University of Alberta

An ontology-driven concept-based information retrieveal approach for Web documents

by

Zhan Li

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Department of Electrical and Computer Engineering

©Zhan Li
Fall 2010
Edmonton, Alberta

Examining Committee

Marek Reformat, Electrical and Computer Engineering Department

Vladik Kreinovich, Department of Computer Science, University of Texas at El Paso

Jozef Szymanski, Civil and Environmental Engineering Department

Witold Pedrycz, Electrical and Computer Engineering Department

Petr Musilek, Electrical and Computer Engineering Department

Vicky H Zhao, Electrical and Computer Engineering Department

Abstract

Building computer agents that can utilize the meanings in the text of Web documents is a promising extension of current search technology. Concept-based information retrieval applies “intelligent” agents to identify Web documents that match user queries. A new concept-based information retrieval framework, *Hybrid Ontology-based Textual Information Retrieval* (HOTIR), is introduced in this thesis. HOTIR accepts conventional keyword-based queries, translates them into concept-based queries, enriches definitions of concepts with supplementary knowledge from a knowledge base, and ranks documents by aggregating “equivalent” concepts identified in them. The concept-based queries in HOTIR are organized in a hierarchy of concepts (HofC) and definitions of concepts are added from a knowledge base to enhance their meanings. The knowledge base is a modified ontology (ModOnt) that can enrich the HofC with concept definitions in the form of related-concepts, terms, their importance values, and their relations. The ModOnt relies on an adaptive assignment of term importance (AATI) scheme that continuously updates the importance of terms/concepts using Web documents. The identified concepts in a Web document that match those in the HofC are evaluated using ordered weighted averaging (OWA) operators, and documents are ranked according to the degree to which they satisfy the HofC. The case studies and experiments presented in the thesis are designed to validate the performance of HOTIR.

Contents

List of Figures

List of Tables

Chapter 1

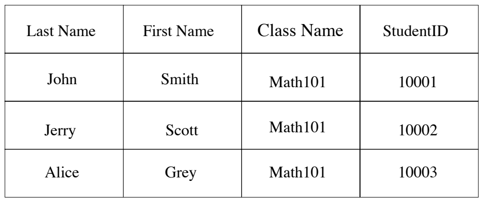
Introduction

1.1 Information on the Web

The World Wide Web (WWW) is currently the largest information repository in the world. Based on Pierre's study [1], almost 1.5 million documents are added to the Web every day. The importance of the WWW as a domain of collected information and knowledge grows constantly. However, discovering useful information on the Web is challenging.

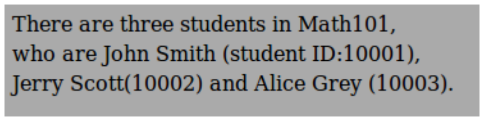
There are multiple reasons for that challenge. First, the primary markup language used for representing documents on the Web, HTML (Hyper Text Markup Language), is designed to present unstructured data in a structured format; this complicates automatic processing activities. In HTML, structured formats are built by using tags to define different document sections, such as headings, titles, paragraphs, lists, and so on, which helps human beings to read. However, there are few tags that provide meanings of the text. For example, for the term "Math101" in Fig. 1.1b, there is no tag indicating that this is a course name. Though humans can guess it is a course name, computer agents can not. This is an example of unstructured data. A database is an example of structured data; each piece of data in a database is stored in a specific field. Handling unstructured data effectively and automatically is an attractive research topic. In Fig. 1.1, the same amount of information is presented in the database table (structured data) as in the HTML-based Web document (unstructured data). Computer agents can utilize information presented in a table (Fig. 1.1a) much more easily than information presented in a Web document (Fig. 1.1b).

A second reason for the challenge to find useful information on the Web is the variable quality of Web documents. Millions of people "post" documents



Last Name	First Name	Class Name	StudentID
John	Smith	Math101	10001
Jerry	Scott	Math101	10002
Alice	Grey	Math101	10003

(a) Structured data



There are three students in Math101,
who are John Smith (student ID:10001),
Jerry Scott(10002) and Alice Grey (10003).

(b) Unstructured data

Figure 1.1: Structured data and unstructured data

on the WWW; some of them are experts, some are not. Some people use the WWW to share ideas and knowledge; some are there just for fun. The cost of putting a document on the WWW is so small that it is rarely a consideration. The variable quality of Web documents makes it difficult for computer agents to automatically retrieve useful information from the Web.

A third factor that impacts the organization of data on the Web is the large number of Web documents that can be seen as relevant to a specific search criterion. In most cases, the search criterion is a simple query, and the matched Web documents are thousands in number. For example, if we use the keywords "Semantic Web" to query the Web using the most popular search engine Google, the number of relevant documents is 8,020,000[1].

These are some of the elements that challenge the "discovery" of knowledge on the Web. Therefore, the necessity of applying information retrieval (IR) techniques to the massive amount of information on the Web is rapidly growing.

1.2 Web-based information retrieval

Information retrieval (IR) is a research topic dedicated to the extraction of useful information from unstructured textual data. The need for efficient IR methods is so overwhelming that the so called "information explosion" should be replaced with the more accurate term "data explosion."

The first step of an information retrieval-based system is to accept a query from a user. This query is a statement describing the user's need for information. Because documents in the repository match the query with different degrees of relevance, information retrieval tools rank documents based on how well they match the query.

Web-based information retrieval applications are called search engines. Most

[1]The search is performed on April, 2010.

engines use either taxonomies built for preset categories – Yahoo, or keyword-matching scheme – Google.

Taxonomy is a hierarchical structure built for classification purposes. In the taxonomy-based approaches, Web documents are manually classified into taxonomies. Because the documents are categorized based on the knowledge embedded in them, users are able to find documents relevant to topics of interest. The major disadvantage of this approach is the extensive human effort required to perform categorization. The conflict between the constantly increasing number of Web documents and the limits of human resources requires substantial improvements in taxonomy-building approaches. Besides, a user wanting to find an interesting document in a specific category has to look among thousands of items in similar categories.

"Keyword matching" is the most popular approach used in current search engines. Its basic structure is presented in Figure 1.2. A *crawler*, also called a spider or a robot, is a program or automated script which "browses the WWW in a methodical and automated manner" [2]. A crawler will grab Web documents from remote sites and copy them into a local repository for further processing, which includes parsing and indexing of the documents plus transferring them from unstructured to structured formats (tables or entries in a database). Parsing is the process of analyzing a text, and indexing is used to optimize the performance of finding relevant documents based on a query. The most widely used index in Web search engines is called *inverted index* [3]. After Web documents are collected and processed, users are able to retrieve documents based on queries, which are normally lists of keywords. There are two main advantages of this approach:

- A Web search engine does not require extensive human labor: collecting documents and keeping them up to date requires minimum effort;
- A query can comprise one to several keywords; this is convenient from a user's perspective.

Though keyword-matching scheme is applied by most of the commercial information retrieval models [4], there is a critical disadvantage to this approach: many words used as keywords have different meanings in different contexts. In addition, users tend to use very few keywords (three or less) in their search queries [5, 6]; this makes it difficult for computers to understand the contexts of the requests.

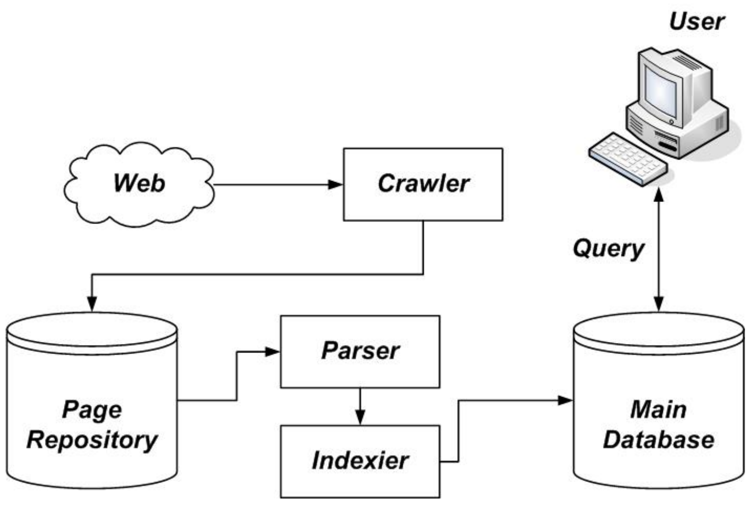


Figure 1.2: A simple structure of Search Engine (SE)

1.3 Motivation

Keyword-based information retrieval approaches provide results, but quite often the results are outside the context of user requirements. Information retrieval that considers the meanings of keywords is a promising modification.

Currently agents of keyword-based information retrieval tools read presented texts from the Web but can not understand the underlying meanings of the texts. Tim Berners-Lee originally expressed the vision of the Web in the next generation as follows [7]:

I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web, the content, links, and transactions between people and computers. ... the day-to-day mechanisms of trade, bureaucracy, and our daily lives will be handled by machines talking to machines. The "intelligent agents" people have touted for ages will finally materialize.

Building computer agents that can discover the meanings of texts in the Web documents is an attractive area of investigation. The application of the "intelligent agents" to Web information retrieval is called concept-based information retrieval.

In concept-based information retrieval, terms (words or phrases) that can be identified directly from texts are mapping into concepts, which are abstract mental representations or meanings of the terms. Unlike the keyword-matching scheme, where documents are regarded as lists of terms, the concept-matching scheme takes documents as lists of concepts. The process to evaluate the relevance of a document and a query is based on the identification of the concepts in the document.

From a cognitive point of view, each term, i.e., a word or phrase, is related to a concept. In real life, a person links the information in a document to a specific topic by recognizing multiple terms that describe concepts associated with this topic. The terms that appear in a document "activate" terms known to the person that are related to the topic's concepts. In such a way, a net of relevant words is activated. The presence of those words in the document determines what the document is about. The more words that are activated, the more confidence the person has that the document contains a specific concept.

Concept-based information retrieval is a new vision. Documents are retrieved through meanings instead of keywords. Concept-based IR models are more "intelligent" than keyword-based models. In concept-based models, search agents try to "understand" the meanings in query texts and attempt to retrieve documents relevant to these meanings.

1.4 Goals and approaches

In this work we attempt to create a complete concept-based information retrieval framework. We first define concepts in a query, then identify these concepts in Web documents.

Concepts are abstract and can not be recognized directly from texts. Concepts must be translated into a set of concrete objects, that is, terms. Thus the identification of an abstract concept becomes the identification of terms in a Web document. The more complete and accurate the definitions of concepts are in a query, the more likely they will be identified in Web documents. We approached this problem as follows:

1. A knowledge base (KB) that contains definitions of concepts used to enrich concepts in queries provided by users was built. In our framework, the knowledge base was built based on an *ontology* (section 2.1.2), that is, a specification of a conceptualization. An ontology contains sets of concepts, terms and their relations to a domain. It provides an organized way to present vocabulary in a specific domain. With a novel adaptive assignment of term importance (AATI) scheme (section 3.1), each piece of knowledge in the knowledge base was assigned values according to its importance. The definitions of a concept includes: (1) terms that define

the concept and the importance of each term in defining the concept[2]; and (2) related-concepts and their importance in defining the primary concept [3].

2. The form of the query is crucial. In a concept-based system, a query must be processed as an organization of concepts. Here the HofC (hierarchy of concepts, section 2.1.4) is applied to represent queries. A HofC contains not only a list of aimed concepts but also their relations. A hierarchical structure presents concepts more meaningful than a flat structure of conventional queries. Moreover, the HofC-based query can be expanded smoothly with the knowledge stored in the knowledge base which is also in a hierarchical structure.

3. The way to evaluate how well a document matches a query is not straightforward. Matched terms in the document provide pieces of information to satisfy definitions of concepts in the query. The satisfaction of the query is the aggregation of the satisfaction of concepts in the query. The hierarchical structure makes queries more complicated, which in turn makes the aggregation more complicated. Ordered weighted averaging (OWA) operators (section 2.1.3) are applied to handle this difficulty. The output of the evaluation is a float number that represents the score of the document. Documents are ranked according to their scores.

1.5 Thesis contributions

We designed and evaluated a new concept-based information retrieval methodology, called Hybrid Ontology-based Textual Information Retrieval (HOTIR). HOTIR is comprised of three components as shown in Fig 1.3: a knowledge management (KMgmt) component that manages the knowledge base; a query processing(QryProc) component that processes queries; and an evaluation (Eval) component that evaluates (i.e., ranks) documents, respectively. In HOTIR, when a query enters, QryProc first translates a keyword-based query into a concept-based query. Then KMgmt provides supplementary knowledge to enrich the query. Finally, Eval ranks Web documents according to the expanded query.

To our knowledge, HOTIR is the only information retrieval model in which both the definitions and the identifications of concepts are in hierarchical structures. That is, hierarchical information is utilized in all three components

[2]In the knowledge base of the proposed framework, it is possible that every concept has concrete terms to define it. See section 4.2.1 for more details

[3]The related-concept is the concept that connects to the current concept.

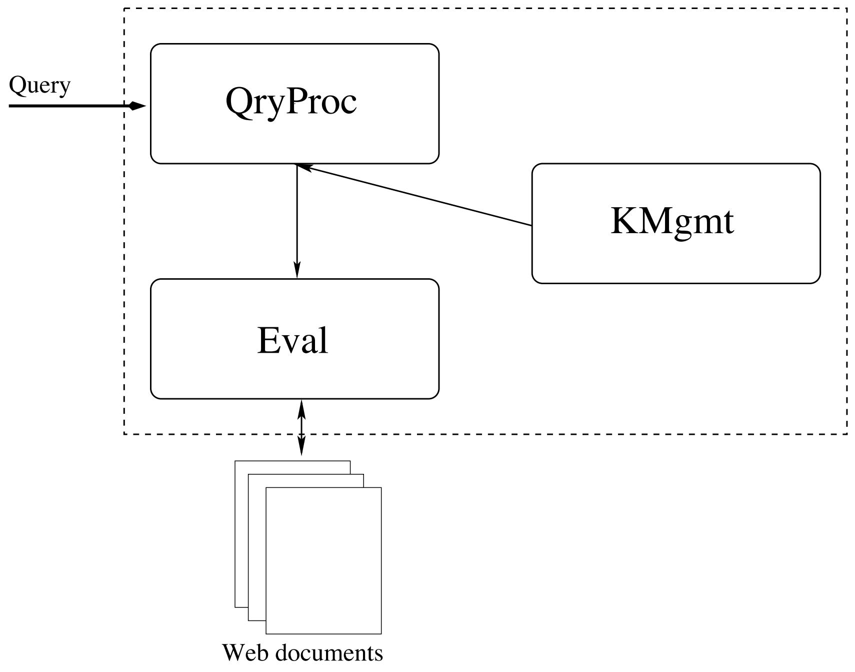


Figure 1.3: Components in the proposed model

(KMgmt, QryProc and Eval).

A hierarchy makes a knowledge base more efficient: the definitions of concepts can be extracted easily to enrich queries. For example, Fig. 1.4 illustrates how the concept *Sports* is defined with and without the help of hierarchies[4].

- In Fig. 1.4a, the concept *Sports* is defined by two related-concepts: *Ball* and *Track and field*. Concept *Ball* is defined by two related concepts: *Soccer* and *Basketball*. Concept *Track and field* is defined by two related concepts: *Race* and *Jump*.
- In Fig. 1.4b, the concept *Sports* is defined by six related-concepts:*Ball*, *Track and field*, *Soccer*, *Basketball*, *Race* and *Jump*.

Although each example includes the same number of concepts, the knowledge base shown in Fig. 1.4a, defines not only the concept *Sports* but also concepts *Ball* and *Track and field*, while the knowledge base shown in Fig. 1.4b only defines *Sports*. Therefore, if a query is provided to retrieve information about the concept *Ball*, the knowledge about *Ball* in the knowledge base in Fig. 1.4a can be directly utilized; while the knowledge about Ball in the knowledge base in Fig. 1.4b cannot.

[4]In Fig. 1.4, we list only concepts and assume that the concrete values (terms) defining them are embedded.

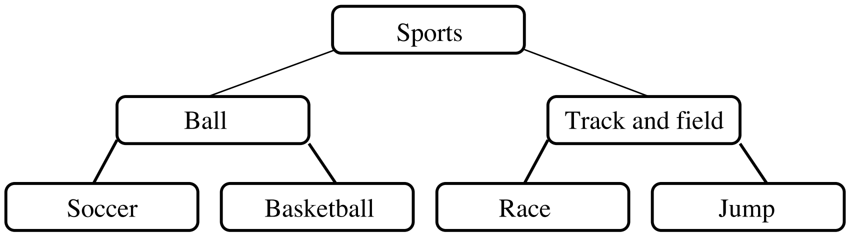
Second, the hierarchy makes query expansion smoother. Because the knowledge base contains hierarchies, it is necessary that queries are in hierarchical structures as well. This ensures that the concepts in queries can be naturally expanded with supplementary knowledge from the knowledge base.

Finally, a hierarchy provides chances to create more flexible criteria to retrieve required information. For example, after a query that contains the concept *Sports* is expanded with the two different knowledge bases presented in Fig. 1.4, it is translated into two queries: q_1, which takes information from Fig. 1.4a; and q_2, which takes information from Fig. 1.4b. Let us assume the single concepts listed in the queries (*Soccer*, *Basketball* etc.) are already identified in document d_1. To evaluate how well a document matches a query, we have to aggregate identifications of the single concepts in the document. Because the hierarchical structures in the two queries are different, the process of aggregation is different:

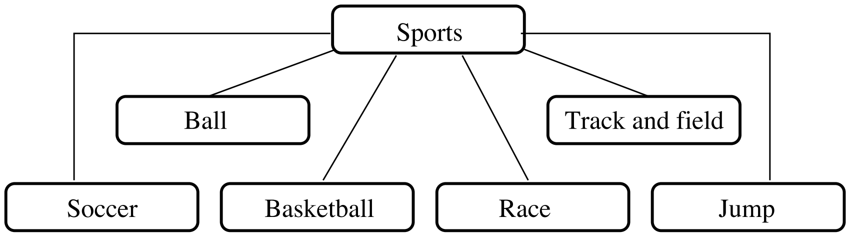
- q_1 includes three aggregations:
    1. an aggregation of the concepts *Soccer* and *Basketball* that contributes to the identification of the concept *Ball*;
    2. an aggregation of the concepts *Race* and *Jump* that contributes to the identification of the concept *Track and field*;
    3. an aggregation of the concepts *Ball* and *Track and filed* that contributes to an identification of the concept *Sports*;
- q_2 includes only one aggregation: an aggregation of all single concepts.

Compared with a "flat" query (like q_2), the evaluation of a hierarchical query (like q_1) is more difficult. However, the hierarchy, integrated with linguistic quantifiers, expands the possibilities of queries to express more complicated requirements. Examples of linguistic quantifiers are *some*, *most* and *all*. In OWA operators (section 2.1.3), which are applied to evaluate documents in HOTIR, linguistic quantifiers are modelled. For instance, a query can be defined as: ***all*** information about *Soccer* and *Basketball* in *Ball*; ***some*** information about *Race* and *Jump* in *Track and field*; and ***most*** information about *Ball* and *Track and field* in *Sports*.

In summary, our first contribution is the design and construction of a new concept-based information retrieval framework – HOTIR. HOTIR takes queries from users, communicates with the knowledge base, expands HofC-based queries (section 2.1.4), parses and analyzes Web documents (in HTML format) through semantic annotation (section 4.2.3), and ranks documents by integrating OWA operators (section 2.1.3). We have created a complete concept-based information retrieval solution.



(a) With hierarchy



(b) Without hierarchy

Figure 1.4: The definition of concept *Sports*

Our second contribution is building the knowledge base for HOTIR, called ModOnt(section 4.2.1), based on the ontology. The ModOnt provides supplementary knowledge to queries, defining *concept* or *class*[5] differently than the classic ontology (discussed in section 4.2.1). The ontology alone was not a sufficient knowledge base for HOTIR because concepts and terms did not contain importance information. Therefore, a new AATI scheme was designed and implemented to handle this problem. The ontology provides lists of concepts, relations and terms that directly define concepts, and the importance of terms as assigned by the AATI which in turn contributes to the importance of concepts. Thus, the ModOnt is capable of presenting complete definitions of concepts. The AATI continuously updates the importance of terms with "unknown" Web documents, making it appropriate for Web applications. Because changing knowledge causes problems for Web-based applications, important terms often become obsolete; for instance, "DOS" in the area of IT. Unlike other approaches, the Web documents the AATI uses for updating importance values have no preset information of the documents; this saves human labor.

A third contribution is implementation of HofCs in query representation. A HofC is capable of storing more information than a conventional query as it includes hierarchies. A query-enrichment process is implemented to expand the HofC with knowledge from the ModOnt. The hierarchy is important for defining or identifying concepts, but it introduces troubles to all processes

[5]Concept or class is a major component in the ModOnt or ontology.

in information retrieval models, such as query construction, query expansion, matching evaluation, and so on. We successfully utilized hierarchy information in HOTIRS to retrieve relevant documents.

The fourth contribution we make to the field is utilizing OWA operators (section 2.1.3) to rank Web documents according to HofC-based queries. OWA operators aggregate pieces of information in documents to satisfy definitions of concepts in queries, which supports different linguistic quantifiers as instructions and hierarchical structures. OWA increases the expressiveness of queries.

1.6 Thesis structure

The thesis is organized as follows:

Chapter 2 contains two sections. The first section provides necessary background knowledge for the work; it includes information retrieval, ontology, ordered weighted averaging (OWA) operators, hierarchy of concepts (HofC), and text categorization. The second section discusses some existing concept-based information retrieval tools.

Chapter 3 introduces a self-adaptive weighting scheme called AATI and presents its validations.

Chapter 4 introduces the methodology of the work. There are three agents in the framework. The KMgmt agent manages the knowledge base, integrating the ontology with AATI. The QryProc agent translates input queries into HofC-based queries and expands HofC-based queries with the knowledge base. The Eval agent ranks documents according to how well they match a query.

Chapter 5 presents case studies that demonstrate how HOTIR ranks documents based on the identifications of concepts; two documents are scored by HOTIR with two queries. The step-by-step calculations in the process expose the details involved in evaluation.

Chapter 6 provides experiments to verify HOTIR. The experiments are in two series. In the first series of the experiments, HOTIR-based approaches and two additional methods rank Web documents in the local repository with six queries. The performances are evaluated by calculating retrieved relevant documents. In the second series of the experiments, five queries are used to compare HOTIR-based approaches with Google. The performance of each approach is evaluated by comparing the ranking of documents with rankings generated by five human experts.

In chapter 7, conclusions derived from the work in this thesis are presented and future projects are suggested.

Chapter 2

Background and related work

2.1 Background

The topics presented in this chapter provide an introduction to the thesis. *Information retrieval* (IR) (section 2.1.1) is the main target of our investigation. We aim to build a model for information retrieval that can deal with the data explosion on the Web. The *ontology* (section 2.1.2) is the knowledge base from which knowledge must be retrieved. It is comprised of sets of concepts and their relations. *Ordered weighted averaging* operators (OWA) (section 2.1.3) aggregate pieces of information under linguistic instructions. The *hierarchy of concepts* (HofC) (section 2.1.4) presents queries in hierarchical structures that can assist information retrieval. *Text categorization*, introduced in section 2.1.5, is used to test the adaptive assignment of term importance (AATI) scheme (chapter 3). Related work is described in section 2.2.

2.1.1 Information retrieval

Information retrieval (IR) extracts useful information out of unstructured data which are mainly in the form of textual documents. Information retrieval follows the acceptance of a query from a user; a query is a statement describing information needs. Some documents in the information repository match the query but with different degrees of relevance. Information retrieval models score documents on how well they match the query, and rank them according to the scores.

Information retrieval approaches can be categorized into three types: boolean, vector space, and probabilistic.

The boolean approach to information retrieval

Queries are the key element of the Boolean approach. A query is a boolean combination of terms. With popular operators like AND, OR, and NOT, the query can be in the form of "t1 AND t2," "t1 OR t2,", "t1 NOT t2," and so on. For the query "t1 AND t2," a document will be retrieved only when it contains both the term "t1" and "t2". For the query "t1 OR t2," a document will be retrieved only when it contains either term "t1" or "t2" or both terms. For the query "t1 NOT t2," a document will be retrieved only when it contains the term "t1" and not the term "t2". In the evaluation of the relationship between query and document, there are only two possibilities (boolean values) for each document, *satisfied* or *not-satisfied*, according to how the query has been dealt with.

Although the classical boolean approach is straightforward, users may still experience difficulties in constructing queries [8], and the strict "yes" or "no" outcome of the retrieved documents cause issues in real world problems.

Salton proposed a system as an extension of the classical boolean approach, which extended queries with the help of statistical results out of documents themselves [9]. After the normal preprocessing, like stemming and removal of the stopping words, Salton obtained a set of terms which were ORed together. Then, Salton looked for pairs (and triples) of the terms that occurred in the multiple documents. Since two or three terms might occur in a document by chance, Salton used a formula for pair correlation to determine if the two or three terms co-occurred more frequently than chance. If the two (a pair) or three (a triple) of the terms had the computed correlation over a predefined threshold, they were grouped with a boolean AND. For instance, if two terms t_i and t_j were regarded as a pair, i.e, they had computed correlation over the threshold, then the query became from (assuming n terms):

t_1 OR t_2 OR ...... OR t_i OR ...... OR t_j OR .... OR t_n

to

t_1 OR t_2 OR ...... OR t_i OR ...... OR t_j OR .... OR t_n OR (t_i AND t_j)

In a classical boolean model, the output is always "yes" or "no." This is an advantage but also a drawback. For example, in a query connecting terms with the AND operator, the model treats the documents including some (not all) of the terms as unfavorably as those including none of the terms. In a query connecting terms with the OR operator, the model treats the documents including all of the terms as favorably as those including one term.

In the extended Boolean model, the ability to rank documents based on queries is developed. That is, documents can be ranked according to their ability to match the query. The model is based on extended (soft) boolean operators. The output of classical boolean operators is evaluated as either false or true, assigning values of zero or one, respectively. However, the extended boolean operators evaluate arguments as float numbers from zero to one. These numbers rank the association between the document and the query.

The most popular extended boolean model was the *p-norm* approach introduced by Salton [10]. The kernel of the p-norm model is the construction of functions for operators AND and OR. The output of this extended operator (the function) is numerical, which guarantees the output of the matching degree is also numerical. Given a query Q_1 of n terms t_1, ... ,t_n with corresponding weights w_{q1} ,..., w_{qn} for each term and a document D_1 with corresponding weights w_{d1} ,..., w_{dn} of the same terms, the extended boolean AND function is:

$$SIM_{AND}(D_1,(t_1,w_{q1})AND...AND(t_n,w_{qn})) = 1-\left(\frac{\sum_{i=1}^{n}((1-w_{di})^p \cdot {w_{qi}}^p)}{\sum_{i=1}^{n} {w_{qi}}^p}\right)^{\frac{1}{p}}$$

where p is a parameter to tune the model, and $0 \leq p \leq \infty$.

The extended boolean OR function is:

$$SIM_{OR}(D_1,(t_1,w_{q1})OR...OR(t_n,w_{qn})) = \left(\frac{\sum_{i=1}^{n} {w_{di}}^p \cdot {w_{qi}}^p)}{\sum_{i=1}^{n} {w_{qi}}^p}\right)^{\frac{1}{p}}$$

where p is a parameter to tune the model, and $0 \leq p \leq \infty$.

The range of p is from zero to ∞. When $p = \infty$, the model turns into a classical boolean model. That is, the AND function will not be zero only when all of the terms are presented; the OR function will be one when any one term is presented. When $p = 1$, the AND and OR functions are identical. When p is one of some normal numbers like 2, 3, or 4, the p-norm shows an ability to overcome the mentioned drawbacks of the classical boolean model. In the AND function, “the presence of all phrase components is worth more than the presence of only some of the components; terms are not compulsory” [10]. At the same time, in the OR function, “the presence of several terms from a class is worth more than the presence of only one term” [10]. Therefore, users can define how strictly they want from an operator in their models through defining a different p. Besides, the p-norm model permits users to define the weights to both query and document terms, which provides a simple way of representing a user’s knowledge/interest in information retrieval. For example, in the query (t_1,0.1) AND (t_2,1), the user gives more weight to the term t_2.

Though experiments show that extended boolean models perform much better than classical boolean models, extended boolean models are too heavy in some cases when the term weights, parameters, and computation are considered.

Vector space approach

classical vector space approach The vector space approach is based on using a set of terms to represent a textual document/query. Normally, the terms are words, phases, or even several letters, which are extracted from a document/query after stemming and eliminating the stopping words. Therefore, each document in the corpus can be represented in a vector. The corpus is a space, in which each term is one dimension.

Each term in the set has a float number value as a term weight. The weight represents how well this term can distinguish one document from another in the corpus. A matrix M is constructed where each row represents one document, while each column represents one term. The element α_{ij} of the matrix is the weight of the jth term in the ith document. If the term does not occur in the document, the value of α_{ij} is zero.

The user's query is also presented by a set of terms. The terms are all from the same set in the corpus so that the query and the document in the corpus are comparable. That is, if new terms occur in the query they will be ignored.

The problem is how to assign weights to the terms. The most popular way to assign weights is $tf \cdot idf$, where tf represents the term frequency and idf represents the inverse document frequency. *Term frequency* (tf) is the frequency that one term occurs in one document. *Inverse document frequency*(idf) is the metric calculated by the frequency of one term in the whole corpus. As presented in [11], the definitions of tf and idf are:

$$tf_{ij} = \frac{n_{ij}}{\sum_k n_{kj}}$$
$$idf_i = \log \frac{n_d}{n_i}$$

where n_{ij} is the frequency of term i in document j; k is the number of different terms; n_i is the number of documents containing term i in the corpus; n_d is the total number of documents in the corpus.

From the definition of idf, the value of idf_i will be zero when every document in the corpus contains the term t_i; and it will be a larger number when fewer documents contains it. Thus t_i has more power to identify a document when it occurs in fewer documents of the corpus. Therefore, the definition of $tf \cdot idf$

Table 2.1: Term weighting scheme

Name	Description
$tf \cdot idf$	$tf \cdot idf$
$logtf \cdot idf$	$\log(1 + tf) \cdot idf$
$tf \cdot idf - prob$	probability idf [14]

is:

$$(tf \cdot idf)_{ij} = tf_{ij} \cdot idf_i$$

In the $tf \cdot idf$ scheme, term t_i in the document d_j obtains heavier weight when it has high frequency in the d_j and low frequency in the rest of the documents in the corpus.

In addition to $tf \cdot idf$, there are other different schemes for evaluating term weights. Lan and Tan [12] and Montanes et al. [13], tried methods to determine word importance extensively, where *TF-IDF*, *logTF-IDF*, *TD-IDF-prob* and *TF-RF* were compared. Table 2.1 presents several usual weighting schemes.

The similarities between two vectors can be calculated after weights of the terms are assigned. *Distance* is the most intuitive metric to evaluate the difference between two vectors. Euclidean Distance [15] is a mathematical way to present the distance between two vectors. Hence, in the space based on the "term" dimension, it is easy to calculate the distance between the query and the given document.

The *inner product* between a query vector and a given document vector is a popular measure [10]. Given that w_{qt_i} is the term weight of the term t_i in the query, and $w_{d_jt_i}$ is the term weight of the term t_i in the document d_j, the inner product of the query and the document d_j is:

$$I_j = q \cdot d_j = \sum_{i=1}^{k} w_{qt_i} \cdot w_{d_jt_i}$$

where I_j is the value of the inner product of the query and document d_j, and k is the number of different terms, that is, the number of dimensions of the corpus. The larger value of inner product represents that two vectors tend to be more similar or relevance.

The n-gram approach to information retrieval Unlike the classical vector-based approach, the n-gram approach is purely statistical [16]. The primary difference between the n-gram approach and the other vector-based approach is the definition of *term*. In the previous sections, the terms are words or phases in the document or query. However, here a term is a string of n consecutive characters.

Damashek built a sliding window approach with the n-gram idea [16, 17]. As the name represents, the approach is like moving a n-character-length window in a document to generate n-character-length terms. Therefore, the first term is the first nth alphabet character in the document, and the second term is the second to the $(n+1)$th alphabet character. Damashek used normalized n-gram frequency, which is the original n-gram frequency divided by the total number of n-gram terms in the document. Then, the inner product was applied to compute the similarity between two documents.

The n-gram approach depends very much on statistics rather than the linguistic knowledge in information retrieval; statistics have been shown to be quite effective. Statistics are valuable in a situation where a system developed in one language has to be extended to a multi-language system. However, this virtue "limits its performance compared to methods that make effective use of language-specific as well as statistical clues" [16].

The probabilistic approach to information retrieval

The probabilistic approach applies probability theory to information retrieval. The probabilistic model translates the posterior probabilistic, i.e., the relevance of the conditional probability of a document given the features of the document, into the prior probability, i.e., the probability that a document in the repository will be relevant. The transformation is based on the Bayes' theorem and some assumptions are made to simplify the problem. The Bayes' theorem is as follows,

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \tag{2.1}$$

Since the simplification usually assumes that each of two terms in the documents are independent, the approach is often called the naive Bayesian approach. In variants of the naive Bayesian approach, the binary independence model(BIM) is the most widely used [18]. In the model, both documents and queries are represented by a set of terms. Let us assume the set of terms has n elements, document d_1 is a vector $\overrightarrow{d_1} = [t_1, t_2, ..., t_n]$, where $t_i = 0$ when term

i does not occur in document d_1 and $t_i = 1$ when term i occurs in d_1. The representation of the query is similar. Another assumption in the BIM is the occurrence of terms in the documents that are independent. This assumption is significant, however, Manning et al. pointed out that "this assumption is far from correct, but it nevertheless often gives satisfied results in practice" [19].

2.1.2 Semantic Web and ontology

The Semantic Web [20], an extension of the World Wide Web (WWW), makes Web contents more machine-processable. The Semantic Web is defined by the World Wide Web Consortium (W3C). Tim Berners-Lee, who invented the WWW in the late 1980s, introduced this special vision of the Web, in which the meaning of the content in Web documents would play a much more crucial role than it does today.

HTML (hypertext markup language) is the predominant language in the current WWW. It satisfactorily fulfills its responsibility, which is presenting information to humans. The Semantic Web will need a more firmly structured language for machine agents to process. XML (extensible markup language) [21], though not sufficient to support the Semantic Web, is a crucial first step for the adaptation of the language. RDF (resource description framework) [22] becomes the current W3C standard for the Semantic Web.

XML is a structured markup language [23]. It allows people to define their own tags to represent information in a structured way that will facilitate machines to better process the information. The RDF is more a model than a language; it is designed to present information about Web resources. RDF presents statement in the form of triples, i.e., *subject-predicate-object*. RDF is applied to build information-sharing models. In the triples, the *subject* denotes the resources, the *object* denotes the properties of the subject, and the *predicate* denotes the relations between the subject and the object. The RDF model can be serialized in XML format.

The most popular definition of an ontology, in the context of the Semantic Web, is "an explicit and formal specification of a conceptualization of a domain of interest" [24]. Ontologies are more than just a vocabulary, they are sources of knowledge of a specific domain. Currently, most of ontologies are implemented in OWL (Web ontology language) which is based on RDF and designed by W3C. There are three types of OWL: OWL Lite, OWL DL and OWL Full. OWL Lite provides a limited feature set, but is relatively efficient. OWL DL, a superset of OWL Lite, is based on a form of first order logic (description logic). OWL Full, a superset of OWL DL, removes some restrictions from OWL DL, which introduces problems with computational tractability.

The most important aspect of ontologies used for Semantic Web applications is related to identifying two ontology layers: *the ontology definition layer*, and *the ontology instance layer*.

The ontology definition layer represents a framework used for establishing an ontology structure and for defining classes (concepts) existing in a given domain. The structure of an ontology is primarily based on a relation *is-a* between classes. This relation represents a *subClassOf* connection between a superclass and a subclass. In such a way, a hierarchy of classes is built.

The ontology definition contains descriptions of all classes of the ontology. The classes are defined using datatype properties and object properties. Both property types provide an accurate and complete description of a class, as described below:

- The *datatype property* focuses on describing features of a class; datatype properties can be expressed as values of data types such as boolean, float, integer, string, and many more (for example, byte, date, decimal, time);
- The *object property* defines other than *is-a* relations among classes (nodes); these relations follow the notion of the RDF that is based on the triple subject-predicate-object, where: *subject* identifies what object the triple is describing; *predicate* defines the piece of data in the object a value is given to; and *object* is the actual value of the property; for example, in the triple John likes books, John is the subject, likes is the predicate and books is the object.

Both types of property are important for defining ontologies. The possibility of defining class properties and relations between classes creates a versatile framework capable of expressing complex situations with sophisticated classes and the multiple different kinds of relationships existing among them.

Once an ontology definition is constructed, its instances, called individuals, can be built. The properties of classes are filled out: real data values are assigned to datatype properties, and links to instances of other classes (individuals) are assigned to object properties.

Ontologies are divided into *top-level* and *domain* types [25]. Top-level (upper) ontologies are the model of most common concepts in a wide range of domains. That is, they describe general concepts that exist in many different domains. Top-level ontologies are normally connected directly or indirectly to many other ontologies. A domain ontology is a model of a specific domain, such as science, animals, sports, etc. Particular meanings of concepts/terms in the represented domain are defined in the domain ontology.

An ontology example is presented in Fig. 2.1. The ontology was developed on *Protege*, an open source ontology editor[1].

The ontology contains three classes, which are *Soccer*, *Soccer_Club* and *Soccer_Class*. Because in Protege every class is inherited from a public class *owl:Thing*, there are four classes presented in Fig. 2.1a. The relations of these four classes are: *Soccer* is the *subClassOf owl:Thing*; and both *Soccer_Club* and *Soccer_Player* are *subClassOf Soccer*. Class *Soccer_Club* includes two individuals: *Club_Chelsea* and *Club_FC_Barcelona*; Class *Soccer_Player* includes two individuals: *Player_Drogba* and *Player_Terry*.

Class *Soccer_Player* is defined by three data properties and one object property, as presented in Fig. 2.1b. The three data properties are *First_Name*, *Last_Name* and *Position*. The one object property is *InClub*.

Class *Soccer_Club* is also defined by three data properties and one object property. The three data properties are *Name*, *Stadium* and *Location*. The one object property is *HasPlayer*.

As shown in Fig. 2.1b, *Player_Drogba*, an individual of class *Soccer_Player*, is described in the ontology by terms “Didier” (the value of the data property *First_Name*), “Drogba” (the value of data property *Last_Name*), “Forward” (the value of data property *Position*), and an instance *Club_Chelsea* (the value of object property *InClub*), which is an instance of class *Soccer_Club*.

2.1.3 Ordered weighted averaging (OWA) operators

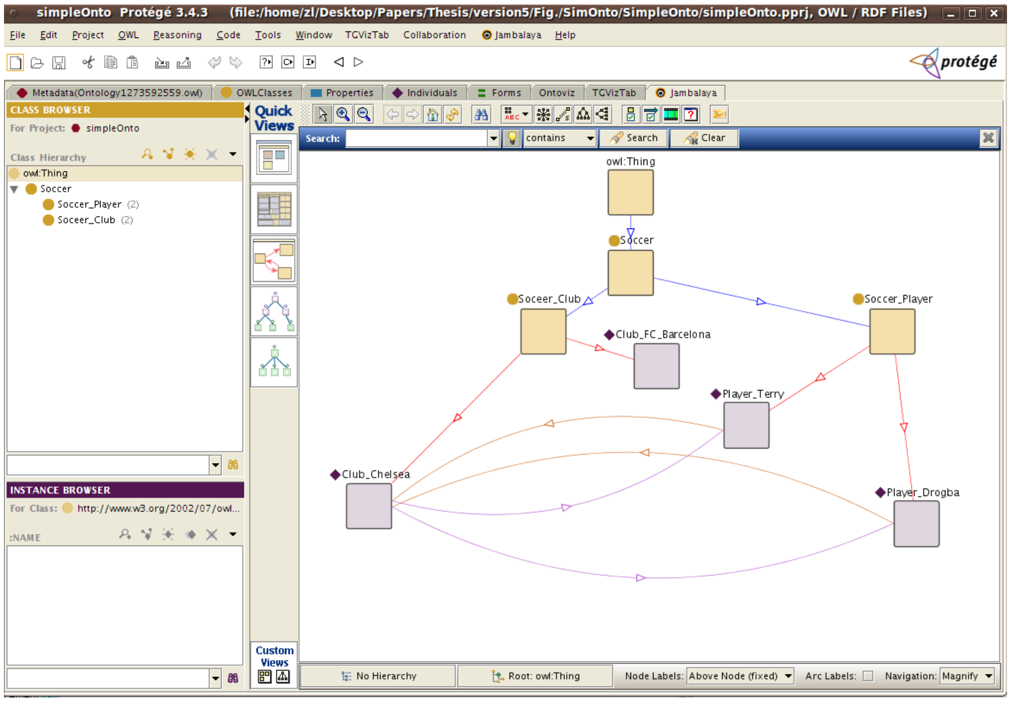
Basic principles of OWA

Aggregation of different pieces of information is a common aspect of any system that has to infer a single outcome from multiple facts. An interesting class of aggregation, ordered weighted averaging (OWA) [26] operators, is a weighted sum over ordered pieces of information.

