A HYBRID KNOWLEDGE-BASED APPROACH TO INFORMATION RETRIEVAL

Yongun YOON, Keysun CHOI, and Gilchang KIM
Department of Computer Science, KAIST, 373-1 Kusong-Dong, Yusong-Gu,
Taejon, 305-701, Republic of Korea {yuyoon, kschoi}@esking.kaist.ac.kr

Dongwook SHIN
Department of Computer Engineering, Chungnam University, 220 Kung-Dong,
Yusong-Gu, Taejon, 305-764, Republic of Korea shin@comeng.cnu.ac.kr

From a couple of years ago, extensive research efforts have been devoted to the knowledge-based approach to information retrieval. These are usually based on thesauri that admit a variety of relations between control vocabularies called index terms. However, there have been few systems to offer sufficient functions enough to convey domain dependent knowledge as well as domain independent one in storing and retrieving information.

This paper proposes a knowledge-based approach that facilitates not only domain independent knowledge but domain dependent one to be represented in the context of information retrieval in a simple and structured manner. It can be accomplished by separating the store of the knowledge into two boxes: domain independent knowledge Box (Ibox) and domain dependent knowledge Box (Dbox). Ibox is used as a graph-based thesaurus that permits “is-a” relation between index terms. Dbox describes domain-specific information, such as synonym relationship, some term relationships other than “is-a” relation, and generation rules for complex compound phrases. Dbox is used as the front-end to Ibox that helps to convert the arbitrary terminology given by an IR user to the index terms implied in Ibox and reflects domain dependent knowledge.

The query evaluation procedure QEP retrieves the documents relevant to a query given as Boolean form, interacting with Ibox and Dbox. It, first, retrieves all related documents and takes the ones of highest rank by Rada’s algorithm. We show that Rada’s algorithm does not raise discontinuity near zero, nor counter-intuitive result in QEP, while it does in general situation.

1. INTRODUCTION

Information retrieval (IR) is a branch of information science that is concerned with the representation, storage, organization, and accessing of information items [13]. During the past decades, IR research has focused on numerical and statistical techniques, such as index term assignment, query formulation and the like. However, recently another direction has been emerged to apply knowledge engineering to IR on the assumption that important heuristics in IR could be utilized by means of knowledge engineering technique.

The characteristics of a knowledge-based IR system are usually determined by the knowledge and its representation method. There are various kinds of knowledge which can be extracted from an IR environment: (1) hierarchical relationship among terms, (2) synonym relationship, (3) domain restriction for a multiple meaning descriptor (polysemy), (4) general rules for a complex compound phrase, (5) affinitive relationships such as "part and whole" or "means and purpose", and (6) conversion rules from a non-index term to the corresponding index term. Putting all these into a knowledge base makes it extremely complex and hard to handle. Hence, most of knowledge based systems only allow to describe one or some of them. For instance, Rada et al. [12] and Kim [10] assume a hierarchical thesaurus that permits only the relation (1). JICST (Japan Information Center of Science and Technology) thesaurus [9] that comprises around fifty thousands terms and Giger’s system [6] allow the relations (1), (3), and (5). However, considering that the effectiveness of a knowledge-based approach strongly depends on how the knowledge is utilized, the aforementioned systems do not take the advantage of knowledge base fully.

The problem of having only one knowledge base mainly comes from that all the domain dependent knowledge should be included in it, as well as domain independent one. Since a domain specific knowledge is heavily dependent on its domain and disjoint from one another, the size of knowledge base increases almost linearly and the structure becomes considerably complicated, whenever
eral situation. Section 3 introduces two types of knowledge bases: Ibox and Dbox. Section 4 describes the structure of the inverted file that plays the role of interface between document set and retrieval system. Section 5 presents query processing, the syntax of a user query and how the query is processed. Finally, Section 6 discusses the contributions and further studies.

2. KNOWLEDGE-BASED APPROACHES

During the past decade, a variety of different knowledge representation schemes, for example frame-based, semantic network and procedural scheme have been studied in order to provide valuable assistance to the human search intermediary who is not an expert in the domain of interest [14].

Most frame representation languages have been developed using some version of LISP, although frame-based languages written in Prolog, like CODER [16], are beginning to appear. TOPIC [7] and ARGON [11] are two knowledge-based text retrieval systems that use frames. They have incorporated concepts from earlier frame systems. The TOPIC system focuses on document analysis, using semantic parsing to map text onto frame representation structures.

