Islamic University of Gaza
Deanery of Post Graduate Studies
Facultyof Information Technology

A Personalized Context-Dependent Web Search Engine Using Word Net (Sama Search Engine)

By:

Samira M. Y. Esbitan

Supervised By:

Dr. Tawfiq SM Barhoom

A Thesis Submitted as Partial Fulfillment of the Requirements
for the Degree of Master in Information Technology

1433 H – March 2012

Abstract

It is a fact that the growth of information resources on the WWW is increasing at every moment. So, it has increasingly become difficult for users to find information satisfies their individual needs. The current search engines are still immature to serve the exact desires of the enormous different users. One problem is that they do not consider the context of queries during searching process. Because of that, a lot of non-relevant results may be retrieved.

In this work, we propose a personalized context-dependent web search engine model titled "Sama Search engine". Various steps are involved in the approach: search results collection, preprocessing submitted query and collected results, concepts extraction, match and index results, and rank the retrieved results according to match and index flags finally. The main difference between the proposed model (Sama Search engine) and other Personalized Context-Dependent Search Engine models that it provides a new algorithm for matching and indexing the concepts extracted from both the submitted query and returned results. Also, effectiveness measures are used to evaluate the search engine. The F-measure obtained by the proposed model achieves with 99.35%. A comparative study between our proposed model and Semantic Tree (ST) model has been conducted. The results show that our proposed system outperforms the ST model.

Keywords: *Web Search, Semantic Trees, Fuzzy Logic, Users' Preferences, Context, Open Directory Project.*

عنوان البحث:

محرك بحث الويب المخصص اعتماداً على سياق الجمل باستخدام وورد نت

(محرك البحث سمـــا)

ملخص:

إنها لحقيقة أن نمو المعلومات على شبكة الانترنت العالمية في تزايد في كل لحظة. هكذا، أصبح من الصعب بشكل متزايد على المستخدمين العثور على المعلومات التي تلبي احتياجاتهم مع النمو المستمر لموارد المعلومات على الشبكة العالمية. العديد من محركات البحث متاحة لمساعدة المستخدمين على استغلال موارد قيمة للغاية في شبكة الانترنت العالمية. ومع ذلك، فإن محركات البحث الحالية لا تزال غير ناضجة لتلبية الاحتياجات الدقيقة لمختلف المستخدمين. إحدى المشكلات هي أن المحركات المتوفرة لا تتعامل مع سياقات الاستعلامات ككل خلال عملية البحث. نتيجة لذلك، تظهر العديد من النتائج عديمة الصلة بالاستعلام المرسل.

في هذا البحث، سنقوم بعرض نموذج محرك بحث الويب المخصص اعتماداً على سياق الجمل (سما). هذا النموذج يتكون من عدة مراحل أهمها: مرحلة جمع نتائج البحث، ومرحلة ما قبل المعالجة، ومرحلة استخراج المفاهيم، ثم مرحلة المطابقة وفهرسة المفاهيم، وأخيراً مرحلة الترتيب واسترجاع النتائج. إن الاختلاف الرئيسي بين النموذج المقترح وغيره من النماذج الخاصة بمحركات البحث الخاصة المعتمدة على سياق الجمل، هو أن نموذج سما يعرض خوارزمية جديدة لمطابقة وفهرسة المفاهيم المستخرجة من الاستعلام المرسل و النتائج المرجعة. في عملية تقييم النموذج، تم استخدام تقييم الفعالية، وقد حصل النموذج المقترح على نسبة فعالية تساوي 99.35%. تم عمل دراسة مقارنة ما بين نموذجنا وبين نموذج الشجرة الدلالية، وقد أظهرت النتائج تفوق النموذج المقترح على نموذج الشجرة الدلالية.

To My Father, Mother, and Sweety Sister…

Om Hamza

ACKNOWLEDGEMENT

To

*my **family** and **friends***

in recognition of their worth

an apology

A feeling bears on itself the scars of its birth; it recollects as a

subjective

emotion its struggle for existence;

it retains the impress of what might have been, but is not.

Dr. Tawfiq SM Barhoom

Process and Reality

and hope

Without their patience, understanding, and support,

the completion of this work

would not have been

possible.

Best Regards

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Dmoz	Directory Mozilla
IR	Information Retrieval
JDK	Java Development Kit
JSON	JavaScript Object Notation
JSP	Java Server Pager
JWNL	Java WordNet Library
NLP	Natural Language Processing
ODP	Open Directory Project
RDF	Resource Description Framework
ST	Semantic Tree

Chapter 1

Introduction

It is a fact that the growth of information on the WWW is increasing at every moment. So, it has become increasingly difficult for users to find information satisfies their individual needs since information resources on the WWW grow continually . Many search engines like GOOGLE[1], YAHOO[2] Search and MSN Live[3] Search are available to help users to exploit extremely valuable resources in the WWW. Despite of that, the current search engines are still immature to serve the exact needs of the enormous different users. One problem is that they simply search keywords separately, but do not consider the contexts of queries. Because of that, a lot of non-relevant results may be resulted. For example, if a user inputs "drawing tables in a document" to MSN Live Search, it will show five useless results relative to the furniture table in the first result page. In fact, to assure the quality of the search results, the exact concept related to a keyword may be determined by the context of the sentence [5].

This chapter provides the statement of current search engine problem and our objectives to deal with it. In addition to that, it shows the significance of this thesis and the scope and limitations of the work. It is also introduce our proposed methodology, and finally, give you a general view of the thesis structure.

1.1 Statement of the Problem

The problem of this research that: the current personalized context-dependent web search engines are still immature to serve the exact needs of the enormous different users with high effectiveness using precision, recall, and fallout measurements.

The sub problems are:

1. How to use WordNet to find semantic relations among the words of submitted query for obtaining the right context of it.

2. How to collect results from other search engines like Google?

3. How to pre-process the submitted query and the collected results?

4. How to extract the content of Open Directory Project (ODP) to categorize the retrieved search results.

5. What would be the used methods to rank the retrieved search results?

6. How to obtain suitable search queries for the experimentations?

[1] http://www.google.com
[2] http://www.yahoo.com
[3] http://www.msn.com/

7. What is the proper approach to develop the search engine?

8. How to evaluate the search engine?

1.2 Objectives

1.2.1 Main Objective

The main objective of this research is to develop a personalized context-dependent web search engine using semantic relations with high effectiveness using precision, recall, and fallout measurements.

1.2.2 Specific Objectives

The specific objectives of this research are:

- To use WordNet 2.1 to find semantic relations between the words of submitted query for obtaining the right context of it.
- To collect results from other search engines like Google as inputs for our search engine.
- To Pre-process the submitted query and the collected results using various tools based on some approaches that will be developed.
- To extract the content of ODP to categorize the retrieved search results.
- To develop a method to rank the retrieved search results.
- To obtain suitable search queries for the experimentations.
- To implement the proposed model.
- To test the capability of the proposed model. Effectiveness measurements as precision, recall, and fallout will be used.

1.3 Significance of the Thesis

The main contributions of this research are:

- Defining the relations between concepts to develop context-dependent web search model will power the area of semantic search.
- Serving the users to retrieve more relevant search results according to the context of the query.

- Saving efforts and time by helping the user to find more related results fit with her/his interest based on the produced search engine.

1.4 Scope and Limitations of the work

There are some limitations which have been considerable during the development of this research as the following:

- The proposed search engine serves only English language.
- Our model depends on WordNet 2.1 to identify the semantic relations between words.
- For web personalization, we focus on the submitted query context, but we don't deal with user preferences.

1.5 Methodology

Many researches and models have been proposed for developing personalized search engines. Personalized search engines still suffer from the disability of identify accurate context for submitted queries which leads to retrieve non relative results to users.

In the proposed model, we shall try to solve the context-dependent of submitted queries based on identifying semantic relations among the terms of submitted query. Five main steps are introduced to build:

1. Search results collection
2. Preprocessing
3. Concepts extraction
4. Matching and indexing
5. Ranking and results retrieving.

In the implementation of the model, Java Server language (JSP) will be used. For the website interface, the user will submit her/his query and the search engine will retrieve the search results.

1.6 Thesis Structure

Chapter two and three provide state of the art and a literary survey of the personalized search engines and current approaches. Chapter four defines in detail the model architecture and the proposed approach including preprocessing, semantic relations extraction, matching and indexing, and ranking. Chapter five includes experiments, the used search engine evaluation techniques and system results. Finally , chapter six presents the conclusions and future works.

Chapter 2
State of the Art

This chapter provides a brief introduction about web search engines, semantic web, personalized web search techniques, the using of Open Directory Project (ODP)[1] to build semantic trees, concepts extraction by WordNet[2], and fuzzy logic which use to model and minimize the effects of uncertainties in rule-based fuzzy logic systems.

2.1. Web Search Engines

A web search engine is designed to search for information on the World Wide Web servers. The search results are generally presented in a list of results often called hits. The information may consist of web pages, images, information and other types of files. Some search engines also mine data available in databases or open directories. [28].

2.1.1 Search Engine Types

The search engines classified according to their types as the following [24][35]:

- **Crawler-Based Search Engines**

Crawler-based search engines use automated software programs to survey and categorize web pages. The programs used by the search engines to access your web pages are called 'spiders', 'crawlers', 'robots' or 'bots'. A spider will find a web page, download it and analyze the information presented on the web page. This is a seamless process. Next , the web page will be added to the search engine's database. Then when a user do a search, the search engine will check its database of web pages for the key words the user searched on to show a list of link results. The results (list of suggested links to go to), are listed on pages by order of which is 'closest' (as defined by the 'bots'), to what the user wants to find online. Crawler-based search engines are constantly searching the Internet for new web pages and updating their database of information with these new or altered pages. Examples of crawler-based search engines are: Google[3] and Ask Jeeves[4].

- **Directories**

A 'directory' uses human editors who decide to what category the site belongs; they place websites within specific categories in the 'directories' database. The human editors check the website comprehensively and rank it, based on the information they find, using a pre-defined set of rules. Good examples of Directories are: Yahoo Directory[5] and Open Directory[6].

[1] http://www.dmoz.org/
[2] http://wordnet.princeton.edu/
[3] http://www.google.com
[4] http://www.ask.com/
[5] http://www.yahoo.com
[6] http://www.dmoz.org/

- **Hybrid Search Engines**

Hybrid search engines use a combination of both crawler-based results and directory results. More and more search engines these days are moving to a hybrid-based model. Yahoo and Google are common examples of hybrid search engines.

- **Meta Search Engines**

Meta search engines take the results from all the other search engines results, and combine them into one large listing. Examples of Meta search engines include: Metacrawler[1] and Dogpile[2].

- **Specialty Search Engines**

Specialty search engines have been developed to cater for the demands of niche areas. There are many specialty search engines, including: Shopping, Local Search, Domain Name Search, and Freeware & Shareware Software Search.

2.1.2 The Work Mechanism of Web Search Engines

A search engine operates by the following order [35]:

1. Web crawling
2. Indexing
3. Searching

Web search engines work by storing information about many web pages, which retrieve from the html itself. These pages are retrieved by a web crawler (sometimes known as a spider) which is an automated web browser following every link on the site. The contents of each page are then analyzed to determine how it should be indexed (for example, words are extracted from the titles, headings, or special fields called meta tags). Data about web pages are stored in an index database to be used later in queries. A query can be a single word. The purpose of an index is to find information as quickly as possible [26][35].

When a user enters a query into a search engine (typically by using keywords), the engine examines its index and provides a listing of best-matching web pages according to its criteria, usually with a short summary containing the title of documents and parts of the text sometimes . The index is formed from the information stored with the data and the method by which the information is indexed. The engine looks for the words or phrases exactly as entered. Some search engines provide an advanced feature called proximity search which allows users to define the distance between keywords. There is also concept-based searching where the research involves using statistical analysis on pages containing the words or phrases you search for [35].

The usefulness of a search engine depends on the relevance of the result set it gives back. While maybe there are millions of web pages include a particular word or phrase, some pages may be more relevant, popular, or authoritative than others. Most search

[1] http://www.metacrawler.com/
[2] http://www.dogpile.com/

engines employ methods to rank the results providing the "best" results first. How a search engine decides which pages are the best matches, and in what order the results should be shown, varies widely from one engine to another. The methods also change over time as Internet usage changes and new techniques evolve. There are two main types of search engine that have evolved: one is a system of predefined and hierarchically ordered keywords that humans have programmed extensively. The other is a system that generates an "inverted index" by analyzing texts it locates. This second form relies much more heavily on the computer itself to do the bulk of the work [35].

Most web search engines are commercial ventures supported by advertising revenue so, some employ the practice of allowing advertisers to pay money to have their listings ranked higher in search results. Those search engines which do not accept money for their search engine results make money by running search related ads alongside the regular search engine results. The search engines make money as soon as someone clicks on one of these ads [35].

2.2 Semantic Web

Tim Berners-Lee described semantic web as "The first step is putting data on the web in a form that machines can naturally understand, or converting it to that form—a web of data that can be processed directly or indirectly by machines." [3]. He crawling towards the first step of semantic web dream has a two-part vision for the future of the web. The first part is to make the web a more collaborative medium. The second part is to make the web understandable, and thus processable, by machines. Tim Berners-Lee's original vision clearly involved more than retrieving Hypertext Markup Language (HTML) pages from web servers[6].

2.2.1 Smart Data Continuum

To achieve Tim Berners-Lee's vision for the future of the web, data had been passed many stages to be more smart. Figure 2.1 shows four stages of the smart data continuum; The four stages in the diagram progress from minimal smart data to data embodied with enough semantic information for machines to make inferences about it. The four stages are as the following [6]:

- **Text and databases (pre-XML):** The initial stage where most data is proprietary to an application. Thus, the "smarts" are in the application and not in the data.

- **XML documents for a single domain:** The stage where data achieves application independence within a specific domain. Now data is smart enough to move between applications in a single domain. An example of this would be the XML standards in the healthcare industry, insurance industry, or real estate industry.

- **Taxonomies and documents with mixed vocabularies:** In this stage, data can be composed from multiple domains and accurately classified in a hierarchical taxonomy. In fact, the classification can be used for discovery of data. Simple

relationships between categories in the taxonomy can be used to relate and thus combine data. Thus, data is now smart enough to be easily discovered and sensibly combined with other data.

