Connectionist Interaction Information Retrieval

Sandor Dominich[*]

Department of Computer Science, University of Veszprem, 8200 Veszprém, Egyetem u. 10, Hungary, E-mail: dominich@dcs.vein.hu; Department of Computing and Information Technology[♦]*, Buckinghamshire Chilterns University College, Queen Alexandra Road, High Wycombe, HP11 2JZ, United Kingdom, E-mail: sdomin01@bcuc.ac.uk*

Tel. +36 88 422022/4711
Fax +36 88 428275

[*] Reprints requests to: 8200 Veszprem, Egyetem u. 10., Hungary
[♦] Research supported by research grants OTKA T030194, AKP 2001-140.

Abstract

Connectionist views for adaptive clustering in Information Retrieval have proved to be viable approaches, and have yielded a number of models and techniques. However there has never been any exhaustive and methodical — i.e., theoretical, formal, practical, simulation- and user-based — evaluation of such a retrieval method and system. The aim of the paper is therefore just this. It suggests a connectionist clustering technique and activation spreading-based Information Retrieval model using the Interaction Information Retrieval method. Theoretical as well as simulation results as regards computational complexity of this method are presented and discussed. Evaluations of relevance effectiveness are also given using standard test collections. Two applications were designed, developed and implemented based on this method. Their relevance effectiveness was evaluated in vivo in experiments carried out with human subjects. The results obtained show that Interaction Information Retrieval based on a connectionist approach proves useful when emphasis is on high precision.

Keywords: Information Search and Retrieval, Retrieval Models, Connectionist Models, Interaction Information Retrieval, Experimentation

1 Introduction

The application of soft computing techniques to Information Retrieval (IR) aims at enhancing retrieval performance by trying to capture aspects that could hardly be modeled by other means numerically, which is important because only this can yield implementable systems. For example, fuzzy set theory allows for expressing the inherent vagueness encountered in the relation between terms and documents, and fuzzy logic makes it possible to express retrieval conditions by means of formulas in the weighted Boolean model; an overview can be found in (Kraft, Bordogna, Pasi, et al., 1998).

Connectionism represents another approach. Basic entities (documents, terms) are represented as an interconnected network of nodes. Artificial Neural Networks (ANN) and Semantic Networks (SN) are two techniques used for this. For example, Wordnet (Miller, 1990) is an online dictionary based on SN. As our technique is based on a spreading of activation, which is characteristic to ANN, we only make reference to this. ANN learning (which allows to model relations between documents, and documents and terms) was used with the principal aim to increase the accuracy of document-term weights (Bartell, Cottrell and Belew, 1995; Belew, 1987, 1989; Bienner, Guivarch and Pinon, 1990; Cunningham et al., 1997; Fuhr and Buckley, 1991; Layaida et al., 1997; Kwok, 1990). ANNs were also applied to query modification aiming at enhancing retrieval performance (Crestani, 1993; Wong and Yao, 1990). An important application area of ANNs is retrieval from legal texts (Rose and Belew, 1991; Rose, 1994; Warner, 1993).

Clustering is a well-known technique applied in Information Retrieval. It is typically used to group documents, in which case the result is a — usually disjoint — set of document groups called clusters, each containing — in some sense — similar documents. Several clustering methods and techniques have been proposed so far based on similarity measures (van Rijsbergen, 1979; Salton and McGill, 1983), neighborhoods (Voorhees, 1985), hierarchies (Lebowitz, 1987; Willett, 1988; Fisher and McKusick, 1989; Crawford et al., 1991; Tanaka et al., 1999), matrices (Deerwester et al., 1990). Retrieval is performed based on a cluster representative which may but need not be one of the cluster members. If the particular retrieval method used associates a cluster representative to a query, then every member of that cluster will be returned. This view of retrieval is based on the well-known *cluster hypothesis* according to which closely associated

documents tend to be relevant to the same request. It is commonly agreed that — *a priori* or fixed — clustering should be stable under growth, description and ordering. Recent research reveals a sound mathematical background for fixed clustering as well as for its evaluation (Hearst and Pederson, 1996; Mather, 2000).

As somewhat opposed to fixed clustering, adaptive clustering (i.e., a clustering in which the cluster structure is being developed in the presence of the query or user) has proved to be a viable approach to IR (Belew, 1989; Rose, 1994; Johnson et al., 1994, 1996; Shaw et al., 1997). Retrieval is then viewed similar to that in fixed clustering: those documents are said to be retrieved which form the same cluster ('nearest' to query). One way of conceiving adaptive clustering is to adopt a connectionist-based view using ANNs (Cohen and Kjeldsen, 1987; Belew, 1989; Kwok, 1989, 1995; Doszkocs et al., 1990; Chen, 1994, 1995; Merkl, 1999; Wermter, 2000). As yet there have not been any exhaustive and methodical — both theoretical and practical — evaluations as regards computational complexity as well as retrieval effectiveness of such a retrieval method using both standard test collections and real users assessing real applications.

Thus the aim of this paper is just this. It proposes a retrieval model using connectionist activation spreading based on the Interaction Information Retrieval (I^2R) paradigm, and reports on its evaluation, analysis and application.

2 Associative Interaction Information Retrieval

The I^2R paradigm was suggested in (Dominich, 1994; van Rijsbergen, 1996) based on the concept of interaction according to the Copenhagen Interpretation in Quantum Mechanics (query: measuring apparatus, documents: observed system, retrieval: measurement).

The documents are represented as a flexibly interconnected network of objects. The interconnections are adjusted each time a new object (e.g., a document) is fed into the network. The query is interconnected with the already interconnected objects. Thus, on the one hand, new connections develop (between the object-query and the other objects), and on the other hand, some of the existing connections can change — this represents an interaction between query and documents. Retrieval is defined as recalled memories: those

documents are retrieved which belong to reverberative circles triggered by a spreading of activation started at the object-query. The reverberative circles correspond to clusters, which are not fixed as they develop in the presence of the query. This model will be referred to as Associative Interaction Information Retrieval (AI^2R). The idea of flexible, multiple and mutual interconnections from AI^2R also appear and are investigated in (Salton, Allan and Singhal, 1996; Salton, Singhal, Mitra and Buckley, 1997; Pearce and Nicholas, 1996; Carrick and Watters, 1997; Liu, 1997; Mock and Vemuri, 1997; Dominich, 1997, 2001).

Any object o_i, $i = 1, 2, ..., M$, is assigned a set of identifiers (e.g., keywords) t_{ik}, $k = 1, 2, ..., n_i$. There are weighted and directed links between any pair (o_i, o_j), $i \neq j$, of objects. The one is the ratio between the number f_{ijp} of occurrences of term t_{jp} in object o_i, and the length n_i of o_i, i.e. total number of terms in o_i:

$$w_{ijp} = \frac{f_{ijp}}{n_i}, \quad p = 1, ..., n_j \qquad (1)$$

Because w_{ijp} is analogous to the probability with which object o_i 'offers' t_{jp} (or equivalently with which t_{jp} is extracted from o_i when being in o_j), the corresponding link may be viewed as being directed from object o_i towards object o_j.

The other weight, w_{ikj}, is the inverse document frequency. If f_{jik} denotes the number of occurrences of term t_{ik} in o_j, and df_{ik} is the number of documents in which t_{ik} occurs, then:

$$w_{jik} = f_{jik} \log \frac{2M}{df_{ik}} \qquad (2)$$

Because w_{ikj} is a measure of how much content of object o_j is 'seen' (or 'mirrored' back) by term t_{ik}, the corresponding link may be viewed as being directed from o_i towards o_j. The other two connections — in the opposite direction — have the same meaning as above: w_{jik} corresponds to w_{ijp}, while w_{jpi} corresponds to w_{ikj} (Figure 1). Figure 2 shows a simple example.

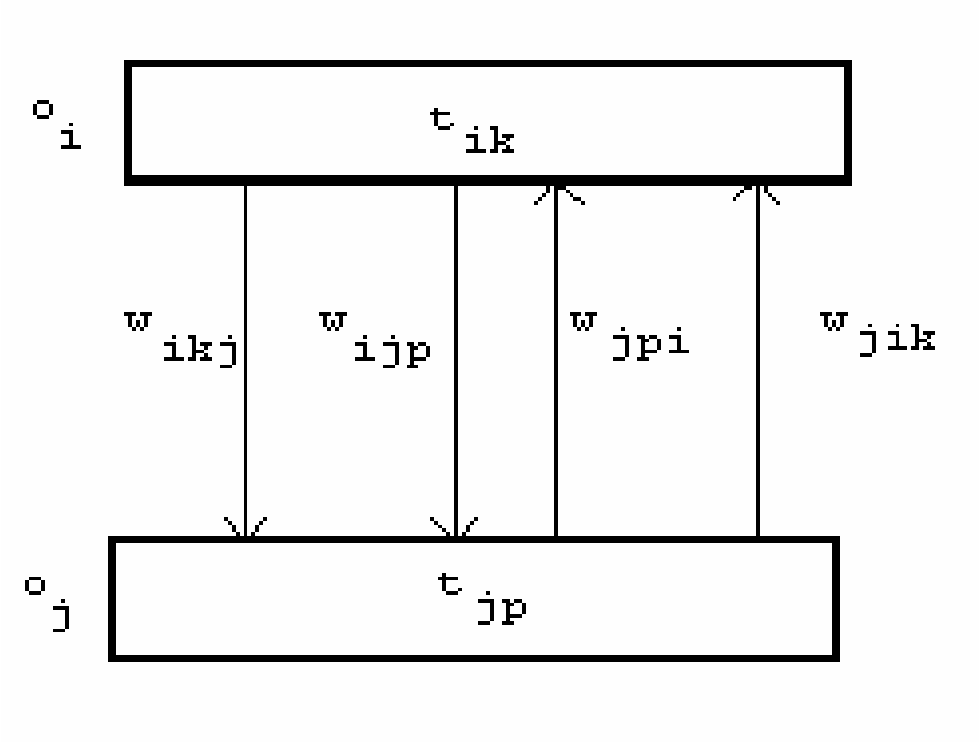

Figure 1. Associative Interaction Information Retrieval (AI^{2}R). Connections between an arbitrary object pair o_i and o_j (see text).