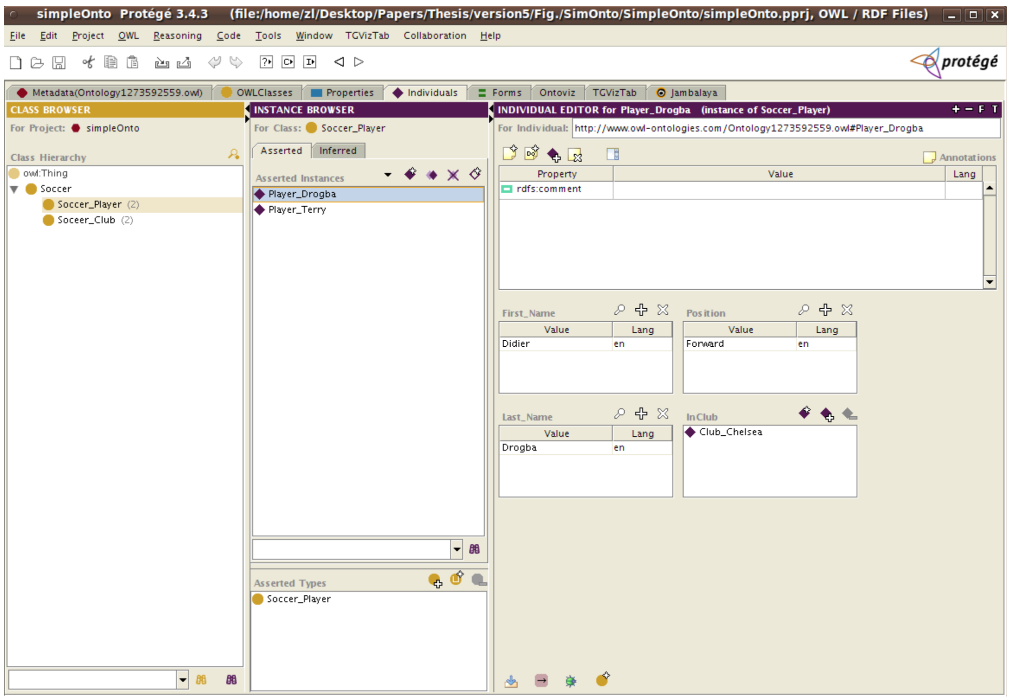
In a formal representation, the OWA operator, defined on the unit interval I and having dimension n (n arguments), is a mapping $F_w : I^n \rightarrow I$ such that

$$F_w(a_1, ..., a_n) = \sum_{j=1}^{n}(w_j \cdot b_j) \tag{2.2}$$

[1]More details can be found in the website:http://protege.stanford.edu/.



(a)



For Individual: http://www.owl-ontologies.com/Ontology1273592559.owl#Player_Drogba

Property	Value	Lang
rdfs:comment		

First_Name

Value	Lang
Didier	en

Position

Value	Lang
Forward	en

Last_Name

Value	Lang
Drogba	en

InClub

Club_Chelsea

(b)

Figure 2.1: An ontology example by Progete

where b_j is the jth largest of all arguments a_1, a_2, ..., a_n, and w_j is a weight such that w_j is in [0,1] and $\sum_{j=1}^{n} w_j = 1$.

If $id(j)$ is the index of the jth largest of a_i then $a_{id(j)} = b_j$ and $F_w(a_1, ..., a_n) = \sum_{j=1}^{n}(w_j \cdot a_{id(j)})$. If W is an n-dimensional vector whose jth component is w_j, and B is an n-dimensional vector whose jth component is b_j, then $F_w(a_1, a_2, ..., a_n) = W^T B$. In this formulation, W is referred to as the OWA weighing vector and B is called the ordered argument vector.

The OWA operator is parameterized by the weighing vector W. A number of interesting observations can be made when different W values are considered. For example, if $W = W_*$, where $w_n = 1$ and $w_j = 0$ for $j \neq n$ then $F_w(a_1, a_2, ..., a_n) = Min_j[a_j]$; If $W = W^*$, where $w_1 = 1$ and $w_j = 0$ for $j \neq 1$ then $F_w(a_1, a_2, ..., a_n) = Max_j[a_j]$; If $W = W_N$, where $w_n = \frac{1}{n}$, then $F_w(a_1, a_2, ..., a_n) = \frac{1}{n}\sum_{j=1}^{n} a_j$, which represents an arithmetic mean(average).

Various other forms of OWA operator can be described. Therefore, $Min[a_j] \leq F_w(a_1, a_2, ..., a_n) \leq Max[a_j]$. In general, it can be said that different values of weights w_j control the level of contribution of single pieces of information toward the final outcome.

At the beginning of the 1980s, Zadeh [27] introduced the concept of linguistic quantifiers. Those quantifiers describe a proportion of objects. According to Zadeh, a person knows a vast array of terms that are used to express information about proportions. Some examples are *most*, *at least half*, *all* and *about one third*. The important issue is to formally represent those quantifiers.

In the mid-1990s, Yager [28] showed how we can use a linguistic quantifier to obtain a weighing vector associated with an OWA aggregation. Yager introduced parameterized families of regular increasing monotone(RIM) quantifiers. These quantifiers are able to guide aggregation procedures by verbally expressed concepts in a description independent dimension. A RIM quantifier is a fuzzy subset Q over $I = [0, 1]$ in which for any proportion $r \in I$, $Q(r)$ indicates the degree to which r satisfies the concept indicated by the quantifier Q [28]. A fuzzy subset Q represents a RIM quantifier if:

1. $Q(0) = 0$
2. $Q(1) = 1$
3. if $r_1 > r_2$ then $Q(r_1) > Q(r_2)$ (monotonic)

Assuming a RIM quantifier, we can associate with Q an OWA weighing vector W such that for $j = 1$ to n,

$$w_j = Q(\frac{j}{n}) - Q(\frac{j-1}{n}) \tag{2.3}$$

where n is a number of pieces of information to be aggregated. This expression indicates that the weighing vector W is a manifestation of the quantifier underlying the aggregation process. Using this expression the values of the weighing vector can be obtained directly from the expression representing the quantifier.

For example, let us take a look at the parameterized family $Q(r) = r^p$, where $p \in [0, \infty)$. Here if $p = 0$, $w_1 = 1$ and $w_j = 0$ for $j \neq 1$, therefore $W = W^*$ and we obtain the $existential(max)$ quantifier, which makes the OWA operator F closer to an *or* operator; when $p- > \infty$, $w_n = 1$ and $w_j = 0$ for $j \neq n$, therefore $W = W_*$ and we have the quantifier $forall(min)$, which makes the OWA operator F closer to an *and* operator; and when $p = 1$ we have $Q(r) = r$, $w_j = \frac{1}{n}$ and we deal with the quantifier *some*.

In [26], an interesting measure for evaluating the degree of the *orness* of the OWA operator was been introduced. Let us assume F is an OWA aggregation operator with a weighing vector $W = [w_1, w_2, ..., w_n]$. The degree of *orness* associated with this operator is defined as:

$$orness(W) = \frac{1}{n-1} \sum_{i=1}^{n} (n-j) \cdot w_j \tag{2.4}$$

The *orness* of the parameterized family $Q(r) = r^p$, where $p \in [0, \infty)$ is approximated by $1/(p+1)$. So, for the quantifiers $forall$ (W_*), *some*$(w = 1/n)$ and $existential(W^*)$, *orness* is equal to 0, 1/2, and 1/3 respectively.

OWA with Argument Importance

In the previous section we showed how a quantifier Q indicating interaction between pieces of information can be used to calculate an OWA weighing vector W. However, not all pieces of information are of the same importance. A user may desire to assign different weights (importance) to different arguments (pieces of information).

Let $m_i \in [0, 1]$ be a value associated with an argument a_1 indicating its importance. In such a case, let M be a n-dimensional importance vector $[m_1, m_2, ...m_n]$, and the weighing vector W has to be calculated based on both Q and M.

The first step is to calculate the ordered argument vector B (Eq. 2.2), such that b_j is the jth largest of all arguments a_1, a_2, ..., a_n. Furthermore, we assume μ_j denotes the importance weight associated with the attribute that has the jth largest value. Thus if a_3 is the largest value, then $b_1 = a_3$ and $\mu_1 = m_3$. The next step is to calculate the OWA weighing vector W using a modified version of Eq.(2.3),

$$w_j = Q(\frac{X_j}{T}) - Q(\frac{X_{j-1}}{T}) \tag{2.5}$$

where $X_j = \sum_{k=1}^{j} \mu_k$ and $T = X_n = \sum_{k=1}^{n} \mu_k$.

So, X_j is a sum of the importance of the jth most satisfied arguments, and T is the sum of all importance. When all arguments have the same importance, Eq. (2.5) simplifies to Eq. (2.3).

2.1.4 Hierarchy of concepts (HofC)

The idea of representing concepts as a hierarchy was introduced by Yager [29], who represented concepts with atomic attributes, words or other concepts. Using this method, a tree-like structure is established where each vertex is a concept, and terminal vertices (leaves) are attributes. The edges of the hierarchy of concepts (HofC) represent relationships that help to define concepts with other concepts and/or attributes. These edges (connections) are of significant importance to the HofC. If we assume that concept C_1 is defined by two other concepts C_2 and C_3, then the hierarchy will have two edges connecting C_2 with C_1, and C_3 with C_1. The concept C_1 is called a superconcept, and C_2 and C_3 are subconcepts. This also means that *activation* of concepts C_2 and C_3 leads to activation of C_1.

The HofC introduces a very important element, the activation of a superconcept by active sub-concepts is fully controlled by a user. There are two controlling components: importance vector M and linguistic quantifier Q. The vector M indicates the significance of each subconcept in defining a superconcept. In other words, M determines the weight of each participating subconcepts in identifying an activation level of a superconcept. The linguistic quantifier Q guides the aggregation of subconcept activations. Both M and Q determine how activation levels of subconcepts should be combined using the OWA operator.

A simple example of HofC is shown in the Fig. 2.2. According to the hierarchy, *conceptA* is defined as:

$$conceptA = (conceptB, conceptC, M_A, Q_A)$$

This means that *conceptA* is defined by *conceptB* and *conceptC*. M_A determines the importance of both of *conceptB* and *conceptC* in defining *conceptA*.

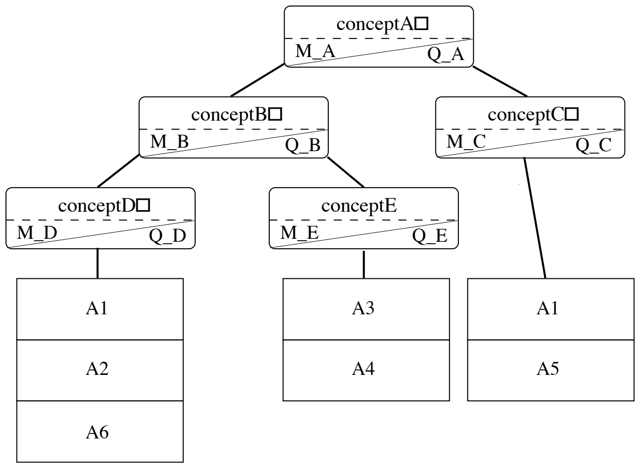


Figure 2.2: A example of a simple HofC

In this case, it is a two-value vector $M_A = [M_{A-B}, M_{A-C}]$ that implies the importance of activations of both subconcepts during calculation of the activation level of the *conceptA*. The quantifier Q_A can be of any type, for example, *most* or *some*, and identifies a mechanism of combining activation levels of subconcepts. The rest of the concepts are defined in the following way,

$$\begin{aligned}
&\textit{conceptB} = (\textit{conceptD}, \textit{conceptE}, M_B, Q_B)\\
&\textit{conceptD} = (A_1, A_2, A_6, M_D, Q_D)\\
&\textit{conceptE} = (A_3, A_4, M_E, Q_E)\\
&\textit{conceptC} = (A_1, A_5, M_C, Q_C)
\end{aligned}$$

As we can see, *conceptD*, *conceptE*, and *conceptC* are defined by attributes only. Activation levels of these concepts are calculated by aggregating activations of attributes. Activation of an attribute means that the attribute is present, for example, in a Web page. The aggregation process for each concept is controlled via the M and Q associated with that concept. For *conceptD*, aggregation of activations of attributes A_1, A_2, and A_6 is performed by the OWA operators with a weighing vector determined by M_D and Q_D.

2.1.5 Text categorization

Text categorization is the process of assigning text documents into preset categories. It is similar to information retrieval in that normal text categorization problems can be called information retrieval in some situations; for instance, when a query itself is regarded as a description of the topic to be searched.

However, text categorization and information retrieval have different goals. The goal of information retrieval is to retrieve documents relevant to a query, while the goal of text categorization is to retrieve documents relevant to a topic or a concept. Unlike information retrieval, where there are key words in a query, there are no direct identifiers in text categorization. The knowledge of the category is obtained through training the machine learning (ML) models on the documents in the category. Information retrieval is directed by specific keywords in a query. Text categorization focuses on the semantics of each category. That is, in the traditional point of view, *text categorization deals with concepts, while information retrieval deals with terms.* Therefore, compared with information retrieval, text categorization has more chances and more motivation to apply computational intelligence techniques as the problems are more abstract.

Until the 1980s, the most widely used approach in text categorization was creating rules for embedding human experts' knowledge of a category. This approach depends on human knowledge in the specific category for which rules were being embedded. The same system did not always perform consistently when applied to a different category. In the 1990s, automatic text categorization (ATC), based on machine learning techniques, began to get the attention of researchers in the field. ATC soon became the dominant approach in text categorization because of its obvious advantages. We will not use the ATC acronym in the rest of the paper; we will use the spelled out form "text categorization" because currently almost all of the text categorization research is based on ATC.

We focus on the text categorization approaches based on the ML techniques that have proved to be effective and practical. Approaches unrelated to ML techniques are beyond the scope of this paper.

The conventional process in text categorization is to first build an ML model based on some precategorized documents called a training set. The model obtains characteristics from the training set. Those characteristics are knowledge of the categories, which is stored in the model. The knowledge is either readable by humans (like rules in the rule-based system) or not readable by humans (like the structures of a neutral network). When the machine encounters a new document, the model decides if the document belongs to the category or not through the knowledge it has obtained from the training set. This ML process is called supervised learning.

The training process is the kernel of the text categorization approach. The training on the precategorized documents provides the category identifier. The features of the model bear the category identifier. For example, if the features are terms, the knowledge of categories normally includes term weights.

For each model, the category status value (CSV) is the output of one given document for one given category. The CSV is the metric that evaluates the matching degree between a document and a category. Normally, the CSV is a float number in the range of $[0, 1]$. Zero means the model decides that the document does not belong to the category, one means the model decides the document belongs to the category. Sometimes, to make it simple, CSV is an integer, zero or one, a binary situation.

Probabilistic approach

The methodologies of probabilistic approaches in text categorization are quite similar to those in information retrieval. The main difference is the rank of the relevance between a document and a topic must be found in text categorization and the rank of the relevance between a document and a query must be found in information retrieval. In text categorization, The lack of query keywords available in information retrieval necessitates a training set of precategorized documents for text categorization.

In the text categorization literature regarding probabilistic approaches [30, 31, 32, 33, 34], people mostly tried to develop probabilistic models that change independence assumptions. Some researchers tried to produce better classifiers by relaxing the independence assumptions [35, 36]. However, these new approaches require too much computation to be practical. Others tried to explain why the independence assumption is not needed. Cooper [37] argues that the independence assumptions can be replaced by a weaker *link dependence* assumption. In the new assumption, these features are not independent but hold the same degree of dependence in the given category. However, the probabilistic model is not good at handling the problem of the high dimensional features, which are a big obstacle for text categorization. The complexity and the efficiency of the model changes a lot with an increase in the number of the features, and the model can hardly be interpreted by humans.

Symbolic model

Symbolic models refer to models based on non-numeric algorithms. The most widely used symbolic models are rule-based and decision-tree models.

Rule-based model The rule-based models is represented by a set of logic rules that are disjunctions of conjunctive clauses. Normally a rule is in an IF...

THEN ... format like:

IF *rule* r_1 *is true* THEN *document* d_1 *in category* c_1

As an simple example, one rule for a category *soccer* could be:

IF *term "Ronaldo" occurs AND term "Score" occurs* THEN *the document is in the category soccer*

Each rule in a rule-based model is a criterion to divide the whole document set into two parts, that is, those documents making this criterion *True* or those making it *False*. The combination of rules will help the model decide which category a document belongs to. In the machine learning approach, the kernel is to find the best combination of the rules from the training set, *best* meaning accurate and not excessively complex [38]. Some variant rule-based models are SWAP-1[39], CHARADE[40], DL-ESC[41], and RIPPER [42].

Decision tree Model Decision tree models have two advantages. The first advantage is the global complex decision region is "approximated by the union of simpler local decision regions at various levels of the tree" [43]. The second advantage is that the decision tree is very efficient, especially in a multicategory situation, because in the normal models every sample is tested against all categories while in decision tree models samples are test only against a subset of categories. Each node in a decision tree model is focused on a certain subset of categories. The most widely used decision tree models include ID3[44], C4.5 [45] and C5 [46].

Rule-based models and decision tree models are transferable, that is, the rule-based models can be transferred into decision-tree models and vice versa. There are differences in the ways these models are created: decision-tree models are "typically built by a top-down, divide-and-conquer strategy", while rule-based models are often built in a bottom-up fashion" [47].

A drawback of decision-tree models is overfitting [48]. *Overfitting* happens when a model memorizes the categorization of a training set instead of learning general rules. In this case, the categorization of the training set will be very accurate while the categorization of unknown documents will be "unknown." The symbolic models are capable of partitioning the training set into a number of classes which are "much larger than the number of actual classes" [43]. This guarantees good results on the training set. However, it is hard to predict how the model will react to a new sample. This issue can be alleviated through feature selection, but that increases the complexity of these models.

Support vector machine(SVM)

The SVM is possibly the most successful text categorization model [49]. The SVM is based on the structural risk minimization principle from computational learning theory [50]. The principle is to find a hypothesis h with the lowest true error. The true error of h is the probability that h makes an error on the new/unknown sample. An upper bound can be used to connect the true error of h with the error of h in the training set and the complexity of H (measured by VC-Dimension), a hypothesis space containing h [50]. That is, the SVM model finds the hypothesis h by minimizing the bound on the true error and controlling the complexity.

The SVM was first introduced into text categorization by Joachims [49], who used the SVM as a learning text classifier, and showed that SVM outperformed several other popular classifiers, such as C4.5 and the Nave Bayesian. He pointed out important advantages of the SVM in text categorization:

1. There is no need for feature selection because SVM uses overfitting protection.
2. The SVM is suited to text categorization because document vectors are normally sparse [51].

Others who applied SVM to the text categorization were Drucker et al. [52], Dumais et al. [53], Dumais and Chen [54], Taira and Haruno [55], Klinkenbert and Joachims [56], Masuyama and Nakagawa [57], and Fu et al. [58].

2.2 Related work

Concept-based models are regarded as a new and promising way to retrieve information from the Web. Unlike the keyword-based systems, where keyword lists are used to describe the contents of documents, concept-based systems employ concepts to describe their contents of documents.

2.2.1 Defining concepts with text categorization classifiers

The most intuitive way to build concept-based information retrieval systems is by utilizing predictive models in text categorization (classifiers) [4]. Classifiers obtain knowledge of each category from precategorized documents (training sets) and store it in the model. The knowledge is either readable by humans (like rules in rule-based systems) or not readable by humans (like structures of a neutral network). When confronted with a new document, the trained classifier decides if the document belongs or does not belong to the category through the knowledge it has gained from the training set. To summarize the classification process: the precategorized documents in the training set represent a set of concepts, the training process defines the concepts and feeds the concepts and their definitions to the classifier, and the trained classifier can then identifies the concepts in Web documents. Classifiers provide additional knowledge for information retrieval.

In [59], decision trees were used to train genre classifiers for Web documents, where each genre was a concept to be identified. The experiments were based on 1539 manually labelled Web documents and 502 selected genre features. With the help of classifiers, the document-retrieving results in [59] obtained average of 17% better precision and 1.6% better accuracy. The authors argued that classification by genre would be a useful addition to search engines.

In [60], text categorization models, like the naive Bayes model, a specialized AdaBoost algorithm, and the SVM, were applied to classify medical papers in the areas of etiology, prognosis, diagnosis, and treatment. SVM performed the best among all the learning methods. The conclusion was that it was possible to retrieve high-quality, content-specific articles using machine learning methods.

Anagnostopoulous et al. [61] presented a framework to combine a SVM model with a search engine to classify documents into different categories. Instead of considering the contents of whole documents, the framework accessed documents through the inverted index of a large scale search engine. Then the authors constructed of short queries via selection and weighting of terms. The goal was to build the *best* short query that characterizes a document class in a large search engines. Surprisingly, they showed that in their experimental set-up the best 10-terms query achieved 90% of the accuracy of the best SVM classifier (14000 terms). When documents were indexed and described by 10 terms, the effectiveness and efficiency increase when retrieving.

Choi and Peng [62] stated that automatic classification of Web pages is an effective way to organize the vast amount of information on the Internet and to assist in retrieving relevant information from the Internet, but the authors addressed the issue that manual classification can hardly “keep up with the growth of the web.” Based on that limitation, they built an automatic classification system that could dynamically add new categories. Their system started from a predefined category tree. Their single-path search algorithm, which was based on breadth-first search, decided whether or not a document belonged to a category based on how well the document feature vector matched a category feature vector. The algorithm also decided if a new category was needed or not needed. The idea is interesting, but in a real Web application, the number of categories could be too huge to process.

In general, the goal of utilizing classifiers in information retrieval models is to provide more knowledge when retrieving relevant Web documents. The knowledge of each category, which can be, for example, genres or topics, is the definition of concepts. This knowledge is added to the classifiers through the training datasets, and saved in the classifiers. With the embedded knowledge, trained classifiers are able to identify concepts in new documents, that is, they can classify documents into categories. However, the requirement of training sets hinders the application of these methods because most of the classifiers require a total reconstruction for every tiny change that occurs.

2.2.2 Defining concepts in concept structures

However, too much human-labor is required to set categories, build training sets, update models, etc. Considering the number of and the speed of change in Web documents, this burden can only get heavier. Moreover, limited categories (concepts) cannot satisfy unlimited requirements (queries). No matter how many categories the system provides to users, they will not feel it is enough. And as more categories are offered, finding the one of interest will become harder. Constructing a background concept structure as a knowledge base to define concepts is a more affordable approach.

Synonym thesauri

Synonym thesauri defines keywords in queries through expanding them with their synonyms. Anick [63] proposed a system that automatically generates an extended condition: "a boolean expression is composed by ORing each query term with any stored synonyms and then ANDing these clusters together." That is, the whole query was ORed together. Each OR term was composed of synonyms from an online thesaurus. This method increases the chance that the useful information will be in the retrieved documents, but it increases the number of the retrieved documents as well. Moreover, since many terms have multiple synonyms with different meanings, an extension with the right synonyms is also an issue.

Conceptual taxonomy

Conceptual taxonomy is applied as a hierarchical organization of concepts. Each concept in aconceptual taxonomy connects both its superconcepts and its subconcepts. Therefore, it provides a topological structure for efficient conceptual search and retrieval. In a project by Sun Microsystems, a conceptual indexing technique was proposed to automatically generate conceptual taxonomies [64].

Ontology

Ontologies are models of a domain or a problem that can be used to provide formal semantics (meanings, concept-based information) to any sort of information, such as databases, Web documents, etc.

A commonly accepted definition of ontology is "an ontology is an explicit and formal specification of a conceptualization of a domain of interest" [24]. In general, an ontology is a representation of a set of concepts and relations between those concepts in a domain.

Examples of existing ontologies are:

> ***WordNet*** is an English lexical ontology built at Princeton University, which includes explanations of the terms and relations between terms (synonyms, antonyms, etc.) [65]. As the most popular linguistic ontology, WordNet is used by many researchers. Normally it is used to expand queries with semantically related terms.
> Voorhees expanded queries with lexical semantic relations in WordNet [66]. Gong et al. utilized WordNet to expand queries in three dimensions including hypernym, hyponymy, and synonym relations [67]. However, the systems applying WordNet suffered from word sense disambiguation (WSD) because of polysemy[2]. Baziz et al. proposed an approach to identify important concepts with two criteria, co-occurrence and semantic relatedness, to eliminate WSD in WordNet [68]. Kolte also proposed an approach to handle WSD based on an unsupervised approach to determine the domain that keywords belong to [69]. Kim et al. transformed WordNet into a matrix, where each term is a vector. The authors then applied singular value decomposition (SVD) to reduce the size of matrix, which helped to relieve the WSD [70]. However, researchers reported difficulties in applying linguistic-based ontologies to non-linguistic applications [71].

[2]Polysemy means more than one words with the same meanings

SENSUS is a natural language-based ontology developed by the Natural Language Group at Information Sciences Institute (ISI) [72]. It is an extension and reorganization of WordNet.
In an OntoSeek [71] project, Guarino et al. utilized SENSUS ontology for concept-based retrieval from online yellow pages and product catalogs. The authors linked conceptual graphs, which were transferred from queries, to the SENSUS ontology by using lexical conceptual graphs.

Gene Ontology is composed of three structured controlled vocabularies describing gene products with their associated biological processes, cellular components and molecular functions [73].
Spasic et al. designed an information retrieval system, KiPar, to facilitate access to the literature relevant to kinetic modelling of a given metabolic pathway in yeast [74]. Instead of free-text descriptions, the input accepted by the system was identifiers used in gene ontology. The back-end ontology expanded the identifiers of the query with specific knowledge about the identifiers. For example, if the name of a specific enzyme was input, the known information about (i) the compounds acting as substrates/products of the reaction catalysed, and (ii) the genes encoding that enzyme could be retrieved. In this way, the query contained extra knowledge to retrieve documents. Finally, documents were retrieved by applying SQL queries over the local database.

Based on different ways to identify concepts from texts and their aggregations, we divide concept-based approaches to ontology into four major solutions.

Solution 1: Regular expressions, rules, and ontology

In this type of approaches, regular expressions are used to identify concepts from texts. Several predefined rules are then used to aggregate the identifications of single concepts.

Regular expression can be regarded as a straightforward way to implement natural language processing. A regular expression is "a string containing a combination of normal characters and special meta-characters or meta-sequences" [75] that provides a concise and flexible representation of texts.

Embley proposed the use of information extraction ontologies, which were formalized over regular expressions [76]. For example, the regular expression "(call|phone)+ a bunch of numbers or hyphens" is used to recognize phone numbers. With regular expressions defined and stored in ontologies, words and phrases in documents can be related to concepts in the ontology. In the phrase "Call 999-999-9999" the number has meaning as a phone number. In an example in [76], with a rule that automobile advertisements must contains textual expressions like “car has year,” “car has make,” and “car has price,” the text “97 CHEVY Cavalier, Red, 5 spd, only 7,000 miles on her. Previous owner heart broken! Asking only $11,995. #1415 JERRY SEINER MIDVALE, Call 566-3800 or 566-3888” can be recognized as a car ad.

Muller et al. constructed an ontology-based information retrieval and extraction system for biological literature, which is called *Textpresso* [77]. In Textpresso, biological concepts (e.g., gene, allele, cell or cell group, phenotype, etc.) were presented in regular expressions. For example, a gene was represented as regular expression “[A-Za-z][a-z][a-z]-\d”, which matched a term with three letters ([A-Za-z][a-z][a-z]), a dash (-), and a sequence of digits (\d). With this format, the phrase “let-60” could be recognized as a gene. Relations between two biological concepts (e.g., association, regulation) and descriptions of one concept (e.g., biological process) were also used in the ontology. The retrieval process in Textpresso was based on concepts and relations/descriptions defined in an ontology. In an example given in the paper, a researcher interested in facts about genetic regulation of cells used concepts “gene”, “regulation”, and “cell or cell group” to retrieve documents.

Solution 2: Natural language processing (NLP) with ontology

The previous solution can be regarded as a simplified *natural language processing* (NLP) approach. Informally, NLP aims to solve problems by making computers understand and process human language. There are two types of approach in NLP: deep approach and shallow approach. *Deep* and *shallow* represent the degrees of expected understanding. As Styltsvig stated, “Deep approaches presume access to a comprehensive body of world knowledge. These approaches are not very successful in practice, mainly because access to such a body of knowledge does not exist, except in very limited domains” [78]. Shallow NLP techniques normally rely on simple rules to do analysis. To our knowledge, shallow NLP techniques are usually integrated with ontologies to fulfill concept-based information retrieval.

The most essential part in NLP is part-of-speech (POS) tagging. In nature language, words are used in grammatical roles such as *nouns*, *adjectives*, *prepositions*, *verbs*, and so on. These categories are called POS. POS tagging is the process of automatically labelling words in the texts with their POS.

Cimiano et al. prsented the LexOnto model consisting of a domain ontology and a corresponding ontology for associating lexical information to entities of the given domain ontology [79]. The lexicon ontology contained three different POS: verbs, nouns and adjectives. It was built on a *subcategorization frame* (linguistic predicate-argument structure) and *relations*, which represents properties or joins between properties defined in the domain ontology.

Several examples of subcategorization frames are: *Transitive_Frame*, which represents a transitive verb (like *wash*) with the arguments subject and object; *Intransitive_PP_Frame*, which represents an intransitive verb with a prepositional complement as argument (like *wait for*); *Transitive_PP_Frame*, which represents a transitive verb with an additional prepositional complement (e.g. bring X to Y); *Noun_PP_Frame*, which represents a noun with a prepositional complement (e.g. capital of) and so on.

For each (instantiated) subcategorization frame, there is a relation in the lexicon ontology representing how it is connected to a property in the domain ontology. Thus, the properties of the domain ontology can be identified after recognizing the subcategorization frame through NLP analysis of the texts.

Morneau et al. proposed SeseiOnto using NLP to identify concepts from texts [80]. Compared with the LexOnto model, this approach involved more shallow NLP. First of all, SeseiOnto removed articles and prepositions from NLP-based user queries (sentences) by through tagging. The remaining words in the queries were matched to the concepts in the ontology. Based on pre-defined transformation rules, the concept-based queries were *converted into concept graphs* (CG) [81]. An example of transformation rules is "when a *noun* (A) is the subject of a *verb* (B) at active voice in a sentence, it should then be converted to a CG stating that a concept of type A is the agent of a concept of type B." A similar process was applied to documents transferring them into CGs at the same time.

Let us assume there are two CGs: one is the CG of the query (CG_1), the other is the CG of a document (CG_2). From CG_1 and CG_2, the common generalization Γ, which contains the information shared by CG_1 and CG_2, is extracted. The projection of Γ in CG_1,represented as $\prod_{\Gamma CG_1}$, is also generated. The projection process is the reciprocal of the generalization process. The subgraph of $\prod_{\Gamma CG_1}$ is $\Omega(\prod_{\Gamma CG_1})$.

The relevance of CG_1 and CG_2 is decided by the *conceptual similarity* and the *relational similarity* they define.

Concept similarity *sc* is:

$$sc = \frac{N_{|c \in \Gamma|}}{N_{|c \in CG_1|}}$$

where $N_{|c\in\Gamma|}$ is the number of concepts of Γ; $N_{|c\in CG_1|}$ is the number of concepts of CG_1.

Relational similarity is calculated by:

$$sr = \frac{\sum_{r \in \Gamma} weight(r)}{\sum_{r \in \Omega(\prod_{\Gamma CG_1})} weight(r)}$$

where weight(r) represents the weight of a relation r. In SeseiOnto, the weights of relations are fixed, and are set empirically by studying a number of graphs.

The relevance score of CG_1 and CG_2 is calculated by:

$$ss(CG_1, CG_2) = 0.5 \times sc + 0.5 \times sc \times sr$$

Utilizing NLP and an ontology may be the most intuitive application to accomplish concept-based systems. However, the discussions on how much NLP should be involved never stops. In LexiOnto [79], NLP exists in the entire process; while in SeseiOnto [80], NLP handles conversion from texts to concept-based structures. Though the NLP techniques allow computer agents to recognize concepts from texts, their complexity cannot be ignored.

Solution 3: vector space and ontology

This type of solution is the most popular. In general, the ontology is utilized to expand the query, and vector space approach is used to evaluate similarity.

Vallet et al. proposd an ontology based information retrieval system applying a vector-space model to retrieve relevant documents [82]. After terms and concepts were connected through ontology-driven weighted annotations on the documents, a classic vector-space model was utilized to evaluate the relevance between documents and queries. The weights were based on the frequency of occurrence of the instances in each document. The structure of the system was very reasonable, however since the ontology as the knowledge base had a hierarchical structure and the query did not have (in a classical vector-space model, a query is a vector), it was difficult to take full advantage of the knowledge in the ontology.

Similarly, Dridi et al proposed an ontology-based framework for semantic information retrieval [83] that used a vector-space model to evaluate similarities. With annotation based on GATE [84] as an information extraction module, metadata for documents were generated. The metadata were extracted concepts from the document text. Based on the term weighting technique $cf \cdot idf$ (concept frequency - inverted document frequency), which is similar to $tf \cdot idf$, concepts in the documents were recognized and indexed. When retrieving relevant documents, concepts in queries were identified with the definition in the ontology, and documents were retrieved with the matched concepts.

The knowledge and information management (KIM) platform was proposed by Kiryakov et al. [85]. It focused on the automatic population of the ontology and the annotation of documents. KIM included a simplistic upper-level ontology that started with some basic philosophic distinctions and moved to the most popular entity types (people, companies, cities, etc.), thus providing many of the inter-domain common sense concepts and allowing easy domain-specific extensions. The semantic annotation, which contained references to classes in the ontology, was achieved. Based on the annotations, keyword-based indexing and retrieval was performed.

Castells et al. [86] presented a complete concept-based information retrieval model. The authors tried to complement KIM [85] with a ranking algorithm. In their model, first a set of root ontology classes were constructed from three main base classes: DomainConcept, Topic, and Document. Documents were annotated with concept instances in the ontology. The annotation process was similar to the one in KIM [85]. The annotations were weighted based on an adaption of the $tf \cdot idf$ scheme(section 2.1.1), and the ranking is then achieved through computing similarity values between queries and documents through the classic vector-space approach (section 2.1.1).

In [87], Tomassen et al. presented an ontology-driven system WebOdIR, where each concept was extended by associating it with a vector of key-phrases describing the concept. The system started from ranking concepts in the ontology according to ontology relevance. It then generated a query for each concept based on relations with other concepts. After submitting the queries to the underlying search engine, a set of documents for each concept was retrieved and then clustered. For each cluster, a set of candidate terms were extracted. After comparing the candidate terms for each cluster of the concept to the candidate terms for neighbouring concepts, the candidate terms with the highest similarity were chosen and used for the final assembly of the feature vector of the concept. Tomassen et al. argued that the altered ontology was more advanced and helped expand users' queries for an in-depth understanding of their needs.

Solskinnsbakk et al. stated that it is not "a straightforward task" to use knowledge in the ontology for information retrieval purposes. To solve this problem the authors proposed definitions of the ontology profile, which was a semantic extension of an ontology where each ontology concept was given a description in terms of a vector of weighted keywords. The ontology profile was constructed based on a document collection covering the same domain as the ontology. Instead of just putting synonymous to the concept, they picked terms based on statistics and added them to ontology profiles; they calculated terms weights by the $tf \cdot idf$ scheme. Then the ontology profiles could be used to expand queries to retrieve relevant documents.

Solution 4: Latent semantic indexing and ontology

Latent semantic indexing (LSI) is similar to the vector space approach that presents documents/queries in the manner of vectors. LSI is applied to handle two severe problems in keyword-based searching: synonymy (more than one word with the same meaning) and polysemy (one word having more than one meaning). Deerwester et al. [88] discussed this method.

Like vector space approaches, LSI relies on a term-document matrix in which each column represents a document and each row lists frequencies of a term in different documents. LSI uses the linear algebra technique singular value decomposition (SVD) to reduce the dimensions of the term-document matrix and approximate the most important part of the original matrix. Let us assume there is an $m \times n$ term-document matrix A (m terms and n documents). Based on SVD, A is decomposed into the product of three matrices,

$$A_{mn} = T_{mr} S_{rr} D_{rn}^T$$

where T is an orthogonal matrix ($T^T \times T = I$), called left singular matrix; S is a diagonal matrix of positive singular values in decreasing order; D is also an orthogonal matrix ($D^T \times D = I$), called a right singular matrix. Please note that T is an m by r matrix of term vectors, where r is the rank of A[3]; S is an r by r matrix; and D is an r by n matrix of document vectors. For the purpose of efficiency, only the first k largest singular values are kept while the remaining ones are set to zero. $\hat{A}_{mn}$ represents the lower rank approximation of A_{mn},

$$A_{mn} \approx \hat{A}_{mn} = T_{mk} S_{kk} D_{kn}^T$$

A document or a query vector d can be projected to the LSI document vector, $d* = d_{1m} T_{mk} S^(- 1)_{kk}$. Therefore, the similarity of a query and a document can be calculated by the inner product of these two vectors.

Because LSI reduces the ranks of the term document matrix from r to k by examining the whole document collection, documents that have many terms in common are regarded as semantically close. The first k largest singular values can be considered as k concepts in the document collection.

Snasel et al. mapped LSI concepts to Wordnet [89]. Wordnet provides synonyms to expand queries: this improves recall through sacrificing precision, because it increased the number of keywords. LSI helps to retrieve the most relevant k terms/concepts from term matrix T_{mk}. The authors argued that the expansion of these k terms instead of all keywords balanced the conflicts between precision and recall.

[3]The rank of a matrix is the number of its unique dimensions.

Chapter 3

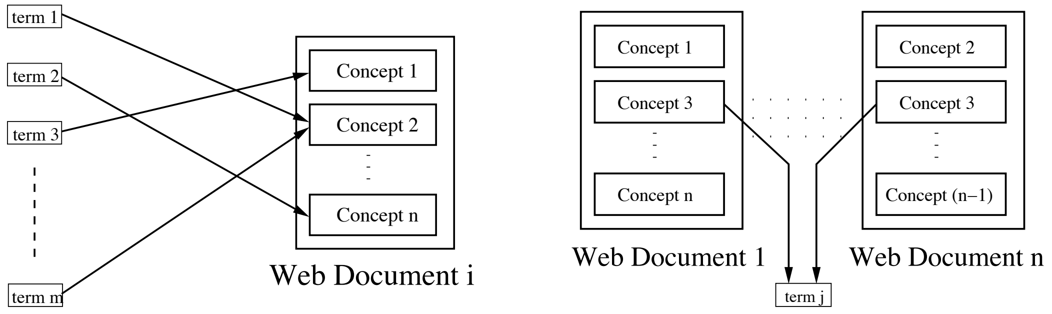
Term importance retrieving scheme

In a concept-based information system, both queries and documents are regarded as lists of concepts, though they are expressed directly by concrete terms. That is, abstract concepts are defined by concrete terms. For example, a concept *Player David Beckham* is described by his first name "David," last name "Beckham," etc. With these concrete terms, computer agents can possibly identify the concept *Player David Beckham* from texts.

However, terms have unique importance in defining a concept. Regarding *Player David Beckham*, if we compare the term "David," with the term "Beckham," intuitively the latter one (the player's last name) is more important in identifying the concept than the former one (the player's first name).

The most popular approach to obtain importance is through $tf \cdot idf$ or its variations (section 2.1.1), however, they are not suitable for concept defining; they are designed to calculate a value of term importance for discriminating between documents, but are not useful when the term importance value is treated as a measure of term contribution to the concept definition. In [90], Soucy also argues that a $tf \cdot idf$ type of term weighting schema is appropriate for information retrieval (keyword-based systems), but not good for text categorization (concept-based systems), because it does not "leverage the information implicitly contained in the categorization task." $tf \cdot idf$ schemes assign the same importance to a term without taking into consideration the concept or category to which this term belongs. This causes a problem when $tf \cdot idf$ is applied to concept-based techniques.

In order to estimate the importance of values of different terms we have designed, we created a special scheme called adaptive assignment of term importance – AATI. The AATI scheme estimates the importance of terms with



(a) TWs contributions to PV (b) PVs contributions to TW

Figure 3.1: Interactions between TW and PV

respect to their levels of contribution toward the defined concept.

3.1 Adaptive assignment of term importance (AATI)

3.1.1 Concept description

The AATI retrieves term importance values in a self-adaptive manner. That is, the AATI keeps updating the importance of terms based on a stream of "unkown" Web documents. It accomplishes this with interactions between two values: term importance (TW) and page values (PV). The two values are connected by concept value (CV). The CV of a concept in a document represents the amount of information related to the concept in the document. Let us assume the concept k is defined by N_k terms, c_{ki} (CV of the concept k) in document i can be calculated as:

$$c_{ki} = \sum_{j=1}^{N_k} t_j \cdot f_{ij} \tag{3.1}$$

where t_j is the TW of term j and f_{ij} is the frequency of term j in document i. If term j is not in document i, $f_{ij} = 0$.

PV, i.e., the page value of a document, is calculated based on the sum of the CVs (Eq 3.1) in it. That is, the PV of a document can be calculated using the TWs of terms presented in the document.

The TWs of terms are updated based on the PVs of documents. This is especially important for Web applications – new documents lead to the up-to-date modification of TWs.