- **Ontologies and rules:** In this stage, new data can be inferred from existing data by following logical rules. In essence, data is now smart enough to be described with concrete relationships, and sophisticated formalisms where logical calculations can be made on this "semantic algebra." This allows the combination and recombination of data at a more atomic level and very fine-grained analysis of data. Thus, in this stage, data no longer exists as a blob but as a part of a sophisticated microcosm. An example of this data sophistication is the automatic translation of a document in one domain to the equivalent (or as close as possible) document in another domain.

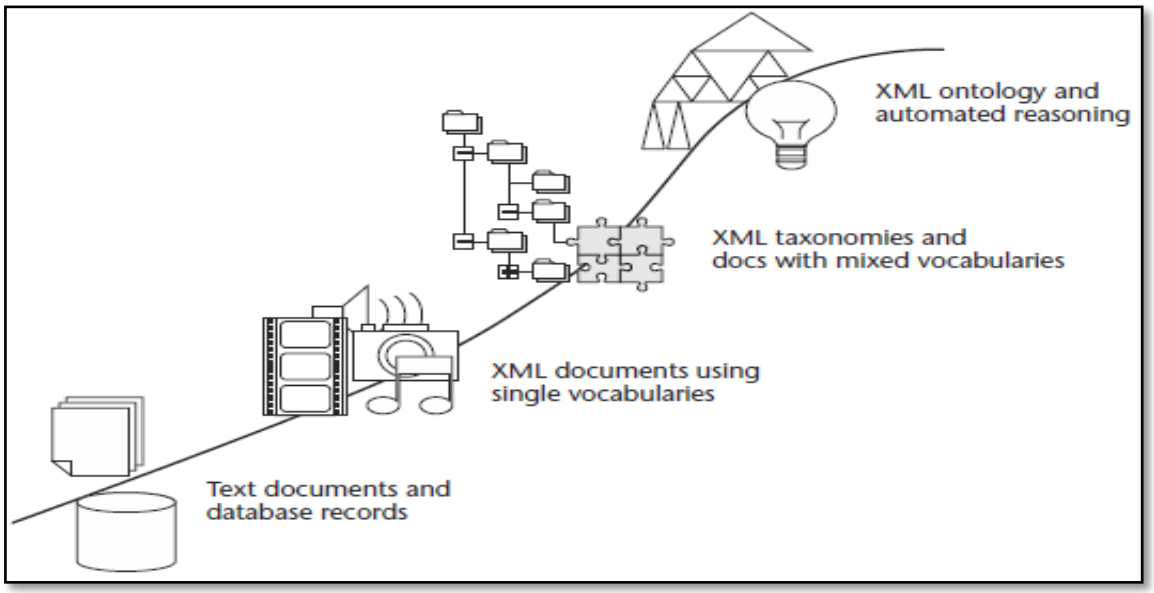

Figure 2.1: The Smart Data Continuum[6]

2.2.2 Resource Description Framework (RDF)

Resource Description Framework is one of the core technologies of the Semantic Web and the current W3C standard to represent data on the web. It is an XML-based language to describe resources. Such a resource is accessed via a Uniform Resource Locator (URL). While XML documents attach meta data to parts of a document, one use of RDF is to create meta data about the document as a standalone entity. In other words, instead of marking up the internals of a document, RDF captures meta data about the "externals" of a document, like the author, the creation date, and type. A particularly good use of RDF is to describe resources, which are "opaque" like

images or audio files. Figure 2.2 displays an application called RDFPic[1], which uses RDF to describe an image resource [6].

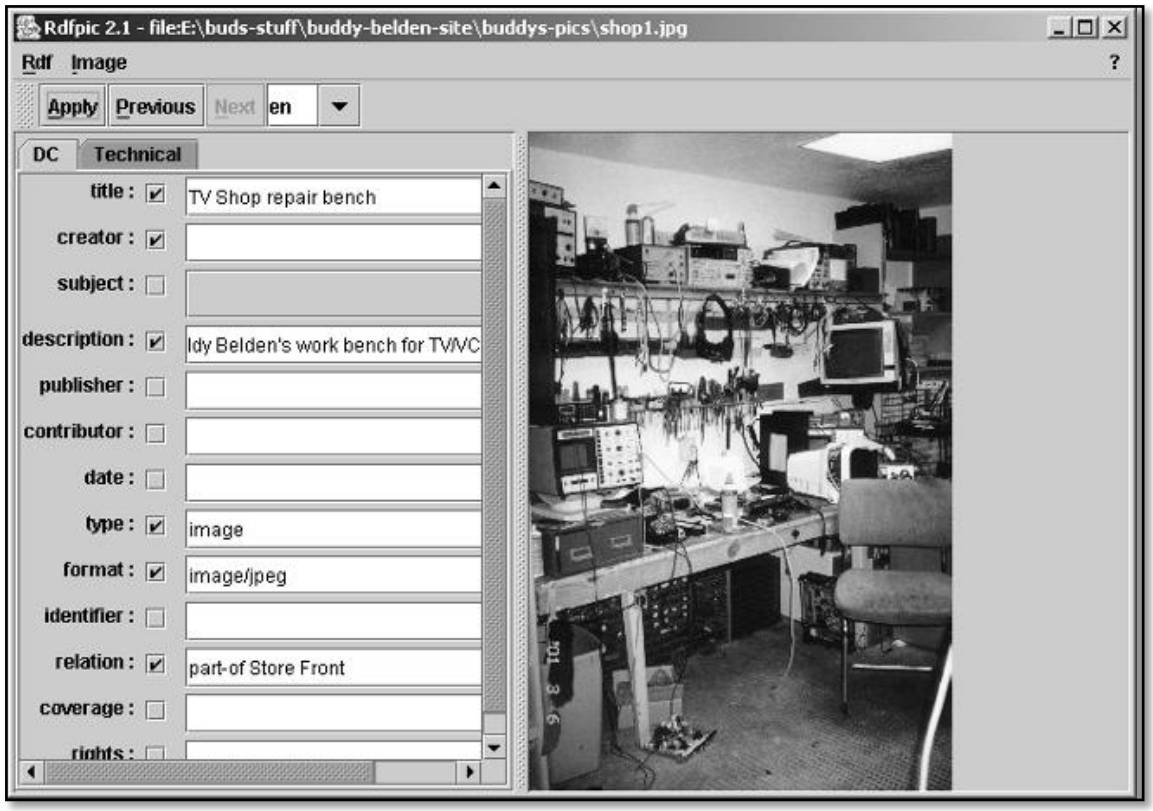

Figure 2.2: RDFPic application describing an image[6]

2.2.3 Ontologies

Ontologies are about vocabularies and their meanings, with explicit, expressive, and well-defined semantics, possibly machine-interpretable [6]. The following is famous Guarino's definition of ontology: [11]

"An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world".

The definition shows that the intention of ontology is to capture, describe knowledge in explicit way, where the domain knowledge is represented as concepts and relationships. As a result, knowledge sharing is attained [36].

Definition 1. Ontology O is defined as:

$$O = <c, p, a>$$

[1] RDFPic is copyrighted by the World Wide Web Consortium. All Rights Reserved. http://www.w3.org/Consortium/Legal/.

Where:

- **c** is a set of concept names ,
- **p** is a set of concept properties ,
- **a** is a set of axioms.

In Definition 1, the axioms are the constraints and rules that are defined on the concepts and properties. There are four types of relationships between the concepts, which are part-of, kind-of, instance-of and attribute-of. In a word, Ontology is a static conceptual model for domain knowledge, which uses terminologies and their relationships agreed upon by wide communities to describe the domain knowledge and its structure [36].

Figure 2.3 shows a simple human resources ontology created in the ontology management tool called Protégé[1]. You'll notice that there are classes such as Person, Organization, and Employee. In an ontology, these are really called concepts, because it is intended that they correspond to the mental concepts that human beings have when they understand a particular body of knowledge or subject matter area or domain, such as the human resources domain [6].

These concepts and the relationships between them are usually implemented as classes, relations, properties, attributes, and values. So what Figure 2.3 depicts primarily are concepts of the important entities of the domain, which are implemented as classes. Examples are Person, Organization, and Employee. Also depicted are the relations between these entity-focused concepts, such as employee_of, managed_by, and manages. Finally, properties or attributes are depicted. Examples include address, name, birthdate, and ssn under the Person class. These properties or attributes have either explicit values or, more often, have value ranges. The value range for the property/attribute of employee_of, a property of the class Employee, for example, is the class Organization. By range we mean that the only possible values for any instances of the property employee_of defined for the class Employee must come from the class Organization [6].

[1] http://protégé.stanford.edu

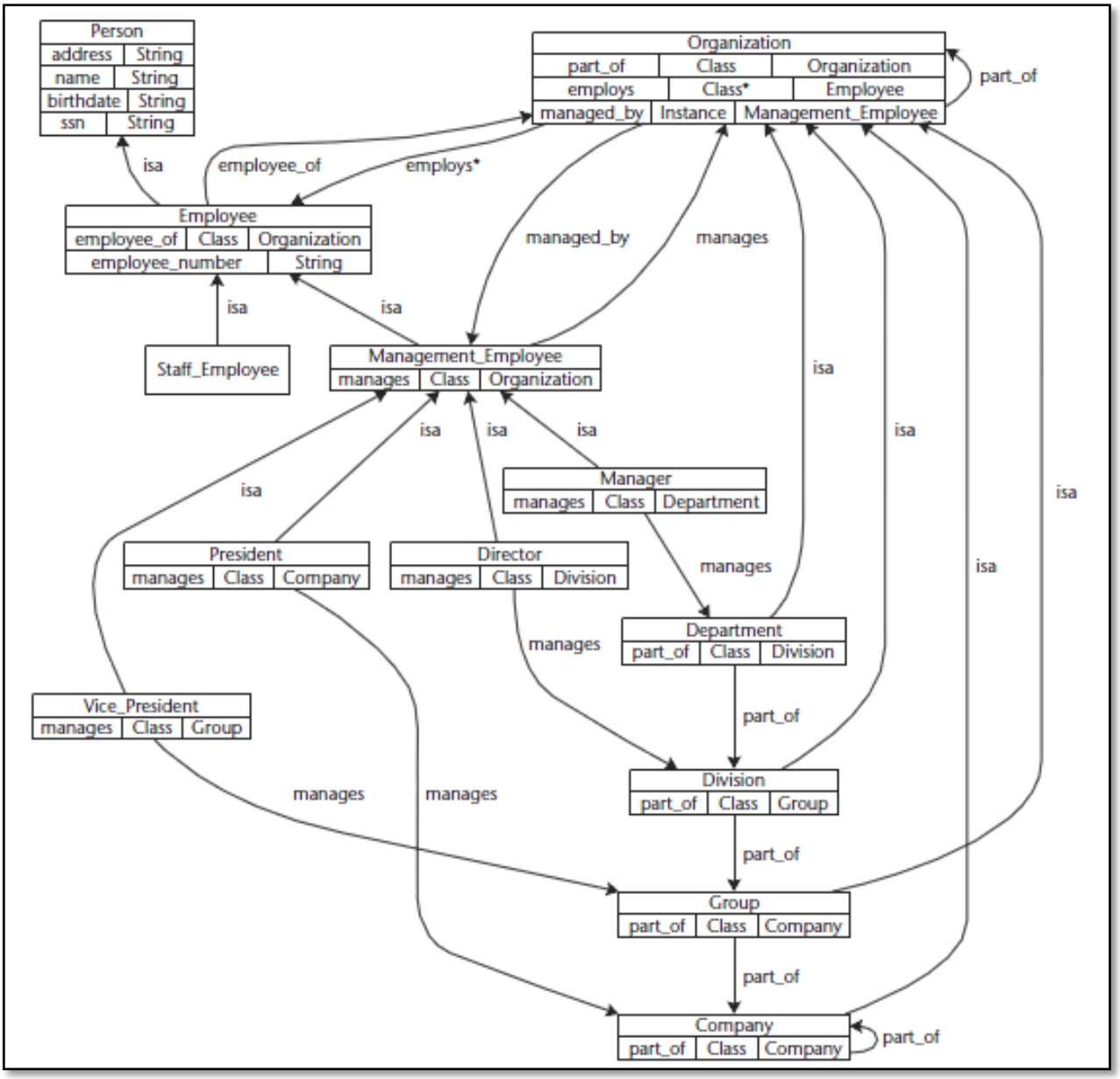

Figure 2.3: Graphical ontology example: Human resources[6]

2.3 Personalized Web Search

With the exponential growth of the available information on the web, a traditional search engine, has difficulty meeting efficiency and effectiveness performance demanded by users searching for relevant information, even it based on sophisticated document indexing algorithms. Users surfing the web in search of resources to satisfy their information needs have less and less time and patience to formulate queries, wait for the results and sift through them. Personalized web environments that build models of short-term and long-term user needs based on user actions, browsed documents or past queries are playing an increasingly crucial role: they form a winning combination, able to satisfy the user better than unpersonalized

search engines based on traditional Information Retrieval (IR) techniques. Recently, several search tools have been developed to tackle the information overload problem in the web. Some make use of effective personalization, adapting the results according to each user's information needs. This contrasts with traditional search engines that return the same result list for the same query, regardless of who submitted the query, in spite of the fact that different users usually have different needs [18].

Users have to undertake three information access paradigms each time they need to meet particular information needs on the web hypertextual environment:

1. Searching by surfing (or browsing)
2. Searching by query
3. Recommendation

Recommendation-based systems suggest items, such as movies, music or products, analyzing what the users with similar tastes have chosen in the past [18].

In browsing, users analyze web pages one at a time, surfing through them sequentially, following hyperlinks. This is a useful approach to reading and exploring the contents of a hypertext, but it is not suitable for locating a specific piece of information. Even the most detailed and organized catalogs of web sites, such as YAHOO! Directory[1] and the Open Directory Project (ODP)[2] do not always allow users to quickly locate the pages of interest. The larger the hypertextual environment is, the more difficulty a user will have finding what he is looking for [18].