CODER [16] uses a Prolog frame representation system for representing documents and domain knowledge. In the CODER system, a factual representation language (FRL) serves as a tool for knowledge engineering and experimentation. The FRL is a hybrid AI language supporting strong typing for attribute values, a frame system, and Prolog-like relational structures. As CODER admits many different modules which can reside on different host computers and which communicate by passing messages to a blackboard structure, it is a distributed expert-based information system. FPR [9], written in Common Lisp, also uses frames to represent concepts. In FPR, concept frames, created primarily from the system’s interaction with the user, contain recognition rules to infer domain knowledge. FPR is also made up of cooperating experts centered around a blackboard/scheduler.

GRANT [4] is an expert system for finding sources of funding for given research proposals. Domain knowledge for research topics is represented as a semantic network where each node represents an interesting research topic and edge represents the semantic relation between research topics. For a given research proposal, relevant funding agencies are found by the constrained spreading activation process, a modified search algorithm.

RUBRIC [15], a commercial system based on production rules and written in Common Lisp, uses a manually built rule base to assist query construction and searching. In the representation, RUBRIC maintains a clear distinction between queries and concept knowledge, and between concept knowledge and document knowledge. This is achieved through a series of lan-
guages (the concept definition language, the evidence definition language, the belief language, and the query language), each of which allows one to express a particular subset of the knowledge.

There are another systems that offer different kinds of schemes coherently, such as Krypton [2], KL-ONE [3], and Sphinx [8]. Many of the concepts in our model are rooted in these systems. For example, Krypton and Sphinx's separation of two representation languages - a terminological one called Tbox and an assertional one termed Abox are paralleled by our separation of Ibox and Dbox for processing domain dependent and independent knowledge differently. Especially, the Ibox and Dbox in our model are very similar to the Tbox and Abox in Sphinx, respectively, in that Ibox and Tbox are graph-based components which admit "is-a" relation, and Dbox and Abox are the components employing Horn clauses.

3. TWO KNOWLEDGE BOXES

The knowledge base is composed of the aforementioned two boxes: Ibox and Dbox. Ibox is concerned of the domain independent knowledge, while Dbox deals with domain dependent knowledge. Ibox is static because domain independent knowledge is hardly changed, whereas Dbox is dynamic in that it is subject to frequent change. The outline of the proposed system comprising two boxes is depicted in Figure 1. Ibox plays the role of thesaurus consisting only of index terms, and Dbox allows to convey the relationship other than "is-a" and several heuristics. The Query Evaluation Procedure (QEP) evaluates a user query, interacting with these boxes. Finally, it accesses the document sets, via the inverted file that keeps the relationship between index terms and documents. This section discusses the details of these boxes such as how knowledge is represented. The descriptions of inverted file and QEP are found in Section 4 and 5, respectively.

3.1. Ibox

Basically, Ibox is a DAG, (Directed Acyclic Graph), in which a term is represented by a node and a relation between two terms by a directed edge. Among the various inter-term relations such as "is-a", "synonym", and "part-and-whole", Ibox only offers the "is-a" relation, since it is heavily concerned about common sense knowledge. According to the research on a comparative study of ranking by human and a computer program, the "isa" relationship is most often used by human when evaluating the relationship of documents and queries [12]. Other relations can be represented in Dbox.

Ibox offers five operators for constructing or navigating the thesaurus:

1. def_concept(C_i)
2. defisa(C_i, C_j)
3. del_concept(C_i)
4. delisa(C_i, C_j)
5. sub_concept(C_i, X)

The first operator lets the concept C_i be generated in Ibox, while the second makes a "is-a" relation from C_i to C_j. The latter two operators play the similar roles as the former two, except that they perform the deletion of concepts or "is-a" relations instead of insertion. The last one gathers all subconcepts below C_i and assigns them to X. The first four operators are mainly used for constructing a thesaurus, whereas the last one is employed for searching related terms in retrieval stage.

Figure 2 depicts an example of Ibox made by referring to JICST Thesaurus, using these operators.
3.2. Dbox

In contrast to Ibox, Dbox lets the domain dependent heuristics be represented. In our system, Dbox is used as the front-end to Ibox that helps to convert the arbitrary terminology given by an IR user to the index terms implied in Ibox and reflects domain dependent knowledge. The usefulness of Dbox is easily demonstrated, considering that it is extremely difficult for an average user to compose a query made up only of index terms without the aid of an expert. Among these heuristics, Dbox allows to represent (1) assertions for synonyms, (2) domain restriction for a multiple meaning descriptor (polysemy), (3) general rules for a complex compound phrase, (4) affinitive relationships other than "is-a", such as "part and whole", and (5) conversion rules from a non-index term to the corresponding index term.