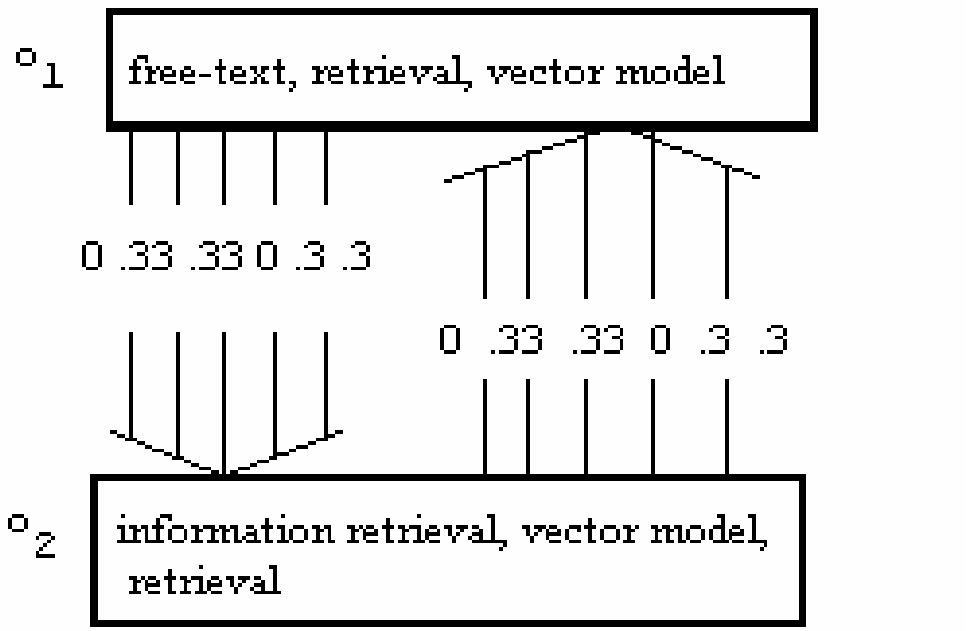

Figure 2. Associative Interaction Information Retrieval (AI^2R). Links with weights between object pair o_1 and o_2 (example). Links pointing in the same direction are grouped together, and shown by one big common arrow.

The process of answering a query (retrieval) is performed in two phases:

(i) Interaction. The query Q is incorporated first into the network. New weighted links appear between Q and the other objects, and some of the existing weights change (formula 2); this may be regarded as an expression of the network 'learning' the query. Figure 3 shows a simple example.

(ii) Retrieval. A spreading of activation takes place according to a winner-take-all strategy. The activation is initiated at the query, say o_j, and spreads over along the strongest connection thus passing on to another object, and so on. The total strength of the connection between any pair (o_i, o_j), $i \neq j$, of objects, and thus between the query and another object o_i is defined as follows:

$$\sum_{p=1}^{n_j} w_{jpi} + \sum_{k=1}^{n_i} w_{jik} \qquad (3)$$

The summations (formula 3) is made possible by the meaning associated to w_{jpi} and w_{jik} (formulas 1, 2); each represents a measure of the extent to which the query, represented by o_j, 'identifies' — the content of — o_i. After a finite number of steps the spreading of activation reaches an object already affected (in the worst case it passes through the entire network and eventually gets back to the query) thus giving rise to a loop called reverberative circle (as a model for short term memory). This is analogous to a local memory recalled by the query. The reverberative circle can be interpreted as an adaptive cluster associated to the query when this is present in the network. Those objects are said to be retrieved which belong to the same reverberative circle, and they are ranked in the order of maximal activation, i.e., in the order in which they are traversed. The same objects may not form the same cluster for a different query.

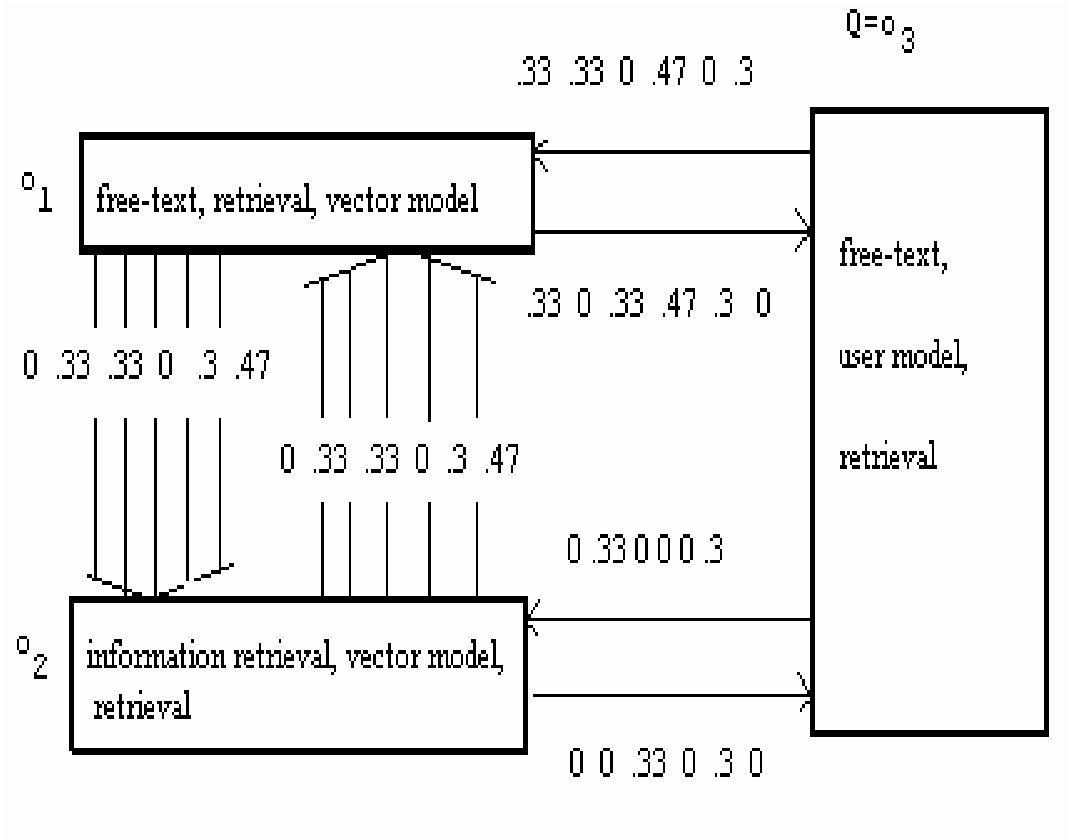

Figure 3. Associative Interaction Information Retrieval (AI^{2}R). All links having the same direction between Q and o_1, and Q and o_3 are shown as one single arrow to simplify the drawing. (a) Interaction. New connections between object-query Q(=o_3) and object-documents o_1 and o_2 are developed, and there is a changed link between o_1 and o_2 (0.47 instead of 0.3, see Figure 2 and formula 2). (b) Retrieval. The activation starts at *Q*, and spreads over to o_1 (total weight = .33+.33+.47+.3=1.43) from which to o_2, and then back to o_1. Q, o_1 and o_2 form a reverberative circle, and thus o_1 and o_2 will be retrieved.

3 Computational Complexity of AI^{2}R

As it could be seen in part 2 an algorithm which implements the method should compute a huge number of weights. Thus the question of tractability and hence complexity of such a computation arises, and it is answered in the following two theorems.

3.1 Complexity of Weights Computation

The complexity of weights calculation is given by the following theorem.

THEOREM 1. The complexity of weights computation is polynomial.

Proof. As it can be seen from the formulas 1 and 2 there are $2 \times (n_i + n_j)$ number of weights between every pair (o_i, o_j), $i \neq j$, of which there are $\binom{M}{2}$, hence $2 \times (n_i + n_j) \times \binom{M}{2}$ has complexity $\mathsf{O}(N^2)$, where O denotes 'big-Oh', and $N = \max_{i,j}(n_i, n_j)$, i.e., the largest of object lengths. The computation of the sums of weights (formula 3) between a given object o_i and all the other objects o_j, of which there are $M - 1$, takes time $(n_i + n_j) \times (M - 1)$, and thus an upper bound for the computation of all sums in the network is $(n_i + n_j) \times (M - 1)^2 = \mathsf{O}(NM^2)$ because i can vary, too, at most $M - 1$ times. Hence an overall upper bound for weights computation is $\mathsf{O}(NM^2) + \mathsf{O}(NM^2) = \mathsf{O}(NM^2) = \mathsf{O}(K^3)$, where $K = \max(N, M)$. ■

In other words the computation of weights is tractable. Once the weights have been calculated.

3.2 Complexity of Retrieval

The complexity of the retrieval process itself is given by:

THEOREM 2. The retrieval process takes polynomial time.