The other dominant information access paradigm involves querying a search engine which it is an effective approach that directly retrieves documents from an index of millions of documents in a fraction of a second. This approach is based on an classic Information Retrieval (IR) model wherein documents and information needs are processed and converted into ad-hoc representations. These representations are then used as the inputs to some similarity function that produces the document result list [26].

In the last few years, the trend of the adaptation of traditional IR system to the web environment has been one of the most important issues of the revolution of web personalization. The former task is accomplished by periodically collecting newly-created documents through re-crawling, keeping the search system's internal document index updated.

The two paradigms, searching by query and browsing, coexist: most of the times, browsing is useful when the user does not know beforehand the search domain keywords. Searching by query is considered as the most popular way that users begin seeking information [12, 29] because it allows them to quickly identify pages containing specific information. For this reason, sophisticated search techniques are required,

[1] http://dir.yahoo.com
[2] http://dmoz.org

enabling search engines to operate more accurately for the specific user. Personalized search aims to build systems that provide individualized collections of pages to the user, based on some form of model representing their needs and the context of their activities. Depending on the searcher, one topic will be more relevant than others tailored to the preferences, tastes, backgrounds and knowledge of the user who expressed it [18].

2.4 Open Directory Project (ODP)

The Open Directory Project (ODP), also known as Dmoz (from directory.mozilla.org, its original domain name), is a multilingual open content directory of World Wide Web links. It is owned by Netscape but it is constructed and maintained by a community of volunteer editors. ODP uses a hierarchical ontology scheme to build semantic trees for organizing site listings. Listings on a similar topic are grouped into categories which can then include smaller categories [34].

Instead of fighting the explosive growth of the Internet, the Open Directory provides the means for the Internet to organize itself. As the Internet grows, so do the number of net-citizens. These citizens can each organize a small portion of the web and present it back to the rest of the population, culling out the bad and useless and keeping only the best content [23].

The Open Directory follows in the footsteps of some of the most important editor/contributor projects of the 20th century. Just as the Oxford English Dictionary became the definitive word on words through the efforts of volunteers, the Open Directory follows in its footsteps to become the definitive catalog of the web. The Open Directory was founded in the spirit of the Open Source movement, and is the only major directory that is 100% free. There is not, nor will there ever be, a cost to submit a site to the directory, and/or to use the directory's data. The Open Directory data is made available for free to anyone who agrees to comply with ODP free use license. ODP data is made available through an RDF dump that is published on a dedicated download server, where an archive of previous versions is also available. New versions are usually generated weekly [23].

2.5 WordNet

WordNet [20] is a lexical inheritance ontology gifted with many different pointers that aim to represent some aspects of the semantics of the lexicon, and the relationships of different lexicalized concepts. WordNet is a thesaurus for the English language based on psycholinguistics studies and developed at the University of Princeton [19]. Princetons WordNet has been under construction for over a decade and several versions were proposed. The last version (WordNet 2.1) contains more than 155000 word forms organized in 117597 word meanings [21].

WordNet was conceived as a data-processing resource which covers lexico-semantic categories called synsets. The synsets are sets of synonyms which gather lexical items having similar significances, for example the words “a board” and “a plank” grouped in the synset {board, plank}. But “a board” can also indicate a group of people (e.g., a board of directors) and to disambiguate these homonymic significances

“a board” will also belong to the synset {board, committee}. The definition of the synsets varies from the very specific one to the very general. The most specific synsets gather a restricted number of lexical significances whereas the most general synsets cover a very broad number of significances [21].

The organization of WordNet through lexical significances instead of using lexemes makes it different from the traditional dictionaries and thesaurus [19]. The other difference which has WordNet compared to the traditional dictionaries is the separation of the data into four data bases associated with the categories of verbs, nouns, adjectives and adverbs. This choice of organization is justified by psycholinguistics research on the association of words to the syntactic categories by humans. Each database is organized different from the others. The names are organized in hierarchy, the verbs by relations, the adjectives and the adverbs by N-dimension hyperspaces [19].

The semantic relations available in WordNet listed as the following [19]: (Note: These relations relate to concepts, but the given examples are based on words).

- **Synonymy:** relation binding two equivalent or close concepts (frail /fragile). It is a symmetrical relation.
- **Antonymy:** relation binding two opposite concepts (small /large). This relation is symmetrical.
- **Hyperonymy:** relation binding a concept-1 to a more general concept-2 (tulip /flower).
- **Hyponymy:** relation binding a concept-1 to a more specific concept-2. It is the reciprocal of hyperonymy. This relation may be useful in information retrieval. Indeed, if all the texts treating of vehicles are sought, it can be interesting to find those which speak about cars or motor bikes.
- **Meronymy:** relation binding a concept-1 to a concept-2 which is one of its parts (flower/petal), one of its members (forest /tree) or a substance made of (pane/glass).
- **Metonymy:** relation binding a concept-1 to a concept-2 of which it is one of the parts. It is the opposite of the meronymy relation.
- **Implication:** relation binding a concept-1 to a concept-2 which resulted from it (to walk /take a step).
- **Causality:** relation binding a concept-1 to its purpose (to kill /to die).
- **Value:** relation binding a concept-1 (adjective) which is a possible state for a concept-2 (poor /financial condition).
- **Has the value:** relation binding a concept-1 to its possible values (adjectives) (size /large). It is the opposite of relation value.
- **See also:** relation between concepts have a certain affinity (cold /frozen).
- **Similar to:** certain adjectival concepts which meaning is close are gathered. A synset is then designated as being central to the regrouping. The relation 'Similar to' binds a peripheral synset with the central synset (moist /wet).
- **Derived from:** indicate a morphological derivation between the target concept (adjective) and the concept origin (coldly /cold).

2.6 Fuzzy Logic

Fuzzy logic is a form of many-valued logic; it deals with reasoning that is approximate rather than fixed and exact. In contrast with traditional logic theory, where binary sets have two-valued logic: true or false, fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false. Furthermore, when linguistic variables are used, these degrees may be managed by specific functions. Fuzzy logic began with the 1965 proposal of fuzzy set theory by Lotfi Zadeh. Though fuzzy logic has been applied to many fields, from control theory to artificial intelligence, it still remains controversial among most statisticians, who prefer Bayesian logic, and some control engineers, who prefer traditional two-valued logic [33].

2.6.1 Type-2 Fuzzy Sets

Type-2 fuzzy sets let us model and minimize the effects of uncertainties in rule-based fuzzy logic systems (FLSs). Unfortunately, type-2 fuzzy sets are more difficult to use and understand than type-1 fuzzy sets; hence, their use is not widespread yet . There are (at least) four sources of uncertainties in type-1 FLSs [13]:

1. The meanings of the words that are used in the antecedents and consequents of rules can be uncertain (words mean different things to different people).
2. Consequents may have a histogram of values associated with them, especially when knowledge is extracted from a group of experts who do not all agree.
3. Measurements that activate a type-1 FLS may be noisy and therefore uncertain.
4. The data that are used to tune the parameters of a type-1 FLS may also be noisy.

All of these uncertainties translate into uncertainties about fuzzy set membership functions. Type-1 fuzzy sets are not able to directly model such uncertainties because their membership functions are totally crisp. On the other hand, type-2 fuzzy sets are able to model such uncertainties because their membership functions themselves are fuzzy .Membership functions of type-1 fuzzy sets are two-dimensional, whereas membership functions of type-2 fuzzy sets are three-dimensional. It is the new third-dimension of type-2 fuzzy sets that provides additional degrees of freedom that make it possible to directly model uncertainties [13].

2.7 Semantic Tree Model

Semantic Tree (ST) is a hierarchical tree, which represents concepts and their relations with other concepts. Every concept is represented as one node in the ST. If concept A can be comprised by concept B, then node A is represented as a child node of node B. Depending on the context of queries, the most relevant concept for specific word can be determined by measuring the distances between concepts associated with the word and concepts associated with other words in the same query. After the distances between concepts are gathered, the fuzzy logic is used for calculating the semantic relations between concepts and words [5].

2.8 Search Engine Evaluation

Evaluation is considered as the essential key to make progress in building better search engines. It is also important to understand if a search engine is being used effectively in a specific application. One of the primary distinctions made in the evaluation of search engines is between effectiveness and efficiency. Effectiveness measures the ability of the search engine to find the right information, and efficiency measures how quickly this is done. Effectiveness and efficiency will be affected by many factors such as the interface used to display search results and techniques such as query suggestion and relevance feedback. It is important to mention that information retrieval research focuses on improving the effectiveness of search, and when a technique has been established as being potentially useful, the focus shifts to find efficient implementations [14].

The two most common effectiveness measures, recall and precision, were introduced in the Cranfield studies[1] to summarize and compare search results. Intuitively, recall measures how well the search engine is doing at finding all the relevant documents for a query, and precision measures how well it is doing at rejecting non-relevant documents. The definition of these measures assumes that, for a given query, there is a set of documents that is retrieved and a set that is not retrieved (the rest of the documents). This applies to the results of a boolean search, but the same definition can also be used with a ranked search, as we will see later. If, in addition, relevance is assumed to be binary, then the results for a query can be summarized as shown in Table 2.1. In this table, A is the relevant set of documents for the query, $\overline{A}$ is the irrelevant set, B is the set of retrieved documents, and B is $\overline{B}$ the set of documents that are not retrieved. The operator ∩ gives the intersection of two sets. For example, A ∩ B is the set of documents that are both relevant and retrieved [4].

Table 2.1: Sets of documents defined by a simple search with binary relevance

	Relevant	Non-Relevant
Retrieved	$A \cap B$	$\overline{A} \cap B$
Not Retrieved	$A \cap \overline{B}$	$\overline{A} \cap \overline{B}$

A number of effectiveness measures can be defined using this table. We are particularly interested in the following two equations:

$$Recall = \frac{|A \cap B|}{|A|} \quad \ldots\ldots\ldots eq\ (2.1)$$

$$Precision = \frac{|A \cap B|}{|B|} \quad \ldots\ldots\ldots eq\ (2.2)$$

[1] http://www.cranfieldprecision.com/

When a document is retrieved, it is the same as making a prediction that the document is relevant. From this perspective, there are two types of errors that can be made in prediction (or retrieval). These errors are called false positives (a non relevant document is retrieved) and false negatives (a relevant document is not retrieved). Recall is related to one type of error (the false negatives), but precision is not related directly to the other type of error. Instead, another measure known as fallout, which is related to the false positive errors [4]:

$$Fallout = \frac{|\overline{A} \cap B|}{|\overline{A}|} \quad \ldots\ldots\ldots eq\ (2.3)$$

From equations 2.1, 2.2 and 2.3, we can conclude the following:

- **Recall is** the proportion of relevant documents that are retrieved.
- **Precision** is the proportion of retrieved documents that are relevant.
- **Fallout** is the proportion of non-relevant documents that are retrieved.

Chapter 3

Related Works

Searching is one of the most important task on the Internet. Search engines are the basic tool of the internet, in which the user can collect information related to her/his submitted query. A perfect search engine is the one which should travel through all the web pages in the web and list the related information based on the given keyword by user . In spite of the recent developments on web search technologies, there are still many conditions in which search engine users obtain the non-relevant search results from the search engines. A personalized web search has various levels of efficiency for different users, queries, and search contexts. Although, personalized search has been a major research area for many years and many personalization approaches have been examined, it is still uncertain whether personalization is always significant on different queries for diverse users and under different contexts [28].

In this chapter, firstly we have a look on some personalized web search approaches focusing on user preferences, then we present with some details other personalized web search approaches focusing on the context of submitted query.

3.1 Personalized User Preferences Web Search Approaches

According to the importance of the search engine for both academic and commercial areas, many studies have been done by researchers for improving search precision and providing better results to users. Kraft et. al [2] , Sugiyama et. al [30], Zhengyu Zhu et al [37], and Bounoy and Walairacht [31] have been contributed in enhancing the search results process by proposing techniques for users' preferences.

Kraft et. al [2] presented an approach to use fuzzy logic for creating user profiles in web retrieval applications. According to the additional facility of intelligent constructing systems, the use of rules in the text framework is very extended. The study of the rules has been done from different perspectives, including the management of uncertainty since user information needs have an inherent nature of ambiguity. From their representation and acquisition, the rules have been classified in a general way. However, different representations retrieve different documents and it is difficult to identify the one working best, besides the additional component of context dependence. From a semantic point of view, documents are related to concepts, but documents are formed by words, not by concepts. On one hand, a concept can be expressed by more than one word appearing or not in the document, as well as by a group of words. On the other hand, the same word can be related to different concepts. Human mind works inferring concepts from the words of a document [1]. However, the simulation of this process by computers automatically is not so direct and represents one of the big challenges of the present and future to enhance retrieval systems. The value of the Kraft et. al technique rely on the ability of using it to expand the queries and extract knowledge related to a group of users with common interests [2].

Sugiyama, Hatano and Yoshikawa [30] proposed several approaches in order to adapt search results according to each user's information need . Their approach allows each user to perform a fine-grained search, which is not performed in typical search engines, but rather by capturing changes in each user's preferences. They conducted experiments in order to verify the effectiveness of the approaches:

(1) relevance feedback and implicit approaches,

(2) user profiles based on pure browsing history,

(3) user profiles based on the modified collaborative filtering.

They evaluated the retrieval accuracy of these approaches. The user profile constructed based on modified collaborative filtering achieved the best accuracy. This approach allows them to construct a more appropriate user profile and perform a fine-grained search that is better adapted to each user's preferences. Although, the combination of these approaches can be applied to situations where users require more relevant information to satisfy their information needs, they need to conduct experiments with a greater number of subjects and attempt to improve their approaches using a longer term of the user's browsing history in order to achieve much more adaptive search for each user [30].

Zhengyu Zhu et al., [37] proposed query expansion based on a personalized web search model. It depends on a representation of personalized web search organization. The novel system, as a middleware connecting a user and a web search engine, is fixed on the client machine. It can study the user's favorite implicitly and then produce the user profile automatically. When the user enters query keywords, more personalized expansion words are produced by the proposed approach, then these words in common with the query keywords are forwarded to a famous search engine such as Google. These expansion words can facilitate search engine retrieval information for a user based on his/her implicit search objectives. The novel web search representation can build an ordinary search engine personalized, specifically all the way through personalized query expansion the search engine can provide different search results to different users who enter the equivalent keywords. The experimental observations demonstrate the consequence and use the proposed work for personalized information service of a search engine [37].