These heuristics are described by the five operators, respectively as:

1. def.syn($C_1$, $C_2$),
2. def.poly($C_1$, $C_2$),
3. def.compound($C, (C_1, \ldots, C_n)$).
4. def.affin($C, (C_1, \ldots, C_n)$), and
5. conv.index($C_1$, $C_2$).

All these are transformed into Horn clauses in Dbox by the rules below, respectively and computed by an extension of SL$\subset$ resolution used in Prolog. In these operators, if an index term appears as the first argument of the first, the third, or the fourth operator, the search of the term should be conducted in Ibox as well as that of the second argument, because every index term related to a query in Ibox should take part in the retrieval. Hence, when an operator among the three is issued by a Dbox constructor, its first argument is investigated in Ibox before the transformation. If it is found in Ibox, the transformation should include the clause that performs the corresponding search in Ibox. Otherwise, it need not do so. Therefore, the following transformation includes the Horn clauses embraced by { and }, only if an index term appears in the corresponding first argument.

Note that, in the fourth operator, terms in the second argument are connected by or operator (\texttt{;}), which means that they are unrelated with one another, whereas each one has relation with the term in the first argument.

1. The operator def.syn($C_1$, $C_2$) is transformed into the Horn clauses:
   { term($C_1$, $X$) :- descriptor($C_1$, $X$). }
   term($C_1$, $X$) :- descriptor($C_1$, $X$).
2. The operator def.poly($C_1$, $C_2$) is transformed into:
   term($C_1$, $X$) :- descriptor($C_1$, $X$).
3. The operator def.compound($C, (C_1, \ldots, C_n)$) is transformed into:
   { term($C_1$, $X$) :- descriptor($C_1$, $X$). }
   term($C_1$, $X$) :- term($C_1$, $X$), \ldots, term($C_n$, $X$).
4. The operator def.affin($C, (C_1, \ldots, C_n)$) is transformed into:
   { term($C_1$, $X$) :- descriptor($C_1$, $X$). }
   term($C_1$, $X$) :- term($C_1$, $X$), \ldots, term($C_n$, $X$).
5. The operator conv.index($C_1$, $C_2$) is transformed into:
   term($C_1$, $X$) :- descriptor($C_1$, $X$).

Now let us illustrate those taking an information retrieval example in Figure 2. As the Ibox in Figure 2 depicts only "is-a" relation among the index terms, other relations or non-index terms should be prepared in Dbox. These are written by using the operators, as follows;

1. def.affin(information retrieval, (thesaurus; indexing; recall precision)).
2. conv.index(IR, information retrieval).
3. conv.index(I.R., information retrieval).
4. def.affin(thesaurus, (automatic classification; dictionary)).
5. def.poly(indexing, indexing[documentation]).

which is transformed into Horn clauses;

1. term(information retrieval, X) :-
   descriptor(information retrieval, X).
2. term(information retrieval, X) :-
   descriptor(thesaurus, X); term(indexing, X);
   term(recall precision, X).
3. term(IR, X) :-
   descriptor(information retrieval, X).
4. term(I.R., X) :-
   descriptor(information retrieval, X).
5. term(thesaurus, X) :-
   descriptor(information retrieval, X).
6. term(thesaurus, X) :-
   term(automatic classification, X);
   term(dictionary, X).
7. term(indexing, X) :-
   descriptor(indexing[documentation], X).

A concept is represented using term/2 predicate whose first argument describes a domain specific term and the second is the variable that contains the corresponding set of documents as its result. descriptor/2 predicate implies that its first argument is used as an index term in Ibox and the second is the same variable as that of term/2 predicate. As are easily seen, Horn clauses in Dbox attempt to convert the terms given by an IR user to the corresponding index terms in Ibox and entail the
A hybrid knowledge-based approach to information retrieval

search of Ibox. Note that the operator def-affin is transformed into two Horn clauses one of which searches the first argument in Ibox, whereas def-poly is converted into only one clause.

As Dbox is dynamic, it also allows the deletion capability, through which one is able to update the knowledge in the mixture of the operators stated above.

6. del_concept(C).

As the five operators stated above, it is also transformed into the corresponding Horn clause:

6. The operator del_concept(C) is transformed into:
abolish(functor name of C, arity of C),
and it is immediately executed, so that all the clauses related to C are deleted from Dbox.