In summary, on the one hand, TWs contribute to CVs and in turn to PVs (Fig. 3.1a). On the other hand, PVs update TWs (Fig. 3.1b).

The AATI starts with randomly initialized TWs of terms. Since we define that the sum of TWs is always equal to one (normalization), the AATI actually modifies the distribution of TWs based on their contributions to PVs. Every time the AATI "reads" a document, it first calculates the PV of the document based on the TWs of terms in the document, and then it uses the PV to update the TWs.

Because the number of Web documents is unlimited, we have to prove that it is possible to obtain TW values. In the next section, we provide the detailed mathematical definitions of TW and PV, and prove theoretically that TWs exist.

3.1.2 AATI: relations between TW and PV

For any concept, there are always two sets of terms: related terms and unrelated terms. The related term set contains terms that represent a given category, called a *target* concept; while the unrelated term set contains terms that do not contribute to the *target* concept. Let us assume there are two sets,

$$\begin{aligned} &T' = \{t'_1, t'_2, t'_3, ..., t'_{n'}\} \\ \textit{and}& \\ &T'' = \{t''_{n'+1}, t''_{n'+2}, t''_{n'+3}, ..., t''_n\} \end{aligned}$$

where each element t'_i in the set T' represents the TW of term i that is related to the target concept, and each element $t''_{i'}$ in the set T'' represents the TW of term i' that is unrelated to the target concept. Hence, a new n-element set T combines T' and T'', where $T = \{T', T''\} = \{t'_1, t'_2, ..., t'_{n'}, t''_{n'+1}, ..., t''_n\} = \{t_1, t_2, t_3, ..., t_n\}$. Set T contains TWs of all the terms contained in Web documents. At the same time, a set $P = \{p_1, p_2, p_3, ..., p_m\}$ represents PVs of m Web documents, where each element p_i in the set P represents the PV of the ith document. Thus, an equation to calculate the PV of the document i is:

$$p_i = \gamma \cdot \sum_{k=1}^{N_i^c} \frac{c_{ki}}{N_k} = \gamma \cdot \sum_{j=1}^{n} \alpha_{ij}\, t_j \tag{3.2}$$

where c_{ki} is the CV defined in Eq.(3.1); N_k is the number of concept k in all documents; n is the number of elements in the set T; α_{ij} represents a relation between p_i and t_j; and γ is a normalization constant that adjusts the relative values between p_i and t_j (the relative values between p_i and t_j should be of the same order of magnitude because p_i is updated by t_j and vice versa).

The value of α_{ij} is obtained in the following way:

$$\alpha_{ij} = \frac{f_{ij}}{N_j} \tag{3.3}$$

where f_{ij} is the frequency of term j in document i. If term j is not in the document i, $f_{ij} = 0$. N_j is the number of term j in all documents.

Therefore, we can represent Eq. (3.2) as:

$$\mathbf{P} = \gamma \cdot \mathbf{A} \cdot \mathbf{T} = \gamma \cdot ([[\mathbf{A}'][\mathbf{A}'']] \cdot \begin{bmatrix} [T'] \\ [T''] \end{bmatrix}) \tag{3.4}$$

where $\mathbf{P}$ is an $m \times 1$ vector; $\mathbf{A}$ is an $m \times n$ matrix; $\mathbf{T}$ is a $n \times 1$ vector; $\mathbf{A}'$ is an $m' \times n'$ matrix; $\mathbf{A}''$ is a $m \times (n - n')$[1]; T' is a $n' \times 1$ and T'' is a $n'' \times 1$, and m is the number of Web documents, n is the total number of terms, n' is the number of terms of the target category, and n'' is the number of terms that do not belong to the target category.

On the other hand, TWs are calculated using PVs:

$$\begin{cases} t'_j = \delta \cdot \sum_{i=1}^{m} \beta_{ji} p_i \\ t''_j = 0 \end{cases} \tag{3.5}$$

where m represents the number of Web documents; β_{ji} represents a relation between p_i and t_j; and δ is the constant parameter with a meaning similar to γ in Eq. (3.2). t'_j represents the TW of a term that is related to the target category; while t''_j represents the TW of a term that is not related to the target category.

β_{ji} in Eq. (3.5) is defined as:

$$\beta_{ji} = \frac{f_{ij}}{N'_i} \tag{3.6}$$

where N'_i is the number of terms in the document i. If term j is not present in the page i, f_{ij} is zero. Therefore, Eq. (3.5) can be shown as:

[1] A number of terms that do not belong to the *target* category will be represented by $n'' = n - n'$.

$$\begin{aligned}\mathbf{T} = \begin{bmatrix} T' \\ T'' \end{bmatrix} &= \delta \cdot \mathbf{B} \cdot \mathbf{P} = \delta \cdot \begin{bmatrix} [\mathbf{B}'] \\ [\mathbf{B}''] \end{bmatrix} \cdot \mathbf{P} \\ &= \delta \cdot \begin{bmatrix} \begin{bmatrix} \\ \mathbf{B}' \\ \\ \end{bmatrix} \\ \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix} \end{bmatrix} \cdot \mathbf{P}\end{aligned} \tag{3.7}$$

where B is a $n \times m$ matrix, B' is a $n' \times m$ matrix, and B'' is a $n'' \times m$ zero matrix.

Combining Eq. (3.4) and (3.7) results in

$$\mathbf{T} = \begin{bmatrix} T' \\ T'' \end{bmatrix} = \begin{bmatrix} t'_1 \\ \cdot \\ \cdot \\ t'_{n'} \\ t''_{n'+1} \\ \cdot \\ \cdot \\ t''_n \end{bmatrix} = \begin{bmatrix} t'_1 \\ \cdot \\ \cdot \\ t'_{n'} \\ 0 \\ \cdot \\ \cdot \\ 0 \end{bmatrix} = \delta \cdot \mathbf{B} \cdot \mathbf{A} \cdot \mathbf{T} = \delta \cdot \mathbf{C} \cdot \mathbf{T} \tag{3.8}$$

where $C = B \cdot A$.

3.1.3 AATI: the solution and its properties

The solution of Eq. (3.8) is set T.

Proposition 1 Given Eq. (3.8), the conditions that one solution T exists are

1. C' is a positive $n' \times n'$ matrix; and
2. The eigenvalue of matrix C' is 1.

Proof Since C' is a positive $n' \times n'$ matrix, then based on the Perron-Frobenius theorem [91], C' has a *unique* right eigenvector $X = \{x_1, x_2, ..., x_{n'}\}$ if $\sum_{i=1}^{n'} x_i = 1$. Therefore

$$\lambda_c \cdot \mathbf{X} = \mathbf{C}' \cdot \mathbf{X} \tag{3.9}$$

where λ_c is the eigenvalue of C'.

Due to the second condition, i.e., $\lambda_c = 1$, after comparing Eqs.(3.8) and (3.9), X as the right eigenvector of matrix C' is the solution of Eq. (3.8). Besides, if $\sum_{i=1}^{n} x_i = 1$, it is unique. **Q.E.D.**

In conclusion, T in Eq. (3.8) exists when C' is a positive $n' \times n'$ matrix with the eigenvalue.

Proposition 2 The matrix C' in Eq. (3.8) is a positive $n' \times n'$ matrix and its eigenvalue is equal to 1.

Proof C' is a positive $n' \times n'$ matrix, so let us focus on its eigenvalue.

Based on Eq. (3.8), we have:

$$\delta \cdot \mathbf{C} = \delta \cdot \mathbf{B} \cdot \mathbf{A} = \delta \begin{bmatrix} \begin{bmatrix} \\ \mathbf{B}' \cdot \mathbf{A}' \\ \\ \end{bmatrix} \begin{bmatrix} 0 & . & . & . & 0 \\ . & 0 & . & . & . \\ . & . & 0 & . & . \\ & . & . & .0 & . \\ 0 & . & . & . & 0 \end{bmatrix} \\ 0 \\ . \\ . \\ 0 \end{bmatrix}$$

$$= \frac{\delta}{\gamma} \begin{bmatrix} \begin{bmatrix} \\ \gamma \cdot \mathbf{C}' \\ \\ \end{bmatrix} \begin{bmatrix} 0 & . & . & . & 0 \\ . & 0 & . & . & . \\ . & . & 0 & . & . \\ & . & . & .0 & . \\ 0 & . & . & . & 0 \end{bmatrix} \\ 0 \\ . \\ . \\ 0 \end{bmatrix} \tag{3.10}$$

Here

$$\begin{aligned} \mathbf{C} &= \mathbf{B} \cdot \mathbf{A} \\ \mathbf{C}' &= \mathbf{B}' \cdot \mathbf{A}' \end{aligned} \tag{3.11}$$

where C is an $n \times n$ matrix, and C' is an $n' \times n'$ matrix.

Based on Eqs. (3.2), (3.3) and (3.4), α_{ij} is the element of row i and column j in the $m' \times n'$ matrix A', thus the sum of each column in the matrix A' is:

$$\sum_{i=1}^{m'} \alpha_{ij} = \sum_{i=1}^{m'} \frac{f_{ij}}{N_j} = \left(\sum_{i=1}^{m'} f_{ij} \right) / N_j = 1 \tag{3.12}$$

Based on Eqs. (3.5), (3.6) and (3.7), β_{ji} is the element of row j and column i in $n' \times m'$ matrix B', thus the sum of each column in the matrix B' is:

$$\sum_{j=1}^{n'} \beta_{ji} = \sum_{j=1}^{n'} \frac{f_{ij}}{N'_i} = \left(\sum_{j=1}^{n'} f_{ij} \right) / N'_i = 1 \tag{3.13}$$

The $n' \times m'$ matrix B' is represented by an $n' \times 1$ vector $B' = [b_1, b_2, ..., b_{n'}]^t$, and each element b_i is the ith row vector in the matrix B', which is a $1 \times m'$ vector, i.e., $b_i = [\beta_{i1}, \beta_{i2}, ..., \beta_{im'}]$. At the same time, $m' \times n'$ matrix A' is represented by an $1 \times n'$ vector $A' = [a_1, a_2, ..., a_{n'}]$, and each element a_i is the ith column vector, which is an $m' \times 1$ vector, i.e., $a_i = [\alpha_{i1}, \alpha_{i2}, ..., \alpha_{im'}]^t$. So we represent Eq. (3.11) in a new form:

$$\begin{aligned}
\mathbf{C'} = \mathbf{B'} \cdot \mathbf{A'} &= \begin{bmatrix} b_1 \\ b_2 \\ ... \\ b_{n'} \end{bmatrix} \cdot [\mathbf{a_1}\ \mathbf{a_2}\ ...\ \mathbf{a_{n'}}] \\
&= \begin{bmatrix} b_1a_1 & b_1a_2 & ... & b_1a_{n'} \\ b_2a_1 & ... & ... & b_2a_{n'} \\ ... & ... & ... & ... \\ b_{n'-1}a_1 & ... & ... & b_{n'-1}a_{n'} \\ b_{n'}a_1 & b_{n'}a_2 & ... & b_{n'}a_{n'} \end{bmatrix}
\end{aligned} \tag{3.14}$$

Therefore, the sum of elements in column i of C' is:

$$\sum_{i=1}^{n'} b_i a_j = (\sum_{i=1}^{n'} b_i) \cdot a_j \tag{3.15}$$

where b_i is an $1 \times m'$ vector, and the sum of the elements of each column in B' is 1, therefore

$$\sum_{i=1}^{n'} b_i = \mathbf{I} \tag{3.16}$$

where I is the $1 \times m'$ vector [1 1 ... 1]. From Eqs. (3.15) and (3.16), we have:

$$\sum_{i=1}^{n'} b_i a_j = [1\ 1\ ...\ 1] \cdot a_j = \sum_{k=1}^{m'} \alpha_{kj} = 1 \tag{3.17}$$

so the sum of the each column in the matrix C' is also 1. Then we have:

$$e \cdot \mathbf{C}' = \mathbf{e} \tag{3.18}$$

where e is the $1 \times n'$ vector $[1, 1, ..., 1]$, therefore

$$e \cdot (\mathbf{C}' - \mathbf{I}) = \mathbf{0} \tag{3.19}$$

$$\Rightarrow \; det(\mathbf{I} - \mathbf{C}') = \mathbf{0} \tag{3.20}$$

where I is the identity matrix. Thus, we have proved that the eigenvalue of C' is 1, that is T' exists and T exists. **Q.E.D.**

In conclusion, because TW set T is the right eigenvector of matrix C as denoted in Eq. (3.9), theoretically the AATI scheme is correct.

3.2 Validation of the AATI scheme

The AATI scheme which assigns term importance, is a crucial part of our research. In the following sections, a number of experiments were designed and deployed to validate the AATI scheme.

The experiments are divided into two groups:

- The first group of experiments illustrates the nature of the AATI. It shows the AATI's ability to continuously update TW values of the terms based on the upcoming stream of documents. The results are presented in section 3.2.3.
- The second group of experiments compares the AATI with the support vector machine (SVM) (secion 2.1.5) when the two methods are applied to categorize documents. Though the AATI is not designed for categorization, it can be adapted for this purpose. We assume that *soccer* is the target category and concepts identified in a document are pieces of evidence supporting a decision whether a document belongs or does not belong to the target (*soccer*) category. The SVM is probably one of the most popular text categorization (TC) techniques (section 2.1.5). It was introduced by Thorsten Joachims [49], and compared with other classifiers, such as k-NN (a k-nearest neighbours classifier), C4.5 (a decision tree based classifier) and Naive Bayes based classifiers. SVM is the best of these techniques according to many researchers[92, 93, 94, 95] .

The AATI scheme is designed to interact with ontologies (Sections 4.2), which provides AATI a list of concepts, a list of terms, and their relations. An ontology is included in the experiments in this chapter (see section 3.2.4).

3.2.1 Evaluation metrics

Four metrics are applied to evaluate results of classification experiments: *precision*, *recall*, *F-Measure*, and *accuracy*. F-Measure combines precision and recall. The definitions of precision, recall, F-Measure, and accuracy are:

$$\begin{aligned}
P_{precision} &= \frac{N_{relevant} \cap N_{retrieved}}{N_{retrieved}} \\
P_{recall} &= \frac{N_{relevant} \cap N_{retrieved}}{N_{relevant}} \\
P_{F-Measure} &= \frac{2 \cdot P_{precision} \cdot P_{recall}}{P_{precision} + P_{recall}} \\
P_{accuracy} &= \frac{N_{retrieved}}{N_{relevant}}
\end{aligned} \tag{3.21}$$

where $N_{relevant}$ is the number of relevant Web documents, and $N_{retrieved}$ is the number of Web documents that are retrieved.

The metric $P_{accuracy}$ is a representative measure evaluating the performance of any classification algorithm. This measure provides good evaluation of a classifier when the distribution of data among categories is balanced. In our experiments, the datasets contained no more than 20% of the soccer-relevant Web documents, and up to 80% of the non-soccer Web documents. If we did not use any classification algorithm but just recognized all Web documents as non-soccer ones, we obtained an accuracy of about 80%, which was meaningless. Therefore, we also used other metrics – precision, recall, and their combination F-Measure – in order to provide a more comprehensive comparison of classification methods.

3.2.2 Datasets

Three collections of Web documents were used in the experiments. The documents were crawled from three news websites: BBC[2], CNN[3], and CBC[4]. The documents collected from those websites were already labelled by editors of those news agencies. In total, 9135 Web documents were collected from BBC,

[2]http://www.bbc.co.uk, crawled in June 2007.
[3]http://www.cnn.com, crawled in June 2007.
[4]http://www.cbc.ca, crawled in August 2009.

907 from CNN, and 6041 from CBC. Different types of document are included in the obtained documents: text documents, files for formating, scripts (which are usually short), etc. We filtered the Web documents and discarded all documents with size less than 10 kilobytes. The prepared datasets for the experiments were:

- Documents from the BBC website were split into two sets–training and testing. The testing dataset – $BBCSet_{TS}$ – contained 7535 web documents: 6059 non-soccer, and 1476 soccer documents; the training dataset – $BBCSet_{TR}$ – contained 1600 web documents: 1280 non-soccer, and 320 soccer ones ;
- There were 907 documents from the CNN website – $CNNSet$ – 802 non-soccer and 105 soccer documents [5];
- Documents from the CBC website were divided into two sets: a testing dataset – $CBCSet_{TS}$ – 4841 web documents, including 4328 non-soccer and 513 soccer documents, and a training dataset – $CBCSet_{TR}$ – 1200 web documents, 1075 non-soccer, and 125 soccer ones.

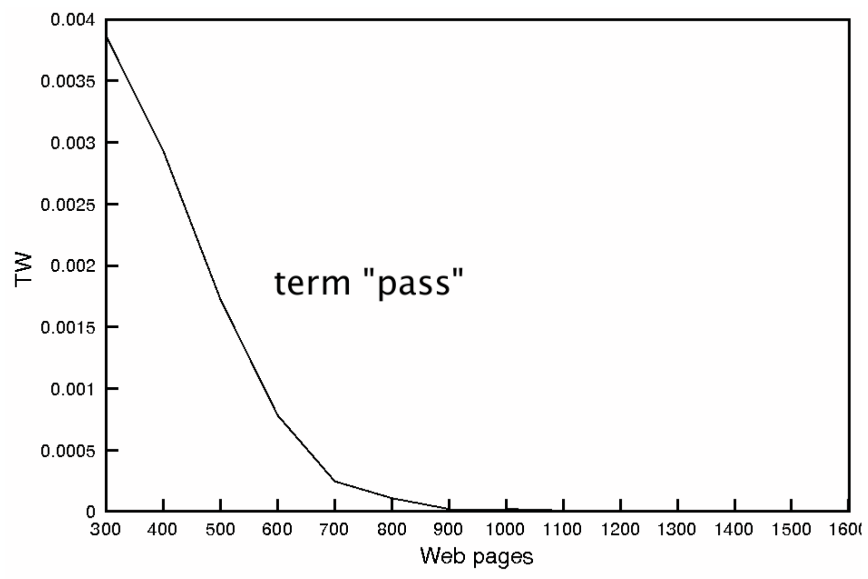
In the experiments, we used $BBCSet_{TR}$, $CNNSet$, and $CBCSet_{TR}$ to construct three (for each dataset) SVM models and three AATI models. Those models were validated using different testing datasets.

3.2.3 Basic experiments

The AATI updates TW every time it reads a new Web document. The changes in the TW values of two different terms "pass" and "midfielder" are shown in Fig. 3.2. It is assumed that the first three hundred Web documents are needed for term weights to gain a discrimination power. The term values after the first three hundred Web documents were 0.004 and 0.002 for "pass" and "midfielder" respectively. However, the value of the term "midfielder", which was more important for the *soccer* category, increased to 0.014, while the value of "pass" dropped to almost zero.

The continuous updates of TWs are "controlled" by upcoming documents. If the contents of Web documents change, the TWs change too. A good illustration of that process is a change in the TW value of the term "Manchester United," shown in Fig. 3.3. This term represents a name of a famous soccer club in England. Initially, the upcoming Web documents were collected from

[5]Due to its small size, the $CNNSet$ dataset was not split into training and testing; the whole set was used to build models.



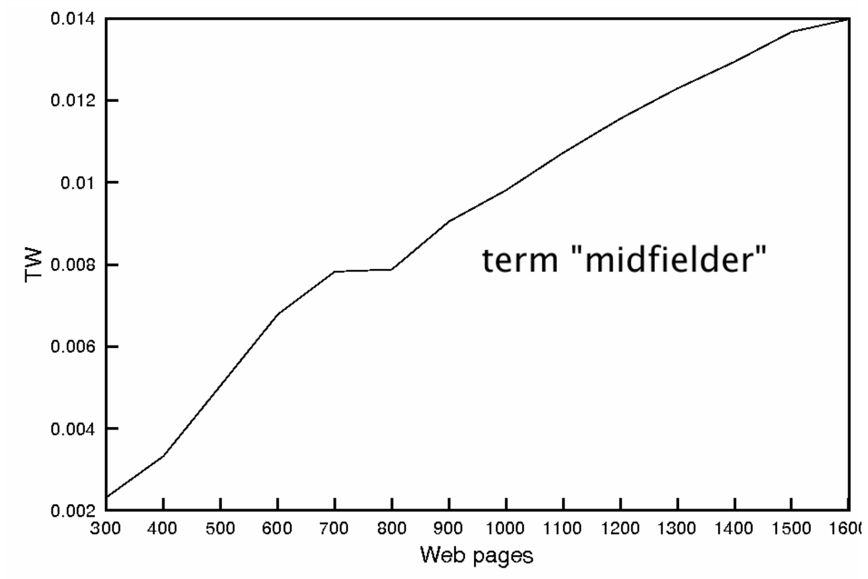


Figure 3.2: Changing of the two different TWs

the BBC website (from England), therefore the TW of this term was rising. However, after page 1000 the source of Web documents changed and the upcoming documents were collected from the CNN site (from USA), where the term "Manchester United" was less popular. This decrease in popularity of the term "Manchester United" translated into a drop of its TW value. The change in TW is easy to recognize in Fig. 3.3 – the dotted line represents what the TW of the term would be if the upcoming stream of Web documents were still from the BBC website.

TW values depend on the frequencies of terms in the Web documents. As shown in Fig. 3.3, the TW of the term "Manchester United" which occurs 42 times in 1000 BBC Web documents initially increases, and then decreases when the term does not occur in the next 600 Web documents from the CNN website.

In general, TW values of terms which do not occur frequently in the documents are small. On the other hand, a high frequency of a given term does not guarantee a high TW value. For example, the top three terms with the highest TW values obtained from the $BBCSet_{TR}$ were "Arsenal" (TW: 0.023), "Milan"(TW: 0.021), and "Liverpool"(TW:0.019), which occurred 183 times, 369 times, and 950 times, respectively, while the top three terms with the highest frequencies – "Cup" (1301 times), "Home"(1186 times) and "International" (1021 times) – had TW values of 0.008, 0.002, 0.003, respectively. In general, if a term does not occur frequently, it is regarded as a non-important term, and it will be assigned a low TW value by the AATI. At the same time, if a term occurs very often, it may not obtain a high TW value unless it occurs constantly in documents with high PV values.

3.2.4 Classification experiments

We classified Web documents in the *soccer* category with the following assumptions.

1. For AATI based classification:
   (a) The ontology defines the category *soccer*.
   (b) A document is recognized as belonging to the *soccer* category when the sum of concept values (Eq. 3.1) for this document exceeds a preset $PageValueThreshold$. The value of $PageValueThreshold$ is manually set based on the results obtained during the training phrase. Its value is set in a way that the values of precision and recall are balanced.

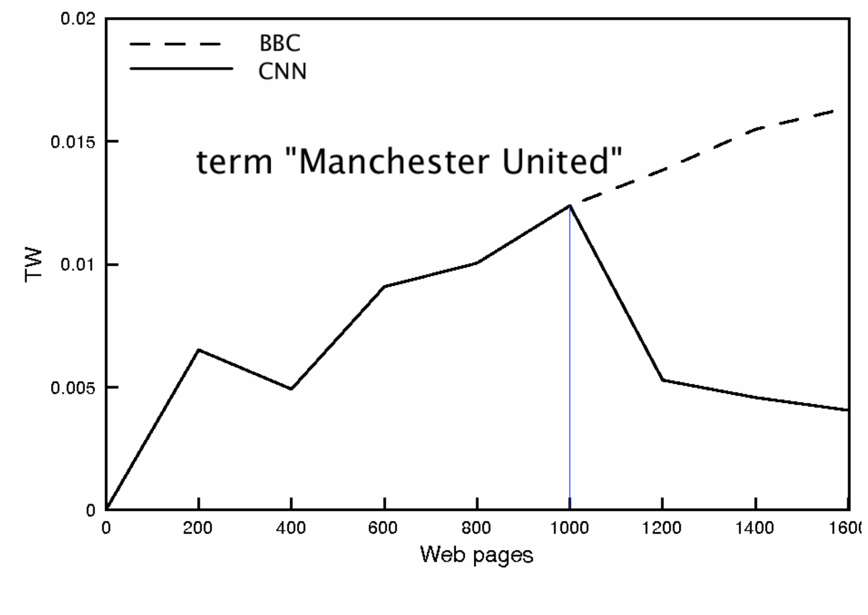


Figure 3.3: TW change of term "Manchester United"

2. For SVM based classification:
   (a) The terms taken from the ontology are used as input features for the SVM. There are more than four hundred terms (486 exactly), such as "referee," "dribbling," "Ronaldo," "AC Milan," and so on. Using all words in documents as input features for SVM models seems ideal but it leads to a computational burden. Many researchers utilize feature selection approaches to reduce the number of input features [49]. Here, we used all terms in the ontology as input features. We did so, because the ontology is a knowledge base that represents human knowledge about the category *soccer*, and the comparison of SVM and AATI built models is fair when the same set of input features/terms is used in both models.
   (b) The trained SVM model decides which category (*soccer* or non-*soccer*) a document belongs to.

3.2.5 AATI training phase

The general training process for a classifier built using AATI is illustrated in Fig. 3.4, where the training was performed using the $BBCSet_{TR}$ (1600 Web documents). The figure shows values of the $P_{F-Measure}$ calculated after evaluation of each upcoming document from the training set. We can say that the results represent cumulative $P_{F-Measure}$ values, and the experiment was

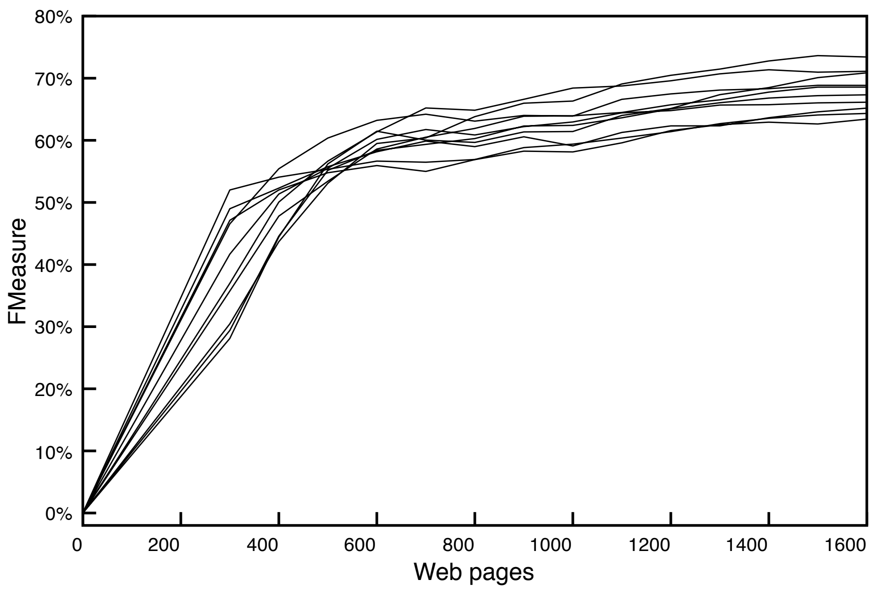


Figure 3.4: F-Measure for a training phase of AATI-based classifier

repeated ten times. In the first 600-700 documents the classifier was in its learning phase. Fig. 3.4 confirms that the values of $P_{F-Measure}$ stabilized after about 700-800 documents, and became similar for all ten runs.

3.2.6 Comparison with SVM

Our comparison of SVM and AATI models included the following three experiments:

- AATI and SVM models were trained using the $BBCSet_{TR}$ dataset. This process was repeated ten times – a ten-fold cross validation technique – and the results were presented as the mean value and its standard deviation. The testing phase was performed with $BBCSet_{TS}$, $CNNSet$ and $CBCSet_{TR+TS}$ (the entire CBC set). The results are shown in Table 3.1.
- The AATI and SVM models were trained with the $CNNSet$. The training was done only once, since the number of Web documents from the CNN site was very small. After the models were trained, their testing was performed with the $BBCSet_{TR+TS}$ (the entire BBC set) and the $CBCSet_{TR+TS}$ (the entire CBC set). The results are shown in Table 3.2.
- AATI and SVM models were trained using the $CBCSet_{TR}$. The experi-

ment was repeated five times – a five-fold cross validation technique – and results were presented as the mean value and its standard deviation. The experiment – its testing phase – is performed with $CBCSet_{TS}$, $CNNSet$ and $BBCSet_{TR+TS}$ (the entire BBC set). The results are shown in Table 3.3.

The experiments presented here were designed in a way that the testing phase was as realistic as possible, mimicking a possible utilization of a text classification system on the Web. The models were tested using datasets containing Web documents from a number of different websites. The frequencies of some terms varied among documents from different sites because of different interests of readers and levels of popularity. For example, involving the category *soccer*, the ratios between soccer vs non-soccer documents varied among all three datasets. Also, the documents were collected at different times – the web pages from BBC and CNN were downloaded in 2007, while from CBC in 2009. We assume such variety makes the comparison more comprehensive. Since $F - Measure$ is the combination of the *precision* and *recall* (section 3.2.1), it was our primary measure to evaluate SVM and AATI. *Accuracy* is not suited as the major metric because the number of soccer documents is much fewer than non-soccer documents.

Tables 3.1, 3.2, and 3.3 contain results obtained during the testing phase for SVM and AATI models. The first rows of Tables 3.1 and 3.3 contain results obtained for testing sets (subscript TS) created using documents crawled from the same sites as the documents used for building the training sets (subscript TR). That is, the $BBCSet_{TS}$ was used to obtain results presented in the first row of Table 3.1, and the $CBCSet_{TS}$ was used to obtain results presented in the first row of the Table 3.3. The results shown in the first row of Table 3.2 are for the $CNNSet$[6]. The results presented in the second and third rows of each table represent performances of the models when tested against documents crawled from different websites. For example, in Table 3.1 which represents performances of the model built using the $BBCSet_{TR}$ dataset, the results in the second row are for the $CNNSet$ dataset and results in the third row are for the $CBCSet_{TS+TR}$ dataset.

At the first glance, the results for SVM models in the first rows of Tables 3.1, 3.2, and 3.3 are good. The values of $P_{F-Measure}$ are close to or even over 90%. This proves that SVM performed very well when documents from the same website were used for testing and training. However, when documents from different websites were used, the $P_{F-Measure}$ values dropped dramatically for

[6]As mentioned earlier, due to the small size of the $CNNSet$ dataset we did not split it into two sets. The results presented in the first row are for the training date set.

Table 3.1: Results for SVM and AATI models trained with $BBCSet_{TR}$

Type	$P_{Precision}$	P_{Recall}	$P_{F-Measure}$	$P_{Accuracy}$
$BBCSet_{TS}$:				
SVM	$83.04 \mp 3.40\%$	$90.14 \mp 2.16\%$	$86.43 \mp 2.59\%$	$94.34 \mp 1.14\%$
AATI	$68.83 \mp 4.20\%$	$68.43 \mp 3.77\%$	$68.61 \mp 3.83\%$	$87.76 \mp 1.56\%$
$CNNSet$:				
SVM	91.37%	41.61%	57.18%	92.75%
AATI	89.88%	76.19%	82.47%	96.25%
$CBCSet_{TR+TS}$:				
SVM	49.09%	60.21%	54.08%	89.24%
AATI	66.49%	77.44%	71.55%	93.49%

Table 3.2: Results for SVM and AATI models trained with $CNNSet$

Type	$P_{Precision}$	P_{Recall}	$P_{F-Measure}$	$P_{Accuracy}$
$CNNSet$:				
SVM	89.16%	88.42%	88.79%	97.46%
AATI	83.80%	83.81%	83.81%	96.25%
$BBCSet_{TR+TS}$:				
SVM	55.57%	40.69%	46.98%	81.96%
AATI	65.90%	66.40%	66.15%	86.62%
$CBCSet_{TR+TS}$:				
SVM	33.96%	78.00%	47.32%	81.64%
AATI	41.48%	88.72%	56.53%	85.58%

Table 3.3: Results for SVM and AATI models trained with $CBCSet_{TR}$

Type	$P_{Precision}$	P_{Recall}	$P_{F-Measure}$	$P_{Accuracy}$
$CBCSet_{TS}$:				
SVM	$98.72 \mp 0.59\%$	$97.28 \mp 0.61\%$	$97.99 \mp 0.44\%$	$99.57 \mp 0.61\%$
AATI	$76.80 \mp 2.98\%$	$80.37 \mp 0.43\%$	$78.05 \mp 1.37\%$	$94.94 \mp 0.37\%$
$BBCSet_{TR+TS}$:				
SVM	74.70%	11.80%	20.38%	81.87%
AATI	63.91%	21.73%	34.43%	82.19%
$CNNSet$:				
SVM	95.00%	15.70%	27.00%	90.18%
AATI	92.86%	49.52%	64.60%	93.71%

Table 3.4: Results for AATI model trained using $CBCSet_{TR}$ at first, and "updated" with $BBCSet_{TR}$

Type	$P_{Precision}$	P_{Recall}	$P_{F-Measure}$	$P_{Accuracy}$
$BBCSet_{TS}$:				
AATI	55.78%	55.81%	55.79%	82.67%
$CBCSet_{TS}$:				
AATI	73.32%	80.90%	76.92%	94.86%
$CNNSet$:				
AATI	91.42%	70.07%	79.34%	95.71%

SVM (the last two rows of each table). It is likely that the SVM model suffered from overfitting. The AATI model performed much better in such cases. Compared with SVM models, AATI based classification led to higher values of $P_{F-Measure}$ when performed against "unseen" and "different" datasets. AATI generated models performed better in all those cases, the differences ranging from a maximum of 37.6%, to a minimum of 9.21%.

The results in Table 3.1 show a better performance for both SVM and AATI models than the results in Tables 3.2 and 3.3. This may be because the training set from the BBC website ($BBCSet_{TR}$) was the largest one. An interesting conclusion can be drawn from Table 3.3. The documents from the CBC website are quite new (crawled in 2009) compared to the documents downloaded from BBC and CNN sites (crawled in 2007). It seems both models are sensitive to changes in popularity of terms over time: the testing results in the second and third rows of Table 3.3 are worse than the results in the second and third rows of Tables 3.1 and 3.2. This demonstrates an undesirable feature of conventional classification models – they have to be retrained to "follow" the changes in the popularity of terms used in Web documents.

Since the AATI does not need a special training phrase, it is suitable as a web-based application. In its original design, the AATI would allow for updating TW values. To illustrate this behavior, one more set of experiments was performed. The AATI model built using the $CBCSet_{TR}$ was "exposed" to Web documents from the $BBCSet_{TR}$ and the TW values were updated. The results in Table 3.4 represent an improvement in the performance of this model for the $BBCSet_{TR+TS}$ and $CNNSet$ datasets compared to the results shown in Table 3.3. The $P_{F-Measure}$ value for the $BBCSet_{TS}$ increased to 55.79%, compared to 34.43% for the model built using $CBCSet_{TR}$. However, the value is smaller when compared with the value obtained for the AATI model built using only $BBCSet_{TR}$. Similarly, the $P_{F-Measure}$ value for the $CCNSet$

increased from 64.60% to 79.34%. For the $CBCSet_{TS}$ we observed a slight decrease in the performance; the $P_{F-Measure}$ value was 76.92% for the new model, and 78.05% for the previous model (Table 3.3). At the same time, the value is better than for the model built using $BBCSet_{TR}$ – 71.55%. The differences are attributed to changes in TW values caused by the $BBCSet_{TR}$.

3.2.7 Conclusion

The fact that the AATI was able to update TWs with "unknown" Web documents makes it suitable for Web-based applications. In the basic experiments (section 3.2.3), the AATI displayed its ability to alter the TWs of the term "midfielder" and term "pass" using upcoming Web documents. Moreover, the experiment involving the term "Manchester United" showed how its TW changed with different sets of Web documents. The basic experiments demonstrated the nature of AATI.

The classification experiments were designed and applied like classic experiments to verify text categorization models. This was not a satisfactory way to employ AATI because it was not designed for categorization. The AATI was designed to obtain TWs (term weight, section 3.1) for deciding whether a document belongs to a category or not through its PV (page value, section 3.1). The PV of a document is the sum of related CVs (concept values, section 3.1), which represents how much information of the related concepts is in the document. The PV is a reasonable definition to satisfy the mathematical requirements that the TW (the importance of terms, section 3.1) set exists, but it is not an optimized measure to assign documents to different categories.

For example, let us assume there are two documents d_1 and d_2: in d_1 there is one related term repeating n times; in d_2, there are m different related terms (attached to m different concepts), and every term is present once. n is much larger than m. The PV of d_1 is equal to the CV of the concept that the term is attached to, and the CV is the TW of the term times n. The PV of d_2 is the sum of the m CVs (the number of the concepts that the m terms are attached to), and every CV is equal to the TW of one term. Since n is much larger than m, the PV of d_1 tends to be larger than the PV of d_2. However, as a value to indicate categories, this result may cause problems because TWs but not PVs are the solutions of the AATI. Therefore, applying AATI on categorization may not appropriate. That is, the experiments on the HOTIR-based system (Chapter 6) is more reasonable than the classification experiments (section 3.2.4), as the former utilizes TWs and the latter uses

PVs.

However, the classification experiments can directly check the results of the AATI based on the assumption that reasonable PVs are generated by properly assigned TWs. We did not expect the AATI to perform better than the SVM, which is one of the best models for text categorization. A reasonable result was acceptable and the experimental results are encouraging.

Though the SVM showed its advantage in categorization, the AATI proved its advantage in adaptability. The SVM performed better when the news pages used for training and testing were obtained from the same website. When the Web documents for training and testing were from different sites, the AATI generated better results than the SVM. The two outstanding features of the AATI were : (1) the Web documents in the training set of the AATI did not need to be categorized; and (2) training of the AATI was easily stopped and started. These two factors dramatically reduce the involved human-labor and help to make the knowledge (TW set) obtained by the AATI up-to-date. In the next chapter, we introduce how deploy ATTI into HOTIR.

Chapter 4

Hybrid ontology-based textual informaton retrieval (HOTIR): Methodology

In this chapter, we introduce a hybrid ontology-based textual information retrieval framework called HOTIR. In section 4.1 three components of HOTIR are introduced: KMgmt (knowledge management), QryProc (query processing), and Eval (evaluation). KMgmt, described in section 4.2, works with a knowledge base to provide extra definitions of concepts. QryProc translates normal keyword-based queries into HofC-based queries, and expands them with knowledge in the knowledge base. The functions and responsibilities of QryProc are explained in section 4.3. Eval, described in section 4.4, evaluates the relevance of queries and documents. Section 4.5 is a summary of this chapter.

4.1 Overview of HOTIR

HOTIR is a concept-based information retrieval framework. Fig. 4.1 displays a sketch of the procedures in HOTIR:

- Firstly, a conventional query (keyword-based) is translated into a concept-based query. In a concept-based system, a query is a list of concepts instead of keywords. Because concepts are abstract, their definitions are needed in the query. The definition of the concept in HOTIR includes: (1) terms that describe the concepts and their importance; (2) related concepts that describe the primary concept and their importance; (3) relations between terms and concepts. Computer agents in HOTIR can

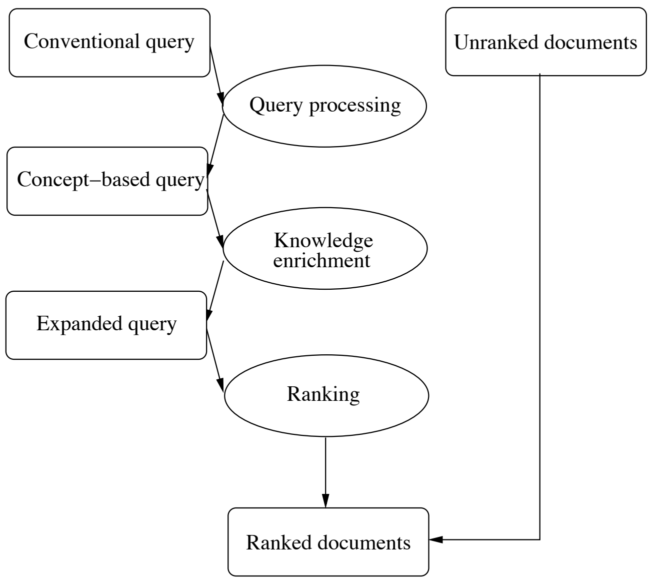


Figure 4.1: The basic procedures in HOTIR

identify the concepts in Web documents from definitions in the query.

- Second, the concept-based query is expanded by extracting knowledge from the knowledge base. Queries are from users and we can not expect all users are experts to provide enough definitions of the concepts. The expanded query contains more information (if it is available in the knowledge base) that is helpful to retrieve relevant documents.
- Finally, documents are ranked according to the satisfied concepts of the HofC in them. To a concept defined only by terms (no subconcepts), satisfaction of the concept is decided by the presence of those terms in the document. To a concept defined by a list of terms and subconcepts, satisfaction of the concept is decided by the presence of those terms and the satisfactions of its subconcepts. In the end, the satisfaction of a query is decided by the satisfactions of concepts defined in it.