Proof. The spreading of activation starts at o_q representing the query, and means finding $\max_i w_{iq}$, $i = 1, ..., M - 1$, i.e., finding the maximum of all the weights linking o_q with all the other objects of which there are $M - 1$ (where $w_{iq} = \sum_p w_{qpi} + \sum_k w_{qik}$, see formula 3). Finding this maximum has complexity $\mathsf{O}(M)$. Let $o_{m'}$ denote the winner object, i.e., the object to which the activation spreads, and let L denote a list

keeping all the winner objects. It should be checked whether m' has already been a winner or not. This is accomplished by checking whether m' is in L or not: if it is we have a reverberative circle, and we stop; but if it is not in L it is written into L in the next available location. Checking L once takes $O(\text{length}(L)) = O(M)$. Because the spreading of activation is carried out at most M times, checking L takes $M \times O(M) = O(M^2)$ time. If all the weights are unique there only is one reverberative circle, but if there are objects, say o_i ($i = j_1, ..., j_k$), from which there are more than one (i.e., multiple), say n_i, maximal weights the number of reverberative circles will be $\prod_i n_i = O(n^k)$, where $n = \max_i n_i$, and thus an upper bound for the overall complexity of retrieval is $O(n^k) \times O(M^2) = O(n^k \times M^2) = O((\max(n^k, M^2))^2)$ ■

Theorem 2 tells us that the retrieval process itself is a tractable computation. These results mean that the computations involved in AI^{2}R have polynomial complexity, and thus the method is tractable (although it may be very computation demanding in practice).

3.3 Probability of Multiple Maxima

After weights summations there are $M - 1$ weighted links — $s_1, ..., s_{i-1}, s_{i+1}, ...,s_M$ — from an object o_i to all the other objects. Because i varies from 1 to M there are at most $M \times (M - 1) = O(M^2)$ links to be evaluated in all (in a search). Depending on the multiplicity (i.e., unique, double, triple maximum, or higher) of the maximum of the sequence $s_1, ..., s_{i-1}, s_{i+1}, ..., s_M$ the number of reverberative circles can increase. The number of retrieved objects depends on two factors: (a) the number of reverberative circles, and (b) the number of objects a reverberative circle contains. In order to render the influence of the first parameter simulations were carried out using a C program written for this purpose. M was taken 100,000, and different number of sequences of weights were generated at random. In each of these cases the maximum and its multiplicity was determined (Table 1).

Table 1. Simulation of the multiplicity of maxima. In 985 sequences out of 1000 sequences there was a unique maximum, in 14 cases there were double maxima, in 3 cases there was one triple maximum, and there were no maxima with multiplicity 4 (and so on).

	Multiplicity of maxima Number of sequences			
Number of generated sequences	**1**	**2**	**3**	**4**
1 000	985	14	1	0
2 500	2469	30	1	0
10 000	9532	439	26	2

Drawing the empirical density function yields a curve represented by the thinner line in Figure 4. The value of the empirical density function on every interval Δx = (0, 1), (1, 2), (2, 3), (3, 4) is calculated separately using the usual ratio:

$$\frac{number\_of\_values}{length \times total\_number\_of\_values} \tag{4}$$

for each of the three cases (Table 1), and then the corresponding values are averaged.

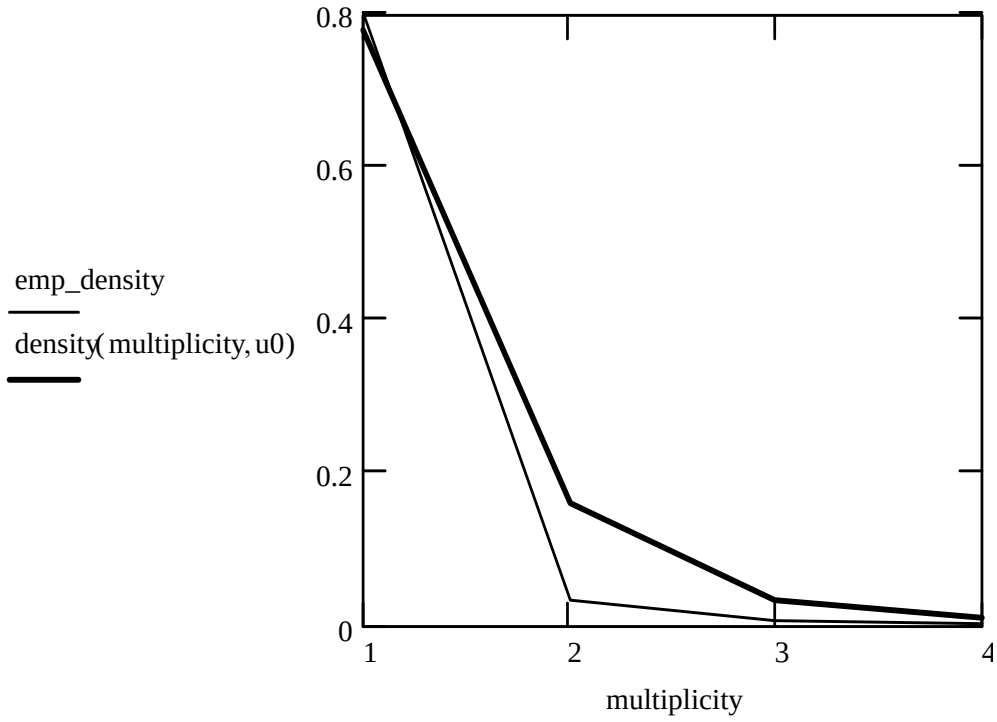

Figure 4. Empirical (thinner line) and estimated (thicker line) density functions for the multiplicity of maxima (see text).

The empirical density function can be approximated by the function:

$$f(x) = u^2 e^{-u^{0.7}x} \tag{5}$$

After curve fitting (calculations carried out using standard Mathcad curve fitting) this becomes:

$$f(x) = 3.864e^{-1.605x} \tag{6}$$

which is an estimated density function, and thus the probability to have maximum with multiplicity 2 or 3 in a random sequence s_1, ..., s_{i-1}, s_{i+1}, ..., s_M of weights can be estimated using the usual definition from probability:

$$\int_2^3 3.864e^{-1.605x}dx = 0.078 \tag{7}$$

The simulation results show that there always are a few multiple maxima, and their proportion is not high. The multiplicity increases with the number of sequences. (The probability of the multiplicity of maximum in a random sequence is an open, interesting and difficult mathematical problem.)

4 Test Collections-based Relevance Effectiveness of AI^{2}R

In order to evaluate the standard retrieval effectiveness of the AI^{2}R technique this was implemented in C++, and tested on the ADI and MEDLINE standard test collections.

Index terms were obtained automatically using standard techniques (stoplist, Porter-stemming). The statistics are shown in Tables 2 and 3, respectively. The standard 11-point recall-precision plots are shown in Figures 5 and 6. These show, for comparison purposes, the results of the correlated search (CS) for the same ADI test collection [Bodner and Song, 1996], and for SMART [Deerwester et al., 1996].

Table 2. Statistics for the ADI test collection.

Subject Area	Information Science
Type	Homogeneous
No. of Documents	82
No. of Queries	35
No. of Terms	736
Mean no. terms/Document	11
Mean no. of Terms/Query	6

Table 3. Statistics for the MEDLINE test collection.

Subject Area	Medical Sciences
Type	Homogeneous
No. of Documents	1033
No. of Queries	30
No. of Terms	5732
Mean no. terms/Document	55
Mean no. of Terms/Query	9

The average precision at seen relevant documents is 0.55 in the AI^{2}R search, whereas 0.43 using SMART search. If we take, as usual, the SMART results as a reference, the AI^{2}R outperforms it by about 20%, and it compares well to CS which it also outperforms at low to middle recall values.

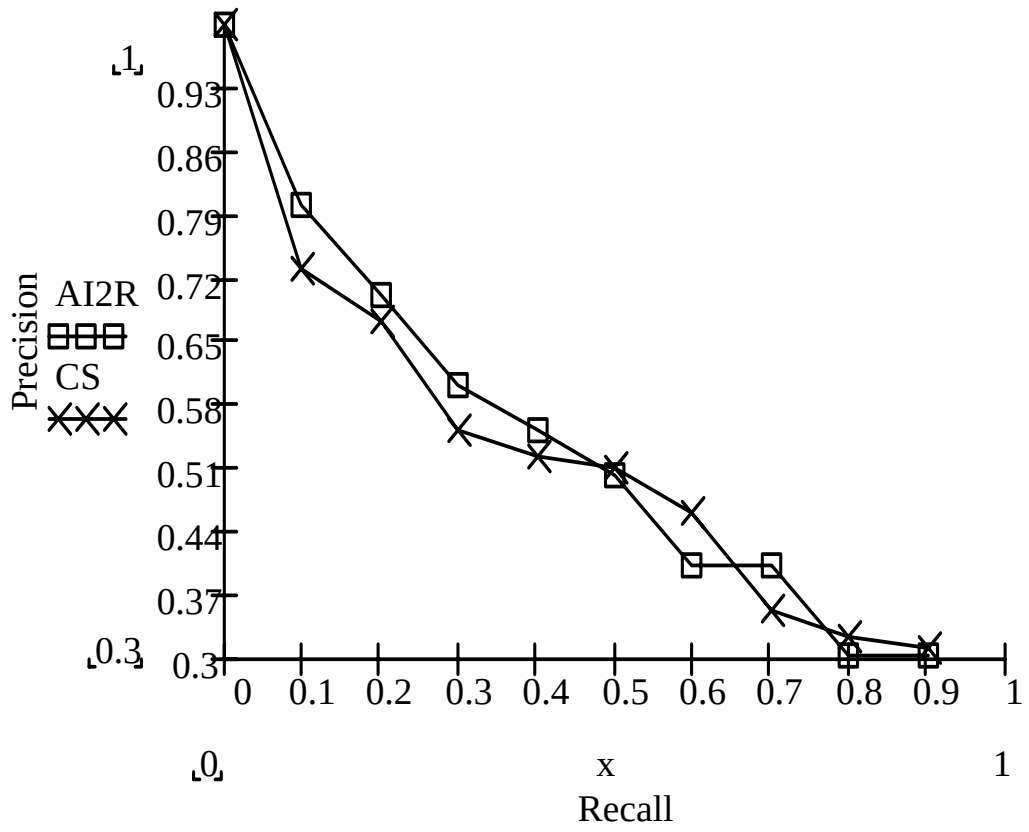

Figure 5. Recall–precision plot (line with squares) of AI^{2}R for the ADI test collection. The plot for correlated search (CS, line with crosses) is also shown for comparison, which AI^{2}R outperforms at low to middle recall values.

