A Multi-Agent Distributed Retrieval System

by
J. Nie, N. Anquetil and J. Vaucher
Dep. d'informatique et r.o.
Université de Montréal
CP 6128, Station "A", Montréal H3C 3J7
Email: {nie, anquetil, vaucher}@iro.umontreal.ca

Various Information Retrieval techniques have been developed in the past, but none of them is perfect in every case. The MARS experimental system was built to test the integration of multiple retrieval methods in a single system in order to offer users more flexible and complementary retrieval tools. The retrieval methods integrated in MARS include not only the classical methods, but also more intelligent ones such as conceptual retrieval based on the use of a Thesaurus, semantic retrieval and conceptual browsing. The integration is facilitated by the highly distributed and flexible architecture of MARS.

1. Introduction

In most Information Retrieval systems, only one query evaluation method is implemented. However, experience shows that no single classical evaluation method is perfect. An evaluation may be suitable in some circumstances and not in others. For example, compared to vector space evaluation, pure Boolean evaluation of AND-expressions is often too strict; but if a document satisfies such a query, it is probably quite relevant. So in the case where there are many potential relevant documents, it may turn out that this method is the more effective. Based on these observations, a trend in Information retrieval is to offer various evaluations in a single system (for example I^3R [Croft and Thompson 87]). A user may thus choose the most appropriate evaluation according to his requirements and the characteristics of the corpus.

These methods, however, are often thrown together without proper integration. Users must re-submit several similar queries for different evaluations. Furthermore, it is often impossible for a casual user to know the properties of different evaluations in order to make the right choice. This observation has been confirmed during several tests of our earlier system by real users. Thus different evaluations should be integrated in a system that allows interaction among them and the system presented here, MARS (Multi-Agent Retrieval System), is an experimental platform designed to test, compare and integrate various information retrieval techniques. It was developed as part of "IT Macroscope", a large-scale collaborative project involving a dozen major companies and two government ministries under the management of the DMR group. This project aims to produce the next generation of software tools and techniques required to support the development of very large information systems. During the initial phase of the Macroscope, several university initiatives were sponsored to explore the practical feasibility of various technologies.

In our case, the general area of research was "reutilisation" with special emphasis on information retrieval of the various documents created during a large project (and particularly design documents as opposed to code). The experimental corpus was provided by one of the industrial partners: it consisted of its corporate data dictionary and several
design models developed with the Excelerator® CASE tool. The language used for the texts was French. A typical document, an "element" from the data dictionary, is shown below. The corpus contains over 5,000 documents and, with indexes, it was estimated that the data would amount to about 50 Mbytes.

<table>
<thead>
<tr>
<th>ELE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Nom_Donnee</td>
<td>ZCBDPT</td>
</tr>
<tr>
<td>Modifie_Par</td>
<td>ADMIN. DONNEES</td>
</tr>
<tr>
<td>Date_De_Modif</td>
<td>00000890609</td>
</tr>
<tr>
<td>Numero_Changement</td>
<td>00000000</td>
</tr>
<tr>
<td>Date_Ajout</td>
<td>000000000000</td>
</tr>
<tr>
<td>Dernier_Projet</td>
<td>Tempo Daniel Dut</td>
</tr>
<tr>
<td>Definition</td>
<td>MONTANT-GRAND-TOTAL-DENOMINATION-BILLET-PRE-PACK-GUICHET</td>
</tr>
<tr>
<td>No_Car_Gauche</td>
<td>00000010</td>
</tr>
<tr>
<td>No_Car_Droite</td>
<td>0000</td>
</tr>
<tr>
<td>Lib_Franco_Court</td>
<td>FORME-COURTE</td>
</tr>
<tr>
<td>Lib_Franco_Long</td>
<td>FORME-LONGUE</td>
</tr>
<tr>
<td>Genre_Cle_Cha</td>
<td>CHA</td>
</tr>
<tr>
<td>Format_N_X_A</td>
<td>N</td>
</tr>
<tr>
<td>Service_Prop</td>
<td>GESTION DE L'ENCAISSE</td>
</tr>
<tr>
<td>Oblig_Opt_Conc</td>
<td>GENEREE PAR LE SYSTEME</td>
</tr>
<tr>
<td>Description</td>
<td>CETTE VARIABLE CONTIENT LE GRAND TOTAL DE TOUTES LES DENOMINATIONS DES BILLETS POUR LES GUICHETS AUTOMATIQUES</td>
</tr>
</tbody>
</table>