Bounoy and Walairacht [31] categorize user preference to build the user profile and general profile base on user's search history and category hierarchy, respectively. Then the search engines use those profiles to determine the interests of each user, execute the search query to obtain a set of relevant documents, and re-ranking the documents in a manner that best reflects their relevance to the user's profile. The user profiles can be constructed automatically from the user's search history and augmented by a general profile which is extracted automatically from a common category hierarchy and used to improve retrieval effectiveness in web search based on a new clickthrough interpretation. The categories that are likely to be of interest to the user are deduced based on his/her query and the two profiles. The experimental results show the accuracy

of using both profiles is consistently better than using the user profile only or only general profile only . The results indicate that using this technique reduces the number of documents retrieved by the search engine into 30% [31].

On the other hand, the current search engines, like GOOGLE and YAHOO Search, adapt to users' requirements by filtering the information that is relevant to the user. The user may explicitly indicate his preferences, or preferences may be inferred automatically from his profile [2, 10, 17].

3.2 Personalized Context-Dependent Web Search Approaches

After user preferences retrieval, context-dependent is the second factor of personalized web search engines. There are many studies proposed by researchers which focusing on context-dependent in order to improve the quality of search. Jie Yu et al [15], and Ohgaya et. al [22], concerned in the other part which focused on the context of the search query.

Jie Yu et al., [15] suggested mining user context based on interactive computing for personalized web search. Personalized web search is a successful way of same query. How to achieve user's real-time information requirement is a key subject in personalized search. Existing approaches focus more on constructing user profile which depends on web pages/documents which influences the effectiveness [9] of search engine. Additionally, dynamics of user profile is frequently ignored. To deal with this problem, the authors have introduced a technique that acquires the user context to perfectly present preferences of users for successful personalized search. Initially, small-term query context is created from web-snippets to take apart role of semantic background of user's search behavior, recognizing associated concepts of the query. Then user context snap is constructed depending on query context based on user's interactive search behavior. Finally, development of user context is taken into account by introducing forgetting factor to combine the independent user context snap in a user session. The experimental outputs completely reveal that this technique can effectively construct user context based on individual user information need [15].

Ohgaya et. al [22] proposed an approach to use Conceptual Fuzzy Set (CFS) model for matching contexts-dependent keywords and the concepts. They developed a navigation system which conceptually matches input keywords and paths using CFSs based on radial basis function (RBF) networks. Taking the meaning of a path into consideration and propagating activations of concepts recursively in CFS unit to associate relative words with input keywords enabled the system to search the path leading to an appropriate category. However, the following are some problems which require further study:

- The scale of the system is small.
- The associations in CFS unit are affected by un-uniformity of the concept base.
- The number of propagation of activation values in CFS unit is empirical.

In addition, due to the fact that numerous combinations of words may appear in queries and documents, it may be difficult to define the relations between concepts in all possible combinations [22].

On other hand, Chen et. al [5] presented a Semantic Tree (ST) model which it is a study combined both, contexts and users' preferences to personalized context-dependent web search agent. Compared to the CFS model proposed by Ohgaya et. al [22], ST model applied the Semantic Tree (ST) to represent the concepts and the relations to other concepts. In the semantic tree, one concept only has the direct relation to its parent node (concept) and children nodes (concept), and the relation between any two concepts can be explored in the tree. Therefore, their search agent can simply define the relations between any two concepts. In addition to that, they applied users' preferences for personalizing search results, then they used the fuzzy logic to determine which factor, semantic relations or users' preferences, will dominate results [5].

ST model [5] applied the following strategy:

1. First, gather the distances between concepts using Semantic Tree (ST), which represents concepts and their relations with other concepts.
2. After the distances between concepts are gathered, the fuzzy logic is used for calculating the relations between concepts and words. The distance membership functions are shown in Figure. 3.1 [5]. The membership functions for semantic relations are shown in Figure. 3.2 [5].

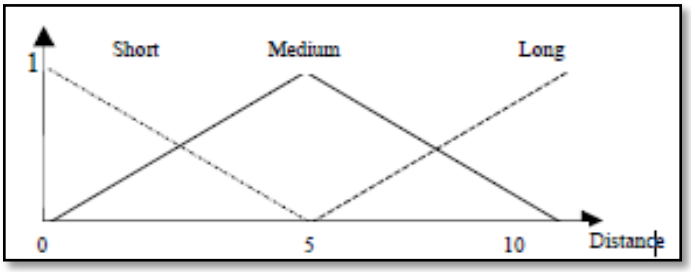

Figure 3.1: Distance Membership Function[5]

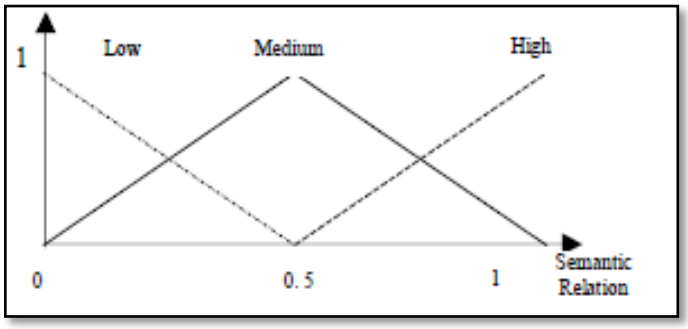

Figure 3.2: Semantic Relations Membership Function[5]

3. Then, use both semantic relations and users' preferences to determine the appropriate concepts associated with queries using query processor agent. The fuzzy rule base is given in Table 3.1. S denotes semantic relations; P denotes preferences; C denotes concept relativity.

Table 3.1: Fuzzy Rule Base for Concept Relativity[5]

	S_Low	S_Medium	S_High
P_Low	C_Low	C_Low	C_Medium
P_Medium	C_Low	C_Medium	C_High
P_High	C_Medium	C_High	C_High

4. Finally, after web pages which are gathered by keyword-based search engines, the page analyzer analyze web pages, and only the associated results returns to the users.

It is worth mentioning that ST model was the main motivator of this thesis . So , its details will be embodied in chapter 5 including the way of implementation , the defects and evaluating its results .

Chapter 4
Sama Search Engine Model

In this chapter, our main view will be the developing of a personalized context-dependent web search engine model using semantic relations. The proposed model which titled "Sama Search Engine" will be described using flowcharts, algorithms, figures and tables. In addition to that, a complete example of Sama Search Engine model will be provided at the end of chapter. Figure 4.1 provides a general view of the architecture of Sama Search Engine model.

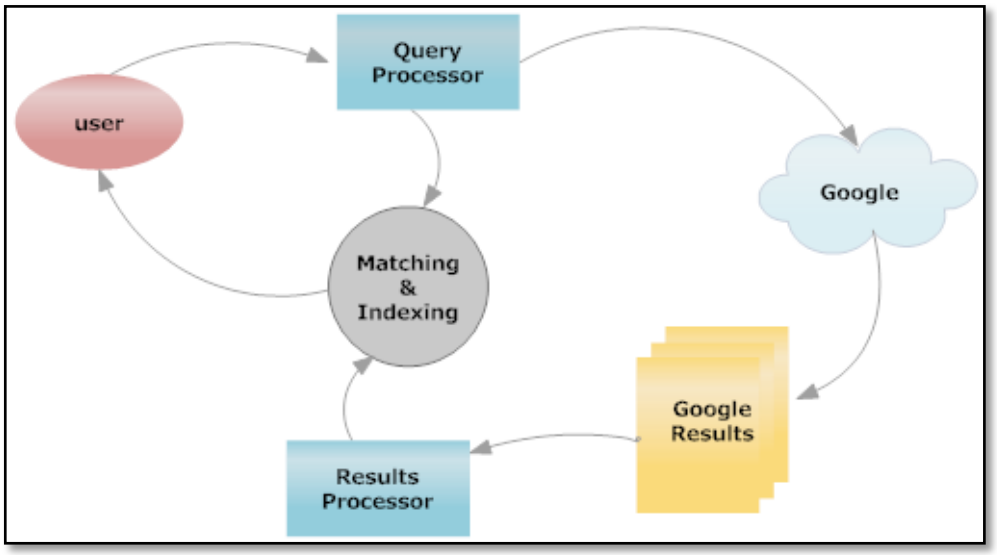

Figure 4.1: Sama Search Engine Architecture

The main requirements of Sama Search Engine listed as the following:

- Achieve a personalized context-dependent search.
- Accuracy.
- Efficiency.

To achieve those requirements, various stages have to be passed. Figure 4.2 shows the main required steps as follow:

1. Search results collection
2. Preprocessing
3. Concepts extraction
4. Matching and indexing
5. Ranking and results retrieving.

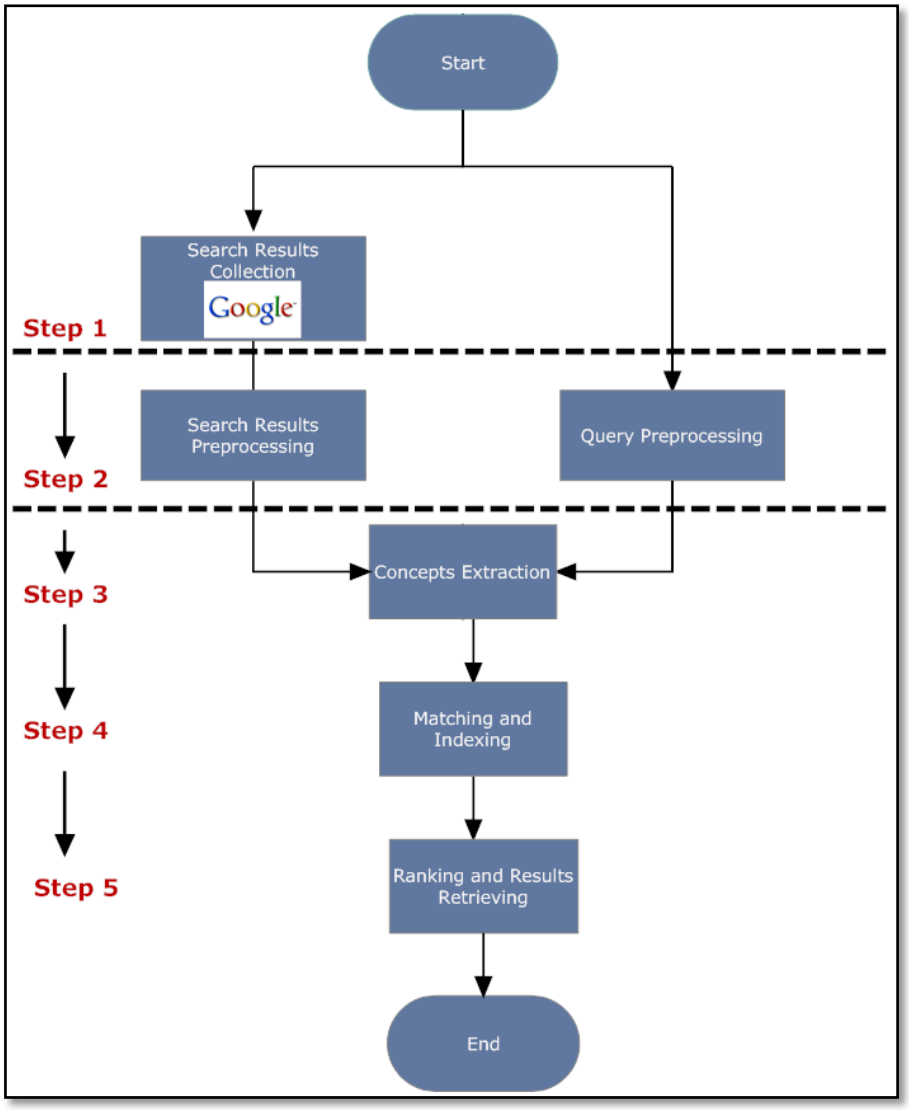

Figure 4.2: Sama Search Engine Flow Diagram

4.1. Search Results Collection

As it is mentioned in section 1.4, the proposed search depends on results collected by Google search engine. Google search engine has been chosen because it is the most popular search engine. Moreover, there are many available APIs more than other search engines which enable developers to use its results in their projects.

At the step of search results collection, the model submits the query to most popular search engine (Google) and stores the first thirty results. These thirty results are considered to be the inputs for Sama search engine which each one contains: Uniform Resource Locator (URL), title, and brief description.

4.2. Preprocessing

While the essential and potential importance is in affecting the outcome of a search, preprocessing applies simply for both submitted query and collected results'

titles and descriptions as an input text and identify ***lemmas*** (the words stems can be found in a dictionary) as output. Then, ***lemmas*** will be used to extract their concepts later.

Figure 4.3 depicts pre-processing stage and shows its sub processes as follows:

1. Special characters and tags will be removed from the input text.
2. Input text will be tokenized into terms.
3. Stop words will be removed.
4. Finally, apply stemming to obtain ***lemmas***.