HOTIR successfully solves three difficulties in utilizing concept matching to rank documents:

- How is the knowledge base constructed? A knowledge base is a source of supplementary knowledge to enrich queries. A concept is defined by terms (including their importance), related-concepts (including their importance) and their relations. A simple example is shown in Fig 4.2, the concept *Soccer Player* is defined by its related-concept *Player Drogba* and *Player Terry*. Concept *Player Drogba* is defined by three terms

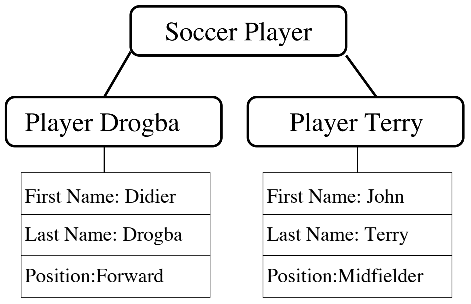


Figure 4.2: Concepts and terms

"Didier" (value of attribute *First Name*), "Drogba" (value of attribute *Last Name*) and "Forward" (value of attribute *Position*). Since it is not reasonable to expect that a user to provide enough terms to create knowledgeable queries, subconcepts and their importance to the primary concept are added to the queries by the knowledge base as a necessary component in HOTIR [1]. The definitions of different concepts are stored in the knowledge base, which can be used to enrich definitions of queries as supplementary knowledge. The knowledge base here is constructed based on an ontology (section 2.1.2) and the new self-adaptive AATI scheme. The former provides lists of terms, concepts, and their relations, and the latter assigns importance to terms, which in turn assigns importance to concepts.

- How are queries built in a concept-based manner? A concept-based system requires concept-based queries. Users generally provide keyword-based queries composed of a few keywords. HOTIR translates keyword-based queries into organized concepts from definitions in a knowledge base, which are in the form of hierarchy of concepts (HofC) (section 2.1.4). Because both the HofC and the knowledge base contain hierarchies, it is easy to expand queries with the definitions of terms and concepts in the knowledge base.
- How can documents be ranked? The knowledge base provides extra definitions for the target concepts and the HofC contains hierarchies of concepts and related subconcepts. In general, the satisfaction of a HofC by a document is an aggregation of satisfactions of concepts defined in the query. And the satisfaction of a concept is an aggregation of satisfactions of terms and subconcepts. A term or a identified subconcept in the document is treated as a piece of information that satisfies a

[1]Concepts related to the primary concepts of the query are called subconcepts (or superconcepts) in the queries. See section 4.3.1 for details.

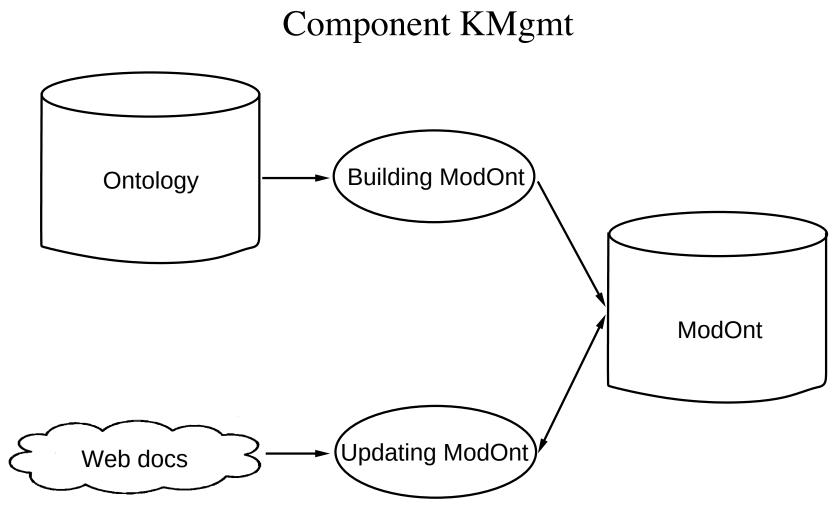


Figure 4.3: Component KMgmt component in HOTIR system

concept in the HofC. Ordered weighted averaging (OWA) operators are applied (section 2.1.3) to aggregate multiple pieces of information and rank documents according to their relevance to a query.

4.2 Knowledge management (KMgmt) component

In HOTIR, KMgmt manages the knowledge base for query expansion. In concept-based models, documents are lists of concepts rather than lists of terms. Because documents are expressed directly through concrete texts instead of abstract concepts, the process of recognizing relevant documents is based on the identification of concepts in the documents that have definitions in the query. That is, concepts cannot be identified until they are defined.

The knowledge base, therefore, stores definitions of concepts. Concept definitions are composed of two parts:

1) Concept definitions include lists of related-concepts, terms and their relations. An ontology, a specification of a conceptualization, is a good choice to provide this type of information.
2) Concept definitions also include the importance of concepts and terms. The importance of a term/concept represents a level of contribution of this term/concept toward the defined concept. The importance value of a concept is calculated based on the importance values of all terms and

concepts attached to it. Human experts can list related concepts/terms to explain one concept, but they can hardly assign numbers to represent their importance. For example, let us assume *Soccer Player* as a primary concept to be defined. Though an expert can find terms and related-concepts to define this primary concept, like the name of the player, the team he plays for, the position he plays, and so on, it will be difficult for the expert to assign importance values to these terms and concepts. Computers use mathematical relations to do this job.

As shown in Fig. 4.3, there are two basic modules (presented in ellipses) in KMgmt, which builds a modified ontology (ModOnt) from an existed ontology and updates the ModOnt, respectively. ModOnt is a new concept proposed in this work; It is the knowledge base for HOTIR (see section 4.2.3 for more details). The ModOnt is generated using input from any native ontologies. The process of updating the ModOnt is accomplished by the new self-adaptive scheme, AATI (Adaptive Assignment of Term Importance), which assigns importance to terms/concepts in the ModOnt, then updates it.

4.2.1 Ontology and modified ontology (ModOnt)

Ontology

In an ontology, a *class* is described with *properties*. There are two types of property: datatype property, and object property (section 2.1.2). In such a case, a concrete piece of information that is called an *individual* is an instance of a class – it is created by assigning values to class *properties*.

Figure 4.4 shows a simple ontology that includes three classes: *Soccer*, *Soccer Player*, *Soccer Club*; and four individuals: *Player Drogba*, *Player Terry*, *Club Chelsea*, *Club FC Barcelona*. Two individuals of the class *Soccer Player* are *Player Drogba* and *Player Terry*. They are described by three datatype properties and one object property. The three datatype properties are *First Name*, *Last Name* and *Position*. The object property is *InClub*. It means that *Soccer Player* is described by three literal values (terms), which are the values of the datatype properties, and one individual, which is the value of an object property.

Individual *Player Drogba* is defined by the term "Didier" as the value of a datatype property *First Name*, by the term "Drogba" as the value of datatype property *Last Name*, by the term "Forward" as the value of datatype property *Position*, and by individual *Club Chelsea* as the value of the object property *InClub*. Similarly, individual *Player Terry* is described by "John" as *First*

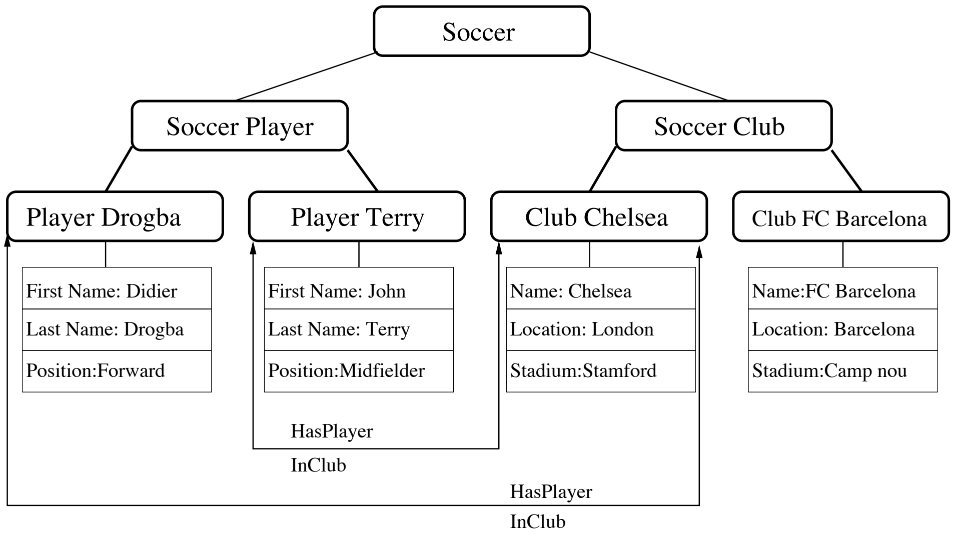


Figure 4.4: A simple ontology

Name, "Terry" as *Last Name*, "Midfielder" as *Position*, and individual *Club Chelsea* as the value of *InClub*.

Individuals of the class *Soccer Club* (*Club Chelsea* and *Club FC Barcelona*) are also described by three datatype properties and an object property. The three datatype properties listed are *Name*, *Location*, and *Stadium*. The object property is *HasPlayer*, which is the inverse property[2] of the object property *InClub*. The values of the datatype properties and the object property for individuals *Club Chelsea* and *Club FC Barcelona* are shown in Fig. 4.4.

Modified ontology (ModOnt)

A native ontology was not a suitable knowledge base for HOTIR, because:

1. A HofC contains only concepts, while an ontology contains both classes and individuals. Thus it was difficult in HOTIR to expand HofC-based queries with the help of a knowledge base that was in the format of an ontology.
2. An ontology does not natively contain importance (represented by vector M), and linguistic quantifiers (represented by Q) which are necessary for OWA operators to rank documents for HofC-based queries (section 2.1.3).

[2]If an object property o_1 points object A from object B, and another object o_2 points B from A, o_1 and o_2 are regarded as inverse properties.

In the classic definition of ontology, classes are different from individuals because classes are abstract and individuals are concrete. That is, class defines properties that have no concrete value; while individuals are described by a list of values of properties.

However, the concepts in the HofC are defined as a combination of classes and individuals in the ontology (section 4.3.1). That is, the concept in the HofC has definitions of attributes and, possibly, concrete values of the attributes.

The difference between an ontology and a HofC lies in their respective responsibilities. Unlike an ontology that is designed to store a large amount of organized knowledge, a HofC represents a list of specific information for retrieval of relevant documents. If a class in an ontology has no individuals, i.e., no concrete values to describe it, it can be filled up later by others. But it is not reasonable that a HofC should contain a concept that has no concrete value to define it in a query. Such a concept is useless because there is no way to identify it in documents.

Therefore, in the knowledge base in HOTIR, all classes and individuals in the ontology were translated into concepts in the ModOnt. The definition of concept in the ModOnt is the same as that in the HofC, which guarantees the smooth expansion of queries by KMgmt.

The following formats and operations pertain to this thesis:

- *Concepts* are used in the HofC and the ModOnt; *classes* and *individuals* are used in the ontology;
- *Attributes* are used in the HofC and the ModOnt; *properties* are used in ontology;
- The values of datatype attributes in the HofC and the ModOnt are called *terms*, which can be recognized directly from Web documents.
- OWA operators are applied to implement aggregations of HofCs; OWA operators require M (importance) and Q (linguistic quantifiers)[3].

The steps performed to build the ModOnt were:

1. All classes and individuals are translated into concepts;
2. A default value of Q is added to each concept; the default Q is *most*;
3. The default value of M is set as zero for each concept and term.

The ModOnt is the knowledge base in KMgmt, which is updated by the AATI and can be used to expand queries in HofC formats.

[3]Details about M and Q are in section 2.1.3 and section 2.1.4

4.2.2 AATI scheme for term importance

ModOnt is a model of the domain, that is, it presents concepts, terms, and their relations in the domain. The AATI scheme integrated with the ModOnt, provides importance values of the terms and concepts in the ModOnt.

In chapter 3, the AATI was theoretically studied and then validated through experiments. In the KMgmt component, the AATI scheme was implemented based on power iteration [96], which is used to find eigenvectors (T in Eq. (3.8)) of a matrix (C in Eq. (3.8)) in linear algebra. The main steps of AATI implementation were as follows:

1. Translate an ontology into a ModOnt;
2. Take a new Web document;
3. Parse the document and annotate terms from the ModOnt in the document;
4. For each term do one of the following:
   (a) *if* TW is not zero (it means the term has already appeared in documents), take this as its new TW;
   (b) *if* TW is zero (the term has not been found in any documents), randomly generate a number between 0 and 1, and assign it as its TW;
5. Calculate the PV of this Web document, based on Eq. (3.4);
6. Update the TWs of those terms found in this Web document using Eq. (3.7);
7. Normalize TWs across all terms (make the sum of TWs equal to 1);
8. If there are no more Web documents, $STOP$; otherwise go back to *step 2*.

In the KMgmt, the AATI continuously updates TWs with the upcoming Web documents. Intuitively, we can say that the PV is high when terms occurring in a document have high TWs (Eq. (3.4)). At the same time, we can state that the TW of a term is high when the PVs of documents that contain the term are high (Eq. (3.7)).

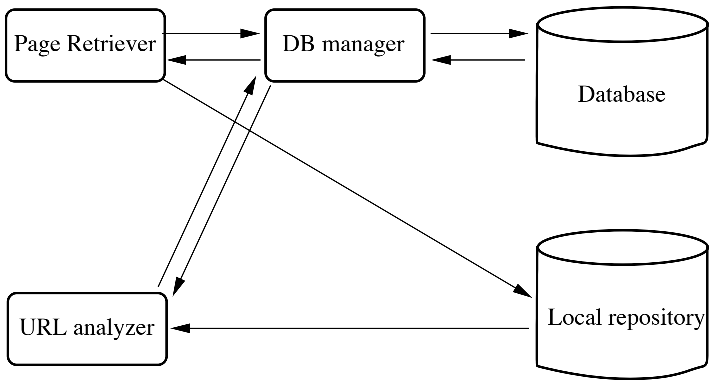


Figure 4.5: Modules in the crawler

4.2.3 Other computer agents in component KMgmt

Web crawler

Web crawler, also called Web robot and Web spider, is a computer agent that automatically retrieves Web documents. We developed our own Web crawler based on Python[4].

As shown in Fig. 4.5, the *database* stores information such as the uniform resource locater (URL –the unique "address" of a Web document), retrieving status, priority, local path of the downloaded document, etc. At the same time, the *local repository* saves retrieved documents. The working procedures of the crawler are:

1. A few URLs (seeding URLs) are entered into the *database*.
2. *Page retriever* obtains URLs from the *Database* through *DB manager* to download documents. After the Web documents are downloaded to the *Local repository*, *Page retriever* returns downloading information such as retrieving status, local paths,etc. to *DB manager*, which is then used to update the saved information in the *Database*.
3. *URL analyzer* sends requests to *DB manager* for the local paths of downloaded documents in the *Database*. With the local paths returned by *DB manager*, *URL analyzer* gets the documents from the *Local repository* and analyzes links in the documents. The link analysis includes extracting links (URLs of other Web documents) in the Web documents and setting priorities. The URLs and their priorities are sent back to the *Database* through *DB manager* for downloading. Priorities of URLs are

[4]Python is a programming language. See its official website at: http://www.python.org/ for more details.

calculated based on the rules we set, which are defined by the crawling purpose. For example, if the crawler is expected to retrieve documents only from a specific domain, the URLs from the domain are assigned a priority value of 1.0; while the URLs from other domains are assigned 0.0. Thus URLs from other domains will not be used to download Web documents by *Page retriever*.

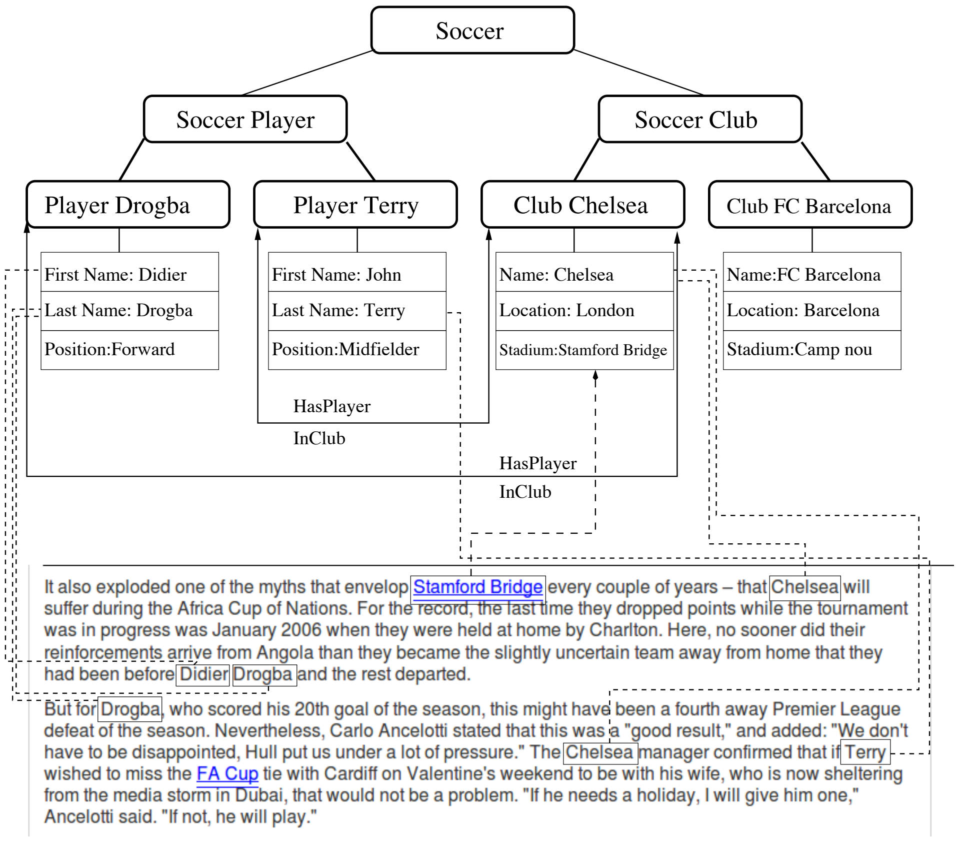
Steps **2** and **3** are repeated until there are no URLs available for downloading. Except for the seeding URLs, *Page retriever* retrieves Web documents with the URLs extracted by *URL analyzer*. On the other hand, *URL analyzer* extracts URLs from Web documents downloaded by *Page retriever*. In this way, the Web crawler we built can automatically retrieve a large number of Web documents from a couple of seeding URLs.

Annotation

We have developed a Java annotation module based on the UIMA (Unstructured Information Management Architecture) library created by IBM [97], which adds extra information to the texts of Web documents.

In HOTIR, annotation is the basis for computer agents to identify concepts. Terms, which are literal values of the datatype attributes in the ModOnt, can be used to identify concepts in texts. Computer agents can locate terms that are organized in the ModOnt from texts in documents, and add extra information to them. The information added to a term can include its position (begin, end) in a document, the concept it belongs to, and the attribute it belongs to. Annotation in HOTIR connects the ModOnt with documents: the frequency of a term presented in a document decides how well the document satisfies the concept, to which the term is attached in the ModOnt.

Fig.4.6 shows how a document is annotated. For example, the term “Chelsea” in the ModOnt is the value of attribute *Name* of concept *Club Chelsea*. The annotation module finds that this term is present twice in the document. So, for the ModOnt and the annotation module, a string “Chelsea,” which to most computer agents is meaningless (nothing but a binary string), has its own semantics/meaning: it is the name of a soccer club.



It also exploded one of the myths that envelop Stamford Bridge every couple of years – that Chelsea will suffer during the Africa Cup of Nations. For the record, the last time they dropped points while the tournament was in progress was January 2006 when they were held at home by Charlton. Here, no sooner did their reinforcements arrive from Angola than they became the slightly uncertain team away from home that they had been before Didier Drogba and the rest departed.

But for Drogba, who scored his 20th goal of the season, this might have been a fourth away Premier League defeat of the season. Nevertheless, Carlo Ancelotti stated that this was a "good result," and added: "We don't have to be disappointed, Hull put us under a lot of pressure." The Chelsea manager confirmed that if Terry wished to miss the FA Cup tie with Cardiff on Valentine's weekend to be with his wife, who is now sheltering from the media storm in Dubai, that would not be a problem. "If he needs a holiday, I will give him one," Ancelotti said. "If not, he will play."

Figure 4.6: An example of annotation in a document

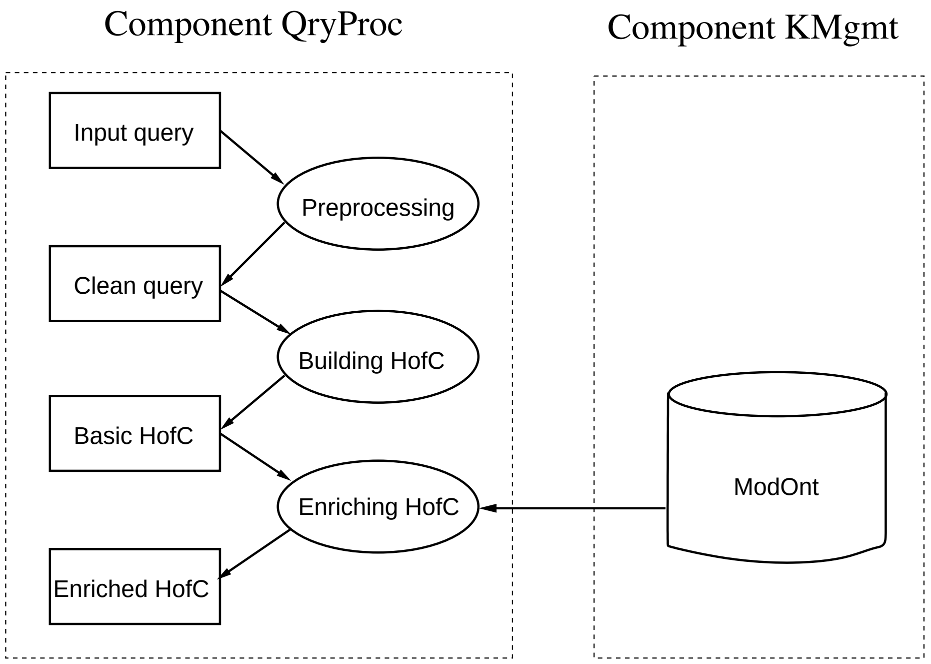


Figure 4.7: Component QryProc in HOTIR

4.3 The Query processing (QryProc) component

The second component in HOTIR, the QryProc, contains three modules which preprocess users' conventional keyword-based queries, build HofCs from conventional queries, and expand the HofCs with knowledge from the ModOnt.

The preprocessing module removes stop words and stems words from queries. Stop words include articles such as "the" and "a" and prepositions such as "in" and "from." The stemming process removes suffixes and prefixes reducing words to their stems. For example, the term "extended" can be stemmed to "extend."

The module to build HofC-based queries translates keyword-based queries into concept-based queries as explained in section 4.3.2. Another module retrieves knowledge from the ModOnt to complete the definitions of concepts in the basic HofC, which helps identify them in Web documents later. The details are presented in section 4.3.2.

4.3.1 HofC as query representation

A query is a statement representing the interest of a user. The user provides keywords that describe objects that the user desires to be retrieved from an information repository. Depending on the applied IR techniques, documents are ranked based on the presence of query keywords in each document. The keywords in the queries are represented as a flat structure, that is, there is no indication that the keywords are related to each other, or that some keywords depend on others. This is a simplification suitable for automatic processing but it is not a realistic representation of a human-like way of finding relevant text. For a human, all keywords are interconnected. They constitute a network of words representing concepts. The activation of a single word or concept initiates activations of related concepts, and finding those related concepts leads to the selection of documents.

In HOTIR, a query is represented by a single HofC – a tree-like structure built of concepts (vertices) and terms[5] (terminal vertices/leaves) (section 2.1.4). The structured form of a query, that resembles a network of words and concepts, provides us with a more intuitive way of expressing things we are looking for. A document that satisfies an HofC-based query is relevant to this query. An aggregation of satisfactions is performed at each vertex of HofC using OWA, and associated with the linguistic quantifier Q, and the vector M of importance values.

As presented in section 2.1.3, a linguistic quantifier Q associated with a concept is used at the time of the aggregation of satisfactions of terms and other concepts (subconcepts) related to the primary concept. The possible quantifiers are *some*, *most*, and so on. The notion of linguistic quantifiers plays an important role in representing different ways of combining satisfactions.

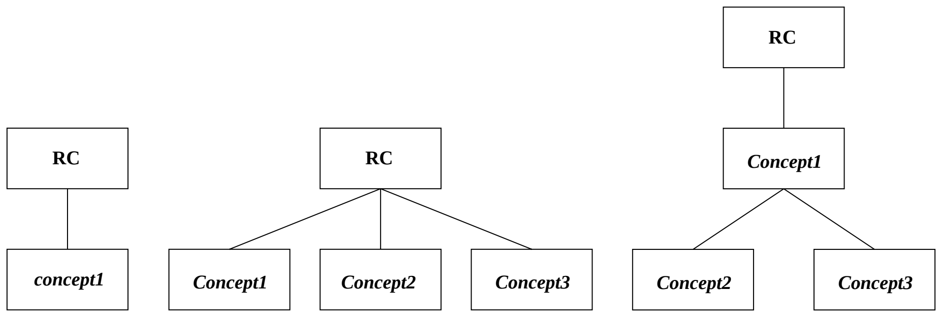
A vector M (section 2.1.3) represents the importance of each term and concept that is taken into consideration during an aggregation process. Not all terms and concepts contribute uniformly to the activation of the primary concept.

Some definitions of the HofC, as used to express a query in HOTIR, are clarified in the following sections.

Root concept (RC) in HofC

A concept-based query contains concepts and terms to be identified in documents. Our goal was to find out if a given document contained terms used

[5]Literal values of datatype attributes, as defined in section 2.1.4, are replaced here with terms.



(a) One Keyword (b) Multiple Keywords (c) Structured Keywords

Figure 4.8: Examples of Applying Root Concept(RC)

to describe concepts in an HofC-based query. This translated into a process of estimating a level of activation (or satisfaction) of the HofC. In order to simplify this process, a special concept, called *Root Concept (RC)*, was automatically added to each HofC-based query. In this case, the activation of the HofC is equivalent to the activation of the top level concept of HofC.

Some possible structures of HofC-based queries are presented in Fig 4.8. In Fig. 4.8a, a query contains one single concept; in Fig. 4.8b, a query contains multiple concepts in a flat structure; and in Fig. 4.8c, a query contains a few structured concepts. Activation of the HofCs was simplified to the activation of a single root concept (RC), regardless the complexity of HofCs.

Concept *versus* individual and class

An HofC is a hierarchical structure built with concepts and attributes. An ontology is built with classes. Each class has zero or more properties that define its features. A concrete instance of a class is called an individual.

A closer look at the meaning of these terms leads to the conclusion that both HofCs and ontologies are built with similar components. HofC concepts have their equivalents in ontology classes. The concept in a HofC has attributes that can be treated as properties of classes. However, there is a small difference here. For a HofC, concept attributes represent concrete information that defines a particular concept. Class properties are just definitions of types of properties. Concrete information – values assigned to properties – are instances of concrete classes (individuals). An individual of a given class is an entity that contains values assigned to the properties of this class.

In general, we translate both *class* and *individual* in ontologies as *concept* in HofCs. Every concept in the HofC can be described by attributes with real values. That is, the difference between individual and class has been fuzzied in the HofC.

Subconcept (Superconcept) ***versus*** related-concept

ModOnts are built from ontologies. To make query expansion smooth, the concept in the ModOnt and the concept in the HofC have the same definition. However, connected concepts are called related-concepts in the ModOnt, and called subconcepts (superconcepts) in the HofC. The difference is decided by their responsibility: ModOnts are knowledge bases; while HofCs are query expressions.

- The connections in ModOnts, like the connections in ontologies, represent different meanings. For example, let us assume there is a ModOnt based on the ontology shown in Fig. 4.4. The connection between the concept *Player Drogba* and the concept *Soccer Player* is different from the connection between the concept *Player Drogba* and the concept *Club Chelsea*. The former connection means "is-a" or "has-a." That is, *Player Drogba* is-a *Soccer Player* or *Soccer Player* has-a *Player Drogba*. The latter connection means "InClub" or "HasPlayer". That is, *Player Drogba* InClub *Club Chelsea* or *Club Chelsea* HasPlayer *Player Drogba*.
- The connections in HofCs only represent relative levels in the hierarchy structure, that is, one concept is a subconcept (superconcept) of another concept. HofCs are built to represent queries where subconcepts contribute to defining their related superconcepts. The connection may be meaningless, because a user can connect any concepts to express his/her interests. For example, let us assume the knowledge in the ontology shown in Fig. 4.4 are correct. A user can create his/her own HofC which has player *Player Drogba* as the superconcept, and *Club FC Barcelona* as the subcocnept if he/she assumes that Player Drogba will transfer to Club FC Barcelona and he/she wants to check if there is any rumour about it. In this case, the satisfaction of *Club FC Barcelona* contributes to the satisfaction of *Player Drogba*, and in turn to the satisfaction of the whole HofC, though it seems the connection should not exist.

In general, subconcepts (superconcepts) are used to describe two connected concepts in the HofC; while the term related-concepts is used to describe two connected concepts in the ModOnt.

Connections in the HofC

Though connections in the HofC only represent layers, we define three types of connections to describe different situations:

1. $TypeI$ connection: connects concepts that are parts of users' queries. That is, the concepts provided directly by users are related to each other through a $typeI$ connection. It is the strongest type of connections.
2. $TypeII$ connection: connects concepts defined in the domain ontology. $TypeII$ connections are "is-a" and "has-a" relations. The first step to expand a HofC-based query is based on this type of connection (section 4.3.2).
3. $TypeIII$ connection: also connects concepts defined in an ontology. $TypeIII$ connections represent all different types of relations except "is-a" or "has-a" relation. This is the weakest connection.

4.3.2 Building and expanding HofC

The process of identifying concepts in a document depends on the ability to determine if HofC concepts can be inferred based on the texts of the document. This depends on the contents of the query HofC itself – the more terms and concepts used to build the HofC, the higher the chance of proper evaluation of the relevance of a document. We do not expect users to provide comprehensive concept-based queries containing a large number of terms and concepts. The model accepts conventional keyword-based queries and automatically translates them into HofC-based queries. That is, the terms are translated into concepts with specific hierarchies. Hierarchical information represents how those concepts are related to each other, which helps to define a more flexible representation for users to express their needs.

First, each keyword in a query is regarded as a concept and connected with a $typeI$ connection, which is called basic HofC in HOTIR. The keywords play as names of the concepts, and are used to retrieve knowledge of equivalent concepts in the ModOnt. The knowledge includes subconcepts, attributes, linguistic quantifiers (Q), and importance vectors (M). The expansion steps are:

1. Search the ModOnt for the equivalent concept of each keyword:
   (a) If the name of a concept in the ModOnt is the same as the keyword, it is called an equivalent concept of the keyword.

   (b) If no concept is found; then check if there is a concept in which a literal value of a datatype attribute (a term) matches the keyword. If there is, the concept is called an equivalent concept of the keyword.
   (c) If no concept is found, there is no equivalent concept of the keyword in the ModOnt.
2. Each concept in the basic HofC for which an equivalent concept in the ModOnt is found is represented as the *primary concept* below:
   (a) Copy all related-concepts of the equivalent concepts from the ModOnt as subconcepts to the *primary concept*, and connect them to the *primary concept* with *typeII* connections.
   (b) Copy all *datatype* attributes (of all copied related-concepts) directly to the corresponding subconcepts with their values as attributes.
   (c) For *object* attributes: all connected concepts are copied and bound as concepts to the *primary concept* through *typeIII* connection.

If no equivalent concept is found in the ModOnt, a string-type attribute "NAME" is added to the concepts built from the keywords. The value of this attribute is the keyword itself. In this way, all concepts in the HofC are created, and ready for evaluation.

A HofC-based query includes concepts and hierarchies. Identification of a concept from the ModOnt through a keyword is based on finding the matched keyword in the names of concepts or in the literal values of attributes of concepts in the knowledge base. Once an equivalent concept is found, all knowledge regarding it in the ModOnt belongs to the knowledge of the query.

It is also possible that a user provides other linguistic quantifiers Q that are different from the default one in the ModOnt. Different linguistic quantifiers mean different ways of aggregation (see section 2.1.3), which provides advanced flexibility.

4.4 Evaluation (Eval) component

Component Eval accepts expanded HofC-based queries (output from QryProc), identifies concepts of those queries from documents, aggregates the identifications and generates rankings of documents, as shown in Fig. 4.9.

A HofC-based query contains not only concepts themselves but also their definitions, so that those concepts can be identified from the texts of documents.

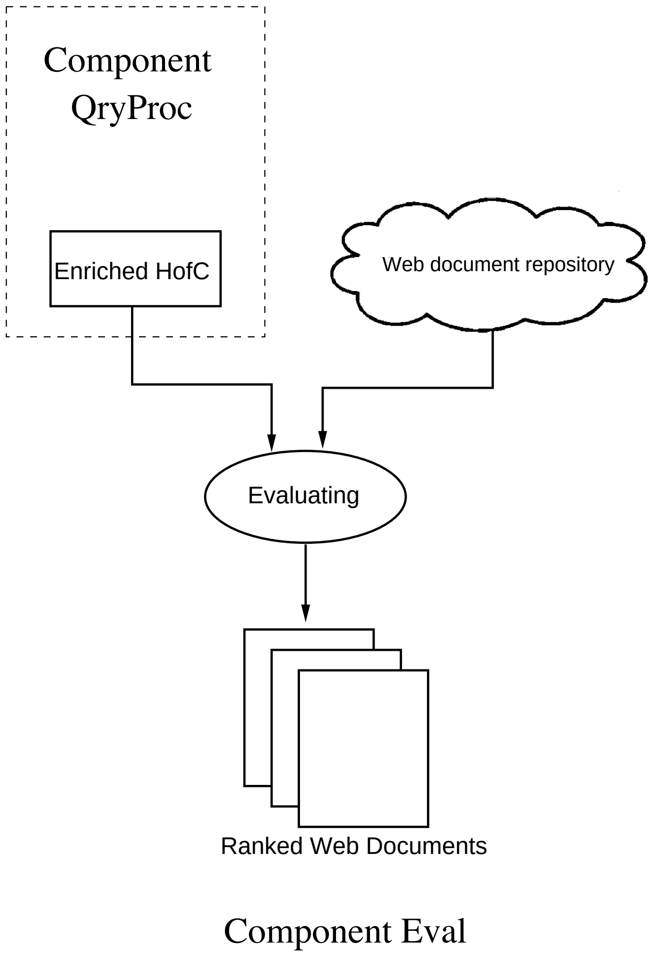


Figure 4.9: Component Eval in HOTIR system

The definitions in the HofC, also known as the knowledge of concepts, include subconcepts, terms, their importance, and their relations.

The evaluation of relevance in Eval is mapped as the "activation" of the HofC based on OWA (section 2.1.3). The more the HofC is activated, the higher the document is ranked.

4.4.1 Satisfaction, importance and aggregation weights

Before explaining how Eval works, three regularly used terms: satisfaction, importance, and aggregation weights, are defined here:

- *Satisfaction* represents how well a criterion in the HofC is satisfied by the text from a document. A *criterion* can be an attribute, a concept or even a complete HofC. The satisfaction of the root concept (RC) (section 4.3.1) is the satisfaction of the whole HofC.
- *Importance* represents how important a criterion is. With the AATI scheme in KMgmt, each entity (terms and concepts) includes a measure of importance in the ModOnt.
- *Aggregation weight* is the weight of a criterion calculated by the OWA operator (section 2.1.3). Aggregation weight depends not only on the importance of the criterion but also on the linguistic quantifier. In section 4.4.2, use of the linguistic quantifier and its importance in calculating aggregation weights is explained.

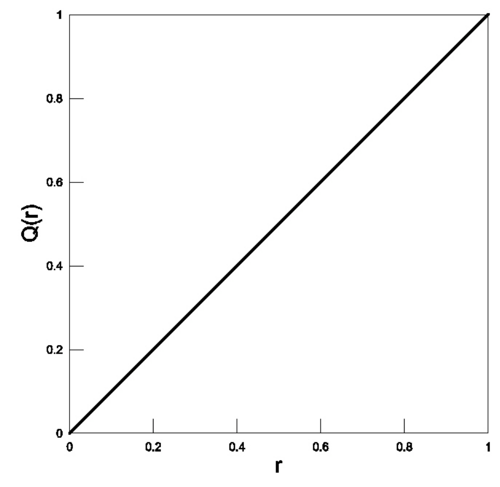
4.4.2 Linguistic quantifier

The concept of linguistic quantifiers was introduced by Zadeh [27] in the early 1980s. Some examples are *all*, *most*, *at least half* and *about one third*. To formally represent those quantifiers, Zadeh suggested using a fuzzy subset $Q(r)$ as a linguistic expression corresponding to a quantifier that indicates the degree to satisfy the concept for any proportion $r \in [0, 1]$.

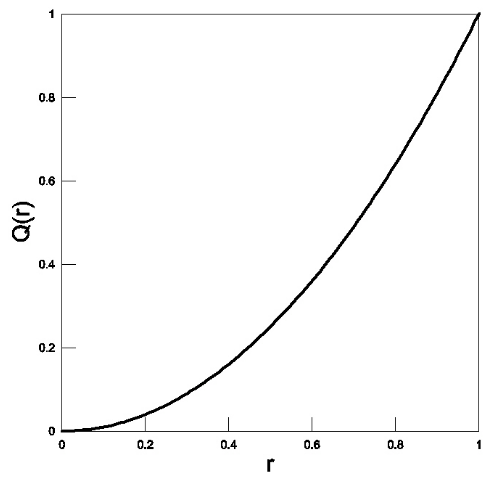
Yager [28] used the linguistic expression $Q(r)$ to obtain a weighting vector W associated with an OWA aggregation. These quantifiers were then able to guide aggregation procedures by verbally expressed concepts in a description independent dimension.

In HOTIR, two linguistic quantifiers were modelled:

- $Q(r) = r$ represents the linguistic quantifier *some*, Fig. 4.10a. The equation means that the aggregation weight of a criterion depends only on its



(a) *some*



(b) *most*

Figure 4.10: Linguistic quantifiers

importance value. That is, the relations between concepts do not change their aggregation weights.

- $Q(r) = r^2$ represents linguistic quantifier *most*, Fig. 4.10b. Using this equation, the aggregation weight of a criterion tends to be large when it is less satisfied (more details are in section 2.1.3).

4.4.3 Aggregation of satisfactions of datatype attributes in a single concept

A criterion in a single concept can be a datatype attribute, an object attribute, or a subconcept. The simplest situation is a single concept containing only datatype attributes.

The literal values of datatype attributes are called *terms* in HOTIR. The presence (more precisely, frequencies) of a term or terms in a Web document to be rated decides how well the criterion (datatype attribute) is satisfied. Mathematically, the satisfaction of a term s_i is:

$$s_i = e^{-\frac{1}{freq_i}} \tag{4.1}$$

where $freq_i$ is the frequency of the term in the document.

The reasons for using Eq. 4.1 are:

1. It is a monotonically increasing function, so when its frequency increases, the satisfaction increases.

2. The function becomes "flatter" with increasing values of the argument, so if a given term occurs many times in a document, its satisfaction does not overshadow the satisfactions of other terms.

Each criterion has an assigned value of importance. Assuming there are n different datatype attributes in a single concept C, the importance vector M contains n elements, where each element represents the importance of a criterion. The satisfaction vector S also contains n elements, where each element is the satisfaction of a criterion by a document. Then the ordered satisfaction vector S' (Eq. 2.2) is generated by S, in which s'_j is the jth largest of all satisfactions s_1, s_2, ..., s_n. Furthermore, we assume μ_j denotes the importance weight associated with the attribute that has the jth largest value. Thus if s_3 is the largest value, then $s'_1 = s_3$ and $\mu_1 = m_3$. Based on this, the OWA weighing vector W is obtained by $Q(r)$ in Eq. 4.2,

$$w_j = Q(\frac{X_j}{T}) - Q(\frac{X_{j-1}}{T}) \tag{4.2}$$

where $X_j = \sum_{k=1}^{j} \mu_k$ and $T = X_n = \sum_{k=1}^{n} \mu_k$. That is, X_j is the sum of the importance of the jth most satisfied arguments, and T is the sum of all importance.

So we have the satisfaction of concept S_c,

$$S_c = \sum_{k=1}^{n} w_k \cdot s'_k \tag{4.3}$$

4.4.4 Aggregation of satisfactions of datatype attributes and concepts

As mentioned, a criterion can be a datatype attribute, object attribute or subconcept. The satisfaction of the object attribute, or subconcept is the satisfactions of concepts. The calculation of the satisfaction of a criterion based on terms (datatype attribute) and the calculation of the satisfaction of a criterion based on concepts (object attributes or sub-concepts) is different. The calculation of the satisfaction of a criterion based on terms is quite simple, as shown in Eq. 4.1, while the calculation of the satisfaction of a criterion based on concepts includes more complicated aggregation. When a concept contains criteria based on both terms and concepts, we aggregate separately the criteria based on terms and the criteria based on concepts, then aggregate these two aggregations.