(Translation of the description: This variable contains the grand total of all bill denominations for automatic teller machines).

The project involved 8 researchers from three universities\(^1\), each with strong opinions as to how to best approach information retrieval and navigation. The methods proposed included:

- classical retrievals by keywords with various weighing schemes,
- retrieval by concept through use of a Thesaurus,
- semantic matching of queries and documents,
- browsing via concept-lattices built automatically from the documents and
- graphical & iconic interfaces.

This divergence of opinions coupled with the geographic dispersion of the participants and the wide variety of languages and software used for development meant that integrating the various approaches was not an obvious task; yet there was much to be gained by collaboration. First, there was the problem of the volume of data (large in terms of our available storage resources especially when various indexes are added). Shared access to unique copies of the data and the indexes would benefit everyone. The solution was to design a loose distributed architecture of collaborating agents that would enable researchers to share resources yet work independently.

In what follows, we start with a description of the distributed architecture, then we present several of the retrieval agents. Then, we show how a "smart" interface can combine the abilities of several agents.

---

\(^1\) G. Laplame, J. Nie, J. Savoy, and J. Vaucher from the Université de Montréal, R. Godin and R. Missaoui from UQAM and G. Mineau from Laval University
2. System architecture

The system was designed as a client-server architecture. The important aspects which contributed to its success were the following:

- telecommunication facilities to enable distributed agents to exchange messages
- standards for message formats and server operations
- a name server so that agents can locate each other
- appropriate database servers
- security considerations

Communication between the various agents of the system is accomplished via messages exchanged through UNIX sockets using the Internet UDP protocols. The basic operation is sending a message (a text of arbitrary length) to an INTERNET address. The messages are sent as datagrams; long messages (up to 40 Kbytes were observed) are split into a sequence of datagrams and sent according to a simple protocol: the receiver acknowledges each sequenced datagram while rejecting messages from other sources with a BUSY indication. So far, reliable transmission has not been a problem and we have not had to implement error checking or recovery. On top of this basic communication layer, we added interfaces for the various languages used on the project: C, Smalltalk, Simula and Prolog.

The second element in the architecture was the specification of standards and conventions. For the format of messages, our specification is loosely inspired by the ASN.1 standard. Messages can be records (or trees) of arbitrary complexity. Each data element (including the message itself) is sent as 3 parts: a length, a type and finally the element itself. All added parts (length and type) are sent as readable text. Simple data types include: integer(i), real(x), text(i), Prolog variable(v); complex types include list(S) and record(M). For complex types, the message includes the number of constituent elements. The command "set(DB1, 128.99.10.1)" would be encoded as:

"26M3 3tset3tDB111t128.99.10.1"

which means that the message is of length 26 and of type "M", and it is composed of 3 constituent elements: "3tset", "3tDB1", and "111t128.99.10.1".