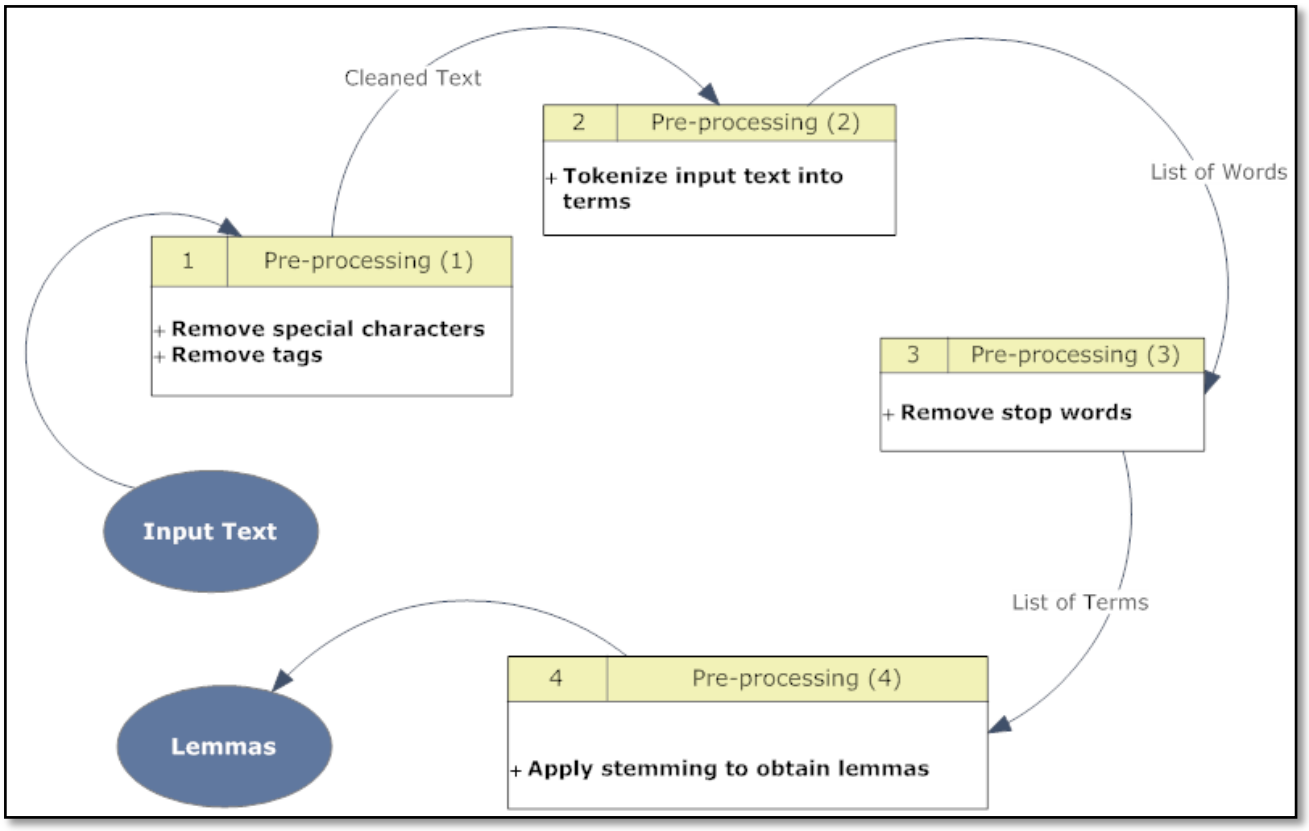

Figure 4.3: Pre-processing Architecture

Next these four steps are explained and discussed.

4.2.1. Pre-processing (1): Special Characters and Tags Removal

At this step, we clean the text by removing special characters as (?<=>#!&). In addition to that, when we retrieve results from Google, it may contain html tags as (
</br>), so we have to ensure that the text will totally be cleaned from

these tags. After applying this step, we obtain cleaned text without special characters or tags.

4.2.2. Pre-processing (2): Tokenization

As soon as a user inputs a query and Google results are collected, Sama Search engine tokenize that streams, i.e., break it down into understandable segments. A token is usually defined as an alphanumeric string that occurs between white space and or punctuation. For each text, the tokens will be stored in a list using a word tokenizer based on the following algorithm:

Algorithm 4.1: Tokenization

1. While (Not End of Text)
   1.1. Tokenize a word
      1.1.1. Word stores in list

4.2.3. Pre-processing (3): Stop Word Removal

This step helps in saving system resources by eliminating the further processing. A stop word list typically consists of those word classes known to convey little substantive meaning, such as articles (a, the), interjections (oh, but), prepositions (in, over), pronouns (he, it), conjunctions (and, but), and forms of the "to be" verb (is, are). To delete stop words, index term candidates in the text compares against a stop word list and eliminates certain terms of inclusion in the index for searching.

Table B.1 in Appendix B shows most common English stop words based on Rainbow[1].

4.2.4. Pre-processing (4): Stemming

A stemming is a process of reducing all words that have the same stem to a common form. It is useful in many areas of computational linguistics and information-retrieval work [16]. Stemming removes word suffixes, perhaps recursively in layer after layer of processing. The process has two goals. In terms of efficiency, stemming reduces the number of unique words in the index, which in turn reduces the storage space required for the index and speeds up the search process. In terms of effectiveness, stemming improves the recalling by reducing all forms of word to a base or stemmed form. After applying stemming, we obtain ***lemmas*** which will be used to extract there concepts later.

It is important to mention that, we adapted this kind of stemming due to the fact that we want only the root of the word and the next steps of our model will be able to obtain all the possible semantic meanings through concepts extraction step.

[1] http://www.cs.cmu.edu/~mccallum/bow/rainbow/

4.3. Concepts Extraction

At stemming sub process in the pre-processing step, the list of ***lemmas*** will be obtained. The main functionality of concepts extraction step is to extract concepts related to each ***lemma*** in the list using WordNet.

Table 4.1 shows some examples of concepts extraction for subset of ***lemmas***.

Table 4.1: Concepts Extraction for Subset of Lemmas

Lemma	Concepts
File	record, line, office_furniture, hand_tool
Sweet	phonetician, course, dainty, taste, taste_property
Snow	precipitation, layer, writer, cocaine
Mouse	rodent, bruise, person, electronic_device
Math	Science
Computer	machine, expert
Beauty	appearance, woman, exemplar
Center	refer, think, move
Color	change, affect, influence, decorate, apologize
Red	chromatic_color, river, radical, sum

4.4. Matching and Indexing

After the extraction of concepts step, three lists of concepts are obtained. The first list stores the concepts extracted from the results collected from Google, the second is for concepts extracted from the submitted query, and the third stores the matched concepts from the previous two lists. According to the proposed algorithm 4.2, each concept from results list will be matched with all concepts in submitted query list for ***typical matching*** (lines 6-18) at the first stage. At the second stage, if there still are un-matched concepts in the result list, ***synonymy matching*** (lines 19-37) will be applied for the rest of the concepts, where *synonymy* is relation binding two equivalent or close concepts. If the matching found, match and index flags will be increased. In addition, the compared concept will be copied to the matched concepts list to prevent comparing it again.

The main difference between match and index flags that match flag will be increased by one when it achieved, but index flag will be increased according to the count of appearance of the matched concept. The value of this flags will be explained in details at the next section.

Algorithm 4.2: Matching and Indexing
Input: 2 Lists, QueryConceptsList & ResultConceptsList
Output: match & index flags

```
1.  match ← 0
2.  index ← 0
3.  MatchedConceptsList ← empty
4.  x ← ResultConceptsList.length
5.  y ← QueryConceptsList.length
6.  for i ← 0 to x
7.      frequent ← count of ResultConceptsList[i] in ResultConceptsList
8.      for j ← 0 to y
9.          if ResultConceptsList[i] not found in MatchedConceptsList then
10.             if ResultConceptsList[i] = QueryConceptsList[j] then
11.                 Add ResultConceptsList[i] to MatchedConceptsList
12.                 match ← match + 1
13.                 index ← index + frequent
14.                 break
15.             end if
16.         end if
17.     end for
18. end for
19. if index < x then // it means there are still un matched concepts then
20.     for i ← 0 to x
21.         frequent ← count of ResultConceptsList[i] in ResultConceptsList
22.         for j ← 0 to y
23.             if ResultConceptsList[i] not found in MatchedConceptsList then
24.                 if ResultConceptsList[i] synonym QueryConceptsList[j] then
25.                     Add ResultConceptsList[i] to MatchedConceptsList
26.                     match ← match + 1
27.                     index ← index + frequent
28.                     break
29.                 else
30.                     if j = y – 1 then
31.                         Add ResultConceptsList[i] to MatchedConceptsList
32.                     end if
33.                 end if
34.             end if
35.         end for
36.     end for
37. end if
```

Note that, the proposed algorithm is able to achieve the personalization through the context by applying the synonymy matching besides the typical matching.

4.5. Ranking and Results Retrieving

At Matching and Indexing step, the differences between match and index flags had been explained. Each result has match and index flags. The value of these flags appears in ranking and results retrieving step by the following.

1. For QueryConceptsList.length > 1, we omit all results with match flag = 1

2. The final results will be ranked according to match values of flag, so result with higher match flag will be displayed before the result with lower flag.

3. The results with equal match flags will be ranked according to their index flags. Result with higher index flag will be displayed before the result with lower flag.

It is noticed that, the importance of ranking and results retrieving step is not only for retrieving the related results, but also it has another value in ordering the retrieved related results according to their flags values.

4.6. Complete Example

In this section, a complete example for the proposed search engine model will be provided. We submitted "drawing tables in a document" query to our model then many related results had been retrieved. Table 4.2 shows three retrieved results from SamaSearch Engine Model.

Table 4.2: Results Retrieved from Sama Search Engine Model

Result No.	Result Title	Result Description
1	How to Draw Tables in a Word 2010 Document - For Dummies	A table is an element you insert into your document, so Word 2010's Table commands are found on the Ribbon 's Insert tab, in the aptly-named Tables group.
2	Drawing tables in PDF document using HTML formatting supported ...	how to add tables to PDF document using built-in HTML formatting support in PDFDoc Scout library
3	Insert or create a table - Word - Office.com	You can insert a table into a document, or you can insert one table into another ... You can create a table by drawing the rows and columns that you want or by ...

In Table 4.3, result no.3 had been picked randomly as an example to illustrate in details how SamaSearch Engine Model works during its steps to calculate match and index flags.

Table 4.3: Complete Example for Sama Search Engine Model

Input query:	drawing tables in a document	
Step 1: Search Results Collection		
Google result no.3:	**URL:**	http://office.microsoft.com/en-us/word-help/insert-or-create-a-table-HA010034300.aspx
	Title:	Insert or create a table - Word - Office.com
	Description:	You can insert a table into a document, or you can insert one table into another ... You can create a table by drawing the rows and columns that you want or by ...
Step 2: Preprocessing		
Pre-processing (1): Special Characters and Tags Removal		
Input query:	drawing tables in a document	
Title:	Insert or create a table Word Office com	
Description:	You can insert a table into a document or you can insert one table into another You can create a table by drawing the rows and columns that you want or by	
Pre-processing (2): Tokenization		
Input query:	drawing, tables, in, a, document	
Title:	Insert, or, create, a, table, Word, Office, com	
Description:	You, can, insert, a table, into, a, document, or, you, can, insert, one, table, into, another, You, can, create, a, table, by, drawing, the, rows, and, columns, that, you, want, or, by	
Pre-processing (3): Stop Word Removal		
Input query:	drawing, tables, document	
Title:	insert, create, table, word, office	
Description:	insert, table, document, insert, table, create, table, drawing, rows, columns	

Pre-processing (4): Stemming	
Input query:	draw, table, document
Title:	insert, create, table, word, office
Description:	insert, table, document, insert, table, create, table, draw, row, column

Step 3: Concepts Extraction

Note: this part of example combine the concepts extraction for both the submitted query and the returned result.

Draw	gully, entertainer, finish, object, playing_card, golf_stroke, run, poker, pull
Table	array, furniture, tableland, gathering, fare
Document	writing, representation, communication, computer_file
Insert	section, artifact, break
Word	language_unit, statement, information, hypostasis, promise, positive_identification, speech, sacred_text, order, computer_memory_unit
Office	place_of_business, administrative_unit, duty, state, staff, rite, occupation
Row	line, dispute, strip, layer, array, sequence, sport
Column	file, tube, array, shape, article, structure, upright

Step 4: Matching and Indexing

Matching and Indexing (1): Typical Matching	
ResultConceptsList	insert, create, table, word, office, insert, table, document, insert, table, create, table, draw, row, column
QueryConceptsList	draw, table, document

match = 0, index = 0, MatchedConceptsList = { }
insert !found in QueryConceptsList
create !found in QueryConceptsList
table found in QueryConceptsList
match = 1, index = 4, MatchedConceptsList = { *table* **}**
word !found in QueryConceptsList

office !found in QueryConceptsList
insert !found in QueryConceptsList
table found in MatchedConceptsList
document found in QueryConceptsList
match = 2, index = 5, MatchedConceptsList = { *table, document* **}**
insert !found in QueryConceptsList
table found in MatchedConceptsList
create !found in QueryConceptsList
table found in MatchedConceptsList
draw found in QueryConceptsList
match = 3, index = 6, MatchedConceptsList = { *table, document, draw* **}**
row !found in QueryConceptsList
column !found in QueryConceptsList

Matching and Indexing (1): Synonymy Matching

ResultConceptsList	insert, create, table, word, office, insert, table, document, insert, table, create, table, draw, row, column
QueryConceptsList	draw, table, document

Query Concepts Synonymies = { *gully, entertainer, finish, object, playing_card, golf_stroke, run, poker, pull* **,** *array, furniture, tableland, gathering, fare* **,** *writing, representation, communication, computer_file* **}**
match = 3, index = 6, MatchedConceptsList = { *table, document, draw* **}**
insert Synonymies = **{** *section, artifact, break* **}**
insert Synonymies ∩ Query Concepts Synonymies = **{ }**
match = 3, index = 6, MatchedConceptsList = { *table, document, draw, insert* **}**
create Synonymies = **{ }**
create Synonymies ∩ Query Concepts Synonymies = **{ }**
match = 3, index = 6, MatchedConceptsList = { *table, document, draw, insert***,** *create* **}**
table found in MatchedConceptsList
word Synonymies = **{** *language_unit, statement, information, hypostasis, promise, positive_identification, speech, sacred_text, order, computer_memory_unit* **}**
word Synonymies ∩ Query Concepts Synonymies = **{ }**
match = 3, index = 6, MatchedConceptsList = { *table, document, draw, insert***,** *create, word***}**
office Synonymies = **{** *place_of_business, administrative_unit, duty, state, staff, rite, occupation* **}**
office Synonymies ∩ Query Concepts Synonymies = **{ }**
match = 3, index = 6, MatchedConceptsList = { *table, document, draw, insert***,** *create, office* **}**
insert, table, document, insert, table, create, table, draw found in MatchedConceptsList
row Synonymies = **{** *line, dispute, strip, layer, array, sequence, sport* **}**
row Synonymies ∩ Query Concepts Synonymies = **{** *array* **}**
match = 4, index = 7, MatchedConceptsList = { *table, document, draw, insert***,** *create,*

office, row **}**

column Synonymies = **{** *file, tube, array, shape, article, structure, upright* **}**

column Synonymies ∩ Query Concepts Synonymies **= {** *array* **}**

match = 5, index = 8, MatchedConceptsList = { *table, document, draw, insert***,** *create, office, row, column* **}**

Step 5: Ranking and Results Retrieving

After having match and index flags for the three results shown in table 4.3, the results will be ranked as the following:

1. Insert or create a table - Word - Office.com **(match = 5, index = 8)**
2. How to Draw Tables in a Word 2010 Document - For Dummies **(match = 3 index = 11)**
3. Drawing tables in PDF document using HTML formatting supported **(match = 3 index = 5)**

In this chapter, we talked about the proposed personalized context-dependent search engine model.