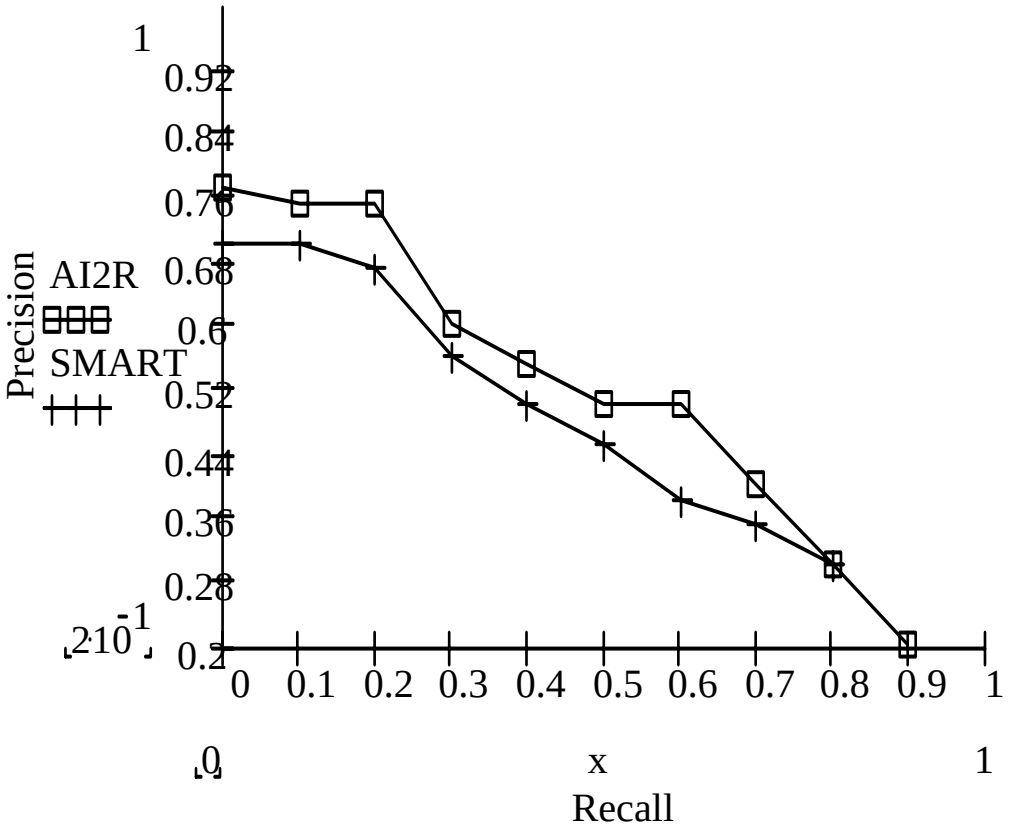

Figure 6. Recall–precision plot of AI^{2}R (line with squares) for the MED test collection. The plot for SMART is also shown (line with pluses) for comparison, which AI^{2}R outperforms very clearly.

5 AI^{2}R-based Applications

5.1 i^{2}rApplication

An application [i2rApplication] was developed which makes it possible to search the M.Sc. theses written (mostly in the Hungarian language) in the School of Technical Informatics at the University of Veszprém, Hungary. The application consists of modules (in C, Visual Basic, CGI, MathCAD, HTML) whose functions are described briefly. The application can run under Windows, Unix, Linux (currently). The complete software consists of installation packs and documentations, and is maintained by [CIR]. Figure 7 shows the architecture of i2rApplication.

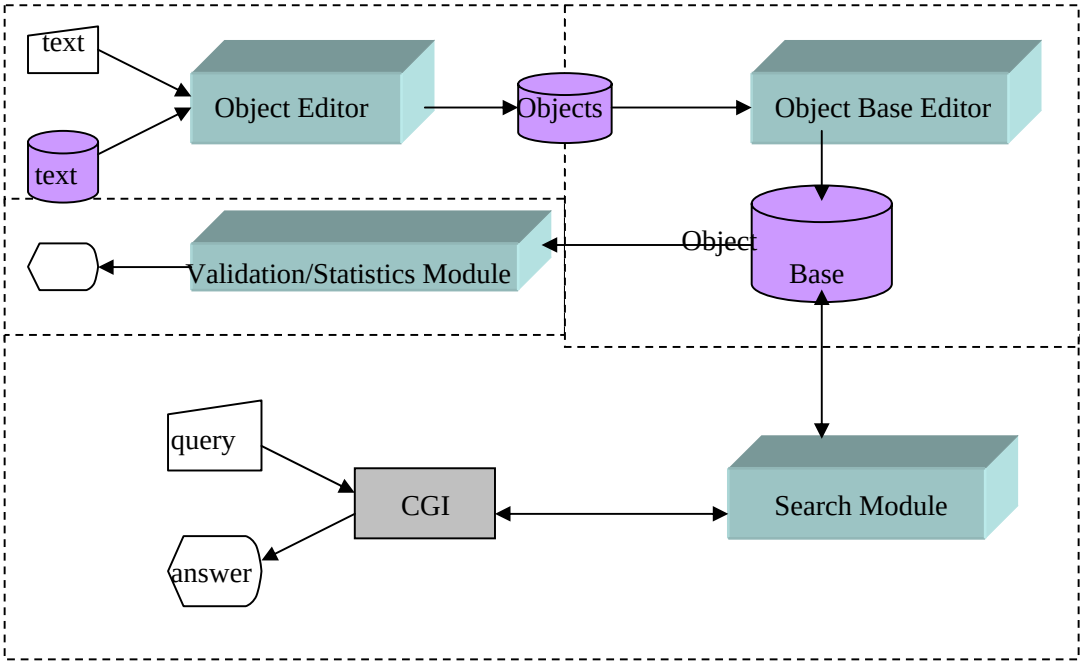

Figure 7. i2rApplication: General architecture; see text for description.

5.1.1 Object Editor

This module (Figure 8) makes it possible to create/edit objects. It is written in Visual Basic 6.0 (which allows for easy string operations and creation of user friendly interface), and is used off line under Windows (once a year in Autumn taking about one month). Its functions are as follows.

- An object is the counterpart of the (traditional) document, and consists of two distinct files: description and keywords.
- The description file contains the text and is stored in HTML format; the saving is modeled by a finite state automaton.
- The other file contains the associated keywords, and is saved as a sequential ASCII file.
- The text can be entered from the keyboard or imported from other files (which is currently done using extended abstracts).
- At present indexing is done manually (a stoplist and stemmer for Hungarian are currently being developed and planned to be used for automatic indexing) by selecting the terms in the text or by keying them in.
- The keywords undergo a small syntactic analysis and convertion for a unified representation (multiple spaces are replaced by one, convertion to small characters, elimination of commas and starting and ending spaces).

