and recall, lack of personalization of information and limited customization to individual users, vocabulary, user search behavior, query formulation, information overload, speed, resources consuming. We will continue to wait problems of new solutions and newer solutions for the previous.

References

[1] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, "modern information retrieval", ISBN 0-201-39829-X, chapter 10 " user interfaces and visualization" pages 257:

[2] Marti Hearst, "search user interfaces" ,ISBN 9780521113793, copyright @ 2009, chapter 3 "models of information seeking process".

[3] Djoerd Hiemstra, "information retrieval models" published in Goker, A., and Davies, J. " Information Retrieval: Searching in the 21st Century. John Wiley and Sons, Ltd., ISBN-13: 978-0470027622, November 2009.

[4] Changwoo Yoon, "Domain – specific knowledge-base information retrieval model using knowledge reduction", a dissertation presented to the graduated school of the university of Florida in partial fulfillment of the requirements for the degree of doctor of philosophy, 2005.

[5] Jyotsna Gharat, Jayant Gadge, "web information retrieval using WordNet", international journal of computer applications (0975-8887), volume 56 – no.13, October 2012.

[6] Diana Inkpen, "Information Retrieval on the Internet", 2008

[7] usabilityetc.com/articles/information-retrieval-concept-matching/, last visit May 28, 2013

[8] Chia-Hui Chang, Ching-Chi Hsu, "Integrating Query Expansion and Conceptual Relevance Feedback for Personalized Web Information Retrieval", Computer Networks volume 30(1-7):621-623.

[9] Queryexpansion,p://en.wikipedia.org/wiki/Query_expansion, last visit June 1, 2013

[10] WordNet, http://en.wikipedia.org/wiki/Wordnet, last visit June 3, 2013

[11] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, "Introduction to Information Retrieval", ISBN: 0521865719, Cambridge University Press. 2008

[12] Harald Reiterer, Thomas M. Mann, "Visual Information Retrieval for the Web", Proceedings of the Ninth International Conference on Human-Computer Interaction, 2001.

[13] Ed H. Chi, "A Taxonomy of Visualization Techniques using the Data State Reference Model", Proceeding INFOVIS '00 Proceedings of the IEEE Symposium on Information Visualization 2000 page 69, ISBN:0-7695-0804-9

[14] Pirooz Chubak, Milad Shokouhi, "Designing a Regional Crawler for Distributed and Centralized Search Engines" ,http://ausweb.scu.edu.au/ aw04/papers/refereed/ shokouhi/ paper.html, last visit June 13, 2013

[15] Vladislav Shkapenyuk, Torsten Suel, " Design and Implementation of a High-Performance Distributed Web Crawler"

[16] Rajashree Shettar, Dr. Shobha G, "Web Crawler On Client Machine", Proceedings of the International Multi-Conference of Engineers and Computer Scientists 2008 Volume II IMECS 2008, 19-21 March, 2008, Hong Kong.

[17] Margaret rouse, "distributed search" april 2008, http://whatis.techtarget.com/definition/distributed-search", last visit June 23, 2013

[18] Mamoon H. Mamoon, Hazem M. El-Bakry, Amany A. Salama, " Interactive Visualization of Retrieved Information", International Journal of Knowledge Engineering and Research, Vol 2 Issue 4 April 2013 ,ISSN 2319 – 832X

[19] Odysseas Papapetrou, George Samaras, "Minimizing the Network Distance in Distributed Web Crawling", Springer-Verlag Berlin Heidelberg 2004, LNCS 3290, pp. 581–596, 2004.