Five stages were adapted Sama Search engine model as the following:

1. Search results collection: after user submitted his/her query to our search engine, the model submits the query to Google search engine and store the first thirty results. These thirty results are considered to be the inputs for our search engine.
2. Preprocessing: simply applies for both submitted query and collected results' titles and descriptions to identify ***lemmas***.
3. Concepts extraction: by using WordNet, concepts of ***lemmas*** will be extracted.
4. Matching and indexing: the extracted concepts of the submitted query and results will be compared to identify match and index flags.
5. Ranking and results retrieving: rank results according to their match flags, which result with higher match flag will be displayed before the result with lower flag. For the results with equal match flags, ranking will be according to indexes flags, which result with higher index flag will be displayed before the result with lower flag

Chapter 5

Experimental Results and Evaluation

In this chapter, before presenting the experiments for Sama Search engine, the results of Semantic Tree (ST) model developed by Chen et al [5] which mentioned in chapter 3 as the main motivator of the proposed search engine will be discussed and criticized in more details. After that, depending on criticizing of ST model, the implementation and results of Sama Search engine will be stated. Finally, the results and enhancement of the proposed search engine will be evaluated comparing to ST model [5].

5.1. Implementation of Sama Search Engine

Figure 5.1 shows the components of Sama Search Engine.

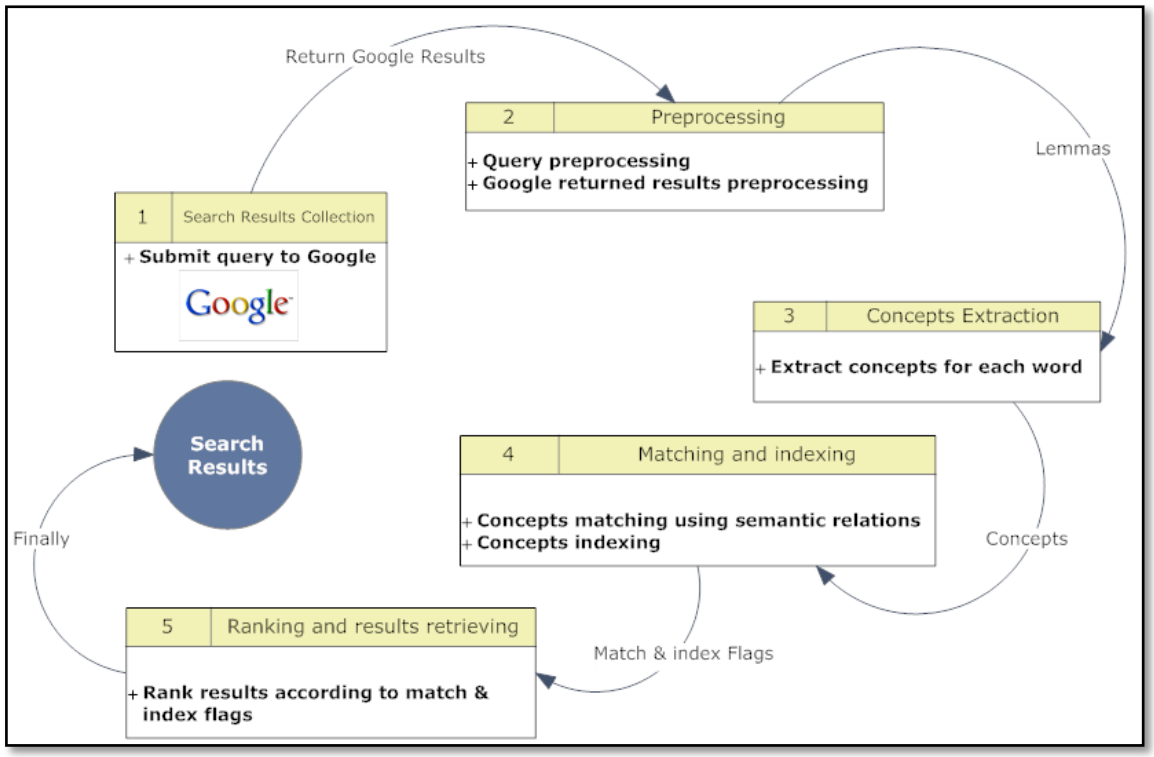

Figure 5.1: Sama Search Engine Components

- We implemented our model in JSP, and deployed it on a TOMCAT web server. JSP was used because the used tools and APIs are found in java language.

- Once a user submits one query to our search engine, the query was transmitted simultaneously to Google and the first thirty results were stored in list.

- We used WordNet for creating the conceptual dictionary, in which words and the associated concepts are stored.

- Matching and indexing was performed using our proposed algorithm in section 4.4.

- Finally, ranking and results retrieving was performed.

Tables 5.1 and 5.2 display part of the code with some explanation; other code examples will be introduced in Appendix A.

Table 5.1 Sort Results According to Match Flag

```
// sort results according to match flag
public ResultBean[] SortResultsAccordingMatches(ResultBean list[])
{
        int n = list.length;
        for (int k= 0; k < n-1; k++ )
        {
           // Find the smallest item in the remaining n-k items.
           ResultBean minVal = list[k];
           int kMin = k;
           for (int j = k+1; j < n; j++)
           {
              Comparable c = minVal.getMatch();
              if (c.compareTo(list[j].getMatch()) < 0)
              {
                   minVal = list[j];
                   kMin = j;
               }
            }

             // Swap the minimum value into location k
             list[kMin] = list[k];
             list[k] = minVal;
          }

          return list;
   }
```

Explanation	This function is responsible for ordering the returned results according to match flags, which results with high flags will be in the top of returned results.

Table 5.2 Return Nonstop Words

```
// Return nonstop words List
public ArrayList<String> getNonStopWords(String str)
{
    ArrayList<String> myArr = new ArrayList<String>();
    String nextToken = "";
```

```
        StringTokenizer st = new StringTokenizer(str, "., ");
        while (st.hasMoreTokens()) {  // Check each token if it stop word or not
            nextToken = st.nextToken();
            if (isStopWord(nextToken))
               System.out.println("yes we have stopword");
           else
           {
               System.out.println("not stopword");
               myArr.add(nextToken.toLowerCase());
           }
         }
        return myArr;
}
```

Explanation	This function is responsible for receiving a complete string, then parting it into tokens to check each token whether it is a stop word or not, finally, it return list of strings of nonstop words.

5.1.1. Sama Search Interfaces

The interface of Sama Search engine is very simple which consists of two pages. The first page contains the following:

- Text Field: to type the query needed to be searched.
- Submit Button: to submit the query to be processed.
- Reset Button: to reset the text field.

The second page appears automatically after press Submit button. It contains the retrieved results related to the submitted query ordering by match and index flags. Each result consist of title, description, and URL. Figures 5.2 and 5.3 show Sama Search engine interfaces.

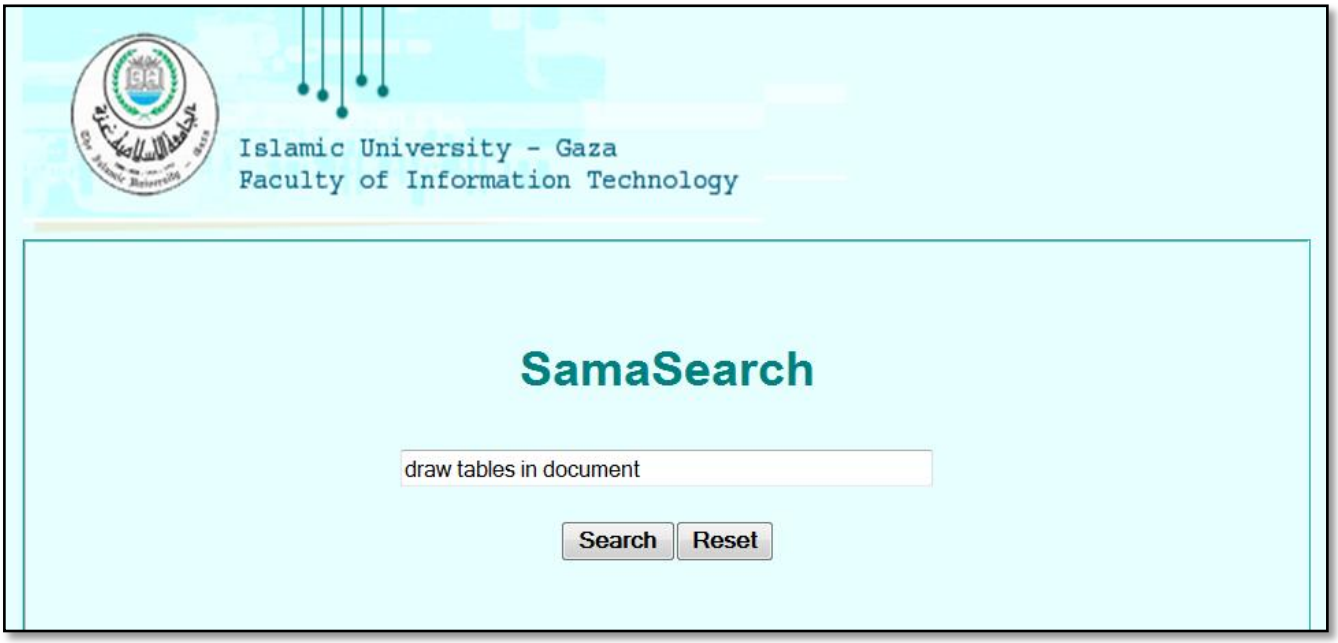

Figure 5.2: First Page of Sama Search Engine Interface

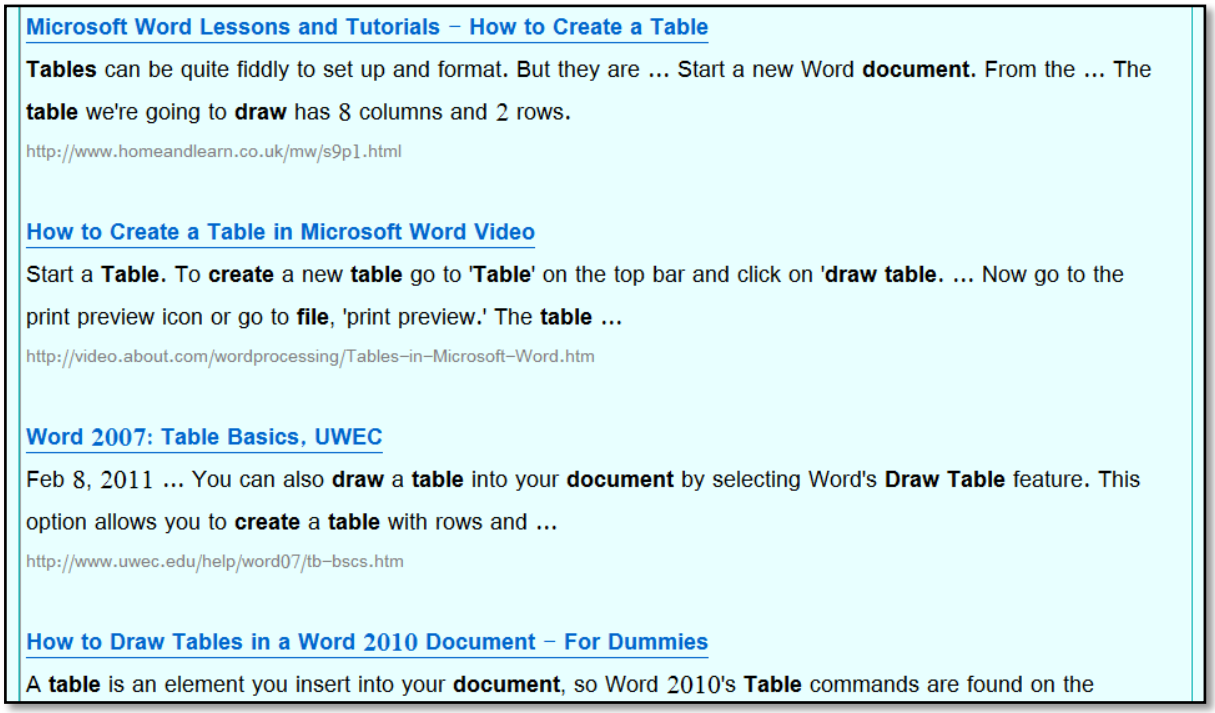

Figure 5.3: Second Page of Sama Search Engine Interface

5.1.2. Tools and Programs:

To implement Sama Search Engine, the following tools had been used:

- Java Development Kit (JDK) 1.6: A software development package from Sun Microsystems that implements the basic set of tools needed to write, test and debug Java applications.
- JCreatorV4.0: this is the program which helps to code the approach using java language.
- Microsoft Office FrontPage 2003: is used to build the interface using Java Server Pages (JSP).
- Internet Explorer: to run the web application.
- Apache Tomcat Jakarta Server 7.0: is used for host the created JSP web application.
- JWNL API[1]: is a Java API for accessing the WordNet relational dictionary to extract concepts.
- JSON Java API[2]: is a Java API for accessing Google results.

5.2. Experiments and Evaluation

The evaluation of the most search engines depends on the most common effectiveness measures such as Recall, Precision, and Fall-Out. As it was mentioned in section 2.8:

- **Recall is** the proportion of relevant documents that are retrieved.
- **Precision** is the proportion of retrieved documents that are relevant.
- **Fallout** is the proportion of non-relevant documents that are retrieved.