4. THE RELATIONSHIP BETWEEN INDEX TERMS AND DOCUMENTS

IR systems usually have a file structure to contain the information on the index terms that appear in a document, or the frequencies of occurrences. For this purpose, inverted files are used in almost every commercially available IR system [13]. The inverted file is a file structure that identifies all documents associated with a given term; that is, for each index term a list of documents indexed by that term is carried in the index [13]. Some systems include certain features, such as frequency feature to enhance the function of the inverted file. The frequency information is represented as the weight of a term that implies the importance of the term. Since the frequency of the usage of a given term correlates with the importance of that term, the weight information usually plays the same role as frequency information.

The proposed system uses a simple inverted file for representing the relationship between the indexed terms and documents without frequency information. Hence, every term is assumed to be equally important. Figure 3 describes a typical inverted file of the proposed system. As shown in this figure, an access to the inverted file for term 1 identifies documents 1 and 3 for retrieval. Generally, if search_index(C,X) is given in query evaluation, where C is an index term and X is a variable, our system searches the inverted file and binds X to a set whose member is represented as a pair (Num,T), where Num is a document number and T is the list of concepts that involves C.

5. QUERY PROCESSING

User queries may be available as natural language formulation or other simple forms, like Boolean expression. To allow natural language formulation, an extensive scope of language understanding is needed. On the other hand, the latter form does not require language understanding. As they usually allow simple operators like &and, &or, and &not, queries can be interpreted easily.

5.1. Query Formulation

The proposed system adopts the Boolean logic to construct queries consisting of a variety of terms using the Boolean operators &and, &or, and &not. These operators are led by & to distinguish from the usual words such as and, or, not.

A legitimate query is a valid expression defined recursively as follows:
1. A term is a valid expression.
2. If E is used as an index term, &not E is a valid expression.
3. If E and F are valid expressions, E &and F, and E &or F are valid expressions.
4. A legitimate query is generated by applying the above rules.

A legitimate query allows an arbitrary combination of &and &or &not, except that &not should be coupled with an index term. If &not appears with an arbitrary expression, even non-index term in a query, it becomes extremely difficult to apply existing ranking algorithm to our query evaluation framework. Any legitimate query can be converted into the minimal disjunctive normal form (MDNF). As the MDNF is easier to deal with than the random query in the evaluation, a legitimate query is, first, converted into the MDNF. Next, the MDNF is again transformed into the standard form by the following rule:

- For each term, q, it is transformed into term(q, X), in which X is the variable that is to contain a set of related document numbers.
- &and, &or, and &not are transformed into ",", and ",", and ",", respectively.

Figure 3. An example of the inverted file in our system
For example, the standard form of the query "indexing & and thesaurus" is "term(indexing, X), term(thesaurus, X)", while that of "document retrieval & or fact retrieval" is "term(document retrieval, X); term(fact retrieval, X)".

5.2. Query Evaluation

The aim of the Query Evaluation Procedure (QEP) is to retrieve the relevant documents in the aid of Ibex, Dbox and the inverted file. It also includes the ranking procedure developed by Rada et al. [12] that ranks the selected documents according to a certain criterion and takes some of highest ranked documents as a result. Rada's algorithm is known to be simple and produces similar results as people do.

If a query is given, it is first, transformed into the standard form. The proposed system assumes the separability condition [1], which guarantees that query constituents can be evaluated one by one and combined afterwards, regardless of their context. Based on this condition, QEP computes the primitives in a query, one by one and combines the results.

Now let us present the algorithm of QEP in detail.

Procedure QEP(S, N, B, D)

S: the standard form of a query
N: the number of documents to be retrieved
B: the set of documents in which the retrieval is conducted
D: the designated Dboxes
Output: a list of document numbers retrieved as the result of computation

1. Do the following until there is no subgoal of the form $\text{term}(C, X)$ in S.

   (a) Select a subgoal of the form $\text{term}(C, X)$ in S,

   (b) If there is a clause whose head is unifiable with $\text{term}(C, X)$ in Dboxes in D, then begin

      i. Find all such clauses $H_i : B_i$, where $1 \leq i \leq m$.

      ii. Replace $\text{term}(C, X)$ by the disjunction of bodies, i.e., $B_1 \cdots B_m$.

    end

   else replace $\text{term}(C, X)$ by $\text{descriptor}(C, X)$.