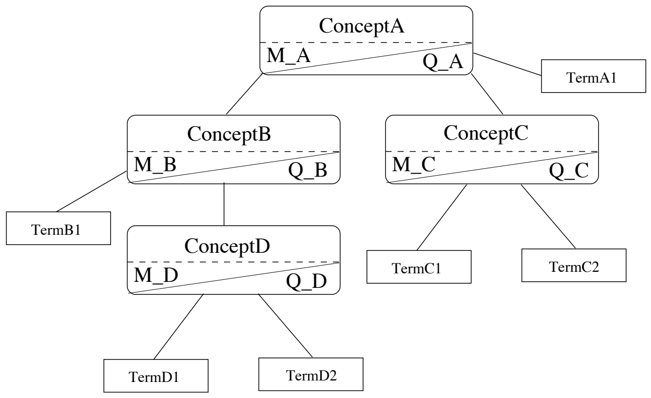


Figure 4.11: An HofC-based query

4.4.5 Satisfaction of HofC

Activation of the HofC is propagated upward as is the process to calculate satisfaction of the HofC. That is, calculation of the satisfaction of the HofC starts from the concepts in the lowest level and ends with the concepts in the highest level. The presence of terms (for example, $TermD1, TermD2, TermC1$, and $TermC2$) (Fig. 4.11) in a document contributes to the satisfaction of the corresponding concepts (*ConceptD* and *ConceptC*, respectively) (Fig. 4.11). The aggregation of the satisfactions of those concepts and other terms contributes to the satisfaction of the higher-level concepts (for example, $TermB1$ and *ConceptD* lead to activation of *ConceptB*) (Fig. 4.11). The process is repeated till the satisfaction of the concept on the top level is calculated.

The structure of HofC determines which terms and concepts contribute to the satisfactions of which concepts – satisfaction of a single concept is calculated based on the satisfactions of all terms and concepts that are attached (from the bottom) to it. The aggregation of satisfactions is performed at each concept of HofC using OWA (Eq. 4.3).

In the next chapter, examples are presented to explain how Web documents are ranked with HofC-based queries in HOTIR.

4.5 Summary

Fig. 4.12 presents an overview of HOTIR. Query coordinator and knowledge base coordinator are two modules in the implementation level that are in charge of communications between modules inside and outside the respective agent.

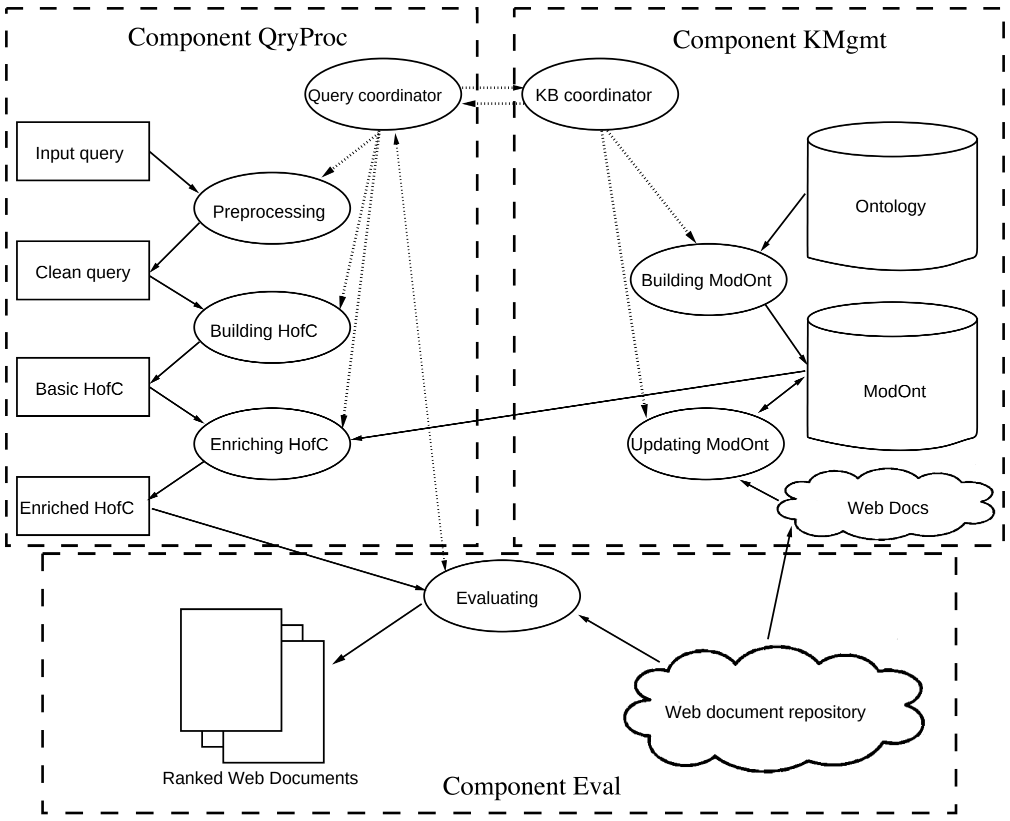


Figure 4.12: HOTIR overview

Component KMgmt builds and manages the ModOnt, the knowledge base in HOTIR, which is built from an ontology. The importance of terms or concepts in the ModOnt are obtained by the AATI scheme, as shown in Fig. 4.12. The *Web Docs* in KMgmt represent a sample of Web documents in the *Web document repository* in component Eval. They are "unknown" to the AATI before they are annotated. That is, there is no need for human experts to work on the documents to generate such knowledge as contained concepts, categories they belong to and so on. HOTIR can automatically generate the collection of *Web Docs* by picking up some percentage, say 10%, of the ranked Web documents to update the knowledge base. Therefore, the process of updating the ModOnt module runs by itself and does not stop. This is an outstanding advantage of HOTIR because no extra human labor is needed to build sample Web documents and the knowledge in the ModOnt can be kept up to date.

Component QryProc translates basic queries into HofC-based queries. QryProc enriches HofC-based queries with knowledge stored in the ModOnt. HofC-based queries provide lists of concepts to be identified and their definitions. The definitions of concepts, also known as the knowledge of concepts, include subconcepts, terms, their importance and their relations. As the output of QryProc, expanded HofC-based queries can be utilized for evaluation.

Component Eval ranks documents. Because the HofC stores hierarchies, the ranking process is not simple. Web documents are semantically annotated first. The annotated terms are pieces of information that satisfy concepts from the HofC. OWA is integrated with the HofC to evaluate the satisfactions of queries by texts in documents. This process is mapped as the aggregation of the activations of the concepts from the HofC.

HOTIR successfully implements concept-based information retrieval. In HOTIR, a term, as a piece of atom information, is not a binary string, but is related to a concept that is a meaningful entity defined in the ModOnt or in a HofC. That is, it has semantics. The search accomplished by HOTIR is based on its semantics instead of its presence. The main techniques in HOTIR, such as an ontology-based knowledge base (ModOnt), an AATI scheme, a HofC query format, and OWA operators for aggregation, are applied for this purpose.

- An ontology provides a specification of a conceptualization. That is, it provides lists of concepts, terms and their relations in a domain.
- An AATI scheme assigns importance to terms and in turn to concepts in the ModOnt by "blindly" reading Web documents.
- Queries in HOTIR are organized into concepts in hierarchical structures (HofC).

- OWA operators calculate the rankings of Web documents. OWA operators aggregate the satisfactions of concepts in a HofC to evaluate the satisfaction of the whole HofC. The HofC accepts different linguistic quantifiers that make HOTIR flexible in defining concepts and representing different user interests.

Chapter 5

HOTIR: Case study

In this chapter cases are presented to illustrate how documents are ranked with HofC-based queries in HOTIR. Two documents Doc.1 and Doc.2, two queries Query.A (Fig. 5.1) and Query.B (Fig. 5.2), and two linguistic quantifiers *most* ($Q(r) = r^2$) and *some* ($Q(r) = r$), are utilized (section 2.1.3). The frequencies of terms in Doc.1 and Doc.2 are listed in Table 5.1. Roughly, Doc.1 contains more information about *Concept Chelsea* and Doc.2 contains more information about *Player Drogba* and *Player Terry*.

5.1 Building and expanding HofC-based query

Both query A and Query B are created from the keywords shown in Fig. 5.3 and Fig. 5.4. Query A is built from the keyword "Chelsea". The keyword "Chelsea" itself is a piece of meaningless text for the machine. However, as soon as HOTIR accepts this binary string, KMgmt tries to figure out the concept embedded in this keyword. It finds the concept *Club Chelsea* in the

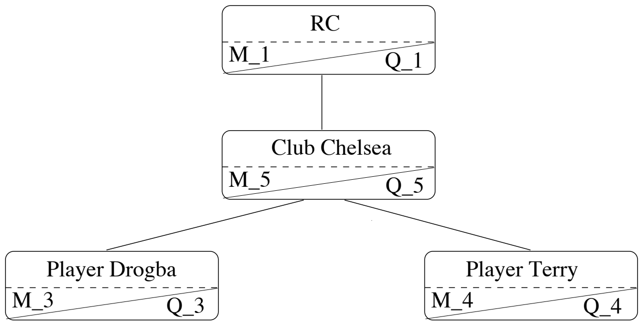


Figure 5.1: Query A.

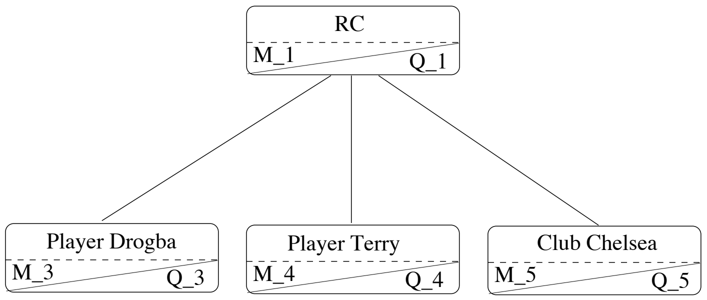


Figure 5.2: Query B.

Table 5.1: Term frequencies in documents

	Doc.1	**Doc.2**
Concept Chelsea		
Name:Chelsea	6	0
Location:London	2	1
Stadium:Stamford	2	1
Concept Player Drogba		
First Name:Didier	0	2
Last Name:Drogba	1	2
Position:Forward	0	1
Concept Player Terry		
First Name:John	0	2
Last Name:Terry	1	3
Position:Midfielder	0	0

ModOnt because the term “Chelsea” is the value of its attribute *Name*. Then definitions of the concept in the knowledge base are added into the query including the values of its datatype attributes such as “London” (attribute *Location*) and “Stamford”(attribute *Stadium*), and the values of the object attribute *HasPlayer* associated with the two concepts: concept *Player Drogba* and *Player Terry*. The connection between concept *Club Chelsea* and concept *Player Drogba* and the connection between concept *Club Chelsea* and concept *Player Terry* are all $TypeIII$ connections (see section 4.3.1 for more details). For simplicity, in this chapter we treat $TypeI$, $TypeII$ and $TypeIII$ connections equally in calculation.

Similarly, Query B is built from keywords “Drogba,” “Terry,” and “ Chelsea,” based on which three concepts in the ModOnt that are retrieved to build a HofC-based query. Since there is no indication about their relations, compo-

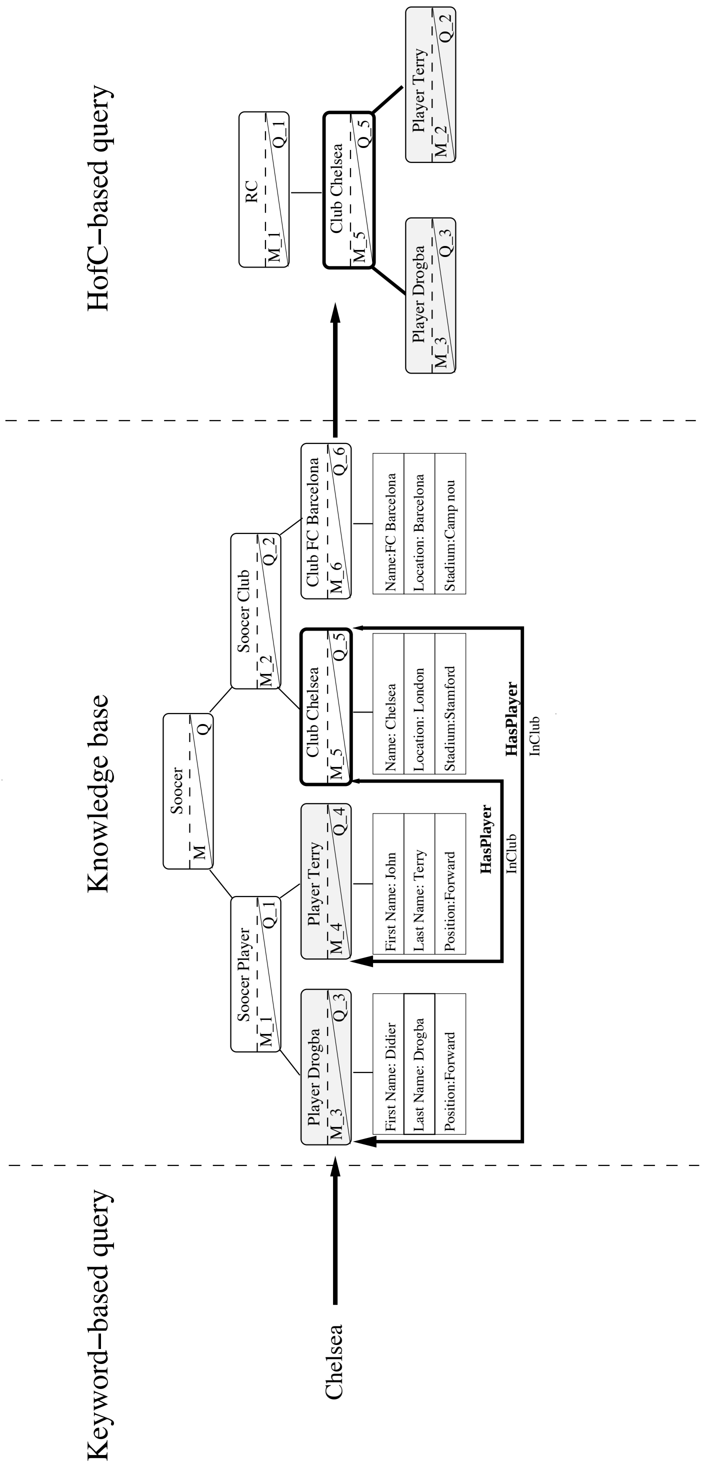


Figure 5.3: Constructing Query A

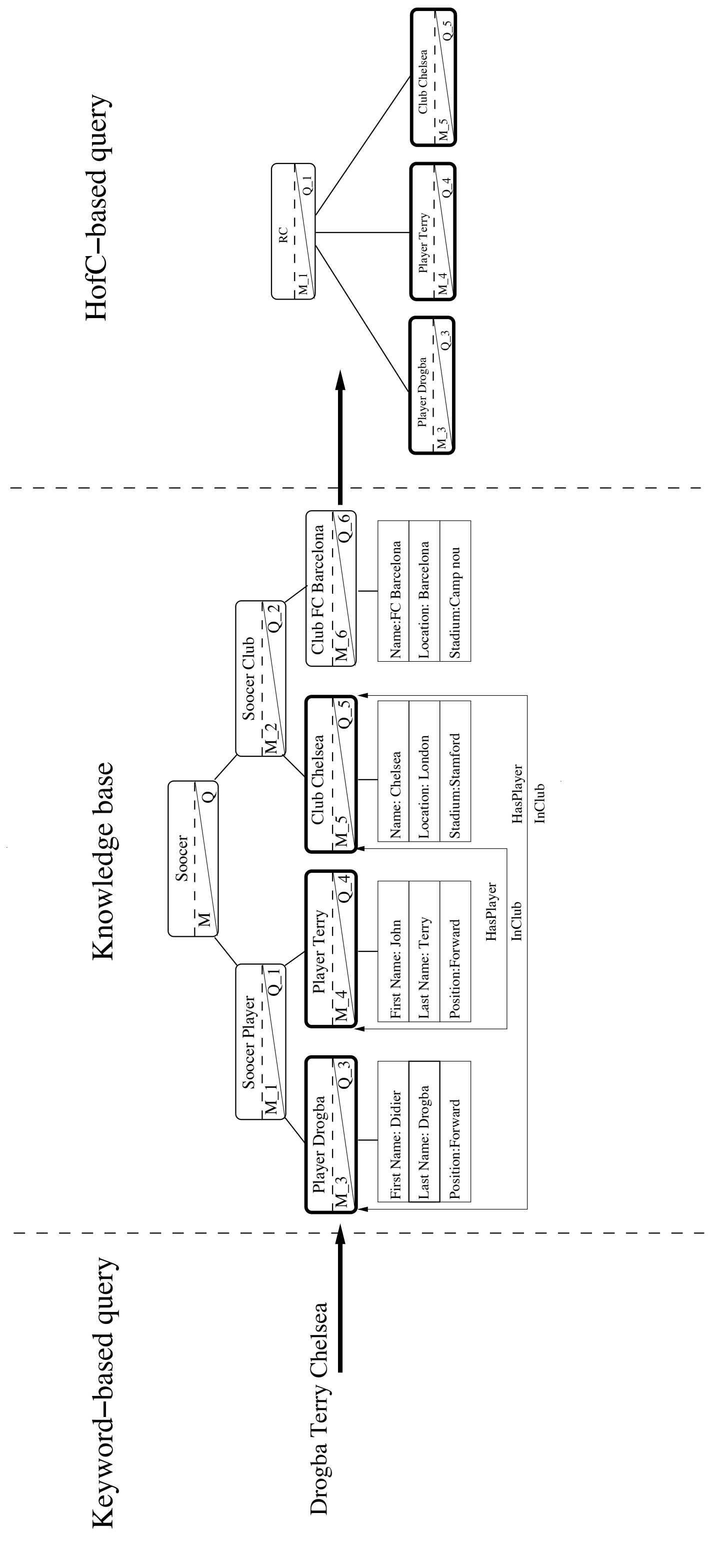


Figure 5.4: Constructing Query B

nent QryProc places the three concepts in a flat (equal level) structure. Query B is created as shown in Fig. 5.4. The connections in this query are $TypeI$.

5.2 Importance of terms in KMgmt

Let us assume the importance (TW) of terms in the ModOnt assigned by the AATI are displayed in the table below:

Concept Player Drogba			
Term	Didier	Drogba	Forward
Importance	0.06	0.11	0.02
Concept Player Terry			
Term	John	Terry	Midfielder
Importance	0.03	0.07	0.06
Concept Club Chelsea			
Term	Chelsea	London	Stamford
Importance	0.15	0.02	0.08

5.3 Evaluation by component Eval

In this section, there are 4 cases using the linguistic quantifiers *most* and *some*: case queryA(most), case queryA(some), case queryB(most), and case queryB(some). Cases queryA(most) and case queryA(some) utilize Query A; cases queryB(most) and case queryB(some) utilize Query B. Table 5.2 shows the design.

Table 5.2: Case study design

Cases	*Query*	*Linguistic Quantifier*
Case queryA(most)	Query A	most
Case queryA(some)	Query A	some
Case queryB(most)	Query B	most
Case queryB(some)	Query B	some

5.3.1 Case queryA(most)

Query A, shown in Fig. 5.3, is generated from the keyword "Chelsea." It is composed of three concepts, which are *Club Chelsea*, *Player Drogba*, and *Player Terry*, as shown in Fig. 5.1. The linguistic quantifier in this case is *most*.

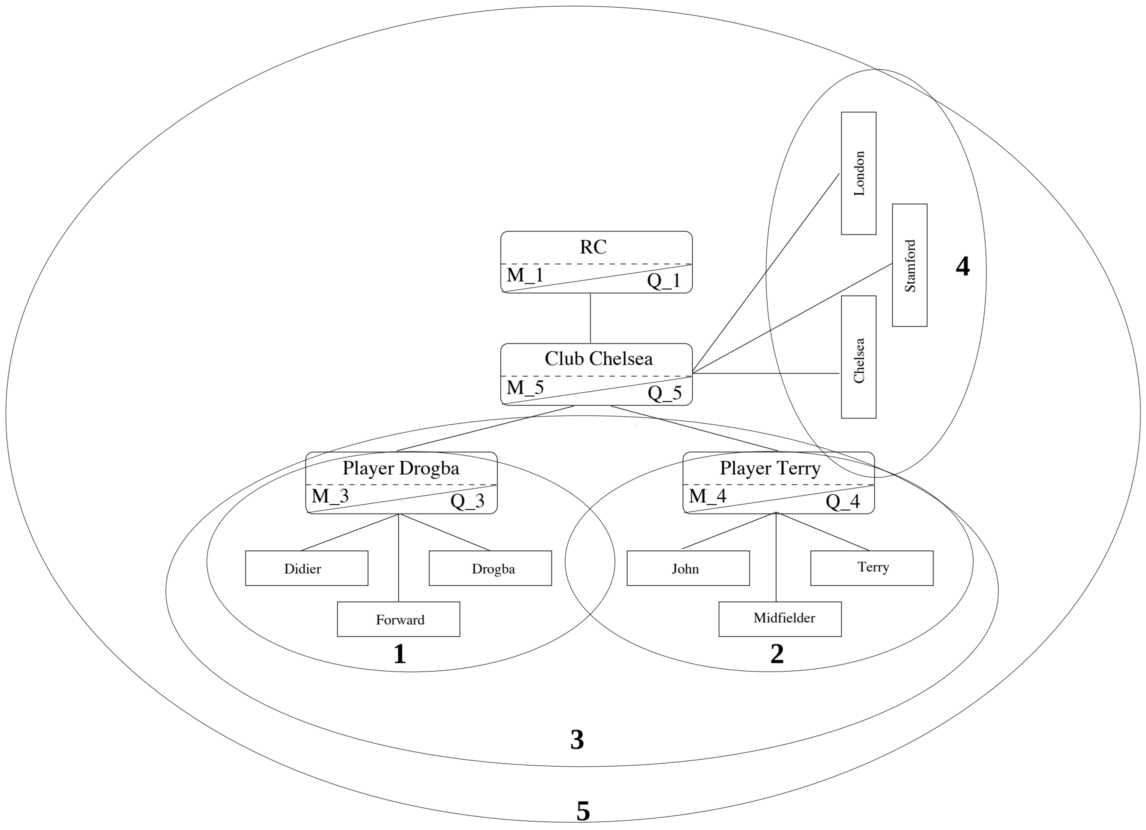


Figure 5.5: Query A in case with queryA(some) and case with queryA(some)

Case queryA(most) for Doc.1

Fig. 5.5 displays the aggregation orders in both cases: queryA(most) and queryA(some). The ellipses in the figure denote the aggregation processes; the numbers in the ellipses are the orders of the processes. As shown in Fig. 5.5, satisfactions of the concepts *Player Drogba* and *Player Terry* are calculated firstly. As subconcepts of the concept *Club Chelsea*, their satisfactions are aggregated. Then satisfactions of the terms in concept *Club Chelsea* are calculated and aggregate with the satisfactions of the aggregated subconcepts, which is the satisfaction of the whole HofC.

Player Drogba has no subconcept, so its satisfaction is decided by attached terms. The term satisfaction is calculated based on Eq. (4.1). We calculate the ordered satisfactions of the terms in *Player Drogba*:

Concept Player Drogba	satisfaction	importance
term: Drogba	0.3679	0.11
term: Didier	0	0.06
term: Forward	0	0.02

Because the linguistic quantifier is *most*, which is defined by $Q(r) = r^2$. For the concept *Player Drogba*, the sum of the importance of all attached terms is $0.11 + 0.06 + 0.02 = 0.19$. Based on OWA, the weights of its criteria/terms are:

term Drogba:

$$w_{tDrogba} = Q(\frac{0.11}{0.19}) - Q(\frac{0}{0.19}) = 0.3352$$

term Didier:

$$w_{tDidier} = Q(\frac{0.17}{0.19}) - Q(\frac{0.11}{0.19}) = 0.4654$$

term Forward:

$$w_{tForward} = Q(\frac{0.19}{0.19}) - Q(\frac{0.17}{0.19}) = 0.1994$$

To Doc.1, concept *Player Drogba* has its satisfaction as:

$$\begin{aligned} s_{cDrogba} &= 0.3352 \times 0.3679 + 0.4654 \times 0 + 0.1994 \times 0 \\ &= 0.1233 \end{aligned} \tag{5.1}$$

Then we calculate the ordered satisfactions of the terms in concept *Player Terry*:

Concept Player Terry	satisfaction	importance
term: Terry	0.3679	0.07
term: John	0	0.03
term: Midfielder	0	0.06

The sum of the importance of all attached terms in concept *Player Terry* is $0.07 + 0.03 + 0.06 = 0.16$. Based on OWA, the weights of its criteria/terms are calculated:

term Terry:

$$w_{tTerry} = Q(\frac{0.07}{0.16}) - Q(\frac{0}{0.16}) = 0.1914$$

term John:

$$w_{tJohn} = Q(\frac{0.10}{0.16}) - Q(\frac{0.07}{0.16}) = 0.1992$$

term Forward:

$$w_{tForward} = Q(\frac{0.16}{0.16}) - Q(\frac{0.10}{0.16}) = 0.6094$$

To Doc.1, the concept *Player Terry* has its satisfaction as:

$$\begin{aligned} s_{cTerry} &= 0.1914 \times 0.3679 + 0.1992 \times 0 + 0.6094 \times 0 \\ &= 0.0704 \end{aligned} \tag{5.2}$$

Defined by the structure of a HofC-based query, satisfaction of the concept *Club Chelsea* is obtained by aggregating the satisfactions of its subconcepts (concept *Player Drogba* and concept *Player Terry*) and terms("Chelsea," "London," and "Stamford"). Because the methods to calculate the satisfaction of subconcepts and terms are different, we calculate the satisfactions separately first and then combine them. The details are as follows.

The ordered satisfactions of sub-concepts in concept *Club Chelsea* are:

Concept Player Chelsea	satisfaction	importance	weights
concept: Drogba	0.1233	0.19	0.2947
concept: Terry	0.0704	0.16	0.7053

Satisfaction of the combination of sub-concepts in concept *Club Chelsea* is:

$$\begin{aligned} s'_{cChelsea} &= 0.1233 \times 0.2947 + 0.0704 \times 0.7053 \\ &= 0.0860 \end{aligned} \tag{5.3}$$

The ordered satisfactions of the terms in *Club Chelsea* are:

Concept Player Chelsea	satisfaction	importance	weights
term: Chelsea	0.8465	0.15	0.3600
term: London	0.6065	0.02	0.1024
term: Stamford	0.6065	0.08	0.5376

Satisfaction of the combination of the terms in *Club Chelsea* is:

$$\begin{aligned} s"_{cChelsea} &= 0.8465 \times 0.3600 + 0.6065 \times 0.1024 + 0.6065 \times 0.5376 \\ &= 0.6929 \end{aligned} \tag{5.4}$$

Satisfaction of *Club Chelsea* is calculated by the combination of subconcepts $s'_{cChelsea}$ and terms $s"_{cChelsea}$ as:

Concept Club Chelsea	satisfaction	importance	weights
comb. of terms	0.6929	0.25	0.1736
comb. of sub-concepts	0.0860	0.35	0.8264

Finally, when applied to Doc.1, satisfaction of the HofC-based query is the same as satisfaction of concept *Club Chelsea*:

$$\begin{aligned} s_{cChelsea} &= 0.6929 \times 0.1736 + 0.0860 \times 0.8264 \\ &= 0.1914 \end{aligned} \tag{5.5}$$

Case queryA(some) for Doc.2

As presented in Table 5.1, there are more information related to *Player Drogba* and *Player Terry* in Doc.2. Similar to the calculations in the previous case, we calculate the ordered satisfactions of the terms for concept: *Player Drogba*:

Concept Player Drogba	satisfaction	importance
term: Drogba	0.6065	0.11
term: Didier	0.6065	0.06
term: Forward	0.3679	0.02

Because the linguistic quantifier is *most*, which is defined by $Q(r) = r^2$. For the concept *Player Drogba*, the sum of the importance of all attached terms is $0.11 + 0.06 + 0.02 = 0.19$. Based on OWA, the weights of its criteria/terms are:

term Drogba:

$$w_{tDrogba} = Q(\frac{0.11}{0.19}) - Q(\frac{0}{0.19}) = 0.3352$$

term Didier:

$$w_{tDidier} = Q(\frac{0.17}{0.19}) - Q(\frac{0.11}{0.19}) = 0.4654$$

term Forward:

$$w_{tForward} = Q(\frac{0.19}{0.19}) - Q(\frac{0.17}{0.19}) = 0.1994$$

To Doc.2, concept *Player Drogba* has its satisfaction as:

$$\begin{aligned} s_{cDrogba} &= 0.3352 \times 0.6065 + 0.4654 \times 0.6065 + 0.1994 \times 0.3679 \\ &= 0.5589 \end{aligned} \tag{5.6}$$

Then we calculate the ordered satisfactions of the terms in *Player Terry*:

Concept Player Terry	satisfaction	importance
term: Terry	0.7165	0.07
term: John	0.6065	0.03
term: Midfielder	0	0.06

The sum of the importance of all attached terms in concept *Player Terry* is $0.07 + 0.03 + 0.06 = 0.16$. Based on OWA, the weights of its criteria/terms are:

term Terry:

$$w_{tTerry} = Q(\frac{0.07}{0.16}) - Q(\frac{0}{0.16}) = 0.1914$$

term John:

$$w_{tJohn} = Q(\frac{0.10}{0.16}) - Q(\frac{0.07}{0.16}) = 0.1992$$

term Forward:

$$w_{tForward} = Q(\frac{0.16}{0.16}) - Q(\frac{0.10}{0.16}) = 0.6094$$

To Doc.2, *concept Terry* has its satisfaction as:

$$\begin{aligned} s_{cTerry} &= 0.1914 \times 0.7165 + 0.1992 \times 0.6065 + 0.6094 \times 0 \\ &= 0.2580 \end{aligned} \tag{5.7}$$

Defined by the structure of HofC-based query, satisfaction of the concept *Club Chelsea* is obtained by aggregating the satisfactions of its subconcepts (*concept Drogba* and *concept Terry*) and terms(“Chelsea”,“London” and “Stamford”).

The ordered satisfactions of subconcepts in *Club Chelsea* are:

Concept Player Chelsea	satisfaction	importance	weights
concept: Drogba	0.5589	0.19	0.2947
concept: Terry	0.2580	0.16	0.7053

Satisfaction of the combination of subconcepts in *Club Chelsea* is:

$$\begin{aligned} s'_{cChelsea} &= 0.5589 \times 0.2947 + 0.2580 \times 0.7053 \\ &= 0.3467 \end{aligned} \tag{5.8}$$

The ordered satisfactions of the terms in *Club Chelsea* are:

Concept Player Chelsea	satisfaction	importance	weights
term: Chelsea	0.3679	0.15	0.3600
term: Stamford	0.3679	0.08	0.4864
term: London	0	0.02	0.1536

Satisfaction of the combination of the terms in *Club Chelsea* is:

$$\begin{aligned} s"_{cChelsea} &= 0.3679 \times 0.3600 + 0.3679 \times 0.4864 + 0 \times 0.1536 \\ &= 0.3114 \end{aligned} \tag{5.9}$$

Satisfaction of the concept *Club Chelsea* is calculated by the combination of the satisfactions of its subconcepts $s'_{cChelsea}$ and terms $s"_{cChelsea}$ as:

Concept Player Chelsea	satisfaction	importance	weights
comb. of subconcepts	0.3467	0.35	0.3403
comb. of terms	0.3114	0.25	0.6597

Finally, when applied to Doc.2, the satisfaction of the query is the same with the satisfaction of the *Club Chelsea*:

$$\begin{aligned} s_{cChelsea} &= 0.3467 \times 0.3403 + 0.3114 \times 0.6579 \\ &= 0.3234 \end{aligned} \tag{5.10}$$

5.3.2 Case queryA(some)

We keep using Query A in case queryA(some). However, another linguistic quantifier, *some*, is utilized.

Case queryA(some) for Doc.1

First, we calculate the ordered satisfaction of the terms in *Player Drogba*:

Concept Player Drogba	satisfaction	importance
term: Drogba	0.3679	0.11
term: Didier	0	0.06
term: Forward	0	0.02

Because the linguistic quantifier is *some*, which is defined by $Q(r) = r$. For *concept Drogba*, the sum of the importance of all attached terms is $0.11+0.06+0.02 = 0.19$. Based on OWA, the weights of its criteria are:

term Drogba:

$$w_{tDrogba} = Q(\frac{0.11}{0.19}) - Q(\frac{0}{0.19}) = 0.5789$$

term Didier:

$$w_{tDidier} = Q(\frac{0.17}{0.19}) - Q(\frac{0.11}{0.19}) = 0.3158$$

term Forward:

$$w_{tForward} = Q(\frac{0.19}{0.19}) - Q(\frac{0.17}{0.19}) = 0.1053$$

so for Doc.1, *concept Drogba* has its satisfaction as:

$$\begin{aligned} s_{cDrogba} &= 0.5789 \times 0.3679 + 0.3158 \times 0 + 0.1994 \times 0 \\ &= 0.2130 \end{aligned} \tag{5.11}$$

Then we calculate the ordered satisfactions of the terms in *Player Terry* as:

Concept Player Terry	satisfaction	importance
term: Terry	0.3679	0.07
term: John	0	0.03
term: Midfielder	0	0.06

The sum of the importance of all attached terms in *concept Terry* is $0.07 + 0.03 + 0.06 = 0.16$. Based on OWA, the weights of its criteria/terms are:

term Terry:

$$w_{tTerry} = Q(\frac{0.07}{0.16}) - Q(\frac{0}{0.16}) = 0.4375$$

term John:

$$w_{tJohn} = Q(\frac{0.10}{0.16}) - Q(\frac{0.07}{0.16}) = 0.1875$$

term Forward:

$$w_{tForward} = Q(\frac{0.16}{0.16}) - Q(\frac{0.10}{0.16}) = 0.3750$$

so for Doc.1, *concept Terry* has its satisfaction:

$$\begin{aligned} s_{cTerry} &= 0.4375 \times 0.3679 + 0.1875 \times 0 + 0.3750 \times 0 \\ &= 0.1610 \end{aligned} \tag{5.12}$$

Defined by the structure of HofC-based query, the satisfaction of *concept Chelsea* is obtained by aggregating those of its sub-concepts (*concept Drogba* and *concept Terry*) and terms("Chelsea","London" and "Stamford").

The ordered satisfactions of sub-concepts in *Club Chelsea* are:

Concept Player Chelsea	satisfaction	importance	weights
concept: Drogba	0.2130	0.19	0.5429
concept: Terry	0.1610	0.16	0.4571

The satisfaction of the combination of sub-concepts in *Club Chelsea* is:

$$\begin{aligned} s'_{cChelsea} &= 0.5429 \times 0.2130 + 0.4571 \times 0.1610 \\ &= 0.1892 \end{aligned} \tag{5.13}$$

The ordered satisfactions of the terms in *Club Chelsea* are:

Concept Player Chelsea	satisfaction	importance	weights
term: Chelsea	0.8465	0.15	0.6000
term: London	0.6065	0.02	0.0800
term: Stamford	0.6065	0.08	0.3200

The satisfaction of the combination of the terms in *Club Chelsea* is:

$$\begin{aligned} s"_{cChelsea} &= 0.8465 \times 0.6000 + 0.6065 \times 0.0800 + 0.6065 \times 0.3200 \\ &= 0.7505 \end{aligned} \tag{5.14}$$

The satisfaction of *Club Chelsea* is calculated by the combination of sub-concepts $s'_{cChelsea}$ and terms $s"_{cChelsea}$ as:

Concept Player Chelsea	satisfaction	importance	weights
comb. of terms	0.7505	0.25	0.4167
comb. of sub-concepts	0.1892	0.35	0.5833

Finally, when applied to Doc.1, the satisfaction of the HofC-based query is the same with the satisfaction of *Club Chelsea*:

$$\begin{aligned} s_{cChelsea} &= 0.4167 \times 0.7505 + 0.5833 \times 0.1892 \\ &= 0.4231 \end{aligned} \tag{5.15}$$

Case queryA(some) for Doc.2

With linguistic quantifier *some*, we calculate satisfaction based on Doc.2.

The ordered satisfactions of the terms in *Player Drogba* are:

Concept Player Drogba	satisfaction	importance
term: Drogba	0.6065	0.11
term: Didier	0.6065	0.06
term: Forward	0.3679	0.02

Because the linguistic quantifier is *most*, which is defined by $Q(r) = r^2$. For *concept Drogba*, the sum of the importance of all attached terms is $0.11+0.06+0.02 = 0.19$. Based on OWA, the weights of its criteria/terms are:

term Drogba:

$$w_{tDrogba} = Q(\frac{0.11}{0.19}) - Q(\frac{0}{0.19}) = 0.5790$$

term Didier:

$$w_{tDidier} = Q(\frac{0.17}{0.19}) - Q(\frac{0.11}{0.19}) = 0.3158$$

term Forward:

$$w_{tForward} = Q(\frac{0.19}{0.19}) - Q(\frac{0.17}{0.19}) = 0.1053$$

so for Doc.1, *concept Drogba* has its satisfaction as:

$$\begin{aligned} s_{cDrogba} &= 0.5790 \times 0.6065 + 0.3158 \times 0.6065 + 0.1053 \times 0.3679 \\ &= 0.5814 \end{aligned} \tag{5.16}$$

Then we calculate the ordered satisfactions of the terms in *Player Terry* as:

Concept Player Terry	satisfaction	importance
term: Terry	0.7165	0.07
term: John	0.6065	0.03
term: Midfielder	0	0.06

The sum of the importance of all attached terms in *concept Terry* is $0.07 + 0.03 + 0.06 = 0.16$. Based on OWA, the weights of its criteria/terms are:

term Terry:

$$w_{tTerry} = Q(\frac{0.07}{0.16}) - Q(\frac{0}{0.16}) = 0.4375$$

term John:

$$w_{tJohn} = Q(\frac{0.10}{0.16}) - Q(\frac{0.07}{0.16}) = 0.1875$$

term Forward:

$$w_{tForward} = Q(\frac{0.16}{0.16}) - Q(\frac{0.10}{0.16}) = 0.3750$$

To Doc.1, *concept Terry* has its satisfaction as:

$$\begin{aligned} s_{cTerry} &= 0.4375 \times 0.7165 + 0.1875 \times 0.6065 + 0.3750 \times 0 \\ &= 0.4272 \end{aligned} \tag{5.17}$$

Defined by the structure of HofC-based query, the satisfaction of *concept Chelsea* is obtained by aggregating the satisfactions of its sub-concepts (*concept Drogba* and *concept Terry*) and terms("Chelsea","London" and "Stamford").

The ordered satisfactions of sub-concepts in *Club Chelsea* are:

Concept Player Chelsea	satisfaction	importance	weights
concept: Drogba	0.5814	0.19	0.5429
concept: Terry	0.4272	0.16	0.4571

The satisfaction of the combination of sub-concepts in *Club Chelsea* is:

$$\begin{aligned} s'_{cChelsea} &= 0.5429 \times 0.5814 + 0.4571 \times 0.4272 \\ &= 0.5109 \end{aligned} \tag{5.18}$$

The ordered satisfactions of the terms in *Club Chelsea* are:

Concept Player Chelsea	satisfaction	importance	weights
term: Chelsea	0.3679	0.15	0.6000
term: Stamford	0.3679	0.08	0.3200
term: London	0	0.02	0.0800

The satisfaction of the combination of the terms in *Club Chelsea* is:

$$\begin{aligned} s"_{cChelsea} &= 0.6000 \times 0.3679 + 0.3200 \times 0.3679 + 0.0800 \times 0 \\ &= 0.3385 \end{aligned} \tag{5.19}$$

The satisfaction of *Club Chelsea* is calculated by the combination of subconcept $s'_{cChelsea}$ and term $s"_{cChelsea}$ as:

Concept Player Chelsea	satisfaction	importance	weights
comb. of sub-concepts	0.5109	0.35	0.5833
comb. of terms	0.3385	0.25	0.4167

Finally, when applied to Doc.2, satisfaction of queryA(some) is the same with satisfaction of *Club Chelsea*:

$$\begin{aligned} s_{cChelsea} &= 0.5833 \times 0.5109 + 0.4167 \times 0.3385 \\ &= 0.4390 \end{aligned} \tag{5.20}$$

5.3.3 Case queryB(most)

If a query "Chelsea Drogba Terry" is provided, a new HofC-based query can be created shown in Fig. 5.2.

Fig. 5.6 displays the aggregation order of Query B in cases queryB(most) and queryB(some). The ellipses in the figure denote the aggregation processes; the numbers in the ellipses are the orders of the processes. As shown in Fig 5.6, the three single concepts (concepts *Player Drogba*, *Player Terry*, and *Club Chelsea*) are processed first. The satisfaction of the three separated concepts are aggregated as the satisfaction of the whole HofC.

In this experiment, we calculate satisfactions on the same two documents (Doc.1 and Doc.2) with linguistic quantifier *most*.