[1] http://jwordnet.sourceforge.net/handbook.html
[2] http://json.org/java/

$$Recall = \frac{Number\ of\ retrived\ relevant\ results}{Number\ of\ relevant\ results} \quad \ldots\ldots\ldots eq\ (5.1)$$

$$Precision = \frac{Number\ of\ retrived\ relevant\ results}{Number\ of\ retrived\ results} \quad \ldots\ldots\ldots eq\ (5.2)$$

$$Fallout = \frac{Number\ of\ retrived\ nonrelevant\ results}{Number\ of\ retrived\ results} \quad \ldots\ldots\ldots eq\ (5.2)$$

5.2.1. Sama Search Engine Evaluation

To evaluate Sama Search engine, Recall, Precision, and Fall-Out have been used. 148 queries from general domains are submitted to Sama Search engine. 4421 results have been retrieved with 4212 relevant retrieved results. Based on equations 5.1, 5.2, and 5.3, Sama Search engine provides the following:

$$Recall = \frac{4289}{4323} = 99.2\ \%$$

$$Precision = \frac{4399}{4421} = 99.5\ \%$$

$$Fallout = \frac{23}{4421} = 0.52\ \%$$

5.2.2. ST Model Evaluation

Chen et al [5] had been also submitted 148 queries from general domains to their engine. 3779 results had been returned. Their experiments recorded the following:

$$Recall = 92\ \%$$

$$Precision = 98\ \%$$

$$Fallout = 2\ \%$$

5.2.3. Comparison Between Sama Search Engine and ST Model

Depending on the experiments of both Sama search engine and ST model, it has been noticed that Sama search engine provided enhancement on ST model included precision, recall, fall-out, and F-measure which balances between recall and precession.

Figure 5.4 shows that the recall of Sama search engine is 99.2% which is higher than the 92% recall of ST model because Sama search engine retrieved more relevant documents that ST model retrieved.

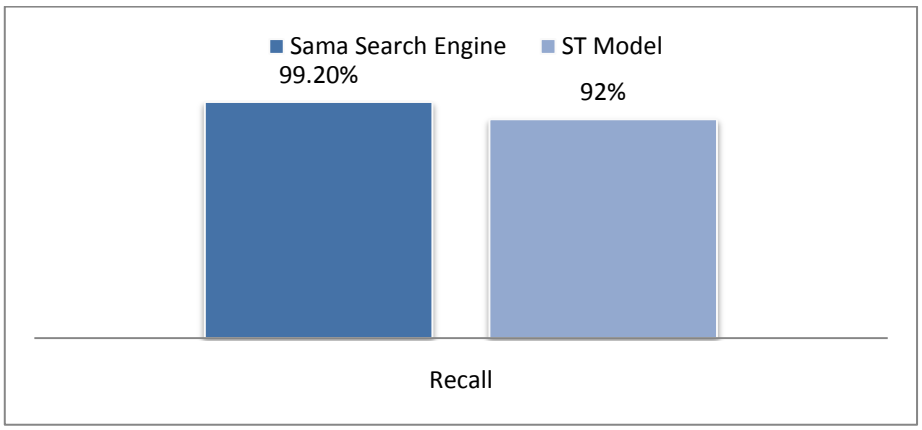

Figure 5.4: Recall Comparison

The precession of Sama search engine is also recorded better results than ST model as it shown in Figure 5.5.

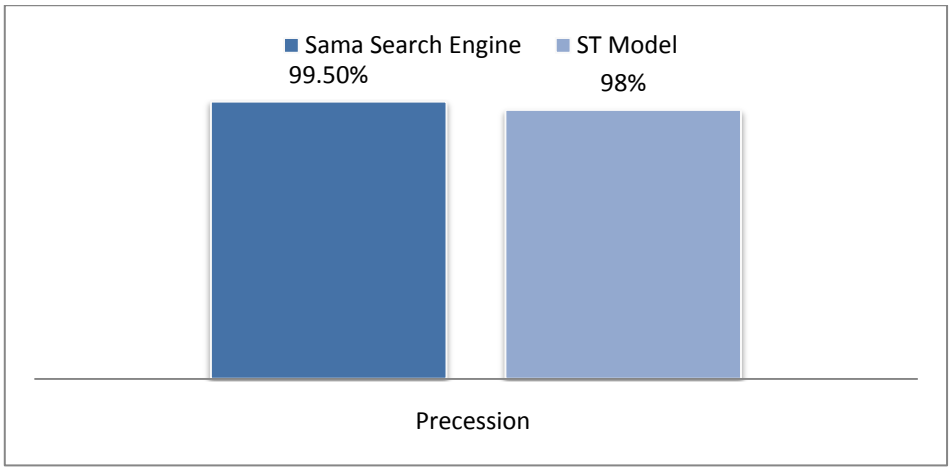

Figure 5.5: Precession Comparison

The non-relevant documents that are retrieved by Sama search engine are less than the documents retrieved by ST model. Because of that, fall-out of Sama Search engine is lower than fall-out of ST model as it shown in Figure 5.6.

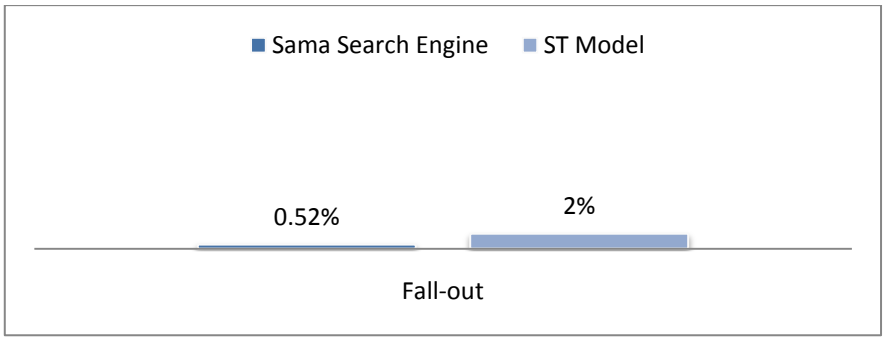

Figure 5.6: Fall-out Comparison

Table 5.3 and Figure 5.7 show, when Sama search engine compared to ST model, we find the precision, recall, and F-measure of Sama search engine are higher, and fall-out is lower.

Table 5.3: Comparison of Sama Search Engine and ST Model

	Precision	Recall	F-measure	Fall-Out
Sama Search Engine	99.5%	99.2%	99.35%	0.52%
ST Model	98%	92%	95%	2%

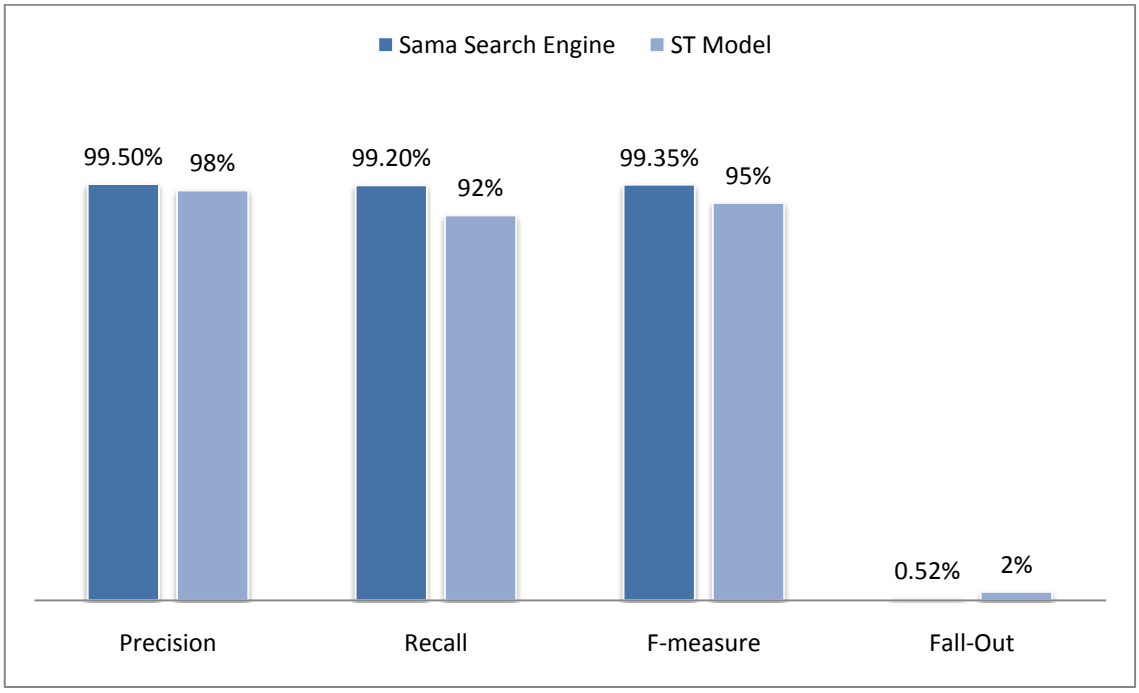

Figure 5.7: Comparison of Sama Search Engine and ST Model

5.3. Discussion

As mentioned in Chen et al [5], once a user submits one query to their search engine, the query was transmitted simultaneously to other search engines and the first thirty results were downloaded and stored in one XML file, in which the titles, abstracts and links of pages were enclosed by meaningful tags. As the core of Chen et al [5] approach is to establishing the Semantic Trees for each link of returned results to match it with the concepts extracted from the submitted query, the following had been noticed when submitted "Drawing tables in document" query:

- As shown in Figure 5.8, there were many links appeared including the top results with no category, it returned Null values. So, how ST would be fully established?

- In addition to that, for the other successful categorized returned results, there was an ambiguity in the used way for match each result with the concepts extracted from the submitted query because most of Semantic Trees leaves had no or very far relation with the submitted query extracted concepts.

- On the other hand, the time execution for each submitted query to return results exceeded 3 minutes because of consuming very long time for establishing the ST.

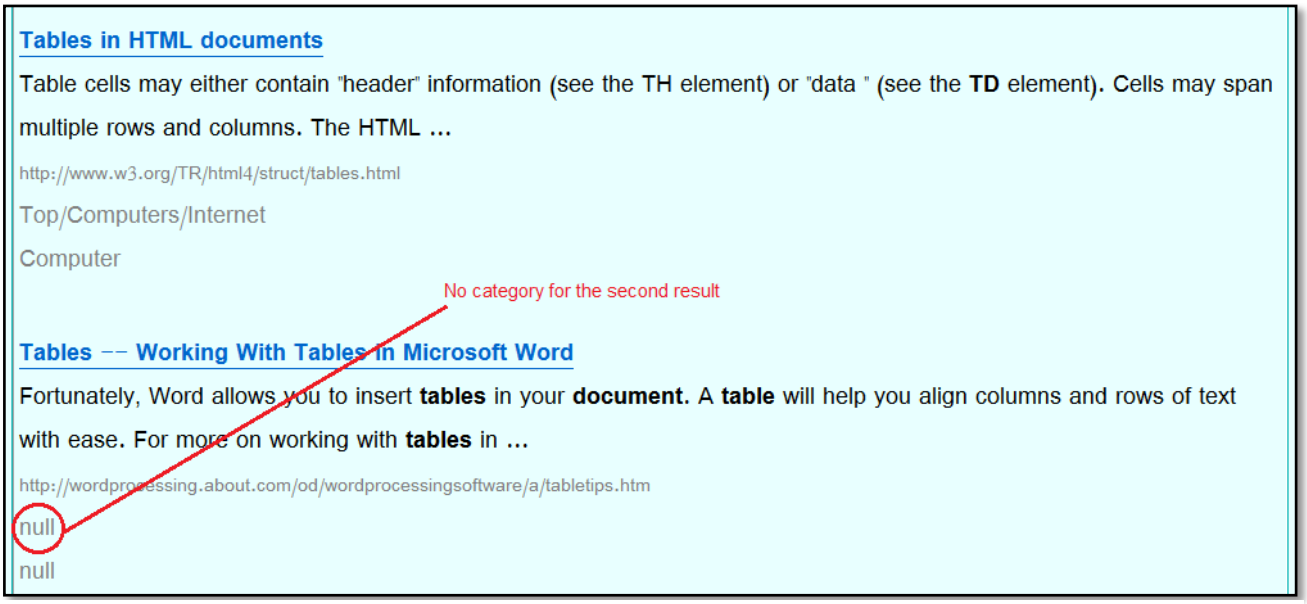

Figure 5.8: Two of Returned Results Using ST Model

From the experiments and comparisons, we found that our results seem higher those of Chen et al [5] because of the following:

- Sama search engine is outperforming ST method based on F-measure, this is due to depending on our new technique for concepts extraction and matching

using WordNet instead of establishing ST which couldn't categorized many results and then affected on the effectiveness of search engine .

- The execution time of Sama Search engine doesn't exceed few seconds because it is does not need to use ODP to establish ST.

Chapter 6

Conclusion and Future Works

6.1. Conclusion

In this thesis, a Personalized Context-Dependent Web Search Engine model has been introduced. The proposed model is called "Sama Search Engine" which based mainly on the context of the submitted query and the returned results.

Five stages are involved in the approach: search results collection, preprocessing submitted query and collected results, concepts extraction, match and index results, and rank the retrieved results according to match and index flags as a final stage.

The main difference between Sama search engine model and other personalized context-dependent search engine models that it does not depend on establishing Semantic Trees (ST) to extract concepts relations in order to identify the context of queries. Instead of that, Sama search engine provides a new technique for concepts extraction using WordNet.

To evaluate Sama search engine model, the most common effectiveness measures such as Recall, Precision, F-measure , and Fall-Out had been used. When Sama Search engine compared to the ST method, it's found the precision, recall, and F-measure of Sama search engine are higher, and it's fall-out is lower. The recall of Sama search engine is 99.2% which is higher than the 92% recall of ST model because it retrieved more relevant documents than ST model retrieved. The precession of Sama Search engine recorded 99.5% and 98% for ST model. The non-relevant documents that are retrieved by Sama Search engine are less than the documents retrieved by ST model. Because of that, fall-out of Sama Search engine is 0.52% and 0.2% for ST model. The F-measure obtained by the proposed model and ST model respectively achieved 99.35% and 95%.

6.2. Future Work

We should apply a number of suggested techniques to enhance the search engine:

- Developing Sama Search engine to serve other languages specially Arabic Language.
- Testing other semantic relations between concepts as Hyperonymy and Hyponymy instead of using only Synonymy.
- In addition to the context of the query, user preferences have to be handled to achieve personalized web search engine fully.