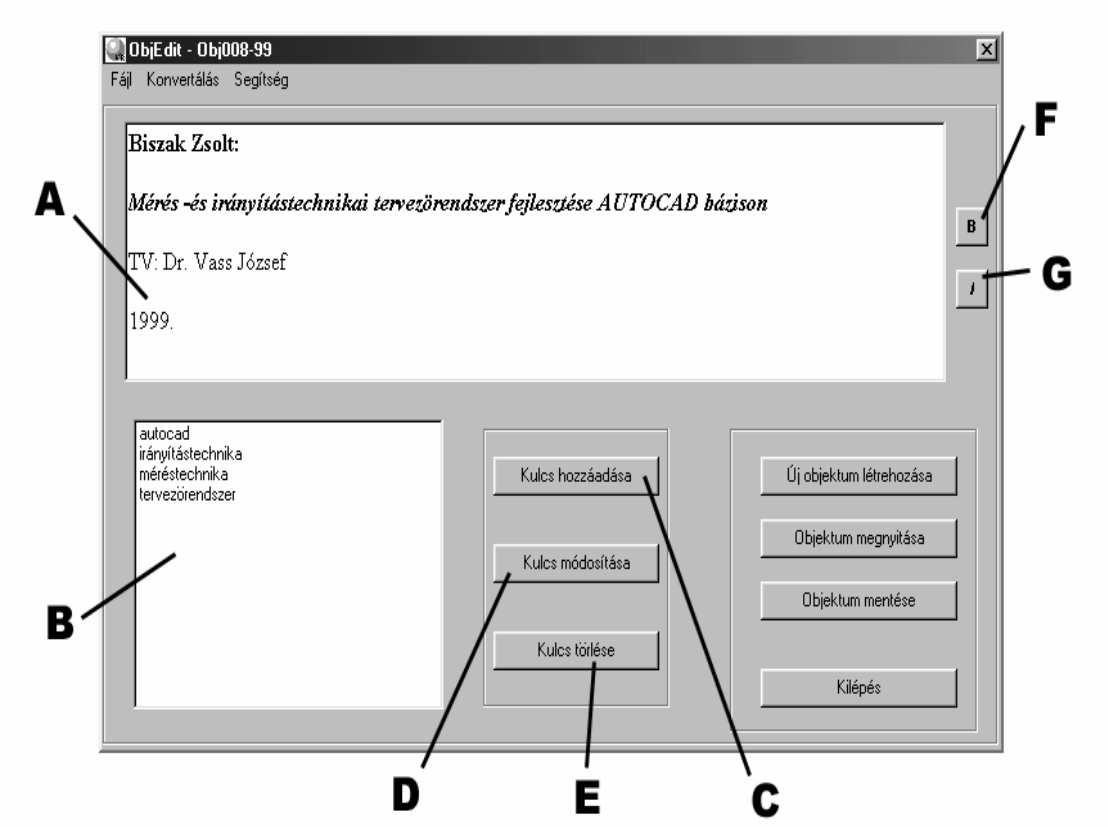

Figure 8. i2rApplication: Object Editor (in Hungarian). The function fields and their meanings are as follows: **A**: Content (e.g., text) of object, it can typed in or imported . **B**: index terms (they can be typed in or selected from field A. **C**: Add index term. **D**: Modify index term. **E**: Delete index term. **F**, **G**: settings of selected text (bold, italic). The other function fields mean object operations (top down): create new object, open for modification, save object, exit editor. Menu (left to right): (i) adds comment to object or HTML hyperlinks for browsing; (ii) import text from file; (iii) Help.

5.1.2 Object Base Editor

This module (Figure 9) is written in C, and makes it possible to create object bases, add/delete objects to/from an object base. It is typically used off line on server once (but can be used as necessary). Its functions are as follows.

- An object base is a collection of objects (pairs of files), and corresponds to a network of documents (in which activation spreading will take place).
- As the objects are created off line on different computers and/or different directories, they must be grouped and stored in one directory on server.
- Also, it is thus possible for an object to belong to multiple object bases (e.g., separate object base for each year, or one big common object base for all years; at present there is only one object base containing all theses from all years).

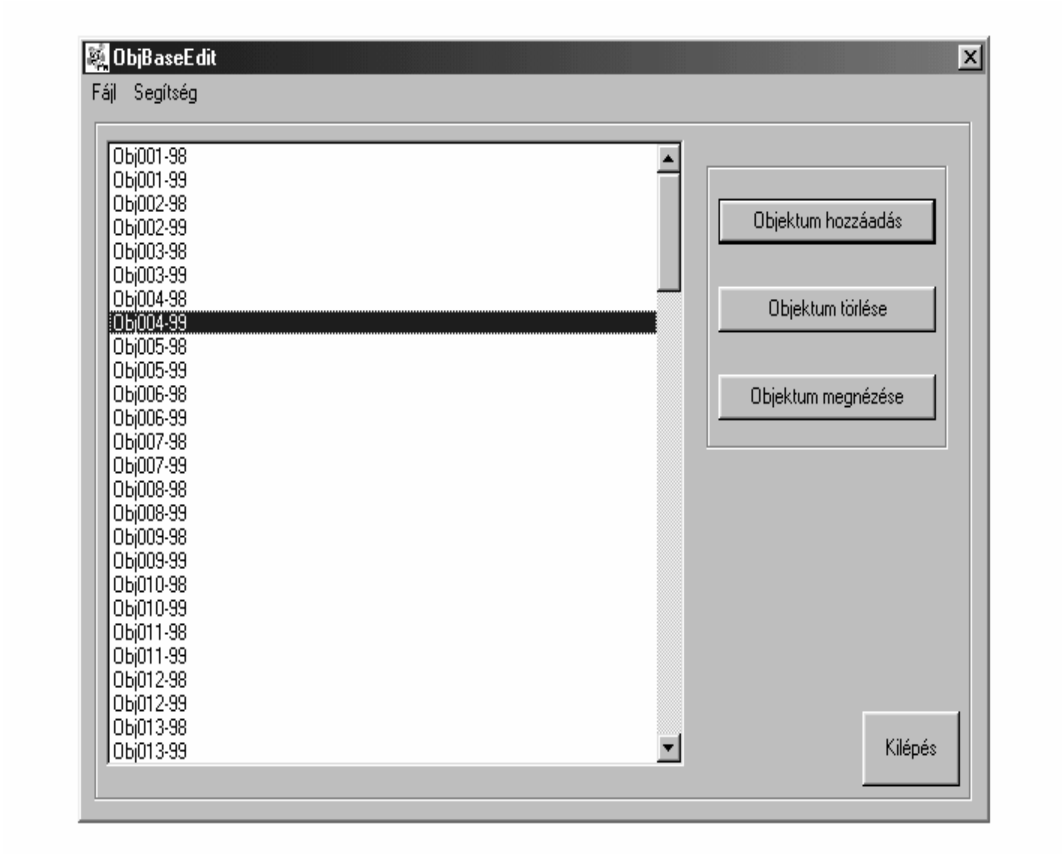

Figure 9. i2rApplication: Object Base Editor (in Hungarian). Object names appear in the white window on the left. They can be selected for viewing, deletion from or adding to an object base. Function buttons (top down on the right): (i) add selected objects to object base; (ii) delete selected objects from object base; (iii) view selected object; (iv) exit object base editor. Menu (from left to right): (i) create, open, save object base; (ii) Help.

5.1.3 Validation and Statistics Module

This module consists of C and Mathcad programs, and is used offline (usually after the use of Object and Object Base Editors) on a given object base. Its functions are as follows.

- It checks the existence of file pairs, computes the number of objects, keywords, distinct keywords, non-zero weights.
- It is also used to set the lengths in bytes for basic types (integer, double, character, pointer), and to calculate the amount of memory needed by the Search Module to run.
- The module was also used for simulations (see part 7).

5.1.4 Search Module

This module is used online on the World Wide Web. It can run under Windows, Linux (currently) or Unix, and consists of a series of CGI, C programs. It consists of

(i) a series of user interfaces,

(ii) of a set of search programs which carry out the spreading of activation and retrieval itself.

The roles of this module are as follows.

- Figure 10 shows the title page.
- Figure 11 shows the HTML page that accepts the user's query, which can contain terms separated by commas; there is no other restriction. The query undergoes a simple syntactic analysis as described at Object Editor.
- This page is sent by browser to the server as a CGI POST FORM.
- Other C programs transform the query into an object, and interconnects it into the network (object base). For the spreading of activation numeric encoding is used for keywords, and objects are represented as vectors of their keywords codes; these yield considerable speed (see part 7.4).
- The retrieved objects are shown as HTML pages as shown in Figure 12 (hyperlinks possible, see Object Editor).

Figure 10. Title page of i2rApplication. English or Hungarian versions can be selected by clicking on the chosen area.

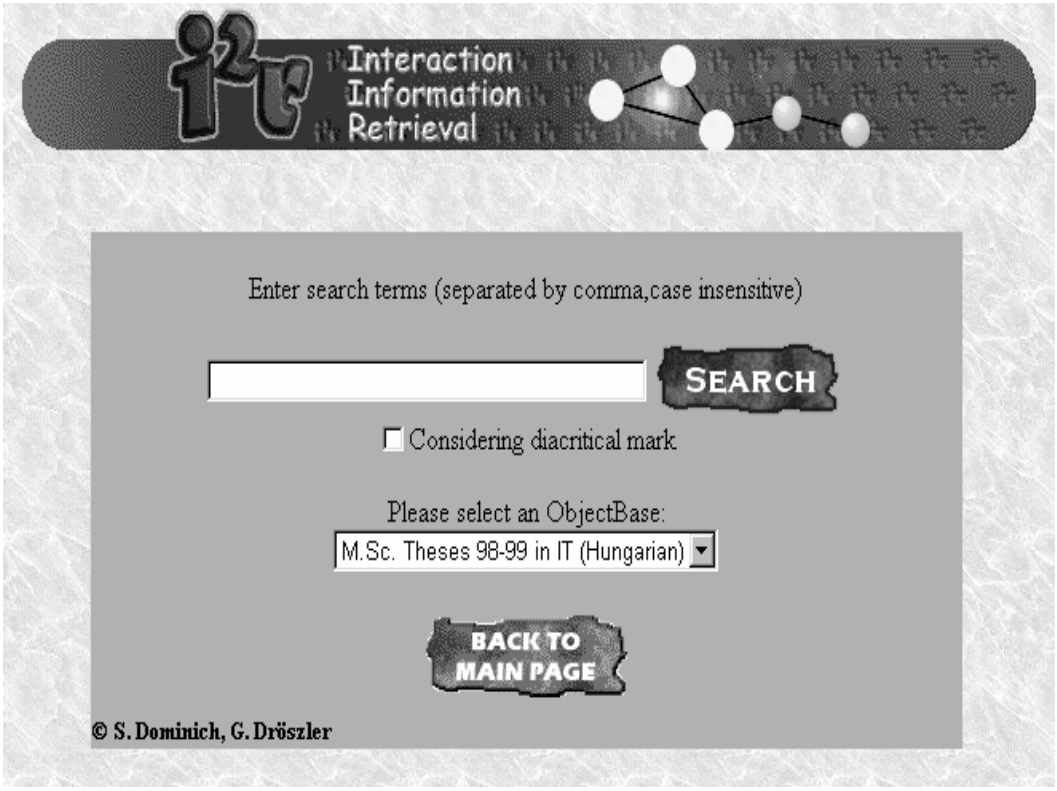

Figure 11. Query formulation in i2rApplication. The query is typed in into the white area to the left of the SEARCH button. Terms should be separated by commas; there is no other restriction. The search is initiated by clicking on SEARCH.