Case queryB(most) for Doc.1

As shown in Fig. 5.2, satisfaction of the RC is calculated based on its three subconcepts: *Club Chelsea*, *Player Drogba*, and *Player Terry*. Therefore, we calculate the ordered satisfaction of the terms in *Player Drogba* as:

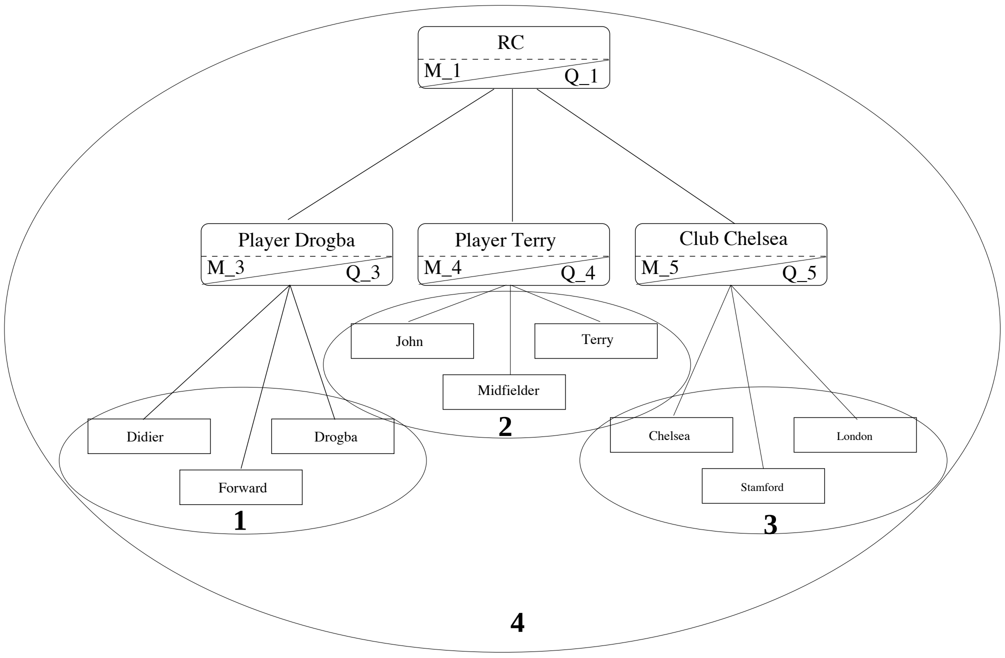


Figure 5.6: Query B in case with queryB(most) and case with queryB(some)

Concept Player Drogba	satisfaction	importance
term: Drogba	0.3679	0.11
term: Didier	0	0.06
term: Forward	0	0.02

Because the linguistic quantifier is *most*, which is defined by $Q(r) = r^2$. For *concept Drogba*, the sum of the importance of all attached terms is $0.11+0.06+0.02 = 0.19$. Based on OWA, the weights of its criteria/terms are:

term Drogba:

$$w_{tDrogba} = Q(\frac{0.11}{0.19}) - Q(\frac{0}{0.19}) = 0.3352$$

term Didier:

$$w_{tDidier} = Q(\frac{0.17}{0.19}) - Q(\frac{0.11}{0.19}) = 0.4654$$

term Forward:

$$w_{tForward} = Q(\frac{0.19}{0.19}) - Q(\frac{0.17}{0.19}) = 0.1994$$

For Doc.1, *Player Drogba* has its satisfaction as:

$$
\begin{aligned}
s_{cDrogba} &= 0.3352 \times 0.3679 + 0.4654 \times 0 + 0.1994 \times 0 \\
&= 0.1233
\end{aligned}
\tag{5.21}
$$

Then we calculate the ordered satisfactions of the terms in *Player Terry* as:

Concept Player Terry	satisfaction	importance
term: Terry	0.3679	0.07
term: John	0	0.03
term: Midfielder	0	0.06

The sum of the importance of all attached terms in *Player Terry* is $0.07 + 0.03 + 0.06 = 0.16$. Based on OWA, the weights of its criteria/terms are:

term Terry:

$$w_{tTerry} = Q(\frac{0.07}{0.16}) - Q(\frac{0}{0.16}) = 0.1914$$

term John:

$$w_{tJohn} = Q(\frac{0.10}{0.16}) - Q(\frac{0.07}{0.16}) = 0.1992$$

term Forward:

$$w_{tForward} = Q(\frac{0.16}{0.16}) - Q(\frac{0.10}{0.16}) = 0.6094$$

For Doc.1, *concept Terry* has its satisfaction as:

$$
\begin{aligned}
s_{cTerry} &= 0.1914 \times 0.3679 + 0.1992 \times 0 + 0.6094 \times 0 \\
&= 0.0704
\end{aligned}
\tag{5.22}
$$

Then, the ordered satisfactions of the terms in *Club Chelsea* are:

Concept Player Chelsea	satisfaction	importance	weights
term: Chelsea	0.8465	0.15	0.3600
term: London	0.6065	0.02	0.1024
term: Stamford	0.6065	0.08	0.5376

Satisfaction of the combination of the terms in *Club Chelsea* are:

$$
\begin{aligned}
s"_{cChelsea} &= 0.8465 \times 0.3600 + 0.6065 \times 0.1024 + 0.6065 \times 0.5376 \\
&= 0.6929
\end{aligned}
\tag{5.23}
$$

RC	satisfaction	importance	weights
concept:Chelsea	0.6929	0.25	0.1736
concept:Drogba	0.1233	0.19	0.3642
concept:Terry	0.0704	0.16	0.4622

Therefore, the satisfaction of the RC, which equals to the satisfaction of queryB(most), is:

$$\begin{aligned} s_{cRC} &= 0.1736 \times 0.6929 + 0.3642 \times 0.1233 + 0.4622 \times 0.0704 \\ &= 0.1977 \end{aligned} \tag{5.24}$$

Case queryB(most) for Doc.2

Similarly, when the new query is applied to Doc.2, the ordered satisfaction of the terms in *Player Drogba* are:

Concept Player Drogba	satisfaction	importance
term: Drogba	0.6065	0.11
term: Didier	0.6065	0.06
term: Forward	0.3679	0.02

Because the linguistic quantifier is *most*, which is defined by $Q(r) = r^2$. For the concept *Player Drogba*, the sum of the importance of all attached terms is $0.11 + 0.06 + 0.02 = 0.19$. Based on OWA, the weights of its criteria/terms are:

term Drogba:

$$w_{tDrogba} = Q(\frac{0.11}{0.19}) - Q(\frac{0}{0.19}) = 0.3352$$

term Didier:

$$w_{tDidier} = Q(\frac{0.17}{0.19}) - Q(\frac{0.11}{0.19}) = 0.4654$$

term Forward:

$$w_{tForward} = Q(\frac{0.19}{0.19}) - Q(\frac{0.17}{0.19}) = 0.1994$$

To Doc.2, *Player Drogba* has its satisfaction as:

$$\begin{aligned} s_{cDrogba} &= 0.3352 \times 0.6065 + 0.4654 \times 0.6065 + 0.1994 \times 0.3679 \\ &= 0.5589 \end{aligned} \tag{5.25}$$

We calculate the ordered satisfactions of the terms in *Player Terry* as:

Concept Player Terry	satisfaction	importance
term: Terry	0.7165	0.07
term: John	0.6065	0.03
term: Midfielder	0	0.06

The sum of the importance of all attached terms in *Player Terry* is $0.07 + 0.03 + 0.06 = 0.16$. Based on OWA, the weights of its criteria/terms are:

term Terry:

$$w_{tTerry} = Q(\frac{0.07}{0.16}) - Q(\frac{0}{0.16}) = 0.1914$$

term John:

$$w_{tJohn} = Q(\frac{0.10}{0.16}) - Q(\frac{0.07}{0.16}) = 0.1992$$

term Forward:

$$w_{tForward} = Q(\frac{0.16}{0.16}) - Q(\frac{0.10}{0.16}) = 0.6094$$

so for Doc.2, *Player Terry* has its satisfaction as:

$$\begin{aligned} s_{cTerry} &= 0.1914 \times 0.7165 + 0.1992 \times 0.6065 + 0.6094 \times 0 \\ &= 0.2580 \end{aligned} \tag{5.26}$$

The ordered satisfactions of the terms in *Club Chelsea* are:

Concept Player Chelsea	satisfaction	importance	weights
term: Chelsea	0.3679	0.15	0.3600
term: Stamford	0.3679	0.08	0.4864
term: London	0	0.02	0.1536

Satisfaction of the combination of the terms in *Club Chelsea* is:

$$\begin{aligned} s"_{cChelsea} &= 0.3679 \times 0.3600 + 0.3679 \times 0.4864 + 0 \times 0.1536 \\ &= 0.3114 \end{aligned} \tag{5.27}$$

The ordered satisfactions of the three concepts are:

RC	satisfaction	importance	weights
concept:Drogba	0.5589	0.19	0.1003
concept:Chelsea	0.3114	0.35	0.4375
concept:Terry	0.2580	0.16	0.4622

Therefore, the satisfaction of the RC, that is, the satisfaction of queryB(most), is:

$$\begin{aligned} s_{cRC} &= 0.1003 \times 0.5589 + 0.4375 \times 0.3114 + 0.4622 \times 0.2580 \\ &= 0.3115 \end{aligned} \tag{5.28}$$

5.3.4 Case queryB(some)

In this case, Query B (Fig. 5.2) and linguistic quantifier *some* are utilized.

Case queryB(some) for Doc.1

First, we calculate the ordered satisfaction of the terms in *Player Drogba* as:

Concept Player Drogba	satisfaction	importance
term: Drogba	0.3679	0.11
term: Didier	0	0.06
term: Forward	0	0.02

Because the linguistic quantifier is *some*, which is defined by $Q(r) = r$. For *Player Drogba*, the sum of the importance of all attached terms is $0.11+0.06+0.02 = 0.19$. Based on OWA, the weights of its criteria/terms are:

term Drogba:

$$w_{tDrogba} = Q(\frac{0.11}{0.19}) - Q(\frac{0}{0.19}) = 0.5789$$

term Didier:

$$w_{tDidier} = Q(\frac{0.17}{0.19}) - Q(\frac{0.11}{0.19}) = 0.3158$$

term Forward:

$$w_{tForward} = Q(\frac{0.19}{0.19}) - Q(\frac{0.17}{0.19}) = 0.1053$$

To Doc.1, *concept Drogba* has its satisfaction as:

$$\begin{aligned} s_{cDrogba} &= 0.5789 \times 0.3679 + 0.3158 \times 0 + 0.1994 \times 0 \\ &= 0.2130 \end{aligned} \tag{5.29}$$

We calculate the ordered satisfactions of the terms in *Player Terry*:

Concept Player Terry	satisfaction	importance
term: Terry	0.3679	0.07
term: John	0	0.03
term: Midfielder	0	0.06

The sum of the importance of all attached terms in *Player Terry* is $0.07 + 0.03 + 0.06 = 0.16$. Based on OWA, the weights of its criteria/terms are:

term Terry:

$$w_{tTerry} = Q(\frac{0.07}{0.16}) - Q(\frac{0}{0.16}) = 0.4375$$

term John:

$$w_{tJohn} = Q(\frac{0.10}{0.16}) - Q(\frac{0.07}{0.16}) = 0.1875$$

term Forward:

$$w_{tForward} = Q(\frac{0.16}{0.16}) - Q(\frac{0.10}{0.16}) = 0.3750$$

To Doc.1, *concept Terry* has its satisfaction as:

$$\begin{aligned} s_{cTerry} &= 0.4375 \times 0.3679 + 0.1875 \times 0 + 0.3750 \times 0 \\ &= 0.1610 \end{aligned} \tag{5.30}$$

The ordered satisfactions of the terms in *Club Chelsea* are:

Concept Player Chelsea	satisfaction	importance	weights
term: Chelsea	0.8465	0.15	0.6000
term: London	0.6065	0.02	0.0800
term: Stamford	0.6065	0.08	0.3200

The satisfaction of the combination of the terms in *Club Chelsea* is:

$$\begin{aligned} s"_{cChelsea} &= 0.8465 \times 0.6000 + 0.6065 \times 0.0800 + 0.6065 \times 0.3200 \\ &= 0.7505 \end{aligned} \tag{5.31}$$

The ordered satisfactions of the three concepts are:

RC	satisfaction	importance	weights
concept:Chelsea	0.7505	0.35	0.4167
concept:Drogba	0.2130	0.19	0.3167
concept:Terry	0.1610	0.16	0.2667

Therefore, the satisfaction of the RC, which is the satisfaction of queryB(some), is:

$$\begin{aligned} s_{cRC} &= 0.4167 \times 0.7505 + 0.3167 \times 0.2130 + 0.2667 \times 0.1610 \\ &= 0.4231 \end{aligned} \tag{5.32}$$

Case queryB(some) for Doc.2

Similarly, when Query B is applied to Doc.2, the ordered satisfaction of the terms in *Player Drogba* are:

Concept Player Drogba	satisfaction	importance
term: Drogba	0.6065	0.11
term: Didier	0.6065	0.06
term: Forward	0.3679	0.02

Because the linguistic quantifier is *most*, which is defined by $Q(r) = r^2$. For *Player Drogba*, the sum of the importance of all attached terms is $0.11+0.06+0.02 = 0.19$. Based on OWA, the weights of its criteria/terms are:

term Drogba:

$$w_{tDrogba} = Q(\frac{0.11}{0.19}) - Q(\frac{0}{0.19}) = 0.5790$$

term Didier:

$$w_{tDidier} = Q(\frac{0.17}{0.19}) - Q(\frac{0.11}{0.19}) = 0.3158$$

term Forward:

$$w_{tForward} = Q(\frac{0.19}{0.19}) - Q(\frac{0.17}{0.19}) = 0.1053$$

To Doc.1, *concept Drogba* has its satisfaction as:

$$\begin{aligned} s_{cDrogba} &= 0.5790 \times 0.6065 + 0.3158 \times 0.6065 + 0.1053 \times 0.3679 \\ &= 0.5814 \end{aligned} \tag{5.33}$$

We calculate the ordered satisfactions of the terms for *Player Terry* as:

Concept Player Terry	satisfaction	importance
term: Terry	0.7165	0.07
term: John	0.6065	0.03
term: Midfielder	0	0.06

The sum of the importance of all attached terms in *Player Terry* is $0.07 + 0.03 + 0.06 = 0.16$. Based on OWA, the weights of its criteria/terms are:

term Terry:

$$w_{tTerry} = Q(\frac{0.07}{0.16}) - Q(\frac{0}{0.16}) = 0.4375$$

term John:

$$w_{tJohn} = Q(\frac{0.10}{0.16}) - Q(\frac{0.07}{0.16}) = 0.1875$$

term Forward:

$$w_{tForward} = Q(\frac{0.16}{0.16}) - Q(\frac{0.10}{0.16}) = 0.3750$$

To Doc.2, *concept Terry* has its satisfaction as:

$$\begin{aligned} s_{cTerry} &= 0.4375 \times 0.7165 + 0.1875 \times 0.6065 + 0.3750 \times 0 \\ &= 0.4272 \end{aligned} \tag{5.34}$$

The ordered satisfactions of the terms in *Club Chelsea* are:

Concept Player Chelsea	satisfaction	importance	weights
term: Chelsea	0.3679	0.15	0.6000
term: Stamford	0.3679	0.08	0.3200
term: London	0	0.02	0.0800

Satisfaction of the combination of the terms in *Club Chelsea* is:

$$\begin{aligned} s"_{cChelsea} &= 0.6000 \times 0.3679 + 0.3200 \times 0.3679 + 0.0800 \times 0 \\ &= 0.3385 \end{aligned} \tag{5.35}$$

The ordered satisfactions of the three concepts are:

RC	satisfaction	importance	weights
concept:Drogba	0.5814	0.19	0.3167
concept:Chelsea	0.4272	0.35	0.4167
concept:Terry	0.3385	0.16	0.2667

Therefore, the satisfaction of the RC, which is also the satisfaction of queryB(some), is:

$$\begin{aligned} s_{cRC} &= 0.3167 \times 0.5814 + 0.4167 \times 0.4272 + 0.2667 \times 0.3385 \\ &= 0.4524 \end{aligned} \tag{5.36}$$

5.4 Conclusion

The results of the cases are listed in Table 5.3.

Table 5.3: The results of cases

Case	*Query*	*Linguistic Quantifier*	*Doc.1*	*Doc.2*
Case with queryA(most)	Query A	most	0.1914	0.3234
Case with queryA(some)	Query A	some	0.4231	0.4391
Case with queryB(most)	Query B	most	0.1977	0.3115
Case with queryB(some)	Query B	some	0.4231	0.4391

In both Doc.1 and Doc.2, there are 12 pieces of information (sum of the frequencies of terms) to satisfy three aimed concepts (Table 5.1). In Doc.1, 10 of 12 are directly related to *Club Chelsea*. In Doc.2, 10 of 12 are directly related to *Player Drogba* and *Player Terry*. In all of the results, the scores of Doc.2 are greater than the scores of Doc.1. This is mainly because different pieces of information have different importance (section 5.2). Moreover, the scores of the documents differ with different hierarchical structures or with different linguistic quantifiers.

In cases queryA(some) and queryB(some), where the linguistic quantifier *some* is applied, the satisfaction of Query A is equal to the satisfaction of Query B when the same document is evaluated. That is, with the linguistic quantifier *some*, the hierarchical structures do not affect the calculation of satisfactions. This is because *some* is defined by $Q(r) = r$ (Fig. 4.10a), which means the aggregation weight of a criterion is decided by its importance value directly. That is, with the linguistic quantifier *some*, the order of aggregation is not considered in calculation.

For information aggregated by the linguistic quantifier *most* (defined by $Q(r) = r^2$,Fig. 4.10b) order is crucial. For example, if there are two pieces of information (i_1 and i_2) with importance $imp_1 = 0.2$ and $imp_2 = 0.3$, respectively, and the order of the aggregation is i_1 then i_2, with *most* their weights are: $wgt_1 = (0.2/0.5)^2 = 0.16$ and $wgt_2 = 1 - (0.2/0.5)^2 = 0.84$, respectively. That is, i_2 obtains larger aggregation weight than i_1. If the order of the aggregation is i_2 then i_1, their weights are: $wgt'_2 = (0.3/0.5)^2 = 0.36$ and $wgt'_1 = 1 - (0.3/0.5)^2 = 0.64$, respectively. That is, i_1 obtains larger aggregation weight than i_2. Thus with *most*, the order of aggregation is important in calculation. Moreover, because the aggregation of pieces of information is in an order from most satisfied to less satisfied, the less satisfied piece of information tends to have larger weights.

The difference in satisfactions between Doc.1 and Doc.2 with the linguistic quantifier *some* is much smaller than that with the linguistic quantifier *most*. For Query A, when the linguistic quantifier changes from *some* to *most*, the difference in satisfactions between Doc.1 and Doc.2 changes from $0.4391 - 0.4231 = 0.0160$ to $0.3234 - 0.1914 = 0.1315$. For Query B, when the linguistic quantifier changes from *some* to *most*, the difference in satisfaction between Doc.1 and Doc.2 changes from $0.4391 - 0.4231 = 0.0160$ to $0.3115 - 0.1977 = 0.1138$. This is because satisfaction with *most* is affected by hierarchies while satisfaction with *some* is not.

The case studies are good examples of how HOTIR, especially component Eval, ranks documents. Moreover, the detailed calculations illustrate the effects of the linguistic quantifiers *some* and *most* in HOTIR. In the next chapter, the designed experiments examine the performance of HOTIR with *some* and *most*.

Chapter 6

HOTIR: Evaluation

6.1 Overview

We set up two different series of experiments, *metrics-based* (MetricsExp) and *human-based* (HumanExp), to perform a thorough examination of the HOTIR system. The experiments were designed in a standard information retrieval way, that is, Web documents were retrieved and ranked with HOTIR according to how well they satisfied a query. The two series of experiments shared the same five conventional queries, while the *Metrics-based* experiment had one more special query which covered all the concepts in the knowledge base.

The experiments were performed on real-world Web documents. The targeted domain was *soccer*, i.e., all queries used in the experiments were related to soccer. The experiments are described below:

- MetricsExp: performed on a local Web repository, where Web documents were crawled from the BBC web site[1]. The details are described in section 6.3.
- HumanExp: designed to compare HOTIR with Google[2]. This set of experiments involved rankings by human beings. Section 6.4 describes HumanExp.

[1]http://www.bbc.co.uk, crawled in September 2009.
[2]http://www.google.com.

6.2 Required knowledge base

Queries in HOTIR were expanded with supplementary knowledge in the domain of interest to build a knowledge base. As soccer was the domain of interest, we constructed a soccer ontology. The linguistic quantifier Q and term importance values M were added to the knowledge base by KMgmt (section 4.2) so the aggregation can be conducted during experiments. Q was linguistic quantifier *most* or *some* (section 4.4.2). The value of M was obtained by the proposed AATI scheme (section 4.2). The soccer ontology was modified by adding Q and M to the ontology. The modified ontology (ModOnt) (section 4.2) was the knowledge base in HOTIR.

The ModOnt contained vocabulary in the domain soccer. In it there were concepts such as *Soccer Player*, *Soccer Club*, *Soccer Events*, etc. Each concept was defined by a number of related-concepts or/and terms (literal values of datatype attributes). For example, the concept *Soccer Player* was defined by related-concepts such as *Player Drogba*, *Player Terry*,*Player Messi*, and so on. The concept *Soccer Club* was related to *FC Barcelona*, *Chelsea*, *AC Milan*, and so on. The concept *Player Terry* (representing the famous soccer player John Terry) was defined by terms “Terry” (last name), “John” (first name), etc. and related concepts *Chelsea* (soccer teams he plays for), *England* (nationality), etc.

6.3 MetricsExp setup

6.3.1 Dataset

There were two datasets in MetricsExp. One was used by the AATI to obtain weights for knowledge in ModOnt (see *Web docs* in KMgmt, Fig. 4.12), the other was to be ranked by the HOTIR-based system (see “Web document repository” of Eval, Fig. 4.12). Web documents were crawled from the BBC News website, where documents were labelled by editors of the BBC News. We knew which documents were relevant to soccer (they were stored in the BBC website “football” folder), and which documents were not relevant. Around 10% of the web documents were picked up for updating by the AATI, the rest were put into web document repository for ranking. In total, the first dataset contained 2554 Web documents, where 276 documents were relevant to soccer, and 2278 web documents were non-relevant ones; the second dataset contained 22957 ones, where 2348 documents were relevant to soccer, and 20609 were non-relevant ones to soccer.

Since different queries were applied, it was necessary to identify documents relevant to them. In this thesis, all queries were related to soccer, so the documents relevant to the queries were subsets of soccer-related documents. To identify the document set for a specific query, the query was expanded with knowledge in the knowledge base. Because the expansion may include some general terms, it is necessary to manually delete them. A Java script was developed to parse all soccer-relevant documents and find the terms matching those in the cleaned expanded-query. Through this process, the number of related-documents were narrowed down, and it was possible for human to decide whether or not a document was relevant to the query.

6.3.2 Procedure

After KMgmt was ready, the experiments were performed. There were four different approaches to rank documents with each query. The queries and ranking mechanisms were modified accordingly:

Naive keyword approach (NaiveKW) This approach performed a naive keyword-matching technique. To make results comparable, queries were expanded with knowledge in the ModOnt. The scores of documents (which were used to rank them) were decided by the sum of the frequencies of the keywords in the documents. The score of a document was calculated by:

$$r_j = \sum_{i=1}^{n_q} f_{d_j t_i}$$

where r_j is the score of document d_j; n_q is the number of different terms in query q; $f_{d_j t_i}$ is the frequency of term t_i in document d_j. If term t_i does not occur in the document, $f_{d_j t_i}$=0. Documents with higher scores are with higher rankings.

Vector space approach (VectorSpace) This approach utilized the classic vector space technique. That is, both queries and documents were presented by vectors. The weights of elements in the vectors were defined by the term frequencies and TWs (obtained by our AATI scheme). The ranking of a document was decided by the inner product (section 2.1.1) between the vector of a query and the vector of a document. The inner product is expressed as,

$$I_j = q \cdot d_j = \sum_{i=1}^{k} w_{qt_i} \cdot w_{d_j t_i}$$

where I_j is the value of the inner product of query q and document d_j; k is the number of different terms, that is, the number of dimensions; w_{qt_i} is the term weight of term t_i in the query; $w_{d_jt_i}$ is the term weight of term t_i in the document d_j. The document with larger inner product value obtained higher rankings.

HOTIR-based approach with some (HOTIR(some)) In this approach, the linguistic quantifier *some* was used to aggregate pieces of information (section 4.4.2). Because of the definition of *some* in OWA, the aggregation process ignores hierarchies (see details in Chapter 5). That is, HOTIR(some) can be regarded as a HOTIR approach without hierarchy.

HOTIR-based approach with most (HOTIR(most)) In this approach the linguistic quantifier most was used to aggregate pieces of information (section 4.4.2).

6.3.3 Evaluation methodology

The results of MetricsExp are presented based on the *TopN* scheme, which has been widely used to evaluate Web-based information retrieval systems [68, 82, 98]. Kobayashi et al in [99] stated that for web-based application, "there is little hope of actually measuring the recall rate, ..., pages retrieved in the top 10 or 20 ranked documents (rather than all relevant pages)" are more important to information retrieval systems. After all, most people do not have the patience to read all relevant documents from the Web. The TopN scheme is focused on the documents with top n rankings. For example, if n is 10, the precision of the top 10 documents is calculated.

The receiver operating characteristic (ROC) curve was also used to illustrate the performances of HOTIR. The ROC curver is somewhat equivalent to the *precision-recall* (PR) curve[3]. In [100], Davis and GoadRich stated that "for any dataset, the ROC curve and PR curve for a given algorithm contain the same points. This equivalence leads to the surprising theorem that a curve dominates in ROC space if and only if it dominates in PR space." Moreover, compared with the TopN scheme, the ROC curve provides a complete analysis on the whole ranking list in the repository. Though not all users are interested on a complete analysis (most users care more about the top N documents returned from a query), the ROC curve provides a good way to study HOTIR.

Because in our experiments both the actual outcome and the predicted outcome are *soccer* or non-*soccer*, the four outcome can be expressed in a 2×2

[3]In precision-recall curve, the Y axis represents precision and the X axis represent recall.

Table 6.1: Confusion matrix

		Actual	
		Soccer	*Non-soccer*
Predicted	*Soccer*	True Positive(TP)	False Positive(FP)
	Non-soccer	False Negative(FN)	True Negative(TN)

matrix, which is called *confusion matrix* in Table 6.1.

In the ROC curve, the Y axis represents the *True Positive Rate* or *TPR*; and the X axis represents the *False Positive Rate* or *FPR*. Based on the confusion matrix, the definitions of *TPR* and *FPR* are:

$$\begin{aligned} TPR &= \frac{TP}{TP+FN} \\ FPR &= \frac{FP}{FP+TN} \end{aligned} \tag{6.1}$$

A perfect system will generate an ROC curve that goes straight upward until all of the relevant documents are encountered, then straight to the right for the remaining documents. A random system will produce a straight line from the origin to the upper right corner.

6.4 HumanExp setup

6.4.1 Objective

Herlocker et al. in [101] stated, "relevance is more inherently subjective." HumanExp involved humans, allowing us to make a human-oriented examination of our technique. The experiment was designed to compare results obtained from Google with results obtained from HOTIR. HumanExp provided an example of how our approach could be used as a value added service for existing keyword-based search engines like Google.

HOTIR performed with the linguistic quantifier *most* (HOTIR(most)) and the linguistic quantifier *some* (HOTIR(some)). As discussed in section 5.4, aggregations with *some* ignore hierarchies. The comparison of these two approaches was interesting as it showed the importance of hierarchal structure.

6.4.2 Procedure

For the purpose of HumanExp, a Java script utilizing Google API was developed. This script retrieved documents through URLs returned from Google with provided queries. The retrieved documents constituted a repository for a query, and the rankings of documents by Google were saved as well[4]. HOTIR ranked the documents according to its own principles and we compared the HOTIR and Google rankings.

Five soccer fans were participants in HumanExp. Soccer fans were selected to ensure participants had good background knowledge of the domain being tested. Participants, thereafter called human experts, were asked to score each document from 0 to 5 according to a given query. The meanings of the score numbers were:

> 1 – WORST – the document is not relevant; it contains no information regarding the query;
>
> 2 – BAD – the document is not very relevant, but it contains a little information;
>
> 3 – HARD TO SAY – the document is somewhat relevant; it contains some information;
>
> 4 – GOOD – the document is relevant; it contains quite a bit of information;
>
> 5 – GREAT – the document is strongly relevant; it contains a lot of information;
>
> 0 – ERROR – the document is not there or can not be be opened.

The arithmetic mean of user scores of a document was called its *combined score*, and the combined score decided the document's ranking. The rankings of documents scored by the human experts were compared with the rankings generated by Google and the two rankings generated by HOTIR.

The experiments employed the following steps for each query:

1. Using Java script, a repository consisting of up to 64 Web documents was created;
2. The rankings by Google of these documents were retrieved and stored;
3. Documents from the repository were ranked with HOTIR(some) and HOTIR(most);

[4]According to the limit of the Google API, there are 64 URLs for each query, i.e. 64 web documents, we can retrieve.

4. Documents from the repository were ranked by human experts;
5. Document rankings from Google, HOTIR(some), HOTIR(most), and human experts were compared.

6.4.3 Evaluation methodology

The methods of evaluation were different in MetricsExp and HumanExp. In HumanExp because here we focused on the orders of the returned documents; while in MetricsExp, we determined whether the retrieved documents were relevant. A normalized distance-based performance measure (NDPM) was used to evaluate HOTIR. NDPM was first proposed by Yao [102] theoretically, and is commonly used to compare two different rankings [101, 103, 104, 105]. NDPM is defined in Eq. 6.2.

$$NDPM = \frac{2C_{-} + C_u}{2C_i} \tag{6.2}$$

Let us assume there are two rankings: one is a system ranking (in our case, Google or HOTIR(some) or HOTIR(most)); the other one is a user ranking (Human experts). C_{-} is the number of *contradictory pairs* between the system ranking and the user ranking. When the system says that document 1 is preferred to document 2, but the user ranking says the opposite, we call them (document 1 and document 2) as a contradictory pair. C_u is the number of *compatible pairs*. When the user says document 1 has a higher ranking than document 2, while the system ranks document 1 and document 2 at equal levels, we call document 1 and document 2 a compatible pair. C_i is the total number of *preferred* pairs in the user's ranking. A preferred pair of documents means that, of two documents, one is rated higher than the other.

6.5 Experiment results

6.5.1 Query A

MetricsExp Query A was "John Terry's teammates in Chelsea." John Terry is a soccer player in the Chelsea club. This query was designed to retrieve Web documents containing information about other soccer players in the Chelsea club. In all methods (NaiveKW, VectorSpace, HOTIR(some), HOTIR(most)), the query was expanded by knowledge in the ModOnt. The expanded query included extra information about the other players in the Chelsea club, such as players' names, positions, etc. The local repository was

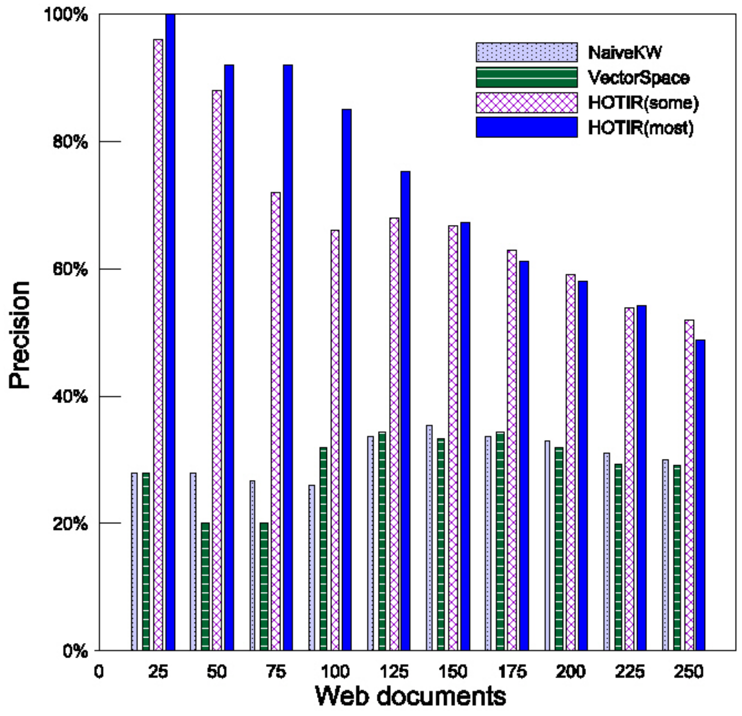


Figure 6.1: Top 250 web documents with Query A

applied, in which there were 22957 Web documents. The number of relevant Web documents was 173, and the number of non-relevant Web documents was $22957 - 173 = 22784$.

Fig. 6.1 shows how different approaches performed by presenting precisions in the top 250 Web documents. Precision is defined as the percent of the relevant documents in the retrieved set. The X axis represents the precision in the retrieved documents, and the Y axis represents the number of retrieved Web documents. The performance of the each approach was evaluated by calculating precisions in its top-ranked documents. For each approach we calculated precisions in the top 25, 50, 75, 100, 125, 150, 175, 200, 225 and 250 documents.

As shown in Fig. 6.1, HOTIR(some) and HOTIR(most) were much better at retrieving documents relevant to the query than NaiveKW and VectorSpace. In the top 25 Web documents retrieved by HOTIR(most), 100% (all 25) of them were relevant documents; for documents retrieved by HOTIR(some), 96%(24) were relevant; while for documents retrieved by both VectorSpace and NaiveKW, 28% (7) are relevant ones. A similar situation was found in the top 50, top 100, and top 150 documents. From the top 175 documents onward, HOTIR(some) began to perform better than HOTIR(most). In the top 175

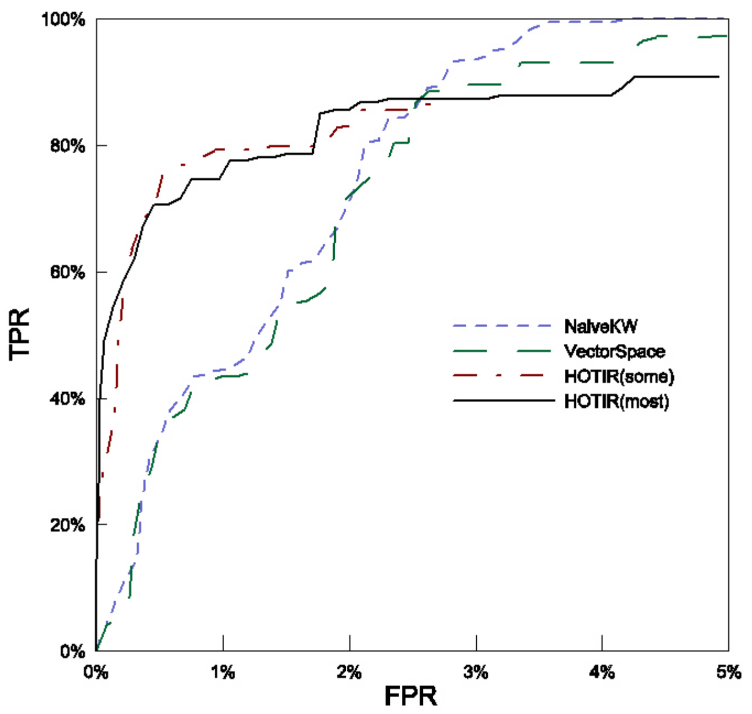


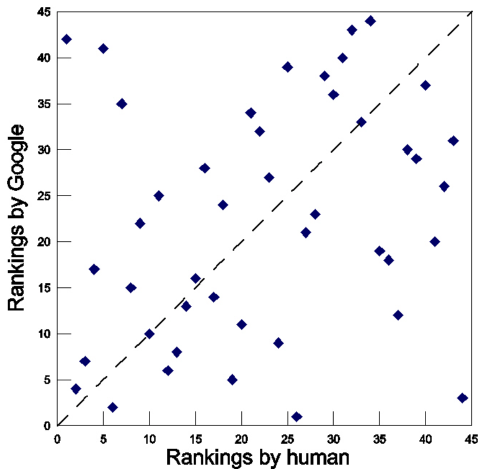
Figure 6.2: ROC curve based with Query A

documents ranked by HOTIR(some), 110 were relevant; while in the top 175 documents ranked by HOTIR(most), 107 web documents were relevant.

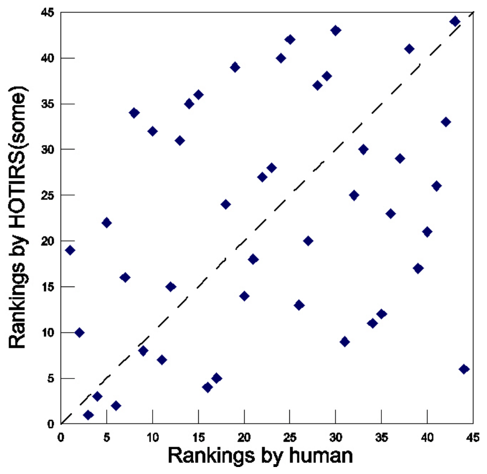
The ROC curve in Fig. 6.2 represents an overview of the rankings of the documents in the whole repository. Although as a Web-based information retrieval tool, the ROC curve may be not as important as the TopN figure, we present it here as an extra results to strengthen our analysis.

In Fig. 6.2, the Y axis represents the percent of retrieved relevant documents over all relevant documents, that is called *TPR*; and the X axis represents the percent of retrieved non-relevant documents over all non-relevant documents, that is the called *FPR* (See detailed definitions of *TPR* and *FPR* in Eq. 6.1).

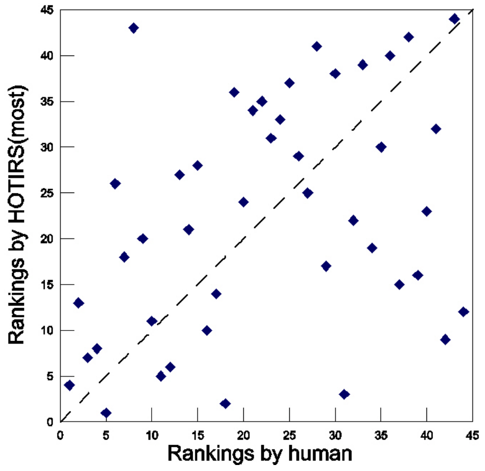
The interesting part in Fig. 6.2 is though HOTIR(some) and HOTIR(most) are better than NaiveKW and VectorSpace in retrieving the first 86.13% (149) relevant documents out of all relevant documents, they take more non-relevant documents to retrieve the rest 13.87% relevant ones. This is a tradeoff between higher precision and higher recall, or between more specific and more general. As we mentioned, precision is the percent of relevant documents in the retrieved ones, and recall is the percent of the retrieved relevant documents in all relevant documents. Since both HOTIR(some) and HOTIR(most) focus more on meanings expressed in users' queries than do NaiveKW and Vec-



(a) Google results



(b) HOTIR(some)



(c) HOTIR(most)

Figure 6.3: Rankings in HumanExp with Query A

torSpace, they retrieved documents with more specific requirements; that is, HOTIR(some) and HOTIR(most) performed with higher precision than recall.

HumanExp Fig. 6.3 includes three scatter plots displaying the rankings of the documents generated by Google, HOTIR(some), and HOTIR(most). In the plots, the Web documents are sorted by rankings from human experts in ascending order on the X axes. The Y axes represent the rankings of documents from Google or HOTIR(some) or HOTIR(most). For example, a point (x,y) in Fig. 6.3a represents one document that has a ranking of x by human experts and a ranking y by Google. The dotted line in the figure represents ideal rankings, which are generated from rankings by human experts.

NDPM measures the performances of the approaches in HumanExp with Query A. Unlike the plots in Fig. 6.3, the NDPM (section 6.4.3) measure

Table 6.2: Summary of HumanExp with Query A

Number of web documents	44
Total preferred pairs	865
Google Results	
NDPM	**0.408**
Contradictory pairs (Google)	353
compatible pairs (Google)	0
HOTIR(some)	
NDPM	**0.417**
Contradictory pairs (HOTIR)	361
compatible pairs (HOTIR)	0
HOTIR(most)	
NDPM	**0.386**
Contradictory pairs (HOTIR)	334
compatible pairs (HOTIR)	0

is a results of the comparing two rankings. A high value of NDPM means the two rankings are far apart; a low value of NDPM means there is not much difference between two rankings. Because we set human-based ranking as the ideal system, non-human rankings with lower NDPM values were desirable.

The results are summarized in Table 6.2. Because the Web documents that human experts failed to open were removed from the repository, there were 44 documents left for Query A. The number of preferred pairs was 865, which is defined to calculate NDPM (C_i in Eq. 6.2, section 6.4.3). The contradictory pairs and the compatible pairs are defined in section 6.4.3. NDPM is defined by Eq.(6.2).

The results in Table 6.2 indicate that HOTIR(most) (NDPM = 0.386) was the most successful approach of the three approaches tested; Google (NDPM = 0.408) performed better than HOTIR(some) (NDPM = 0.417).

In the daily life of using search engines, the searching results listed on the first page are the most important to most users. Because in Google the default number of results per page is 10, the top 10 returned documents of the three approaches and their scores by human experts are presented in Table 6.3. In each row of the table, there are:

- *Rank* in *System* category: ranking of a document by Google, HOTIR(some), or HOTIR(most);
- *Usr1* – *Usr5* in *User Score* category: scores of a document by the five human experts;

- *Combined* in *User Score* category: the arithmetic mean of the scores of a document by the five human experts; this number is regarded as the ideal score of the document;
- *Rank* in *User* category: ranking of a document by human experts; the rankings is based on the *combined score* of the document.