2. Change the final form of Step 1 into the MDNF form and let it be $S_n$.

3. For each subgoal, $\text{descriptor}(C_i, X) \in S_n$, do the following.

   (a) Call sub_concept($C_i, C$) in Ibex.

   (b) Replace $\text{descriptor}(C_i, X)$ by the disjunction of $\text{search_index}(C_j, X)$, where $C_j$ is in $C$.

4. In the resulting query, remove redundancy, i.e. reduce the form of "q,q" and "q;q" to "q".

5. Do the following for each $\text{search_index}(C, X)$ in the resulting form of Step (4), that is, the form having only $\text{search_index}(C, X), ",", \text{and } ","$.

   (a) Select a subgoal of the form $\text{search_index}(C, X)$.

   (b) Retrieve all the document numbers related to $C$, referring to the inverted file and assign the set to $X$.

6. If there is a negated form "\neg q", do the following

   (a) If it is "conjunctive with" another goal, $q_j$, i.e. of the form "\neg q, q_j" or "q_j, \neg q", do set difference operation of the set assigned to the second argument of $q_i$ from that of $q_j$.

   (b) If it is not conjunctive with another goal, i.e., solely of the form, \neg $q_i$, do set difference operation from $B$.

7. Do the following nondeterministically for the operators ",," and ",;" in S.

   (a) For ",," do join operation for the sets assigned to the occurrences of $X$ in $\text{search_index}(X, C)$.

   (b) For ",;", do union operation for the sets assigned to the occurrences of $X$ in $\text{search_index}(X, C)$.

8. If $N = \infty$, then return $X$.

9. For the resulting set $X$, each member of which is represented as the pair $(\text{Num}, C)$, where $\text{Num}$ is the document number and $C$ is the conjunction of index terms related to the document, do the following.

   For each member, calculate the distance between $S_n$ and $C$ by the following equation:

   \[
   \text{Distance}(C_1, \cdots, C_k, C) = \min \text{ Distance}(C_n, C),
   \]

   \[
   \text{Distance}(C_1, C_2) = \frac{1}{m} \sum_{i = 1}^{m} \sum_{j = 1}^{k} d(t_i, t_j),
   \]

   where

   \[
   d(t_i, t_j) = \text{minimum number of edges in a path from } t_i \text{ to } t_j \text{ in Ibex}.
   \]

   \[
   \text{Distance}(C_1, \neg C_2) = \text{Distance}(C_1, C_2^{-1}),
   \]

   where

   $v^{-1} = \{w \in V | \text{ Distance}(v, w) = \max_{u,v} \text{ Distance}(u, v)\}$.  

Let us illustrate QEP, taking an example. Suppose that a query "I.R." is given, in the presence of Ibox and Dbox in Section 3. First it is converted into standard form "term(I.R., X)" and begins to work with Dbox. Referring to Dbox, as there exists only one clause that can be unified, i.e., the fourth one, it is changed into the body part of the clause and the initial query becomes "term(information retrieval, X)". Next, as this goal can be unified with the first two clauses, it is replaced by the disjunction of body parts, i.e., "descriptor(information retrieval, X); term(thesaurus, X); term(indexing, X); term(recall precision, X)". Now supposing that the goal "term(recall precision, X)" is selected, as it does not have unifiable clauses, it is directly replaced by "descriptor(recall precision, X)" by Step 1-b). Continuing this way, the query ends in "descriptor(information retrieval, X); descriptor(thesaurus, X); descriptor(automatic classification, X); descriptor(dictionary, X); descriptor(indexing[documentation]); descriptor(recall precision, X)" in Step 2. Step 3 searches all subconcepts of each resulting index term in Ibox, which leads to "search_index(information retrieval, X); search_index(retrospective search, X); search_index(document retrieval, X); search_index(fact retrieval, X); search_index(data retrieval, X); search_index(chemical structure search, X); ...". From this query, Step 5 accesses the inverted file one by one and gathers the set of document numbers. Finally Step 7 does join or union operation for "",";" or "",";" respectively and entails the result.

In Step 6, QEP evaluates the negation by set difference operation. If the negation is conjunctive with another concept, i.e., of the form $C_1 \wedge \neg C_2$, in 5-a), the operation is simply done by deleting every concept of $C_2$ from $C_1$. However, if the negation is not, as in 5-b), the deletion should be performed from the whole set $B$, which usually carries an enormous amount of computation. It can be improved by taking a rather small set that only contains one's domain of interest by performing the goal, at the beginning stage of retrieval:

$$B = QEP(q, \infty, Universe),$$

where $q$ is the most general query among those of interest and $Universe$ represents the whole document sets.