The third element of the architecture is the NAME-SERVER, the only component in the whole system which has a fixed and known address (Host & Port). This server operates as a LISP property list and responds to three basic messages:

- SET( NAME, VALUE [, password])
  - which adds the couple Name=Value to its property list. For clients from a different host, a password is required.
- GET( NAME)    - which returns the Value associated to Name - or ERREUR, if Name is unknown.
- LIST( PREF )  - returns all Name=Value couples whose Name starts with PREF

Names and Values are arbitrary texts. The following extract from LIST( "" ) shows the formats used for addresses.
The fourth component of the architecture is a flexible database system compatible with the telecommunication infrastructure. For this we adapted SIMDBM, a demonstration database system written in Simula [Mäkiälä 1975]. The system handles records with a known structure but flexible size coded similarly to our messages. Records are retrieved by Key and Type. Record schemes are stored in the database and can be accessed like other records. The system also supports "M to N" relationships between records. We modified SDB, a SIMDBM query program, to act as a server program, registering its database with the name-server and doing IO for all usual user operations via TCP messages instead of the keyboard.

An important final point is security. Whenever useful data is put on the INTERNET, there is always the possibility of its being found by hackers systematically scanning network addresses. To discourage this, all our databases are protected by password and login is required. The password on name-server SET operations is meant to discourage Trojan Horse techniques registering a fake server to obtain passwords. The most important technique that we have adopted is that of activity logs: all important messages received by any server are stored (with time stamp and sender address) in LOG files that we periodically scan. This is not only good for security, it is also useful for gathering experimental data on user requests.

![Diagram](image_url)

Figure 1: MARS - Information retrieval system
The MARS system is shown in Figure 1. The double vertical line in the center of the drawing represents the INTERNET backbone. The main document database is shown at the top left. There are also several other databases (shown on the right) connected to the network; the more active agents are shown on the left.

3. Retrieval agents

Mars incorporates several standard retrieval methods as well as more innovative techniques. First the documents can be accessed by type and key, in the same way they were retrieved in the original CASE system. Furthermore, to ease exploration, whenever a partial key or a wrong key is specified, the system returns a short list of the closest existing keys.

Secondly, several variations of weighed keyword retrieval are available. These all use the same index database whose records use index terms as keys and contain a list of documents (specified by key and type) along with a weight which combines term frequency within each field of the document with an importance rating for the fields. In the "ZCBDPFT" example shown in the introduction, "definition" rates as 2, "description" rates as 1, fields with readable data like "Service_Prop" rates as 0.5. The number of references can vary from one to several thousands and the index database must be able to store records of arbitrary lengths. The database weights are not adjusted for the term frequency in the database; this is done as needed by the various algorithms. At present, the classical retrievals implemented include extended Boolean evaluation and vector space evaluation with various weighing schemes.

A very successful alternative to the use of weighed terms is the use of weighed concepts. Until now, thesauri used for IR were always automatically constructed by machine based on co-occurrences of terms. However, various experiments have shown that such thesauri do not improve system performance [Spark Jones 91]. In our opinion, this is due to the fact that term co-occurrence does not always coincide with semantic similarity. This leads to a gap between relationships established in the thesaurus and real relationships. In contrast, manually made thesauri are often of higher precision (although not exhaustive), and are more and more available in electronic form. So the use of man made thesauri in IR is a subject that is worth serious investigation. For this, we use a Thesaurus based on the hierarchical categories of the French Larousse [Péchoin 91]. The translation from a sequence of words to a sequence of concepts is carried out by a special Mars agent which returns either all concepts for a given term or the best concept according to the context. Each concept is identified by its category number in the hierarchy. In this way "client" may become [827,20]. Translation is used both for the descriptions on indexing and for queries on retrieval. All routines used for term retrieval are used for concept retrieval; the difference is that queries are first translated by the Thesaurus and that a concept index database is used. Compared with keyword retrieval, conceptual retrieval offers more flexibility to extend or restrict the retrieval by replacing a concept in a query by a super-concept or a sub-concept, as well as by a neighboring concept.

We now consider two more innovative methods also imbedded in Mars: semantic matching and browsing a lattice of conceptual clusters.