The arithmetic mean **Avg.** of the *Combined score* of the top 10 documents obtained in each approach appears at the bottom of each table. Higher Avg values indicate better performance of the system on the top 10 documents.

The average scores for the top 10 documents were: Google 2.86, HOTIR(some) 2.94, and HOTIR(most) 3.02 (Table 6.3,). Thus, in the top 10 documents, HOTIR(most) performed better than HOTIR(some), which performed better than Google.

6.5.2 Query B

MetricsExp Query B was "Chelsea", which was designed to retrieve information about the famous soccer club Chelsea. In all methods (NaiveKW, VectorSpace, HOTIR(some), HOTIR(most)), the query was expanded by the knowledge in the ModOnt. The expanded query included extra information such as location of the team, stadium of the team, players of the team, etc. The local repository of 22957 Web documents was applied. The number of relevant documents was 185, and the number of non-relevant documents was $22957 - 185 = 22772$.

HOTIR(some) and HOTIR(most) performed much better than NaiveKW and VectorSpace as shown in Fig. 6.4. From the top 25 to the top 250 Web documents, the precision of HOTIR(most) was 95% to 54%, the precision of HOTIR(some) was 92% to 55%, the precision of NaiveKW was from 28% to 36%, and the precision of VectorSpace was from 28% to 34%. Before the top 125 Web documents, the performance of HOTIR(most) was better than that of HOTIR(some); while after the top 125 Web documents, HOTIR(some) performed better than HOTIR(most). The performance difference between HOTIR(some) and HOTIR(most) was small. For example, in the top ranked 25 documents: of those retrieved by HOTIR(most), 96% (24) were relevant; of those retrieved by HOTIR(some), 92% (23) were relevant. In the top ranked 150 web documents: of those retrieved by HOTIR(most), around 71% (107) were relevant; in those by HOTIR(some), 74% (111) are relevant.

The results in Fig. 6.5 show that HOTIR(some) was the best one of the four approaches. The performance of HOTIR(most) with Query B was sim-

Table 6.3: Top 10 documents by the three approaches with Query A

Top 10 documents ranked by Google

System	User score						User
Rank	Usr.1	Usr.2	Usr.3	Usr.4	Usr.5	Combined	Rank
1	2	2	2	3	2	2.2	26
2	2	2	5	4	4	3.4	6
3	1	1	1	2	2	1.4	44
4	3	3	4	5	4	3.8	2
5	2	2	3	3	3	2.6	17
6	3	2	2	4	4	3.0	11
7	4	3	4	4	3	3.6	3
8	2	2	4	5	2	3.0	11
9	2	1	4	4	1	2.4	20
10	2	1	4	5	4	3.2	9
						Avg.:2.86	

Top 10 documents ranked by HOTIR(some)

System	User score						User
Rank	Usr1	Usr2	Usr3	Usr4	Usr5	Combined	Rank
1	4	3	4	4	3	3.6	3
2	2	2	5	4	4	3.4	6
3	3	3	4	5	3	3.6	3
4	1	2	4	4	3	2.8	16
5	2	2	3	3	3	2.6	17
6	1	1	1	2	2	1.4	44
7	2	3	4	4	3	3.0	11
8	3	2	4	3	4	3.2	9
9	1	2	4	1	2	2.0	31
10	3	3	4	5	4	3.8	2
						Avg.:2.94	

Top 10 documents ranked by HOTIR(most)

System	User score						User
Rank	Usr1	Usr2	Usr3	Usr4	Usr5	Combined	Rank
1	4	4	4	3	3	3.6	3
2	3	2	2	3	3	2.6	17
3	1	2	4	1	2	2	31
4	5	5	2	5	5	4.4	1
5	2	2	4	4	3	3	11
6	3	2	2	4	4	3	11
7	4	3	4	4	3	3.6	3
8	3	3	4	5	3	3.6	3
9	1	1	2	2	2	1.6	39
10	1	2	4	4	3	2.8	16
						Avg.:3.02	

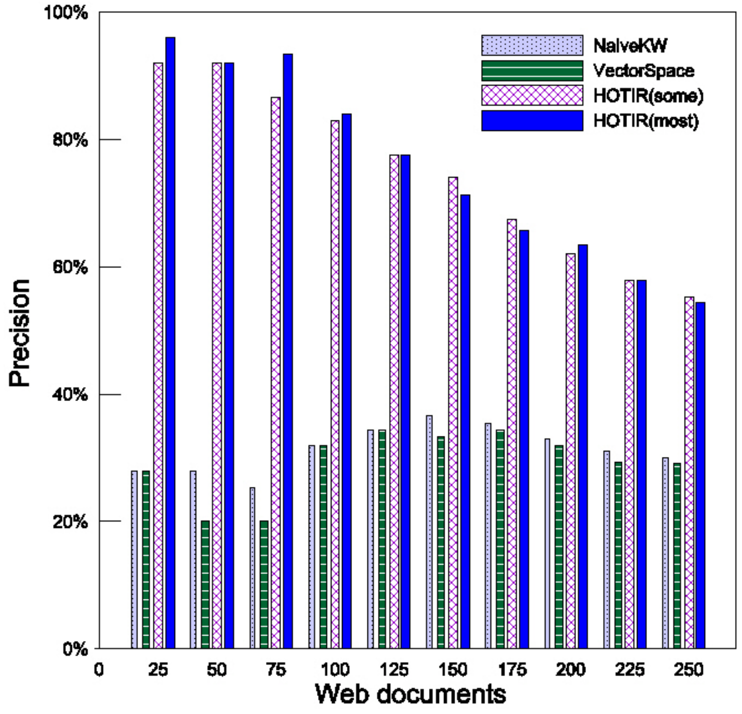


Figure 6.4: Top 250 web documents with Query B

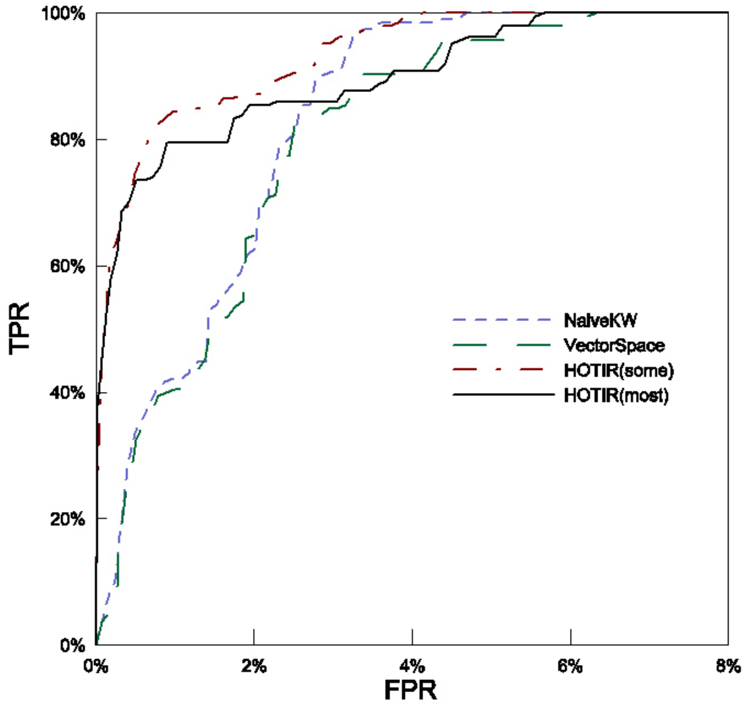
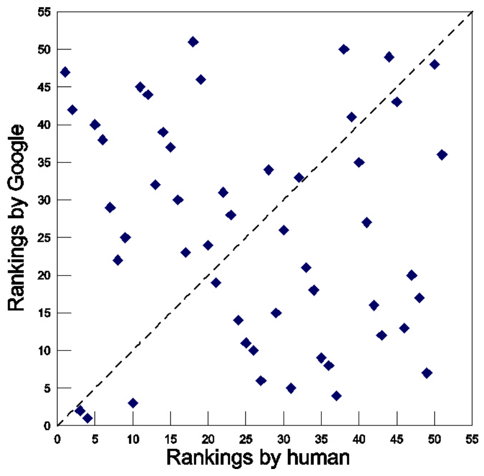
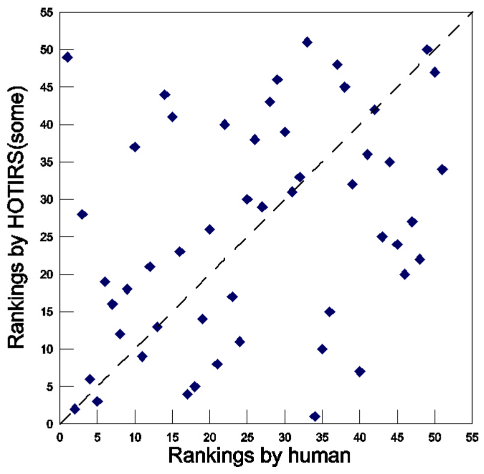


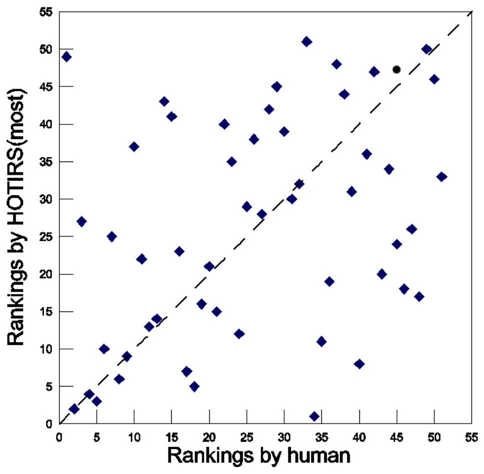
Figure 6.5: ROC curve with Query B



(a) Google results



(b) HOTIR(some)



(c) HOTIR(most)

Figure 6.6: Rankings in HumanExp with Query B

ilar to its performance with Query A: HOTIR(most) performed as well as HOTIR(some) and much better than NaiveKW and VectorSpace; then the precision of HOTIR(most) dropped and could not keep up with that of HOTIR(some). Before 85.96% relevant documents were retrieved, HOTIR(most) performed better than VectorSpace and NaiveKW. After that NaiveKW performed better in retrieving the rest of the relevant documents than VectorSpace and HOTIR(most).

HumanExp Three scatter plots Fig. 6.6a, Fig. 6.6b and Fig. 6.6c compare the document rankings by human experts with document rankings by Google, HOTIR(some), and HOTIR(most), respectively.

The results of HumanExp with Query B are presented in Table 6.4. There were 51 valid Web documents in the repository. The total number of preferred pairs

Table 6.4: Summary of HumanExp with Query B

Number of web documents	51
Total preferred pairs	1202
Google Results	
NDPM	**0.527**
Contradictory pairs (Google)	633
compatible pairs (Google)	0
HOTIR(some)	
NDPM	**0.369**
Contradictory pairs (HOTIR)	437
compatible pairs (HOTIR)	12
HOTIR(most)	
NDPM	**0.366**
Contradictory pairs (HOTIR)	434
compatible pairs (HOTIR)	12

was 1202. The NDPM values for Google, HOTIR(some), and HOTIR(most) were 0.527, 0.369, and 0.366, respectively. The contradictory pairs of Google, HOTIR(some), and HOTIR(most) were 633, 437, and 434, respectively. The compatible pairs of Google, HOTIR(some), and HOTIR(most) were 0, 12, and 12, respectively. HOTIR(some) and HOTIR(most) performed better than Google when system rankings were compared with rankings by human experts in this experiment. HOTIR(most) performed only slightly better than HOTIR(some). It seems that with Query B the information hierarchy did not make much difference.

The average scores for the top 10 documents were: Google 2.64, HOTIR(some) 3.04, and HOTIR(most) 3.32 (Table 6.5). Therefore, in the first 10 returned documents, HOTIR(most) performed better than HOTIR(some) which performed better than Google.

6.5.3 Query C

MetricsExp Query C was "famous soccer players". It was designed to retrieve information about famous soccer players. In all methods (NaiveKW, VectorSpace, HOTIR(some), HOTIR(most)), the query was expanded by the knowledge in the ModOnt. The expanded query included information about famous soccer players, such as their names, clubs they play in, etc. The local repository of 22957 documents was applied. The number of relevant documents was 473, and the number of non-relevant web documents was

Table 6.5: Top 10 documents by three approaches with Query B in HumanExp

Top 10 documents ranked by Google							
System	User score						User
Rank	Usr.1	Usr.2	Usr.3	Usr.4	Usr.5	Combined	Rank
1	5	3	5	2	5	4.0	3
2	5	3	5	2	5	4.0	3
3	5	5	2	5	1	3.6	9
4	1	2	3	3	1	2	3.2
5	1	4	2	3	1	2.2	30
6	1	4	3	1	4	2.6	23
7	3	1	1	1	1	1.4	47
8	1	2	3	2	2	2.0	32
9	1	3	2	1	3	2.0	32
10	3	3	3	1	3	2.6	23
						Avg.:2.64	
Top 10 documents ranked by HOTIR(some)							
System	User score						User
Rank	Usr1	Usr2	Usr3	Usr4	Usr5	Combined	Rank
1	1	2	3	2	2	2.0	32
2	4	4	4	5	4	4.2	1
3	3	4	3	5	4	3.8	5
4	3	3	4	4	2	3.2	14
5	3	4	1	4	3	3.0	18
6	5	3	5	2	5	4.0	3
7	1	2	2	2	2	1.8	38
8	3	3	4	1	4	3.0	18
9	3	4	3	4	3	3.4	11
10	1	3	2	1	3	2.0	32
						Avg.:3.04	
Top 10 documents ranked by HOTIR(most)							
System	User score						User
Rank	Usr1	Usr2	Usr3	Usr4	Usr5	Combined	Rank
1	1	2	3	2	2	2.0	32
2	4	4	4	5	4	4.2	1
3	3	4	3	5	4	3.8	5
4	5	3	5	2	5	4.0	3
5	3	4	1	4	3	3.0	18
6	4	4	4	4	3	3.8	5
7	3	3	4	4	2	3.2	14
7	1	2	2	2	2	1.8	38
9	3	4	4	5	2	3.6	9
10	3	4	4	5	3	3.8	5
						Avg.:3.32	

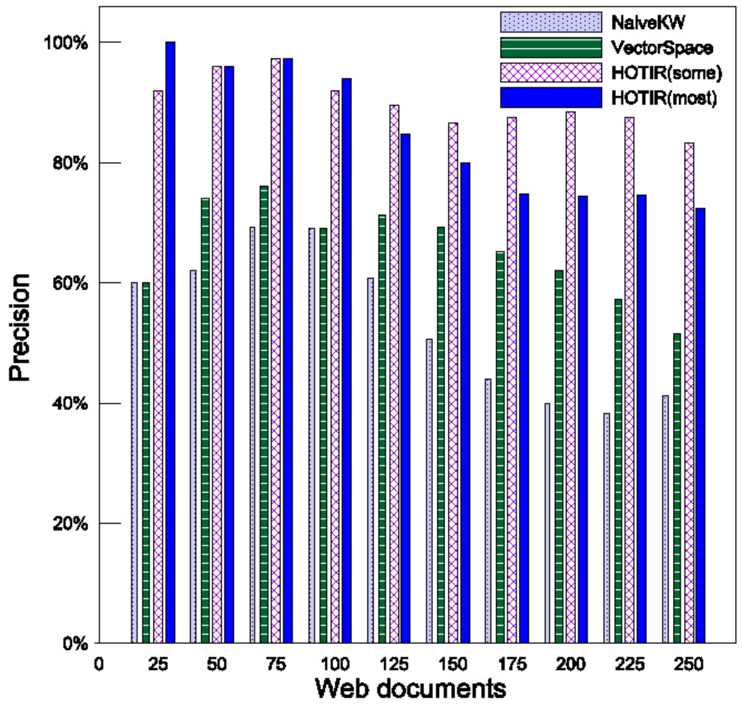


Figure 6.7: Top 250 web documents with Query C

$22957 - 473 = 22484$.

NaiveKW performed the worst of the four IR systems tested with Query C (Fig. 6.7). Although NaiveKW had the same number of relevant documents (15 out of 25) as VectorSpace in the top 25 web documents, in the rest VectorSpace had higher precision. HOTIR(some) and HOTIR(most) performed better than both NaiveKW and VectorSpace, and it was difficult to determine which of HOTIR(some) and HOTIR(most) performed the best. Before the top 125 documents (top 25, top 50, and top 100), HOTIR(most) had higher precision. As shown in Fig. 6.7, the precision of HOTIR(most) was 100% precision in the top 25 (25 out of 25), 96% in the top 50 (48 out of 50), 97% in the top 75 (73 out of 75), and 94% in the top 100 (94 out of 100). The precision of HOTIR(some) was 100% in the top 25 (23 out of 25), 96% in the top 50 (48 out of 50), 97% in the top 75 (73 out of 75) and 92% in the top 100 (92 out of 100). However, after the top 100 documents, HOTIR(some) performed better than HOTIR(most).

An analysis of Fig. 6.8 leads to a similar conclusion: HOTIR(some) performed better than HOTIR(most) because more relevant documents were collected for the same amount of documents retrieved according to the rankings by HOTIR(some). The only advantage of HOTIR(most) was that there are more

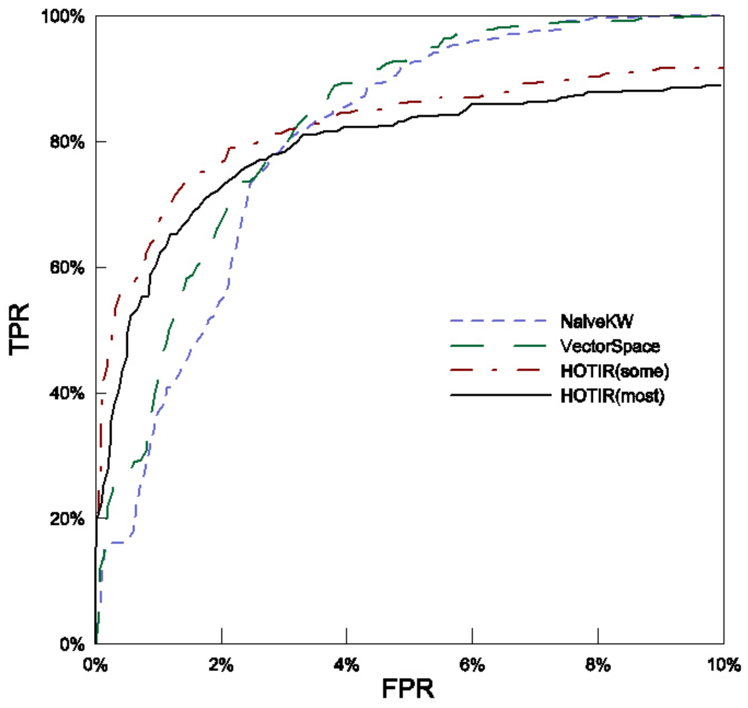


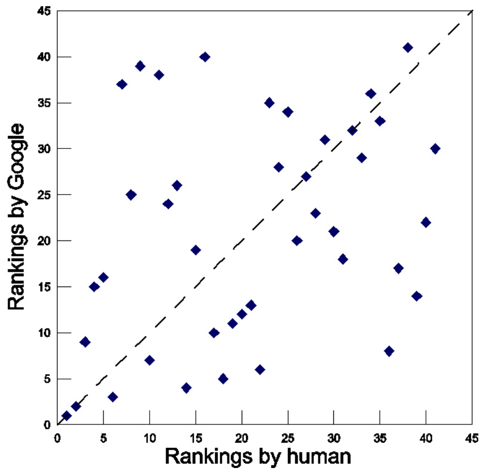
Figure 6.8: ROC curve with Query C

relevant documents in the top 100 documents it retrieved. VectorSpace had the highest recall. VectorSpace performed better after around 78% of the relevant documents were retrieved, but it was not as good as HOTIR(some) and HOTIR(most) before that point.

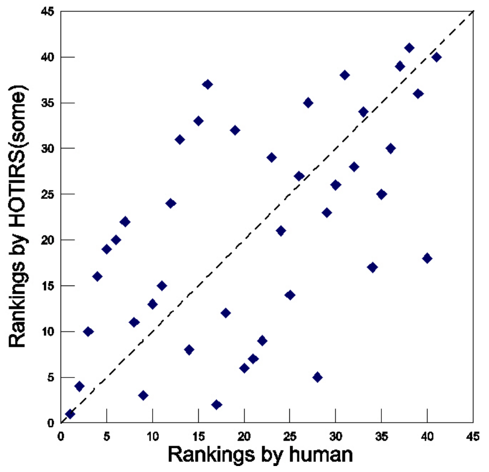
HumanExp Three scatter plots are shown in Fig. 6.9 displaying the comparison of the rankings by human experts and the rankings by Google, HOTIR(some) and HOTIR(most).

The results of HumanExp with Query C are presented in Table 6.6. There were 41 valid documents in the repository. The total number of the preferred pairs was 792. The NDPM values for Google, HOTIR(some), and HOTIR(most) were 0.374, 0.307, and 0.320, respectively. The contradictory pairs of Google, HOTIR(some), and HOTIR(most) were 296, 243, and 253, respectively. The compatible pairs of Google, HOTIR(some), and HOTIR(most) were 0, 1, and 1, respectively. Both HOTIR(some) and HOTIR(most) performed better than Google in this experiment; HOTIR(some) performed better than HOTIR(most).

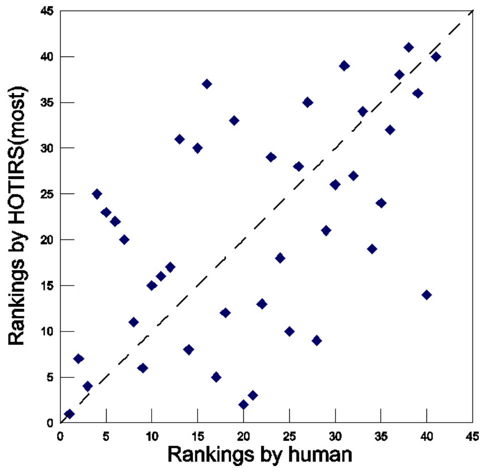
As shown in Table 6.7, the average score for the top 10 documents were: Google 3.48, HOTIR(some) 3.34, and HOTIR(most) 3.30. Therefore, Google



(a) Google results



(b) HOTIR(some)



(c) HOTIR(most)

Figure 6.9: Rankings in HumanExp with Query C

Table 6.6: Summary of HumanExp with Query C

Number of web documents	41
Total preferred pairs	792
Google Results	
NDPM	**0.374**
Contradictory pairs (Google)	296
compatible pairs (Google)	0
HOTIR(some)	
NDPM	**0.307**
Contradictory pairs (HOTIR)	243
compatible pairs (HOTIR)	1
HOTIR(most)	
NDPM	**0.320**
Contradictory pairs (HOTIR)	253
compatible pairs (HOTIR)	1

performed better than HOTIR(some), and HOTIR(some) performed better than HOTIR(most).

6.5.4 Query D

MetricsExp Query D was defined as "Manchester > Ronaldo". Manchester is where the soccer club Manchester United is located. Ronaldo is now a player in another club (Real Madrid), but he previously played for Manchester United. Because the knowledge base did not contain the knowledge that these two concepts (Manchester and Ronaldo) are related, while user knew it, this query showed that knowledge from a user and knowledge from the knowledge base can be combined in HOTIR. ">" is a new defined operator in HOTIR that connects two objects and means the latter object supports the former object; that is, the concept defined by the right-side keyword is a subconcept of the concept defined by the left-side keyword. This query was designed to retrieve information about club Manchester United and player Ronaldo. Player Ronaldo helped define club Manchester United. In all methods (NaiveKW, VectorSpace, HOTIR(some), HOTIR(most)), the query was expanded by the knowledge in knowledge base ModOnt.

The local repository of 22957 Web documents was applied. The number of relevant Web documents was 169 and the number of non-relevant Web documents was $22957 - 169 = 22788$.

HOTIR(some) performed the best of the four IR systems tested in this ex-

Table 6.7: Top 10 documents by three approaches with Query C in HumanExp

Top 10 documents ranked by Google

System	User score						User
Rank	Usr1	Usr2	Usr3	Usr4	Usr5	Combined	Rank
1	5	5	5	5	5	5.0	1
2	5	4	5	5	5	4.8	2
3	4	3	5	4	5	4.2	3
4	4	3	4	2	3	3.2	14
5	1	3	2	5	1	2.8	17
6	3	2	3	3	2	2.6	20
7	3	3	5	5	3	3.8	9
8	1	1	1	3	1	1.4	34
9	4	4	5	4	4	4.2	3
10	3	2	3	4	2	2.8	17
						Avg.:3.48	

Top 10 documents ranked by HOTIR(some)

System	User score						User
Rank	Usr1	Usr2	Usr3	Usr4	Usr5	Combined	Rank
1	5	5	5	5	5	5.0	1
2	3	2	3	4	2	2.8	17
3	3	4	4	5	3	3.8	9
4	5	4	5	5	5	4.8	2
5	1	1	2	4	1	1.8	28
6	1	2	4	2	4	2.6	20
7	1	2	4	2	4	2.6	20
8	4	3	4	2	3	3.2	14
9	3	2	3	3	2	2.6	20
10	4	4	5	4	4	4.2	3
						Avg.:3.34	

Top 10 documents ranked by HOTIR(most)

System	User score						User
Rank	Usr1	Usr2	Usr3	Usr4	Usr5	Combined	Rank
1	5	5	5	5	5	5.0	1
2	1	2	4	2	4	2.6	20
3	1	2	4	2	4	2.6	20
4	4	4	5	4	4	4.2	3
5	3	2	3	4	2	2.8	17
6	3	4	4	5	3	3.8	9
7	5	4	5	5	5	4.8	2
8	4	3	4	2	3	3.2	14
9	1	1	2	4	1	1.8	28
10	2	1	3	4	1	2.2	25
						Avg.:3.30	

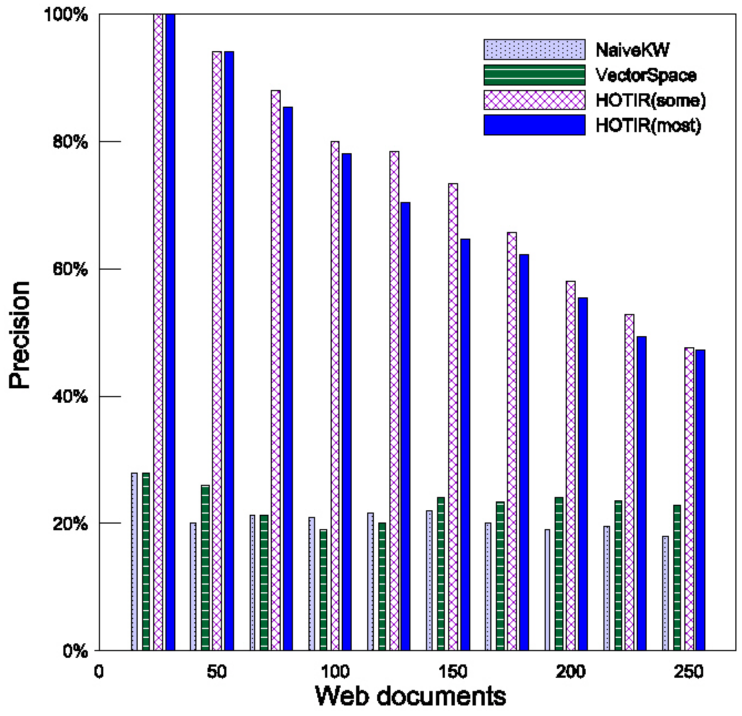


Figure 6.10: Top 250 web documents with Query D

periment (Fig. 6.10). In the top 25 Web documents in HOTIR(some) and HOTIR(most), 100% (25 out of 25) of them were relevant, and in the top 50, 94% (47 out of 50) were relevant. After 50 Web documents were retrieved, HOTIR(some) performed better than HOTIR(most). In the top 75 documents retrieved, 88% (66 out of 75) of the documents by HOTIR(some) were relevant and around 85% (64 out of 75) of the documents by HOTIR(most) were relevant. In the top 100 Web documents retrieved, 80% (80 out of 100) of the documents by HOTIR(some) were relevant and 78% (78 out of 100) of the documents by HOTIR(most) were relevant. NaiveKW and VectorSpace did not perform as well as HOTIR(some) or HOTIR(most). In NaiveKW and VectorSpace, the average precision was around 20%, while in HOTIR(some) and HOTIR(most), it was 47% to 100%.

Fig. 6.11 presents an overview of the performance of four approaches over the whole repository. Before around 80% of the relevant documents were retrieved, HOTIR(some) and HOTIR(most) performed much better than NaiveKW and VectorSpace. After 80% of the relevant documents were retrieved, it is difficult to see in the figure which one, HOTIR(some) or HOTIR(most), performed better. After 80% of the relevant documents were retrieved, both NaiveKW and VectorSpace performed better than HOTIR(some) and HOTIR(most). Between NaiveKW and VectorSpace, VectorSpace performed better than NaiveKW

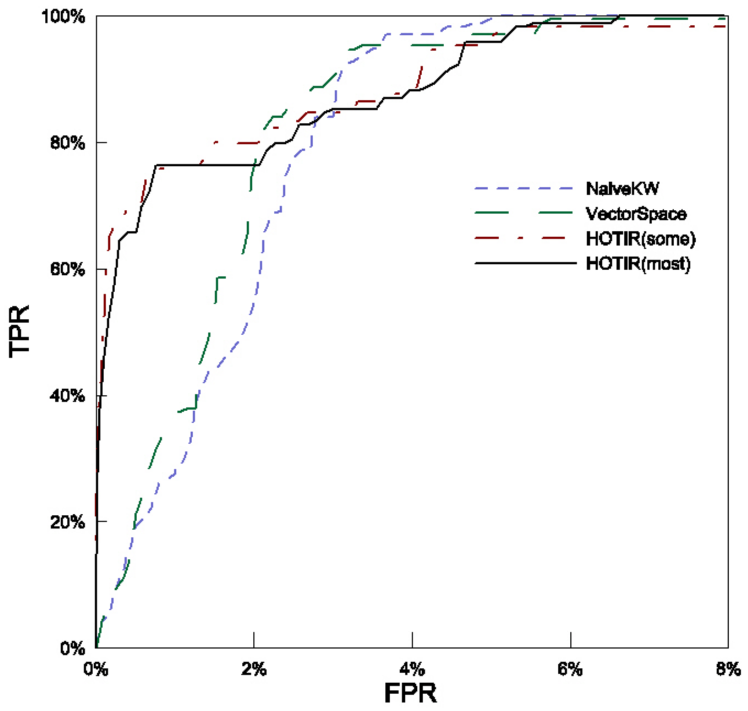


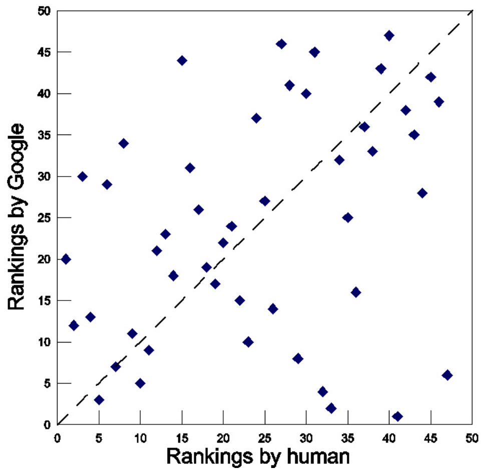
Figure 6.11: ROC curve with Query D

before 95.27% of the relevant documents were retrieved; after 95.27% of the relevant documents were retrieved, NaiveKW performed better than VectorSpace in retrieving the rest of the relevant documents.

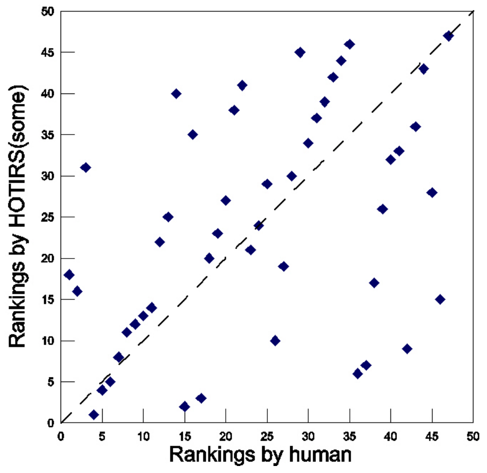
HumanExp In Fig. 6.12 three scatter plots compare document rankings by human experts with the document rankings by Google, HOTIR(some) and HOTIR(most).

The results of HumanExp with Query D are presented in Table 6.8. There were 47 valid documents in the repository. The total number of preferred pairs was 970. The NDPM values for Google, HOTIR(some), and HOTIR(most) were 0.375, 0.365, and 0.356, respectively. The contradictory pairs of Google, HOTIR(some), and HOTIR(most) were 364, 354, and 345, respectively. The compatible pairs of Google, HOTIR(some), and HOTIR(most) were 0, 0, and 0, respectively. In summary, HOTIR(some) and HOTIR(most) performed better than Google and HOTIR(most) performed better than HOTIR(some)

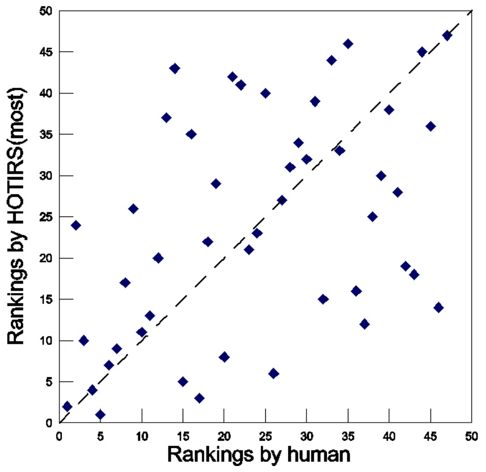
As shown in Table 6.9, the average scores for the top 10 documents were: Google 2.48, HOTIR(some) 2.62, and HOTIR(most) 3.16. Therefore, in the first 10 returned documents, HOTIR(most) performed better than HOTIR(some), and HOTIR(some) performed better than Google.



(a) Google results



(b) HOTIR(some)



(c) HOTIR(most)

Figure 6.12: Rankings in HumanExp with Query D

Table 6.8: Summary of HumanExp with Query D

Number of web documents	47
Total preferred pairs	970
Google Results	
NDPM	**0.375**
Contradictory pairs (Google)	364
compatible pairs (Google)	0
HOTIR(some)	
NDPM	**0.365**
Contradictory pairs (HOTIR)	354
compatible pairs (HOTIR)	0
HOTIR(most)	
NDPM	**0.356**
Contradictory pairs (HOTIR)	345
compatible pairs (HOTIR)	0

6.5.5 Query E

MetricsExp Query E was defined as "Owen > Rooney". Micheal Owen and Wayne Rooney, are soccer players in England. Although there was no direct relation between them in the knowledge base, a user could provide a query that connects them. Query E presented a user's interest in both players. However, because *Rooney* was a subobject of *Owen* in the query, *Rooney* was included in the definition of *Owen*.

The semantics of Query E depended on the linguistic quantifier used. If a linguistic quantifier that ignores hierarchical structures, like *some*, was utilized, Query E represented an information requirement regarding *Owen* or *Rooney*. If a linguistic quantifier that does not ignore hierarchical structures, like *most*, was utilized, *Rooney* in Query E, as a new requirement, was added to define object *Owen*. That is, the satisfaction of *Rooney* contributed to the satisfaction of *Owen*, and the satisfaction of *Owen* was the final criterion to retrieve web documents.

The local repository of 22957 Web documents was applied. The number of relevant Web documents was 144, and the number of non-relevant Web documents was $22957 - 144 = 22813$.

HOTIR(most) gave the best performance of the four IR approaches (Fig. 6.13) except for the retrieval of the top 25 documents, where HOTIR(some) performed a bit better. Both HOTIR(some) and HOTIR(most) performed better than VectorSpace and NaiveKW.

Table 6.9: Top 10 documents by three approaches with Query D in HumanExp

Top 10 documents ranked by Google

System	User score						User
Rank	Usr1	Usr2	Usr3	Usr4	Usr5	Combined	Rank
1	1	1	2	3	2	1.8	36
2	2	2	2	2	2	2.0	30
3	3	2	4	4	3	3.2	4
4	2	1	3	2	2	2.0	30
5	3	2	4	4	3	3.2	4
6	2	1	1	1	1	1.2	46
7	3	2	4	4	3	3.2	4
8	2	1	4	2	2	2.2	27
9	3	2	4	4	3	3.2	4
10	2	1	4	4	2	2.6	23
						Avg.:2.46	

Top 10 documents ranked by HOTIR(some)

System	User score						User
Rank	Usr1	Usr2	Usr3	Usr4	Usr5	Combined	Rank
1	2	2	4	5	3	3.2	4
2	3	2	4	4	2	3.0	15
3	2	1	5	4	2	2.8	17
4	3	2	4	4	3	3.2	4
5	2	2	5	4	3	3.2	4
6	2	1	2	2	2	1.8	36
7	1	1	3	2	2	1.8	36
8	3	2	4	4	3	3.2	4
9	1	1	2	2	2	1.6	42
10	2	2	3	3	2	2.4	26
						Avg.:2.62	

Top 10 documents ranked by HOTIR(most)

System	User score						User
Rank	Usr1	Usr2	Usr3	Usr4	Usr5	Combined	Rank
1	3	2	4	4	3	3.2	14
2	4	4	5	5	4	4.4	1
3	2	1	5	4	2	2.8	17
4	2	2	4	5	3	3.2	4
5	3	2	4	4	2	3.0	15
6	2	2	3	3	2	2.4	26
7	2	2	5	4	3	3.2	4
8	5	3	2	2	2	2.8	17
9	3	2	4	4	3	3.2	4
10	3	2	5	4	3	3.4	3
						Avg.:3.16	

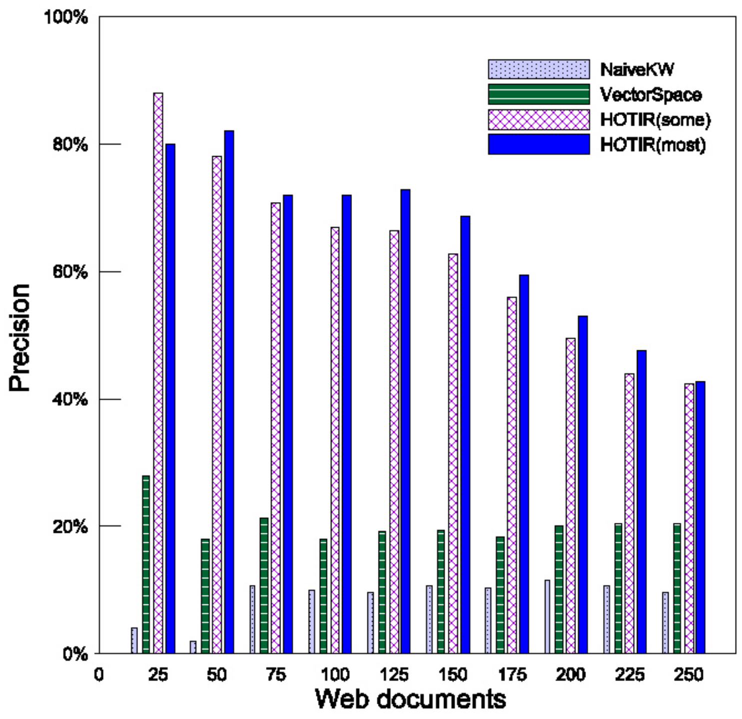


Figure 6.13: Top 250 web documents with Query E

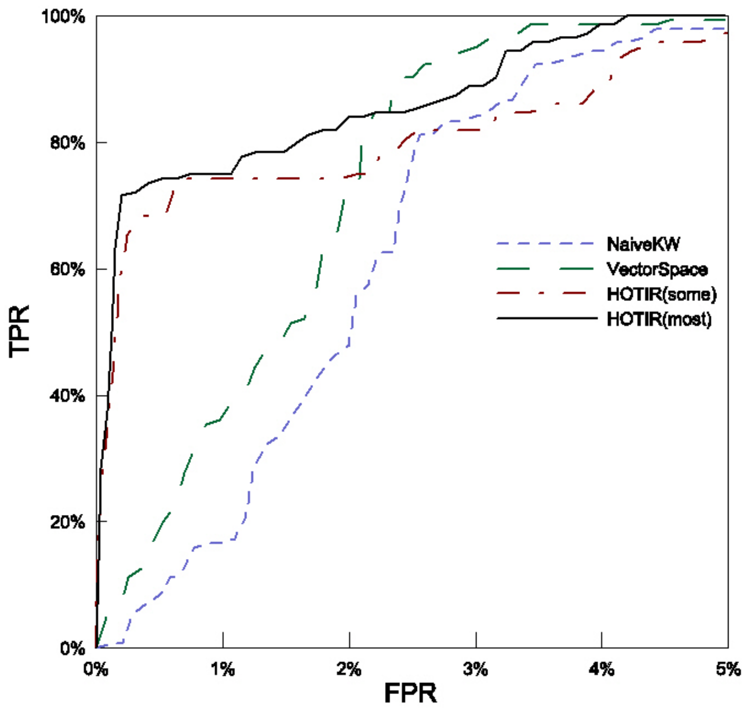
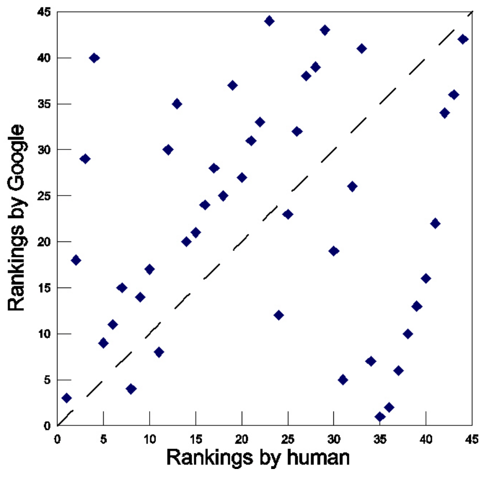
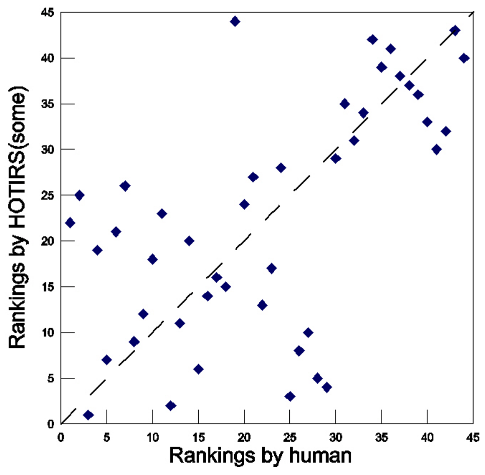


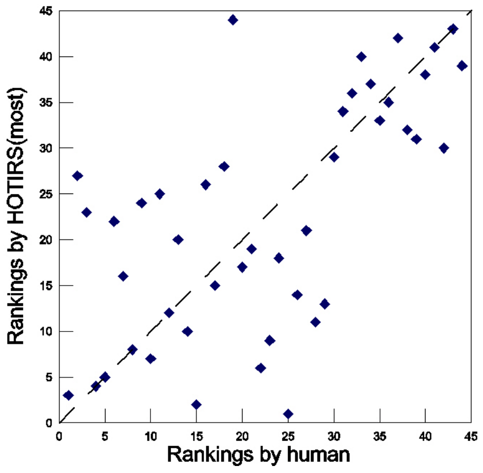
Figure 6.14: ROC curve with Query E



(a) Google results



(b) HOTIR(some)



(c) HOTIR(most)

Figure 6.15: Rankings in HumanExp with Query E

Fig. 6.14 presents an overview of the four approaches over the whole repository. Before around 85% relevant documents were retrieved, HOTIR(most) was the best approach. After that point, VectorSpace performed the best. However, just before all relevant documents (around 97% to 100%) were retrieved, HOTIR(most) returned to give the best performance. HOTIR(some) was the second best approach before 75% of the relevant documents were retrieved. In the end, HOTIR(most) retrieved all relevant documents firstly, VectorSpace was the second one, NaiveKW was the third one and HOTIR(some) was the last one.