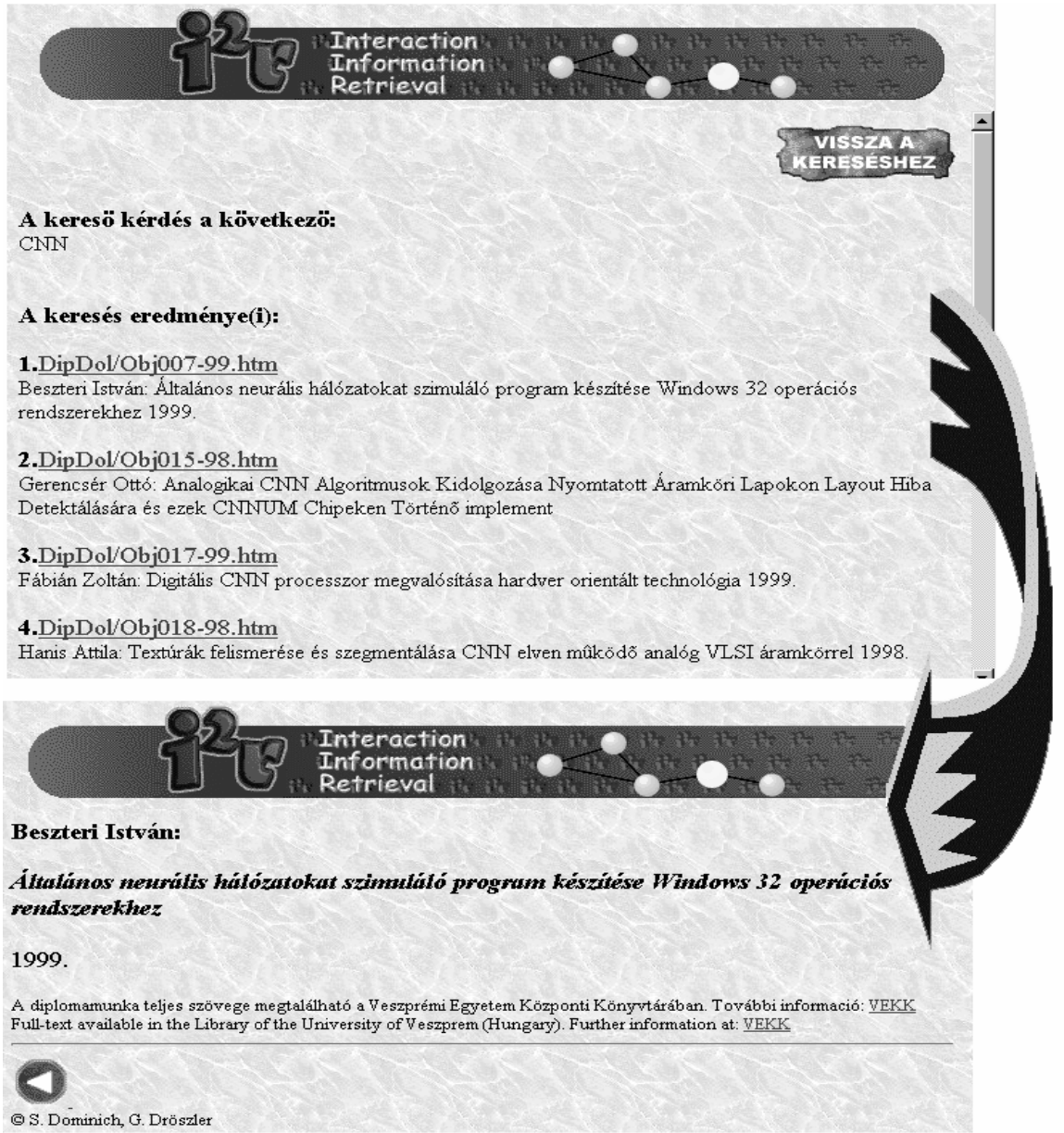

Figure 12. Search results in i2rApplication. Top page: List of hits. Bottom page: Details shown after clicking on a link.

5.2 i2rMeta

i2rMeta is a Web meta-search engine using the AI^{2}R method [i2rMeta]. It consists of two software modules (in C and PERL) whose functions are described briefly. The application runs under Linux currently, and is maintained by [CIR]. Figure 13 shows the general architecture.

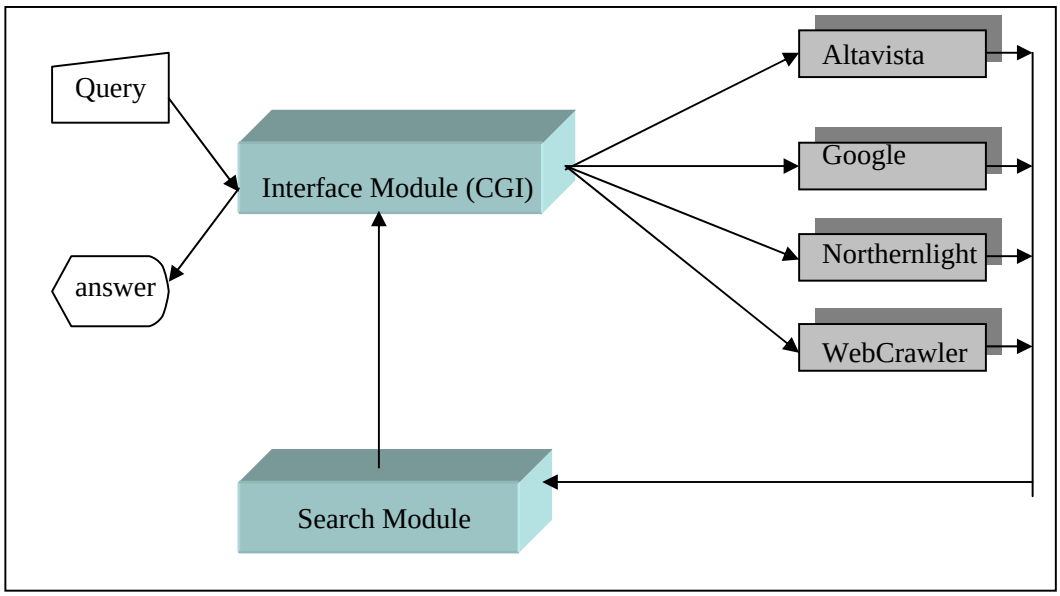

Figure 13. i2rMeta: General architecture; see text for description.

5.2.1 Interface Module

It is written in PERL, works online, the communication with the Web server is performed by CGI (Figure 14). Its roles are as follows.

- The query is entered as a set of terms (separated by commas), they are Porter-stemmed, and then sent to four commercial spider-based Web search engines (Altavista, Google, Northernlight, WebCrawler) as HTTP requests.
- The first twenty elements from the hit list of each Web searcher are considered, the corresponding Web pages are downloaded in parallel (Parallel User Agent) for speed. Non-existing Web pages, non-responding hosts or responding with an error, or pages not downloading in a predefined time slot are all treated as transparent cases (i2rMeta continues working by passing on to a next page, if any), thus i2rMeta never frustrates the user by occasionally not working.
- Each Web page undergoes the following processing: tags are removed, words are identified, stoplisted and Porter-stemmed.
- The result will be an object base on the server disk (see 5.1 for object base), which also contains the corresponding URLs
- The object base is then searched by the Search Module (below).
- The Interface module is also used for displaying the results returned by the Search Module.