3.1. The semantic approach

Traditional IR systems based on the concept of "keyword" do not provide a high performance. Their precision-recall ratios are always around 20-30%. This is especially problematic in software reuse environments where the objective is more to find the most
relevant object than to retrieve all relevant objects. This suggests that a more accurate retrieval mechanism should be studied. The objective of a semantic-based IR is twofold:

1) to recognize the semantic relations among descriptors expressed in documents; and
2) to extend the retrieval operation to those documents that are described by different but related descriptors.

To illustrate, let us consider a document describing "the code of the company of a client". A classical system represents this as "code AND company AND client", whereas a semantic-based system would tend to recognize the relations between the words. This leads to a representation like "attribute_of(code, company) AND works_in(client, company)". This representation is more accurate than the classical one. If a query is also represented in this way, the comparison between document and query will rule out non-relevant documents such as those about "the code of clients of a company", "the code of the company which is client", etc. Thus precision will be increased.

On the other hand, "company" is not the only descriptor corresponding to the concept "company". Other descriptors such as "enterprise", "corporation", "organization", "establishment", etc. express a similar meaning. The relations among these words are part of domain knowledge. The second aspect of semantic-based retrieval is to exploit the domain knowledge during retrieval operation so that for a given query, documents described with different but related descriptors will also be considered. This will increase the recall ratio.

Although the principle of semantic-based retrieval may be easily accepted, its implementation remains difficult due to its complexity and domain dependency. Some alternative approaches based on syntactic analysis have thus been proposed [Fagan 88, Lewis and Croft 90]. Although the complexity and domain dependency are reduced, these approaches do not show improvement on system performance. In our opinion, the radical solution remains in a semantic approach. This kind of approach is practiced in limited specific domains. Banking is one of such domains in which both the vocabulary and semantic ambiguity are limited.

The semantic approach is particularly suitable in software reuse environment in which, unlike in general IR environments, the retrieval is highly precision oriented. This has been demonstrated in several test sessions of our system by real users using the Boolean and extended Boolean keyword approaches. Typical queries would provide very long lists of documents (often more than a hundred). The users, however, look at only the first few documents (less than 10). If there is no relevant document among the first ones, they begin another search with an alternative query2. This shows that it is useless to include in the result all the potential relevant documents, but just the most relevant ones. In other words, retrieval should be precision oriented; semantic retrieval fits this requirement.

A semantic approach involves two things: 1) the choice of a semantic representation of both documents and queries and 2) an algorithm for document-query comparison. The basic ideas described in this semantic approach have been successfully experimented in the RIME project developed in the Laboratoire de Génie Informatique de Grenoble in France [Berrut 90, Chiaramella and Nie 90]. We adapted this approach to our application context.

---

2 This behaviour was observed from the analysis of the LOG files of interactions.
Semantic representation

In IR environment with many documents, a representation oriented solely towards accuracy is not practical because the representation will be too complex. A suitable representation should be a compromise between accuracy and simplicity.

For our study, we have chosen a tree structure in which a node is a semantic relation and a leaf is a basic concept. This structure is simple to manipulate and at the same time rich enough to express interesting parts of document contents. For example, the phrase "the code of client's company" will be represented as follows:

```
attribute_of
   code
     where_works
       company
       client
```

This tree should be read as: the code which is an attribute of company where_works client.

It has been observed that any business domain may be specified by a limited set of interconnected aspects [Zachman 82]. This observation implies that any word in a specific domain has a restricted meaning. Semantic ambiguity is thus rare. A derived result of this observation is that we can classify the concepts of a specific domain into interconnected classes. Thus our approach consists of:

- grouping the concepts of the corpus into classes,
- and then extracting the relations that may exist among the classes.