HumanExp Fig. 6.15 includes three scatter plots that compare rankings of the documents by human experts with rankings by Google, HOTIR(some) and HOTIR(most).

Table 6.10: Summary of HumanExp with Query E

Number of web documents	44
Total preferred pairs	879
Google Results	
NDPM	**0.484**
Contradictory pairs (Google)	425
compatible pairs (Google)	0
HOTIR(some)	
NDPM	**0.266**
Contradictory pairs (HOTIR)	234
compatible pairs (HOTIR)	0
HOTIR(most)	
NDPM	**0.234**
Contradictory pairs (HOTIR)	206
compatible pairs (HOTIR)	0

The results of HumanExp with Query E are presented in Table 6.10. There were 44 valid Web documents in the repository. The total number of preferred pairs was 879. The NDPM values of Google, HOTIR(some) and HOTIR(most) were 0.484, 0.266, and 0.234, respectively. The contradictory pairs of Google, HOTIR(some), and HOTIR(most) were 425, 234, and 206, respectively. The compatible pairs of Google, HOTIR(some) and HOTIR(most) were 0, 0, and 0, respectively. In summary, HOTIR(some) and HOTIR(most) performed better than Google; HOTIR(most) performed better than HOTIR(some).

As shown in Table 6.11, the average score for the top 10 documents were: Google 2.08, HOTIR(some) 2.78, and HOTIR(most) 3.04. Therefore, in the first 10 returned documents, HOTIR(most) performed better than HOTIR(some), and HOTIR(some) performed better than Google.

6.5.6 Query F

Query F was a special query that contained all the knowledge of soccer in the knowledge base. That is, Query F included all concepts, their attributes, and their relations. Query F was only tested in MetricsExp.

The local repository of 22957 Web documents was applied. The number of relevant Web documents was 2348 and the number of non-relevant Web documents was 20609.

Fig. 6.16 shows how the four different approaches performed by presenting the

Table 6.11: Top 10 documents by the approaches with Query E in HumanExp

Top 10 documents ranked by Google

System	User score						User
Rank	Usr1	Usr2	Usr3	Usr4	Usr5	Combined	Rank
1	1	1	1	1	1	1.0	35
2	1	1	1	1	1	1.0	35
3	4	4	4	5	4	4.2	1
4	3	3	4	4	2	3.2	8
5	1	1	2	2	1	1.4	31
6	1	1	1	1	1	1.0	35
7	1	1	1	2	1	1.2	34
8	3	3	3	4	2	3.0	11
9	3	4	4	5	3	3.8	5
10	1	1	1	1	1	1.0	35
						Avg.:2.08	

Top 10 documents ranked by HOTIR(some)

System	User score						User
Rank	Usr1	Usr2	Usr3	Usr4	Usr5	Combined	Rank
1	3	4	5	5	3	4.0	3
2	3	3	4	4	1	3.0	11
3	2	2	2	2	2	2.0	24
4	2	2	2	3	1	2.0	24
5	3	3	2	5	1	2.8	14
6	3	4	4	5	3	3.8	5
7	2	2	1	4	1	2.0	24
8	3	3	4	4	2	3.2	8
9	1	2	1	4	2	2.0	24
10	2	3	3	5	2	3.0	11
						Avg.:2.78	

Top 10 documents ranked by HOTIR(most)

System	User score						User
Rank	Usr1	Usr2	Usr3	Usr4	Usr5	Combined	Rank
1	2	2	2	2	2	2.0	24
2	3	3	2	5	1	2.8	14
3	4	4	4	5	4	4.2	1
4	3	4	5	5	3	4.0	3
5	3	4	4	5	3	3.8	5
6	2	2	2	3	2	2.2	20
7	3	3	3	4	3	3.2	8
8	3	3	4	4	2	3.2	8
9	2	2	3	3	1	2.2	20
10	2	3	3	4	2	2.8	14
						Avg.:3.04	

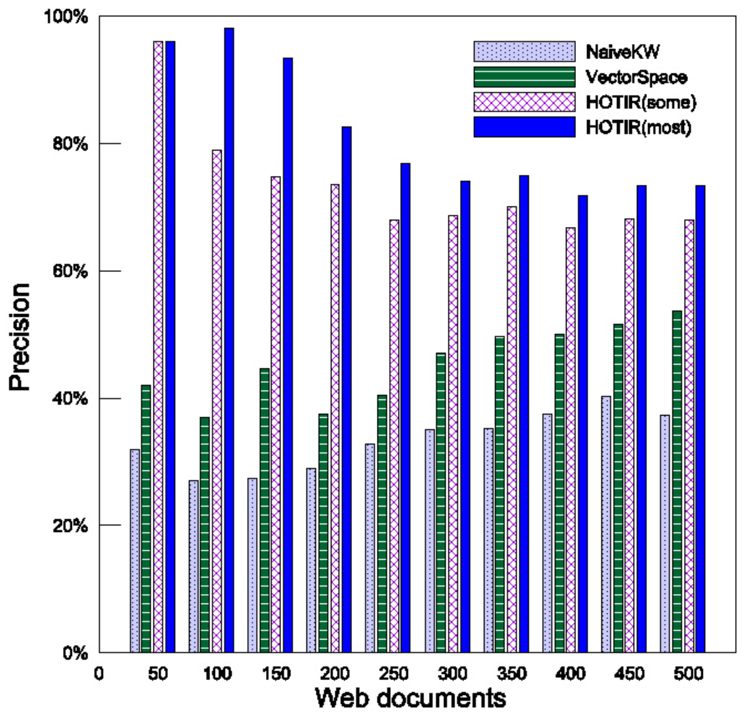


Figure 6.16: Top 500 documents with Query F

precision of relevant Web documents in the top 500 Web documents. Here we considered the top 500 documents instead of the top 250 documents because of the increasing number of the relevant Web documents. In the experiment with Query F, we calculated precisions for each approach in the top 50, 100, 150, 200, 250, 300, 350, 400, 450 and 500 documents.

As shown in Fig. 6.16, HOTIR(most) and HOTIR(some) performed better than NaiveKW and VectorSpace. Intuitively, HOTIR(most) performed the best; HOTIR(some) performed the second best; VectorSpace performed better than NaiveKW but worse than HOTIR(some).

In Fig. 6.17, it can be seen that NaiveKW was not as successful at information retrieval for Query F as the other three approaches. The performances of VectorSpace, HOTIR(some), and HOTIR(most) are close to each other in Fig. 6.17, but HOTIR(some) appears to have given the best performance of these three approaches, while VectorSpace gave the worst.

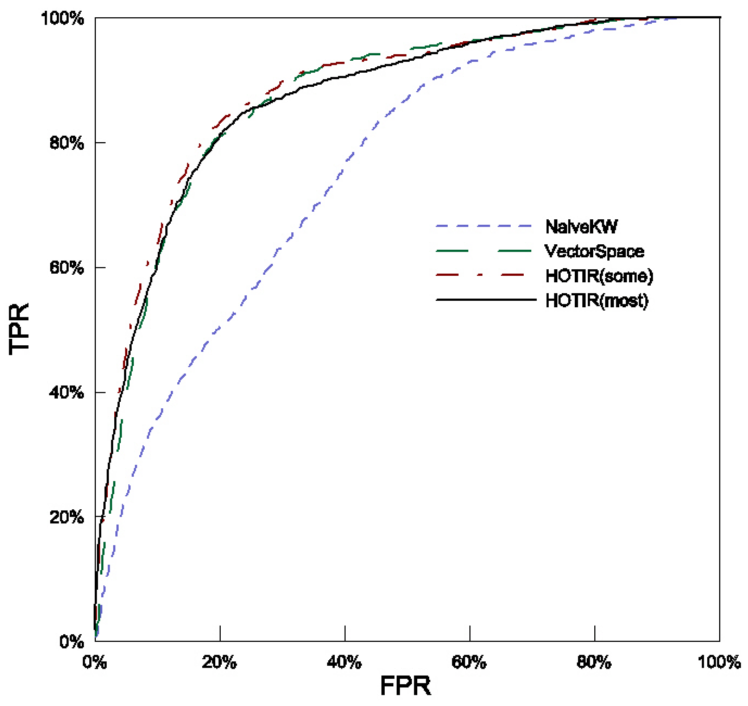


Figure 6.17: ROC curve with Query F

6.6 Conclusions

To examine HOTIR, we designed two series of experiments and applied six queries, five conventional ones (Query A, Query B, Query C, Query D, and Query E) and one special query (Query F).

The first series of experiments, MetricsExp, was based on a large local repository of the crawled Web documents. For each query, Web documents were first labelled as *relevant* or *non-relevant*. Then four different approaches (NaiveKW, VectorSpace, HOTIR(some), and HOTIR(most)) (section 6.3.2) were applied to score the documents by evaluating their relevance to a query. The results were presented in TopN figures and ROC curves (section 6.3.3).

The second series of experiments, HumanExp, included rankings of human experts. For each query, the returned documents by Google were crawled. Human experts then scored every document by evaluating its relevance to the query. The scores ranged from 1 (most non-relevant) to 5 (most relevant)[5]. The arithmetic mean of the scores (from human experts) was regarded as the ideal score of the document. HOTIR generated two scores by using linguistic quantifier *some* and *most*. In summary, a document had four rankings: one

[5]Besides, 0 means cannot open or not exist.

from human experts, one from Google, and two from HOTIR. The evaluation was mainly based on the NDPM measure (section 6.4.3).

From the results of the experiments, we drew several conclusions as described below.

1. HOTIR performed better compared to other information retrieval approaches.
   - In MetricsExp, the two HOTIR-based approaches, HOTIR(some) and HOTIR(most), outperformed two classical information retrieval approaches (NaiveKW and VectorSpace) on all 6 queries in the TopN figures. That is, if a user checked the returned first 250 (Query A-E) or 500 (Query F) documents, those returned by HOTIR(some) and HOTIR(most) would be found to be more relevant to the query. After studying ROC curves, we concluded that for all five conventional queries, HOTIR(some) and HOTIR(most) obtained better performance in retrieving around 80% or more of the relevant documents. NaiveKW and VectorSpace performed better in retrieving the last 10% - 20% of the relevant documents, this was reasonable because HOTIR(some) and HOTIR(most) were designed to better represent users' needs; that is, they expressed more specific meaning to a given query. This specificity sacrificed retrieval of some generally relevant objects. We deem this trade-off to be acceptable because precision is more important than recall considering the huge amount of Web documents available.
   - In HumanExp, two measures were used to compare HOTIR(some) and HOTIR(most) with Google, currently the most successful search engine. The first measure was NDPM (section 6.4.3). All ranked documents were used to calculate NDPM values. HOTIR(some) and HOTIR(most) outperformed Google with four out of five conventional queries[6]: Query B, Query C, Query D, and Query E. With Query A, HOTIR(most) performed better than Google, and HOTIR(some) performed worse than Google. The second measure was a comparison of arithmetic means of the scores of the top 10 documents by each approach. The scores were given by human experts, and each approach got its own top 10 document list. The average scores for the top 10 documents represented the satisfaction of the human experts for the top 10 documents. The results indicated that HOTIR(some) and HOTIR(most) performed better

[6]The special query, Query F, was not applied in HumanExp.

than Google with four out of five queries: Query A, Query B, Query D, and Query E; while with Query C, Google performed better than HOTIR(some) and HOTIR(most).

2. A comparison between HOTIR(some) and HOTIR(most) with respect to information retrieval was also interesting. As we discussed in section 5.4, linguistic quantifier provides instructions on how pieces of information are aggregated. With linguistic quantifier *some*, HOTIR ignores hierarchy information when aggregating. With linguistic quantifier *most*, HOTIR utilizes hierarchy information. Moreover, when *most* is used, a piece of less satisfied information tends to be more important in aggregation, because of the way *most* is modelled (section 5.4). Generally, though the HOTIR approaches with linguistic quantifier *some* and *most* (HOTIR(some) and HOTIR(most)) beat their competitors (NaiveKW and VectorSpace in MetricsExp and Google in HumanExp), and their (HOTIR(some) and HOTIR(most)) performances were about equal:

   – In MetricsExp, roughly speaking, according to the TopN figures (Fig 6.1, Fig. 6.4, Fig. 6.7, Fig. 6.10, Fig. 6.13, and Fig. 6.16), HOTIR(most) performed better with Query A, Query B, and Query E; HOTIR(some) performed better with Query D. It was hard to tell which one was better with Query C and Query F. According to the ROC curves (Fig 6.2, Fig. 6.5, Fig. 6.8, Fig. 6.11, Fig. 6.14 and Fig. 6.17), HOTIR(most) performed better with Query E; HOTIR-(some) performed better with Query B, Query C and Query D. It was hard to tell which one was better with Query A and Query F. In summary, a better performance of HOTIR(most) (than that of HOTIR(some)) in a TopN figure showed that HOTIR(most) was better at retrieving "very relevant" documents; while a better performance of HOTIR(some) (than that of HOTIR(most)) in an ROC curve shows that HOTIR(some) was better at retrieving "generally relevant" documents. We assumed that because the aggregation instructed by *most* is more specific than the aggregation instructed by *some*; the former was not as good as the latter when it was applied to binary categorization; while it was better at ranking systems.

   – The results of HumanExp supports our assumption (the aggregation instructed by *most* is more specific than the aggregation instructed by *some*; the former was not as good as the latter when it was applied to binary categorization; while it was better at ranking systems.). HOTIRS(most) showed better performance than HOTIR(some), obtaining smaller NDPM values with Query A, Query B, Query D and Query E (four out of five conventional queries) (see Table 6.2, Table 6.4, Table 6.8, and Table 6.10). With Query C, HOTIR(some) performed better than HOTIR(most) as evidenced by a smaller NDPM value (Table 6.6). HOTIR(most) had a higher arithmetic mean score for the top 10 documents retrieved than HOTIR(some), with Query A, Query B, Query D, and Query E (four out of five conventional queries) (see Table 6.3, Table 6.5, Table 6.9, and Table 6.11). With Query C, HOTIR(some) had a larger arithmetic mean than HOTIR(most) (Table 6.7).

In summary, we compared HOTIR with several other classic information retrieval approaches, and studied the performances of HOTIR with the linguistic quantifiers *some* and *most*. As a concept-based approach, HOTIR proved its ability to retrieve and rank Web documents.

Chapter 7

Conclusion and futurework

7.1 Conclusion

HOTIR, a new concept-based information retrieval framework for Web documents, is introduced in this thesis. It accepts conventional keyword-based queries, translates them into concepts, and organizes them into a hierarchy (HofC). HOTIR then enriches the definitions of the concepts with knowledge in a modified ontology (ModOnt), identifies equivalent concepts in Web documents, aggregates document concepts, and ranks the documents according to their relevance to the query.

The identification of concepts in Web documents can be explained as the process of discovering the meanings of texts in them. In order to make computers utilize the meanings of texts, HOTIR is comprised of several parts to make information retrieval more intelligent.

The knowledge base in HOTIR, called ModOnt, was built from an ontology. In the ModOnt concepts are defined by related concepts, terms, their importance values and by the relations between these concepts and terms. In the process of building concept-based queries from keyword-based queries, the knowledge base provides extra definitions of the concepts in queries. The vocabulary (terms and concepts) related to a domain and the relations between the terms and concepts in the vocabulary are manually stored in the ontology. The ontology is a collection of concepts and interconcept relations. Each concept is defined using a set of related-concepts and terms. However, since every concept or term contributes differently to the concept it defines, without importance information it is not possible to identify concepts in documents. A new AATI (adaptive assignment of term importance) scheme was proposed to assign importance weights to concepts and terms through reading "unknown"

Web documents. “Unknown” means there is no need for humans to read the documents and interpret their meanings to add extra knowledge for the AATI. Examples of extra knowledge are: topics of the documents, categories the documents belong to, concepts mentioned in the documents. The AATI scheme dramatically saves human labor. Integration of the ontology and the AATI scheme provides the knowledge base for HOTIR.

Queries inside HOTIR are in the form of a hierarchy of concepts (HofC). The HofC includes concept definitions that can be enriched with knowledge in the knowledge base, where hierarchies also exist. The organization of information – concepts, terms and their relations – into hierarchies allows knowledge to move through HOTIR smoothly. The application of hierarchies increases the complexity of calculation but improves the accuracy and efficiency of information retrieval.

Evaluation of how well a document matches a query is mapped as activation of the HofC-based query by information in the document. Ordered weighted averaging (OWA) operators are utilized to implement evaluation. The activation performs in a “bottom-up” manner. That is, it starts from the lowest level of the HofC, and the activation of the sublevel nodes contributes to the activation of the superlevel nodes. The activation level of the top level concept, the root concept (RC) (section 4.3.1), represents how well the document satisfies the HofC.

We have contributed to the information retrieval field by constructing a complete concept-based information retrieval framework, HOTIR. The process of information retrieval – query presentation, query expansion, knowledge base management, and evaluation of documents that match the query – can be abstracted as the definitions and identifications of concepts in the framework. In HOTIR, both definitions and identifications utilize hierarchical information:

- ModOnt, the ontology-based knowledge base provides supplementary knowledge to define concepts in user queries. ModOnt is based on an ontology.
- AATI scheme assigns importance to terms and concepts; it updates the ModOnt.
- Concepts, terms and their relations are organized into hierarchies (HofC) as queries. A query connects the user with computer agents in information retrieval: a query is a formal statement of a user’s information need, and is the basis for computer agents to retrieve documents from the WWW. In every HofC-based query, concepts and their definitions are in a hierarchical structure.

- Documents are ranked according to how well they satisfy a query. Satisfaction is mapped as the activation of an HofC by documents. The activation is bottom-up, and is calculated using ordered weight averaging (OWA) operators. Normally, a concept is defined by several subconcepts and terms organized in a HofC. The activation of the HofC is the process of aggregation of the satisfactions of subconcepts and terms. Linguistic quantifiers introduced in OWA operators can be integrated with hierarchies in the HofC to satisfy more flexible information requirements.

The experiments in chapter 6 were designed to validate HOTIR. They compare HOTIR-based approaches with different linguistic quantifiers, and with other popular information retrieval approaches. Case studies presented in chapter 5 showed how a HOTIR-based approach was affected by the use of the linguistic quantifiers *some* and *most*. Mathematical definitions of these linguistic quantifiers indicate that *most* utilizes hierarchical structures in queries and *some* ignores them. We found that if one document was applied to two queries which contained the same set of concepts but with different hierarchical structures, the document tended to have different scores for satisfying the two queries when the linguistic quantifier *most* was used, and the documents got equal scores for satisfying the two queries when the linguistic quantifier *some* was used. Generally, where documents were preset only as relevant or non-relevant, the use of *some* in HOTIR was a better choice than other approaches (including HOTIR with *most*). Where documents were ranked by human experts, HOTIR with *most* showed better performance than other methods (including HOTIR with *some*). These results convinced us that hierarchical information helps to express queries more precisely. By ignoring hierarchical information, HOTIR utilizing the linguistic quantifier *some* was a more general information retrieval tool than HOTIR utilizing *most*. A general IR tool is a better option if the aimed documents are binary categorized (e.g., relevant or non-relevant). In our experiments, HOTIR utilizing *most* took hierarchical information into consideration, making it a more specific IR tool than HOTIR utilizing *some*; the linguistic quantifier *most* showed its power when the evaluation was made by comparing rankings.

Ideas, and even techniques, in HOTIR can be adapted easily by other intelligent systems to realize Semantic Web (section 2.1.2), where computers can accomplish more complicated jobs through utilizing the meanings of the contents of Web documents. HOTIR provides a complete set of processes for computer agents to identify aimed concepts from texts. For everyday life, retrieving Web documents through search engines is a common beginning for many projects: papers may be needed for a background study; financial news can be accessed to build an investment plan; reviews of a product may be required before shopping; and so on. Similarly, the identification of concepts from texts by computer agents is the basis of many different applications.

7.2 Future work

Several areas of future research are presented as:

- User interface (UI): A more sophisticated UI will help a HofC-based query better express users' interests. Queries connect users and computer agents. A query is a formal statement of a user's information needs. HOTIR uses information presented in the query to retrieve documents. One of the reasons for applying a HofC to represent queries is that it is able to contain more information – a set of concepts in hierarchical organization is more meaningful to a computer agent than a set of unordered concepts. However, in the current HOTIR-based system, users' keyword-based queries are inputs to QryProc which do not have complicated hierarchical structures. HofC-based queries that are built from these queries do not have complicated hierarchical structures either. Although the starting keyword-based query can use a new operator ">" to construct some simple relations (the concept defined by the right-side keyword is a subconcept of the concept defined by the left-side keyword, see section 6.5), and HofC-based queries can obtain hierarchical information from the knowledge base (ModOnt), it is still far from what HofC can do. The development of a UI that accepts more special operators and on which a graphical user interface (GUI) can be built. With a GUI, users could freely choose concepts of interest and organize these concepts in a way that would point the search in the desired direction.
- Natural language possessing (NLP): NLP [106] can help computer agents utilize the meanings of unstructured data. NLP systems attempt to translate human language into representations that computer agents can manipulate. Introducing ideas/techniques of NLP into HOTIR would be helpful. Currently in HOTIR, the prepossessing module in QryProc (section 4.7) removes stop words and stems words in queries entered by users. The application of NLP modules to QryProc would provide more intelligent preprocessing.
- Query expansion: Finding and applying supplementary knowledge from a knowledge base is worth further study. In HOTIR, the keywords input by users are used to retrieve "equivalent" concepts in the ModOnt.

We use a somewhat naive approach to find concepts in the ModOnt: if a keyword matches the name of a concept or matches a literal value (term) of a datatype attribute of a concept, the concept is recognized as "equivalent" to one of the keywords. After finding "equivalent" concepts, we take it for granted that the concept is equal to what the keyword represents, so all definitions of the concept in the ModOnt are added to the query. However, the corresponding concept may not be an "equivalent" concept but merely a "related" concept. Therefore, not all definitions of the concept in the ModOnt should be added to the query; a selective filter is needed here.

- Linguistic quantifier: In the thesis, two linguistic quantifiers, *some* and *most*, were implemented and studied in HOTIR. Like hierarchies, linguistic quantifiers provide possibilities for the improvement of information retrieval. Linguistic quantifiers instruct the aggregation process. In some special scenarios (purposes, popularity of the aimed concepts, etc.), some linguistic quantifiers are more productive than others. There are several other linguistic quantifiers that could be tested in HOTIR for their ability to improve the aggregation process.

Bibliography

[1] M. J. Pierre. Practical issues for automated categorization of web pages. In *Proceedings of ECDL 2000 Workshop on the Semantic Web*. 2000.

[2] Wikipedia. Web crawler, Oct 2008. URL `http://en.wikipedia.org/wiki/Web_crawler`.

[3] P. E. Black. Inverted index, Oct 2008. URL `http://www.nist.gov/dads/HTML/invertedIndex.html`.

[4] H.M.Haav and T.-L. Lubi. A survery of concept-based information retrieval tools on the web. In *Proceedings of 5th East-European conference ADBIS*, volume 2, pages 29–41. Springer Berlin, Vilnius, Lithuania, 2001.

[5] J. A. Gulla, P. G. Auran, and K. M. Risvik. Linguistics in large-scale web search. *In Book:Natural Language Processing and Information Systems*, 2553:218–222, 2002.

[6] A. Spink, D. Wolfram, M. B. J. Jansen, and T. Saracevic. Searching the web: the public and their queries. *Journal of the American Society for Information Science and Technology*, 52(3):226–234, 2001.

[7] T. Berners-Lee and M. Fischetti. *Weaving the Web*. Harper San Francisco, 1999.

[8] D. Harman. User-friendly systems instead of user-friendly front-ends. *Journal of the American Society for Information Science*, 43(2):164–174, 1992.

[9] G. Salton. *Automatic text processing: The transformation, analysis and retrieval of information*. Addison-Wesley, Reading, MA, 1989.

[10] G. Salton, E. Fox, and H. Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

[11] Wikipedia. Tf-idf, Sept. 2008. URL `http://en.wikipedia.org/wiki/Tf-idf`.

[12] M. Lan and C.-L. Tan. A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. In *Proceedings of International World Wide Web Conference*, pages 1032–1033. ACM Press, New York, NY, 2005.

[13] E. Mantanes, I. Diaz, J. Ranilla, E. Combarro, and J. Fernandez. Scoring and selecting terms for text categorization. *IEEE Intelligent Systems*, pages 40–47, 2005.

[14] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information precessing management*, 24(5):512–523, 1988.

[15] Wikipedia. Euclidean distance, Sept. 2008. URL `http://en.wikipedia.org/wiki/Euclidean_distance`.

[16] E. Greengrass. Information retrieval: a survery, Sept. 2008. URL `http://clgiles.ist.psu.edu/IST441/materials/texts/IR.report.120600.book.pdf`.

[17] M. Damashek. Gauging similarity with n-grams: Language-independent categorization of text. *Science*, 267:843–848, 1995.

[18] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[19] C. D. Manning, P. Raghavan, and H. Schutze. *Introduction to infomation retrieval.* Cambridge University Press, New York, NY, 2008.

[20] G. Antoniou and F. van Harmelen. *A Semantic Web Primer(2nd Edition).* The MIT Press, Cambridge, Massachusetts, London, England, 2008.

[21] L. A. Cunningham. Language, deals and standards: The future of xml contracts, Feb. 2007. URL `http://ssrn.com/abstract=900616`.

[22] B. McBride. Rdf primer, Aug. 2008. URL `http://www.w3.org/TR/REC-rdf-syntax`.

[23] Wikipedia. Xml, Aug. 2008. URL `http://en.wikipedia.org/wiki/XML`.

[24] T. E. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.

[25] O. Dridi. Ontology-based information retrieval: Overview and new proposition. In *Proceedings of second international conference on Research Challenges in Information Science, 2008. RCIS 2008.*, pages 421–426. IEEE, Marrakech, June 2008.

[26] R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18:183–190, 1988.

[27] L. Zadeh. A computational approach to fuzzy quantifiers in natural language. *Computers and Mathematics with Applications*, 9:149–184, 1983.

[28] R. Yager. Families of owa operators. *Fuzzy Sets and Systems*, 59:125–148, 1993.

[29] R. Yager. A hierarchical document retrieval language. *Information Retrieval*, 3:357–377, 2000.

[30] M. Maron. Automatic indexing: An experimental inquiry. *Journal of the ACM*, 8:404–417, 1961.

[31] N. Fuhr. Models for retrieval with probabilistic indexing. *Information Processing and Management*, 25(1):55–72, 1989.

[32] D. Lewis. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of SIGIR-92, 15th ACM International Conference on Research and Development in Information Retrieval*, pages 37–50. Copenhagen, Denmark, 1992.

[33] D. Lewis and M. Ringuette. Comparison of two learning algorithms for text categorization. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieal(SDAIR'94)*, pages 81–93. ACM, USA, Las Vegas, USA, 1994.

[34] D. Lewis. Naive(bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 5–15. Chemnitz, Germany, 1998.

[35] W. B. Croft. Boolean queries and term dependencies in probabilistic retrieval models. *Journal of the American Society for Information Science*, 37(2):71–77, 1986.

[36] L. S. Larkey and W. B. Croft. Combining classifiers in text categorization. In *19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 289–297. ACM, USA, Zurich, Switzerland, 1996.

[37] W. S. Cooper. Some inconsistencies and misidenfified modeling assumption in probabilistic information retrieval. *ACM Transactions on Information Systems*, 13(1):100–111, 1995.

[38] C. Apte, F. Damerau, and S. M. Weiss. Automated learning of decision rules for text categorization. *ACM Transactions on Information Systems (TOIS)*, 12(3):233–251, 1994.

[39] S. Weiss and N. Indurkhya. Optimized rule induction. *IEEE Expert*, 8(6):61–69, 1993.

[40] I. Moulinier and J. Ganascia. Applying an existing machine learning algorithm to text categorization. In S.Wermter, E. Riloff, and G. Schaler, editors, *Proceeding of Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pages 343–354. Springer Verlag, Heidelberg,Germany, 1996.

[41] H. Li and K. Yamanishi. Text classification using esc-based stochastic decision lists. In *Proceedings of CIKM-99, 8th ACM International Conference on Information and Knowledge Management*, pages 122–130. Kansas City, MO., 1999.

[42] W. Cohen and Y. Singer. Context sensitive learning methods for text categorization. *ACM Trans. Information System*, 17(2):141–173, 1999.

[43] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE Trans. on Systems,Man and Cybernetics.*, 21(3):660–674, 1991.

[44] J. Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

[45] J. Quinlan. C4.5:programs for machine learning. *Machine learning*, 16(3):235–240, 1993.

[46] rulequest. Data mining tools see5 and c5.0, Oct 2008. URL `http://www.rulequest.com/see5-info.html`.

[47] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys(CSUR)*, 34(1):1–47, 2002.

[48] Wikipedia. Overfitting, Oct. 2008. URL `http://en.wikipedia.org/wiki/Overfitting`.

[49] T. Joachims. Text categorization with support vector machine:learning with many relevant features. In *Proceedings of ECML-98, 10th European Conference on Machine learning*, pages 137–142. Chemnitz, Germany, 1998.

[50] N. Vapnik. *The nature of statistical learning theory*. Springer New York, New York, 1995.

[51] J. Kivinen, M. Warmuth, and P. Auer. The perceptron algorithm vs winnow: Linear vs. logarithmic mistake bounds when few input variables are relevant. In *In Conference on Computational Learning Theory*. 1995.

[52] H. Drucker, V. Vapnik, and D. Wu. Automatic text categorization and its applications to text retrieval. *IEEE Trans. Neutral Netw*, 10(5):1048–1054, 1999.

[53] S. T. DUMAIS, J. PLATT, D. HECKERMAN, and M. SAHAMI. Inductive learning algorithms and representations for text categorization. In *In Proceedings of CIKM-98, 7th ACM International Conference on Information and Knowledge Management*, pages 148–155. Bethesda, MD, 1998.

[54] S. Dumais and H. Chen. Hierarchical classification of web content. In *In Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263. Athens, Greece, 2000.

[55] H. Taira and M. Haruno. Feature selection in svm text categorization. In *In Proceedings of AAAI-99, 16th Conference of the American Association for Artificial Intelligence*, pages 480–486. Orlando, FL, 1999.

[56] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *In Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 487–494. Stanford, CA, 2000.

[57] T. Masuyama and H. Nakagawa. Two step pos selection for svm based text categorization. *IEICE Transactions on Information and Systems*, E87-D:373–379, 2004.

[58] P. Fu, D. Zhang, Z. Ma, and H. Dong. Svm-based semantic text categorization for large scale web information organization. In *Proceedings*

of Second International Symposium on Neural Networks, pages 931–936. Springer Verlag, Chongqing, China, 2005.

[59] V. Vidulin, M. Lustrek, and M. Gams. Training a genre classifier for automatic classification of web pages. *Journal of Computing and Information Technology*, 15(4):305–311, 2007.

[60] Y. Aphinyanaphongs, I. Tsamardinos, A. Statnikov, D. Hardin, and C. F. Aliferis. Text categorization models for high-quality article retrieval in internal medicine. *Journal of the American Medical Information Association*, 12(2):207–216, 2005.

[61] A. Anagnostopoulos, A. Broder, and K. Punera. Effective and efficient classification on a search-engine modeling. *Knowledge and information systems*, 16(2):129–154, 2008.

[62] B. Choi and X. Peng. Dynamic and hierarchical classification of web pages. *Online Information Review*, 28(2):139–147, 2004.

[63] P. Anick. Adapting a full-text information retrieval system to the computer troubleshooting domain. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 349–358. 1994.

[64] W. A. Woods. Conceptual indexing: a better way to organize knowledge, 1997. URL `http://research.sun.com/techrep/1997/abstract-61.html`.

[65] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[66] E. M. Voorhees. Query expansion using lexical-semantic relations. In *In: Proceedings of the 17th Annual ACM SIGIR conference on research and development in information retrieval*, pages 61–69. Springer-Verlag New York, Inc., New York, NY, USA, 1994.

[67] Z. Gong, C. Cheang, and L. H. U. Web query expansion by wordnet. *Lecture notes in computer science*, 3588:166–175, 2005.

[68] M. Baziz, M. Boughanem, N. Aussenac-Gilles, and C. Chrisment. Semantic cores for representing documents in ir. In *in: Proc. of 2005 ACM Symposium on Applied Computing*, pages 1011–1017. Santa Fe, New Mexico, 2005.

[69] S. Kolte and S. Bhirud. Word sense disambiguation using wordnet domains. In *Emerging Trends in Engineering and Technology, 2008. ICETET '08. First International Conference on*, pages 1187–1191. IEEE, Nagpur, Maharashtra, July 2008.

[70] Y.-B. Kim and Y.-S. Kim. Latent semantic kernels for wordnet: Transforming a tree-like structure into a matrix. In *Advanced Language Processing and Web Information Technology, 2008. ALPIT '08. International Conference on*, pages 76–80. IEEE, Dalian Liaoning, July 2008.

[71] N. Guarino, C. Masolo, and G. Vetere. Ontoseek: Content-based access to the web. *IEEE Intelligent Systems*, 14, issue 3:70–80, 1999.

[72] K. Knight and R. Whitney. Ontology creation and use: Sensus, 1997. URL `http://www.isi.edu/natural-language/resources/sensus.html`.

[73] S. E. Lewis. Gene ontology: looking backwards and forwards. *Genome Biology*, 6(1):103, 2005.

[74] I. Spasic, E. Simeonidis, H. L. Messiha, N. W. Paton, and D. B. Kell. Kipar, a tool for systematic information retrieval regarding parameters for kinetic modelling of yeast metabolic pathways. *BIOINFORMATICS*, 25(11):1404–1411, 2009.

[75] T. Stubblebine. *Regular Expression Pocket Reference*. O'Reilly Media, Sebastopol, California, 2003.

[76] D. Embley. Towards semantic understanding - an approach based on information extraction ontologies. In *In: Proceedings of the Fifteenth Australasian Database Conference (ADC2004)*, pages 3–12. Australian Computer Society, Inc. Darlinghurst, Australia, Dunedin, New Zealand, 2004.

[77] H.-M. Muller, E. E. Kenny, and P. W. Sternberg. Textpresso: An ontology-based information retrieval and extraction system for biological literature. *PLoS Biology*, 2(11):1984–1998, 2004.

[78] H. B. Styltsvig. Roskilde University.

[79] P. Cimiano, P. Haase, M. Herold, M. Mantel, and P. Buitelaar. Lexonto: A model for ontology lexicons for ontology-based nlp. In *In: Proceedings of the OntoLex (From Text to Knowledge: The Lexicon/Ontology Interface) workshop at ISWC07 (International Semantic Web Conference)*. Busan, South-Korea, 2007.

[80] M. Morneau, G. W. Mineau, and D. Corbett. Lexonto: A model for ontology lexicons for ontology-based nlp. In *In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*.

[81] J. F. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks Cole Publishing Co., Pacific Grove, CA, 2000.

[82] D. Vallet, M. Fernandez, and P. Castells. An ontology-based information retrieval model. In *In: Proceedings of 2nd European Semantic Web Conference, ESWC 2005*, pages 455–470. Springer Berlin, Grete, Greece, June 2005.

[83] O. Dridi and M. B. Ahmed. Building an ontology-based framework for semantic information retrieval: application to breast cancer. In *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, volume 2, pages 1–6. April 2008.

[84] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, C. Ursu, M. Dimitrov, M. Dowman, N. Aswani, I. Roberts, Y. Li, A. Shafirin, and A. Funk. Developing language processing components with gate, April 2009. URL `http://gate.ac.uk/sale/tao/index.html`.

[85] J. Mayfield and T. Finin. Semantic annotation, indexing, and retrieval. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(1):49–79, 2004.

[86] P. Castells, M. Fernandez, and D. Vallet. An adaptation of the vector-space model for ontology-based information retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 19(2):261–272, 2007.

[87] S. L. Tomassen. Searching with document space adapted ontologies. *In Book:Emerging Technologies and Information Systems for the Knowledge Society*, 5288:513–522, 2008.

[88] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[89] V. Snasel, P. Moravec, and J. Pokomy. Wordnet ontology based model for web retrieval. In *In: Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration*. April 2005.

[90] P. Soucy. Beyond tfidf weighting for text categorization in the vector space model. In *In Proceedings of the Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, pages 1130–1136. Edinburgh, Scotland, 2005.

[91] Wikipedia. Perron-frobenius theorem, Nov 2008. URL `http://en.wikipedia.org/wiki/Perron-Frobenius_theorem`.

[92] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49. ACM, Berkeley, California, 1999.

[93] A. Sun, E.-P. Lim, and W.-K. Ng. Performance measurement framework for hierarchical text classification. *Journal of the American Society for Information Science and Technology*, 54:1014–1028, September 2003.

[94] B. Goertzel and J. Venuto. Accurate svm text classification for highly skewed data using threshold tuning and query-expansion-based feature selection. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1220–1225. Vancouver, BC, Canada, 2006.

[95] A. C. Tantug and G. Eryigit. Performance analysis of naive bayes classification, support vector machines and neural networks for spam categorization. *Advances in Soft Computing*, 34:495–504, 2006.

[96] Wikipedia. Power iteration, May 2008. URL `http://en.wikipedia.org/wiki/Power_method`.

[97] D. Ferrucci and A. Lally. Uima: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, 2004.

[98] Y.-C. Chang and S.-M. Chen;. A new query reweighting method for document retrieval based on genetic algorithms. *IEEE Transactions on Evoluationary Computation*, 10(5):617–622, 2006.

[99] M. Kobayashi and K. Takeda. Information retrieval on the web. *ACM Computing Surveys*, 32:144–173, 2000.

[100] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *In: Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM New York, NY, USA, Pittsburgh, Pennsylvania, USA, 2006.

[101] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[102] Y. Yao. Measuring retrieval effectiveness based on user preference of documents. *Journal of the American Society for Information Science*, 46:133–145, 1995.

[103] M. Balabanovic and Y. Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[104] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, volume 1, pages 133 – 142. ACM New York, Alberta, Canada, 2002.

[105] H. Wu, H. Lu, and S. Ma. *Multilevel Relevance Judgment, Loss Function, and Performance Measure in Image Retrieval in Image and Video Retrieval*, volume Volume 2728/2003 of *Lecture notes in computer science*. Springer Berlin, January 2003.

[106] C. D. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.