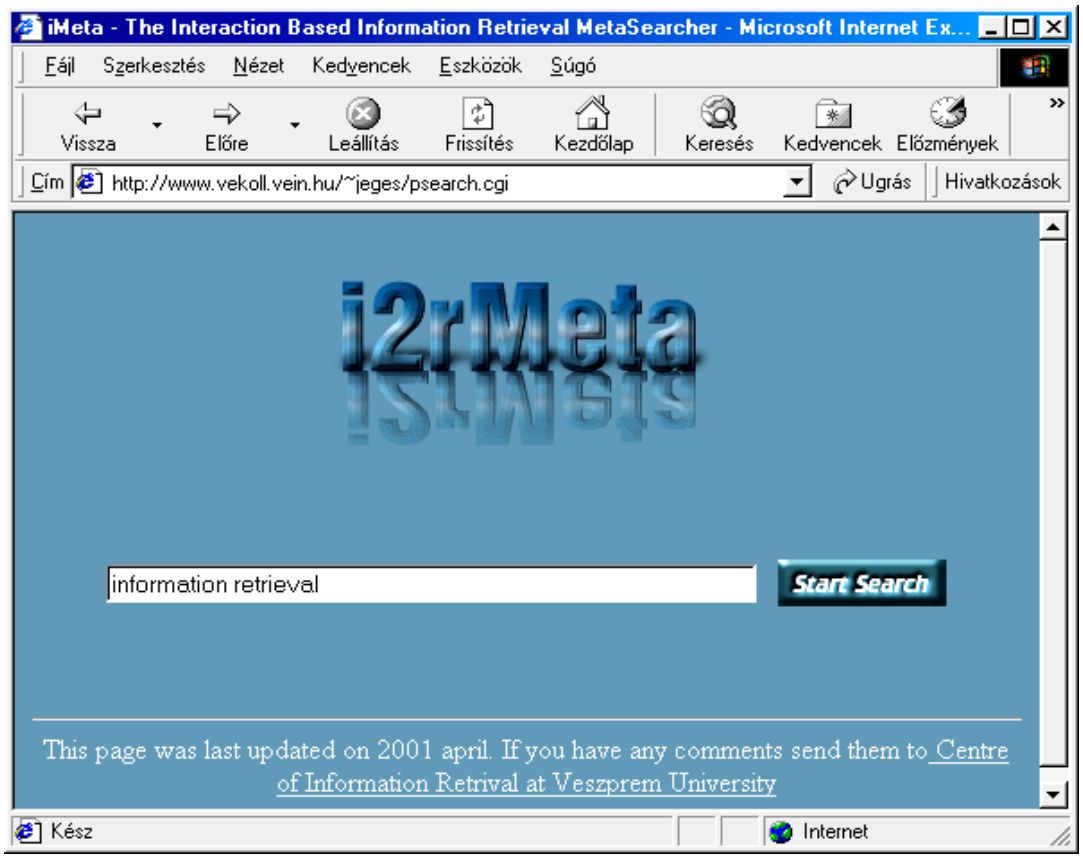

Figure 14. i2rMeta Web meta-searcher: search screen. The query is typed as terms separated by commas. The search is initiated by clicking on START SEARCH. (The language of the words in the title and tools bars depends on the browser being used.)

5.2.2 Search Module

This module (Figure 15) is written in C, works online, and implements the AI^2R method using the object base and query from the Interface Module. The implementation is similar to that used in Search Module of point 5.1 with specific differences (e.g., presence of URLs).

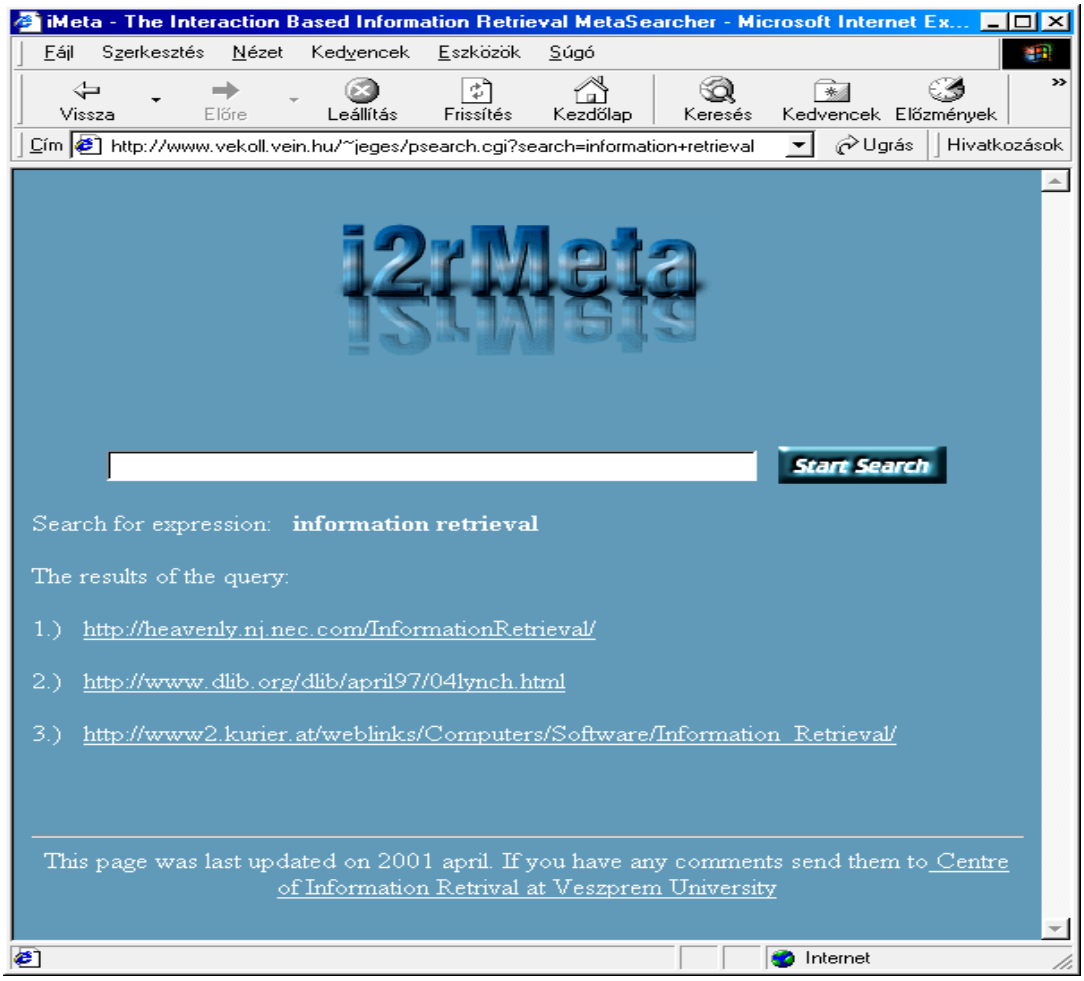

Figure 15. i2rMeta Web meta-searcher: search results (see text). (The language of the words in the title and tools bars depends on the browser being used.)

6 Relevance Effectiveness of the Applications

6.1 User Evaluation of i2rApplication

i2rApplication was evaluated *in vivo* by its typical users, i.e., undergraduate students in computer science. They were asked to perform searches with queries at their choice, and used questionnaires to qualify the returned documents as to how relevant they found them on a scale of four values as follows: not satisfied = 1, too few relevant documents = 2, satisfied = 3, very satisfied = 4. The results are shown in Figure 16. The searches were performed in June 2000 by 45 students.

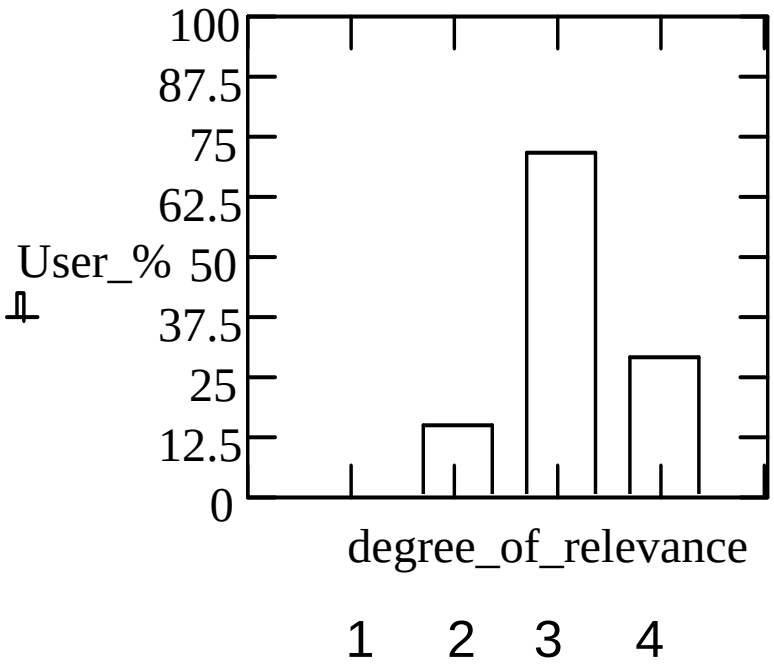

Figure 16. User satisfaction with relevance in i2rApplication.

The results show that almost 63% of the users were satisfied with the returned documents. 25% were very satisfied, and 12% thought that there were too few relevant documents in the answers. These confirmed what could be expected based on the results of the simulation and test collections, namely that the users are satisfied with the retrieval application based on AI^2R which favors high precision at low recall.

6.2 User Evaluation of i2rMeta

The i2rMeta application was evaluated *in vivo* by users. A number of forty — undergraduate and graduate — users, coming from a variety of fields (computer science, chemistry, electrical engineering, tourism, material sciences, theatre, philology, English, German, environmental sciences, management) were asked to perform searches using i2rMeta. Each user performed four searches which means 160 searches in all. The searches were conducted in the second half of March 2001. The queries were their choice, and covered a variety of topics; example queries:

Western digital specification, horned cattle, beggarly, country wedding, tesco, lyrid meteors, wireless markup language, mp3, regular expression in perl, HPLC, perl tutorial, driver, development kit, Nokia Mobile Phones, Search Engines.

Every user was asked to write down the queries used and give the total number of hits/query returned by i2rMeta as well as the number of hits considered relevant (per query). Thus precision could be estimated in each case. The average values obtained are as follows:

average_number_of returned_documents = 6 (per search)

average_number _of_relevant_documents = 4 (per search)

average_precision = 0.66.