This analysis results in a set of rules that describe the application domain. Here are some of the classes and relations that we extracted:

```
class
person
attribute
establishment

concept
client, spouse, ...
address, code, ...
company, establishment, government, ...

Relations
works_in
attribute_of

Classes
person
establishment
attribute
establishment
```

During the semantic analysis, these rules are used in a way very similar to the syntactic rules in syntactic analysis. The process which interprets a sentence into a semantic representation is as follows:

```
sentence/phrase \rightarrow morphological/lexical analysis \rightarrow syntactic analysis \rightarrow semantic analysis
```

The morphological/lexical analysis recognizes each word form in a sentence or phrase, then associates syntactic and semantic information with it. The syntactic analysis identifies the syntactic structure for a phrase or sentence. Finally the semantic analysis, based on
domain rules, maps a syntactic structure into a semantic tree (for more detail, see [Nie et al. 93]). Both documents and queries are analyzed with the above process, creating a semantic tree for them.

We adopt the bottom-up fashion in both syntactic and semantic analyses, because texts in our corpus are not always grammatically correct sentences. Bottom-up parsing allows maximal analysis of a sentence or phrase even if the whole is ill-formed. This robustness is critical in practice.

A disadvantage of the techniques presented here (compared to classical IR methods) is that additional information is required: mainly, a dictionary containing both syntactic and semantic information for all the important words in the corpus. The gain of the semantic evaluation is proportional to the covering of the domain vocabulary by the dictionary. In the extreme case where the dictionary is empty, we return to a keyword retrieval.

**Comparison between document and query**

The objective is to see if the query tree is "implied" in a document tree. "Implied" means that the query tree has a same or more generic meaning than the document tree. If a query tree is implied by a document tree, then the corresponding document has a very high potential to be relevant.

There are two cases of implication. The first case is that the query tree is a "sub-tree" of document tree. The definition of "sub-tree" is slightly changed from the classical one in order to take into account the particularity of our trees. We do not give the formal definition here, but just show an example. Among the following trees, the tree (1) is a sub-tree of tree (2), because the former may be extracted from the latter. However, the tree (1) is not a sub-tree of the tree (3) which corresponds to the phrase "the code of a client who works in a company".

```
(1)  
attribute_of
  code
  company

(2)  
attribute_of
  code
  type
  company
  public

(3)  
attribute_of
  code
  works in
  client
  company
```

The second case of implication relies on the existence of a hierarchy of concepts. For example, the concept "company" is implied by more specific concepts such as "government". Thus, the tree (1) is also implied by tree (4).

```
(4)  
attribute_of
  code
  government
```

Such specificity relationships among basic concepts may be obtained from the Thesaurus module.
3.2. Hierarchical conceptual graph

In this section we describe the conceptual graph agent. The originality of this part of the Mars system lies in the construction of the graph, whereas the browsing through it is more classical: basically following the links in an entity relationship database.

We now describe a method for classifying documents into a conceptual graph. This method automatically finds concepts from the descriptions of the documents to be classified. It then sets up these concepts into a hierarchical graph and arranges the documents into it.

Finding out the concepts

The concepts are discovered by clustering, the action of grouping together objects that share some feature. Typically, the members of a group—a cluster—are more similar to each other than to the members of any other cluster. The most popular clustering methods are the hierarchical ones. They consider each object as a singleton cluster and iteratively gather clusters (the two most similar at the moment) into larger clusters until only one remains.

We define a concept as a set of words that can be found together in the natural language description of a components, providing that they occur in the same sentence. This definition is similar to the Lexical Affinity (LA) described in [Maarek 91]. Maarek defines LAs as pairs of words separated by at most five words in a single sentence. To insure better results, she only considers open-class words, i.e. nouns, verbs, adjectives and adverbs.

Our concepts are different in that:

- we do not limit the distance between the pair to five words,
- we do not limit the number of words to two. A concept may contain from one to as many open-class words as are contained in the longest sentence.