If $\infty$ is given as the value of $N$, QEP returns the set of all document numbers related to the query, without cutting the low ranked documents. Hence, after taking this, a user is able to use it as the base set of QEP, until his interest is shifted to another domain. For instance, supposing that a user is going to retrieve documents related to computer domain, he can, first, narrow down his domain of interest, by issuing a query, "$B = QEP(computer, \infty, Universe)". Next, by using $B$ as his base set until his interest is shifted, he can improve the response time of his query that involve negation.

Another thing to note is that QEP adopts a ranking algorithm developed by Rada et al. [12], which is characterized by simplicity and satisfaction of metric properties, such as zero, symmetry, positive and triangle inequality property. Even though, in general, it reveals discontinuity near zero and counter-intuitive results, it does not invoke these problems in QEP.

For instance, the first problem occurs when considering the distance of the query $(c_0, c_{-3}, c_8)$ and a related document represented by $(c_0, c_{-1}, c_1)$.

$$\text{Distance}((c_0, c_{-3}, c_8), (c_0, c_{-1}, c_1)),$$

in the presence of the thesaurus in Figure 4. In Rada's algorithm, the distance becomes larger as $i$ goes from one to four, but when $i$ reaches five, it suddenly drops to zero. However, in QEP, the retrieval stage already cuts off the documents having higher concepts than $c_{-3}$ and $c_8$, the ranking algorithm never computes the distance from $(c_0, c_{-1}, c_1)$, where $1 \leq i \leq 4$, which prevents the discontinuity near zero.

Secondly, a counter-intuitive result takes place, supposing that a query $q = (c_1, c_2, c_3, c_4)$ is compared with the documents represented by $d_1 = (c_1)$ and $d_2 = (c_1, c_2, c_3, c_4)$, respectively, in the given thesaurus below. Rada's algorithm produces the counter-intuitive result that $d_1$ is closer to $q$ than $d_2$ is, because the distance from $d_1$ is lower than from $d_2$. On the other hand, in QEP, the problem does not occur, because the retrieval stage does not take the document $d_1$, which removes the computation from $d_1$. 

In Figure 5, the discontinuity problem is shown. However, in QEP, the retrieval stage already cuts off the documents having higher concepts than $c_{-3}$ and $c_8$, the ranking algorithm never computes the distance from $(c_0, c_{-1}, c_1)$, where $1 \leq i \leq 4$, which prevents the discontinuity near zero.
6. CONCLUSION AND FURTHER STUDIES

This paper proposes a knowledge-based approach to information retrieval that utilizes domain dependent information as well as domain independent information. Although these informations are also useful in retrieving, or storing information, their characteristics differ from each other. That is, as domain independent information is usually common to everyone, it is hardly considered as heuristics. Furthermore, once it is collected, it is rarely changed, independently of domain and the knowledge of a user. On the other hand, most heuristics have domain dependent nature. They tend to change, dependently on domain and the degree of the knowledge of a user. Hence, to utilize these two kinds of knowledge effectively, we argue that they should be treated differently from each other.

In this paper, domain independent information is stored in Ibox as a DAG (Directed Acyclic Graph) that has index terms as nodes and "is-a" relation as edges. In contrast, domain dependent information is preserved in Dbox that has information as Horn clause form. As the information in Dbox is easily changed, Dbox allows one to change, add, delete the information freely. Dbox is used as the front-end to Ibox that helps convert the arbitrary terminology given by an IR user to the index terms implied in Ibox and reflects domain dependent knowledge. The effectiveness of Dbox is easily shown, considering that it is considerably difficult to make a query composed only of index terms by an average user without the aid of an expert.

The query evaluation procedure QEP retrieves documents relevant to a query given as Boolean form, interacting with Ibox and Dbox. It, first, retrieves all related documents and takes highest ranked ones by Rada's algorithm. We showed that Rada's algorithm does not raise discontinuity near zero, nor counter-intuitive result in QEP, while it does in general situation.

We are now implementing the whole system in Sixtus Prolog and expecting to finish it several months later. Even though the current system permits only Boolean queries, we believe that, in principle, it can accept natural language queries in some degree because Dbox is able to reason some amount of natural language processing. It can be achieved if the available codes written in Prolog that parses natural languages are combined in Dboxes.

REFERENCES