References

1. Apte, C., Damerau, F., and Weiss, S.M., "Automated learning of decision rules for text categorization", ACM Transactions on Information Systems 12 (3), (1994).
2. Bautista, Kraft, D., and Rules, M., "Fuzzy Rules in Text: Concept, Extraction and Usage", International Journal of Approximate Reasoning, Vols. 34, pages 145–161, (2003).
3. Berners-Lee, Tim., "Weaving the Web", Harper San Francisco, (1999).
4. Bruce Croft, W., Metzler, D., and Strohman, T., "Evaluating Search Engines", Search Engines Information Retrieval in Practice, (2010).
5. Chen, Y., Hou, H. and Qing Zhang, Y., "A personalized context-dependent Web search agent using Semantic Trees", Annual Meeting of NAFIPS, (2008).
6. Daconta, M., Obrst, L., and Smith, K., "The Semantic Web:A Guide to the Future of XML, WebServices, and Knowledge Management", Canada, Wiley Publishing, (2003).
7. Development Team of Sun Microsystems, "JavaServer Pages(TM) Tutorial", Sun Microsystems Company, (2001).
8. Development Team of VisualBuilder.com, "JSP Tutorial", (2001).
9. Fang Liu, C. Yu and Weiyi Meng, "Personalized Web search for improving retrieval effectiveness", IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 1, Pp. 28 – 40, (2004).
10. Google personalized search, http://www.google.com/ig, [Online] [Cited: January 7, 2012].
11. Guarino, N., "Formal Ontology and Information Systems", Proceedings of FOIS, (1998).
12. H¨oscher, C., Strube, G., "Web search behavior of internet experts and newbies", Amsterdam, Netherlands : Proceedings of the 9th World Wide Web Conference (WWW9), (2000).
13. John, Jerry M. Mendel and Robert I. Bob, "Type-2 Fuzzy Sets Made Simple", IEEE Transactions On Fuzzy Systems, VOL.10, NO. 2, (2002).
14. Ljosland, M., "Evaluation of Web search engines and the search for better ranking algorithms", SIGIR99 Workshop on Evaluation of Web Retrieval, Norway, (1999).
15. Liu, Jie Yu and Fangfang, "Mining user context based on interactive computing for personalized Web search", 2nd International Conference on Computer Engineering and Technology (ICCET), Vol. 2, Pp. 209-214, (2010).
16. Lovins, Julie Beth, "Development of a stemming algorithm", Mechanical Translation and Computational Linguistic, (1968).
17. Ma, Z., Pant, G., and Sheng, L., "Interest-Based Personalized Search", ACM Transactions on Information Systems (TOIS), ACM Press, (2007).
18. Micarelli, A. , Gasparetti, F., Sciarrone, F., and Gauch, S., "Personalized Search on theWorld Wide Web", [book auth.] A. Kobsa, and W. Nejdl (Eds.) P. Brusilovsky. The Adaptive Web. s.l. : Springer-Verlag Berlin Heidelberg, pp. 195–230, (2007).

19. Miller, G. A., "Nouns in WordNet: A Lexical Inheritance System", International Journal of Lexicography, Vol. 3, (1990).
20. Miller, G. A., "WordNet: An On-Line Lexical Database", In Special Issue of International Journal of Lexicography, Chongqing, China, (1990).
21. Mimoun, Bentaallah Mohamed Amine and Malki, "WordNet based Multilingual Text Categorization", Djillali Liabes university, AlGeria, (2007).
22. Ohgaya, R., Fukano, K., Taniguchi, K., Takagi, T., Aizawa, A., and Nikravesh, M., "Conceptual Fuzzy Sets-Based Menu Navigation System for Yahoo!", IEEE, the North American Fuzzy Information Processing Society – The Special Interest Group on Fuzzy Logic and the Internet NAFIPS-FLINT, pp. 274-279, (2002).
23. Open Directory Project, http://www.dmoz.org/about.html, [Online] [Cited: January 16, 2012].
24. Page, Sergey Brin and Lawrence, "The Anatomy of a Large-Scale Hypertextual", Stanford, USA, (1998).
25. Popovic, Willett, "The Effectiveness of Stemming for Natural-Language Access to Slovene Textual Data", Journal of the American Society for Information Science, (1992).
26. Rae, Brian Clifton and Nikki, "How Search Engines Optimization Works (White Paper)", (1999).
27. Robertson, "Theories and models in information retrieval", Journal of Documentation, (1977).
28. Sathiyabama, T., and Dr. Vivekanandan, K., "Personalized Web Search Techniques - A Review", Global Journal of Computer Science and Technology, Volume 11 Issue 12, (2011).
29. Savoy, J., Picard, J., "Retrieval effectiveness on the web", Information Processing & Management, pp 543–569, (2001).
30. Sugiyama, K., Hatano, K., and Yoshikawa, M., "Adaptive Web Search Based on User Profile Constructed Without any Effort From Users", Proceedings of the 13th International Conference on World Wide Web, pages 675–684, ACM Press, (2004).
31. Walairacht, A., and Bounoy, T., "User Preference Retrieval using Semantic Categorization for Web Search", IEEE, (2010).
32. Widom, Glen Jeh and Jennifer, "Scaling Personalized Web Search", National Science Foundation. (2002).
33. Wikipedia, http://en.wikipedia.org/wiki/Fuzzy_logic, [Online] [Cited: January 9, 2012].
34. Wikipedia, http://en.wikipedia.org/wiki/Open_Directory_Project, [Online] [Cited: January 15, 2012].
35. Wikipedia, http://en.wikipedia.org/wiki/Web_search_engine, [Online] [Cited: January 19, 2012].
36. Xiong, Qizhi Qiu and Qianxing, "An Ontology for Semantic Web Services", Wuhan University of Technology, China, (2000).

37. Zhengyu Zhu, Jingqiu Xu, Xiang Ren, Yunyan Tian and Lipei Li, "Query Expansion Based on a Personalized Web Search Model", Third International Conference on Semantics, Knowledge and Grid, Pp. 128 – 133, (2007).

Appendix A

Sama Search Engine Code

Table A.1 to A.3 display parts of code using to implement the model.

Table A.1 Sort Results According to Index Flag

```
// sort results according to index flag
 public ResultBean[]SortResultsAccordingIndexes(ResultBean list[])
 {
        int n = list.length;
        for (int k= 0; k < n-1; k++ )
        {
           // Find the smallest item in the remaining n-k items.
           ResultBean minVal = list[k];
           int kMin = k;
           for (int j = k+1; j < n; j++)
           {
              Comparable c = minVal.getMatch();
              Comparable cc = minVal.getIndex();
              if ( (c.compareTo(list[j].getMatch()) <= 0) &&
                   (cc.compareTo(list[j].getIndex()) < 0) )
              {
                   minVal = list[j];
                   kMin = j;
               }
            }

             // Swap the minimum value into location k
             list[kMin] = list[k];
             list[k] = minVal;
           }

          return list;
            }
```

Explanation	This function is responsible for ordering the returned results according to index flags, which results with high flags will be in the top of returned results.

Table A.2 Retrieve results from Google code

```
// Retrieve Google Results
private static String[] makeQuery(String query, int start, boolean completRound)
throws Exception{
  try
  {
   // Convert spaces to +, etc. to make a valid URL
   query = URLEncoder.encode(query, "UTF-8");

   URL url = new URL(http://ajax.googleapis.com/ajax/services/search/web?
                              +" start=" + start + "&rsz=large&v=1.0&q=" + query);

   URLConnection connection = url.openConnection();

   connection.addRequestProperty("Referer", HTTP_REFERER);

   // Get the JSON response
   String line;
   StringBuilder builder = new StringBuilder();
   BufferedReader reader = new BufferedReader(
           new InputStreamReader(connection.getInputStream()));
   while((line = reader.readLine()) != null) {
          builder.append(line);
   }

   String response = builder.toString();
   JSONObject json = new JSONObject(response);

   JSONArray ja = json.getJSONObject("responseData")
   .getJSONArray("results");

   String[] Results = new String[8];

   for (int i = 0; i < ja.length(); i++) {
          searchCount++;
          ResultBean newResult = new ResultBean();
          if (i == 2 && completRound == false)
                 break;
          JSONObject j = ja.getJSONObject(i);
          Results[i] = j.getString("url");
          newResult.setURL(j.getString("url"));
          newResult.setTitle(j.getString("titleNoFormatting"));
          newResult.setDesc(j.getString("content"));

          ResultsList.add(newResult);
   }
   return Results;
  }
```

```
catch (Exception e) {
  e.printStackTrace();
  String[] error={"error"};
  return error;
 }
}
```

Explanation	This function is responsible for retrieve results from Google using JSON API.

Table A.3 Extract Word Concepts Code

```
// Get possible concepts for specific word
private ArrayList<String> getWordConcepts(String sw)
throws JWNLException
{
      // Get all the hyponyms (children) of the first sense of <var>word</var>
     // Dictionary object
      //Dictionary wordnet;
      // Look up all IndexWords (an IndexWord can only be one POS)
      IndexWordSet set = Dictionary.getInstance().lookupAllIndexWords(sw);
      // Turn it into an array of IndexWords
      IndexWord[] iwords = set.getIndexWordArray();
      // Make the array of POS
      POS[] pos = new POS[iwords.length];
      for (int i = 0; i < iwords.length; i++)
     {
         pos[i] = iwords[i].getPOS();
      }

     IndexWord word = Dictionary.getInstance().getIndexWord(pos[1], sw);
     int SenseCount = word.getSenseCount();
     int i;
    ArrayList<String> ConceptsArr = new ArrayList<String>();
    for (i=1; i<SenseCount+1; i++)
    {
         PointerTargetNodeList hypernyms =
         PointerUtils.getInstance().getDirectHypernyms(word.getSense(i));
         List sl = hypernyms.subList(0,1);
         String s = sl.get(0).toString();
         int j = s.indexOf("W");
         String s2 = s.substring(j+7, s.indexOf("-")) ;
         s2 = s2.replaceAll(",", " ");

         if (s2.contains(" "))
             s2 = s2.substring(0, s2.indexOf(" ")) ;
```

if (!(ConceptsArr.contains(s2)))
ConceptsArr.add(s2);
}
return ConceptsArr;
}	

Explanation	This function is responsible for extraction word concepts using JWNL API.

Appendix B

Most Common English Stop Words

Table B.1 shows most common English stop words based on Rainbow[1] project.

Table B.1 Most Common English Stop Words

a, able, about, above, according, accordingly, across, actually, after, afterwards, again, against, all, allow, allows, almost, alone, along, already, also, although, always, am, among, amongst, an, and, another, any, anybody, anyhow, anyone, anything, anyway, anyways, anywhere, apart, appear, appreciate, appropriate, are, around, as, aside, ask, asking, associated, at, available, away, awfully, b, be, became, because, become, becomes, becoming, been, before, beforehand, behind, being, believe, below, beside, besides, best, better, between, beyond, both, brief, but, by, c, came, can, cannot, cant, cause, causes, certain, certainly, changes, clearly, co, com, come, comes, concerning, consequently, consider, considering, contain, containing, contains, corresponding, could, course, currently, d, definitely, described, despite, did, different, do, does, doing, done, down, downwards, during, e, each, edu, eg, eight, either, else, elsewhere, enough, entirely, especially, et, etc, even, ever, every, everybody, everyone, everything, everywhere, ex, exactly, example, except, f, far, few, fifth, first, five, followed, following, follows, for, former, formerly, forth, four, from, further, furthermore, g, get, gets, getting, given, gives, go, goes, going, gone, got, gotten, greetings, h, had, happens, hardly, has, have, having, he, hello, help, hence, her, here, hereafter, hereby, herein, hereupon, hers, herself, hi, him, himself, his, hither, hopefully, how, howbeit, however, i, ie, if, ignored, immediate, in, inasmuch, inc, indeed, indicate, indicated, indicates, inner, insofar, instead, into, inward, is, it, its, itself, j, just, k, keep, keeps, kept, know, knows, known, l, last, lately, later, latter, latterly, least, less, lest, let, like, liked, likely, little, ll, look, looking, looks, ltd, m, mainly, many, may, maybe, me, mean, meanwhile, merely, might, more, moreover, most, mostly, much, must, my, myself, n, name, namely, nd, near, nearly, necessary, need, needs, neither, never, nevertheless, new, next, nine, no, nobody, non, none, noone, nor, normally, not, nothing, novel, now, nowhere, o, obviously, of, off, often, oh, ok, okay, old, on, once, one, ones, only, onto, or, other, others, otherwise, ought, our, ours, ourselves, out, outside, over, overall, own, p, particular, particularly, per, perhaps, placed, please, plus, possible, presumably, probably, provides, q, que, quite, qv, r, rather, rd, re, really, reasonably, regarding, regardless, regards, relatively, respectively, right, s, said, same, saw, say, saying, says, second, secondly, see, seeing, seem, seemed, seeming, seems, seen, self, selves, sensible, sent, serious, seriously, seven, several, shall, she, should, since, six, so, some, somebody, somehow, someone, something, sometime, sometimes, somewhat, somewhere, soon, sorry, specified, specify, specifying, still, sub, such, sup, sure, t, take, taken, tell, tends, th, than, thank, thanks, thanx, that, thats, the, their, theirs, them, themselves, then, thence, there, thereafter, thereby, therefore, therein, theres, thereupon, these, they, think, third, this, thorough, thoroughly, those, though, three, through,

[1] http://www.cs.cmu.edu/~mccallum/bow/rainbow/

throughout, thru, thus, to, together, too, took, toward, towards, tried, tries, truly, try, trying, twice, two, u, un, under, unfortunately, unless, unlikely, until, unto, up, upon, us, use, used, useful, uses, using, usually, uucp, v, value, various, ve, very, via, viz, vs, w, want, wants, was, way, we, welcome, well, went, were, what, whatever, when, whence, whenever, where, whereafter, whereas, whereby, wherein, whereupon, wherever, whether, which, while, whither, who, whoever, whole, whom, whose, why, will, willing, wish, with, within, without, wonder, would, would, x, y, yes, yet, you, your, yours, yourself, yourselves, z, zero