Maarek selects the most important LAs by weighting them and comparing the weights to a fixed threshold. By using a clustering algorithm, we avoid the unpleasant and difficult task of choosing such a threshold. Moreover, there exists clustering algorithms, known as hierarchical clustering algorithms, which simultaneously arrange the discovered clusters into a binary tree, a feature which significantly reduces the task of setting up the concepts into a hierarchical graph. The counterpart is that the complexity of Maarek's algorithm is linear in the number of words to process, whereas ours is quadratic. We must therefore pay particular attention to the number of words in each documents. This means that, with a thousand documents to classify, only a few sentences (less than 20 open-class words) will be considered for each one. We intend to suppress these limitations by using other algorithms, like heuristic clustering algorithms [Salton 73, Croft 77]. We also plan to adapt related work on Galois lattices by Godin et al. [Godin 91].

From a binary tree of clusters to a hierarchical conceptual graph

Given a binary tree of clusters, we must promote it into a hierarchical conceptual graph. This is done in two steps. First we extend it to a general conceptual tree, then we further transform this tree into a hierarchical conceptual graph.

The transition from the binary tree of clusters to a general conceptual tree is simple. Each cluster consists of a set of documents whose intersection is a list of words. By gathering all the clusters whose lists of words are identical, we get a unique object identify by this list of
words: a concept. This is easily done by recursively comparing each node of the tree of clusters to its sons. As a side effect, the binary tree becomes a general tree.

The general conceptual tree thereby obtained, can be seen as a spanning tree of the conceptual graph that we are looking for. The only task left is to add the "missing" links between the concepts. Following our definition of the concepts, we will say that two concepts must be linked if one of them is included in the other, the "father" (the conceptual graph is hierarchical thus oriented) being the smallest of them. For example, the concepts conceptual graph and graph must be linked, graph being the "father".

Had we preferred Maarek's method over a clustering algorithm to find out the important concepts, we may have generated the graph using the algorithm proposed by Lieberherr in [Lieberherr 91] which infers a class inheritance graph from a set of instances. But in this case the concepts would have been limited to two words. We also could have used Lieberherr's algorithm on its own, with the result that every possible concepts would have appeared in the resulting graph. Our solution selects the most pertinent concept, thus preventing the user to being swamped with irrelevant information.

The resulting concept lattice is stored in one of our databases with the Entity-relationship model: concept are stored as entity records and the arcs implemented as relationships. To navigate in the lattice, one can retrieve either all "sons" or all "parents" of a given node.

4. Integration

The implementation of Mars as a set of collaborating agents, each with a well defined function, facilitates integration. For development, access by service name (via the name-server) meant that new methods could be added under new names without perturbing existing agents (which don't need to know about them). Better versions of existing agents could be integrated to the running system "on-the-fly" by simply changing entries in the name-server.

The provision of data-servers also helped. Most retrieval techniques rely heavily on pre-computed data: massive tables or index files. The algorithmic complexity lies mainly in the programs which build these tables; whereas, the algorithms which use this data to answer queries are usually quite simple. In Mars, the various sources of data are implemented as independent databases. The remaining portion of the retrieval code for the various techniques is quite small and easy to combine within one agent (the retrieval agent shown on the top left of figure 1).

We have already alluded to tighter integration between the various methods. The option of using terms or concepts being orthogonal to the choice of combination technique (Boolean, vector space retrieval as well as different weighing schemes). Another example is the way that semantic matching is integrated with classical retrieval.

The most problematic aspect of the semantic retrieval operation is the complexity in tree comparison. To reduce this complexity, the semantic approach is presented (and used) as a post-processing filter. A query is first evaluated by a classical method (for example, an extended Boolean evaluation); then a relatively small subset (the top 30) of the ordered documents are submitted to the semantic evaluation. The semantic module modifies the earlier document ordering according to the semantic correspondence between the query and the documents. This interaction allows improvement in document ordering while keeping the time needed for semantic evaluation to a few seconds.
Similarly, concept-browsing and standard retrieval are integrated so that at any point the user may switch from one technique to the other. As the concept lattice is huge, it is often difficult to explore it from the root concept. Other retrieval agents may give the browser an initial position in the concept lattice. Semantically related document classes can then be explored. Similarly, a user may choose first to browse the conceptual graph in order to determine the right terms to be used in his further queries submitted to other retrieval agents. The interaction in this direction is particularly important for casual users.

We are also working on a collaboration between the Thesaurus and the other modules. The objective is query modification. Depending on the number of documents returned from a query, it will be possible to replace original terms or concepts by related concepts to either broaden or narrow the search.

5. Results and conclusions

MARS was tested by analysts from the Banque nationale. One person defined a set of typical tasks and other analysts attempted the searches using the various features available. We did a short demonstrations of the system and, after that, all interaction was done remotely in our absence. The testers were on their own. After exploring the features on-line during an hour, there were 3 test sessions of two hours each over 2 weeks. During each session, a different method was tried.

The fact that our industrial partners were about to be reassigned to other projects meant that the system was still in pre-beta stage when these tests were done. Several modules were finalized during the test period and some were installed or modified even while the testers were active: a tribute to the flexible nature of Mars. We also found out post-facto that some of our original databases contained erroneous data.

In spite of all the difficulties, the main result was that our testers were delighted. They were able to find what they were looking for very quickly. They also found many pertinent documents that they never even suspected existed. Mars gave them a new efficient and interesting way to view the database that they used everyday but had never fully mastered. The Thesaurus approach had some notable successes when the pertinent documents used words different from the query (ex. règlement for paiement and espèces for devises). Extensive use of menus, care in the choice of default values and the fact that all our methods used a common set of interface routines also helped the users learn to navigate in the system.

Semantic-based IR allows to distinguish the most relevant objects from the others. The results with tester's queries have shown that the documents satisfying semantic evaluation are effectively of high relevance. This means that the precision ratio is very high as expected. For the tests, we combined the semantic retrieval with an extended Boolean retrieval. This method has shown an average precision improvement of 13% (61% v.s. 54%). However, the tests also indicate that the recall ratio would be too low if the semantic evaluation were not combined with classical techniques; for many queries, no document can be matched to the query via semantic comparison. There are many reasons for this but the most important factor is that the concept of "implication" used currently for semantic evaluation is too strict. So in our next version, tree comparison will be more flexible to allow less perfect matching. Another reason is the poor covering of the vocabulary by our current dictionary. It may be expected that a higher improvement may be obtained when the dictionary is completed. There is not yet direct interaction between the semantic retrieval agent and the thesaurus. This will be added in our next version, and will contribute to a higher recall.
Conceptual graphs were deemed interesting by our testers especially to discover the important concepts of a new area. The method has been successfully tested on a sample of over 5000 descriptions of entities from an entity-relationship model, the descriptions consisting of "sentences" of 11 open-class words on average. Over 2600 concepts were discovered, 76 of them are at the first level of the hierarchy. Having been tested by users, this graph has proven its worth; most of the concepts discovered correspond to commonly accepted concepts in the domain. In practice, the main problem arises from the number of arcs between one node and another. At the top level, for example, there are 76 concepts and users find this number overwhelming. A branching factor of 10 to 20 would appear to be acceptable; in future, we will achieve this by arbitrarily grouping small clusters into "Miscellaneous" categories.

With the rapid growth of information sources and the multiplicity of retrieval techniques, it is no longer reasonable to store all information in a single database, nor to gather all evaluation methods in a single program. Combining information retrieval techniques and network facilities is a good way to face the new requirements. We are convinced that future information retrieval systems should have the highly distributed architecture and multiple tools as found in Mars.

Acknowledgments

This research has been funded mainly by the IT Macroscope Project, managed by the DMR Group Inc. Other funding was provided by NSERC. We should also like to thank Renaud Champagne, Stan Rolland, Daniel Dutil, Pierre Bourque and the Banque nationale du Canada for their help and collaboration on this project.

Bibliography


Nie, J.Y., Paradis, F., Vaucher, J., "Un système de recherche d'informations pour la réutilisation", ICO93, Montréal, mai 1993.