6.3 Comparison of Rankings

A set of experiments were conducted during April 2001 to compare rankings. The hits returned by i2rMeta as well as their rank in the hit list were compared with those of the Web search engines it used. The same query was submitted first to i2rMeta, and then, in sequence, to Altavista, Google, Northernlight, WebCrawler. The five lists were compared with each other. For example, the query “Download ICQ Message Archive” got 5 answers in i2rMeta, of which the first was found on the hit list returned by Altavista where it was ranked third, the second found on the Google list and ranked first, the third found on Altavista’s list and ranked fourth, the fourth found on the Northernlight list and ranked second, and the fifth found on Google’s list and ranked third. Table 4 shows the results. The query “Nokia Mobile Phone 5110 3210” was not found within the first fourty hits (which are taken into account by i2rMeta; see above) returned in either of the search engines. In 70% of the cases search engines ranked the queries within the first ten. The majority of hits came from Google (21 cases), and then decreasingly from WebCrawler (14 cases), Altavista (12 cases), Northernlight (7 cases). It is very important to note that in all but one case there were hits ranked within the first three, in other words in 92% of the cases three out the hits returned by i2rMeta got top ranking by one of the search engines. This indicates that, as expected, the precision of i2rMeta (relative to these four commercial search engines) is indeed high.

Table 4. Results of experiments to compare rankings.

Query	**Number of hits returned by i2rMeta**	**Search engine**	**Rank**
Information Retrieval	4	Altavista	6
		Altavista	1
		Google	4
		Altavista	4
Porter's Algorithm	5	Google	3
		Altavista	6
		Google	7
		WebCrawler	4
		Northernlight	2
Regular Expression Tutorial	4	Google	1
		Altavista	1
		WebCrawler	2
Perl Language Tutorial	4	Northernlight	2
		Google	6
		WebCrawler	12
		WebCrawler	10
Siemens Mobile Phone C35i	4	Google	1
		WebCrawler	3
		Google	4
		Altavista	2
Nokia Mobile Phone 5110 3210	4	Northernlight	3
		Northernlight	7
		Northernlight	2
Free SMS Server	5	Google	2
		WebCrawler	5
		Altavista	8
		Google	1
		Northernlight	3
Search Engines	3	Google	9
		Google	3
		Google	2
Spider Based Search Engines	5	Google	5
		Google	8
		Google	7
		WebCrawler	21
		WebCrawler	4
Interaction Based Information Retrieval	3	Google	2
		Google	6
		WebCrawler	13
Access Control Systems	5	WebCrawler	23
		WebCrawler	15
		Google	3
		Google	8
Download ICQ Message Archive	5	Altavista	3
		Google	1
		Altavista	4
		Northernlight	2
		Google	3
English Hungarian Dictionary	6	Altavista	3
		WebCrawler	23
		Altavista	7
		Altavista	4
		WebCrawler	24
		WebCrawler	19

6.4 First Five/Ten Precision in i2rApplication

The i2rApplication was evaluated for its ability to put relevant pages within the first five and ten links returned. Experiments were carried out involving 50 computing students during September 2000 based on a method suggested in (Leighton and Srivastava, 1999). In our academic setting, the first five and ten links represent the quality a typical undergraduate would expect; but the method and formulas can be easily adapted for different values in another setting. Relevance categories were established prior to evaluating any links. Separate searches were performed for each query. Each of the thus returned lists (HTML pages) was evaluated (the persons the information needs originated from were not involved in these evaluations, which is not an uncommon practice as this would be hardly feasible): placement in a relevance category, calculation of numeric precision. The categories are as follows: (i) inactive links: file not found, forbidden, server not responding, (ii) duplicate links: the same link as an earlier one (e.g, duplicate theses titles due to multiple authors), (iii) category zero: irrelevant thesis, (iv) category one: relevant hit (either technically, i.e., the thesis title contains the search expression, or judged to be relevant due to its content). A document is either in a category or not in the category. In these tests precision was calculated, which is preferred in undergraduate context. Other measures of user satisfaction (such as screen layout, etc.) were not assessed.

First five precision. The classes and their weights for the first five precision case are as follows: (a) Class 1: contains the first two links, and has weight 10, and (b) Class 2: contains the next three links, and has weight 5. A weighted sum is computed as follows: $Links_{1,2} \times 10 + Links_{3,4,5} \times 5$. In order to obtain a final value for precision the above weighted sum should be divided by the total weighted sum corresponding to the case when there are five answers. This is expressed as $2 \times 10 + 3 \times 5 = 35$. If there are fewer links than five then 35 is decreased by 5 for every missing link (denoted by *mis_link)*. Thus the formula for the denominator is as follows: $35 - mis\_link \times 5$. Hence the final formula for the computation of the final value of precision is as follows:

$$\frac{Links_{1,2} \times 10 + Links_{3,4,5} \times 5}{35 - mis\_link \times 5} \tag{8}$$

First ten precision. The classes and their weights for the first ten precision case are as follows: (a) Class 1: contains the first two links, and has weight 20, (b) Class 2: contains the next three links, and has weight 17, and (c) Class 3: contains the last five links, and has weight 10. Taking into account the above, and following the line for the first five precision case the final formula in this case is as follows:

$$\frac{Links_{1,2} \times 20 + Links_{3,4,5} \times 17 + Links_{6-10} \times 10}{141 - mis\_link \times 10} \tag{9}$$

In the case of duplicate links the above formulas were used twice each: once when a duplicate link was counted as being just one link (without penalty), and once when one of the duplicate links was counted as being irrelevant (with penalty). Figure 17 shows the averages for the first five/ten precision values.

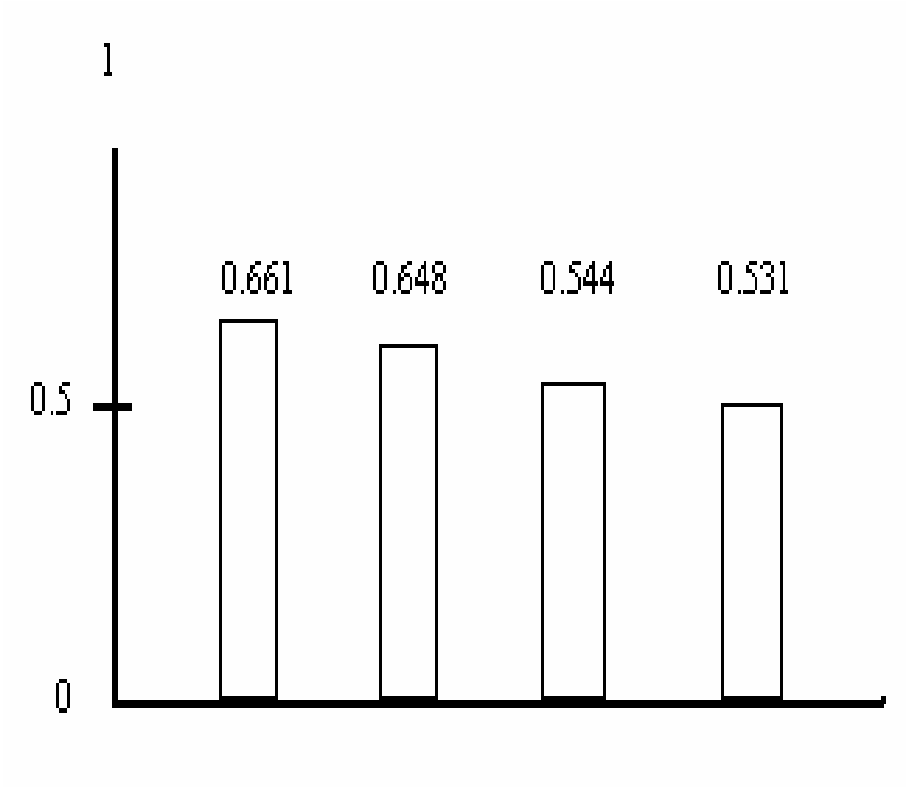

Figure 17. Average values for first five/ten precision (see text).

The average for the first five precision is 0.661 without penalty and 0.648 with penalty. The average for the first ten precision is 0.544 without penalty and 0.531 with penalty. The mean of these four values is 0.596. Precision effectiveness was much affected by irrelevant links the proportion of which was about 20%. As the proportion of duplicate links is relatively low in this application (it is not typical for a thesis to have several authors) their existence does not have a considerable affect on precision.

7 Discussion

7.1 AI^{2}R and Hopfield Net

Typically, ANN-based IR models consists of two or three layers (Kwok, 1989, 1995; Bienner, Giuvarch and Pinon, 1990) with inter- and intra-layer connections. The AI^{2}R model is single-layered just like, e.g., the Hopfield network (Hopfield, 1982). Unlike in the Hopfield net, however, where nodes are activated in parallel and then relaxed until a stable state is reached, in AI^{2}R the nodes are sequentially activated according to a winner-take-all strategy (Feldman and Ballard, 1982). Convergence is represented by a stable state in the Hopfield net, and by reverberative circles in AI^{2}R.

7.2 AI^{2}R and NP-Completeness

The suggested retrieval process, i.e., finding the reverberative circles, reminds of the Longest Circle Problem (Garey and Johnson, 1979) which is known to be NP-complete. (The Longest Circle problem reads as follows: Given an *n*-node undirected graph, and a positive integer *k*. Does the graph contain a simple cycle having at least *k* nodes?) However, the two problems are not equivalent: first of all because of Theorem 2, and on the other hand because retrieval means precise traversal (follow the maximum!) of the nodes starting from a fixed one (the query). The parallel or similarity might be caused by the following: the graph obtained after (i) having performed retrieval, (ii) having identified the circles, (iii) having kept only the links

corresponding to maxima (but unweighted), allows for formulating the Longest Circle Problem — but this is no longer the initial retrieval problem.

7.3 Average Precision of AI^{2}R

The average precision values were 0.555 for ADI, and 0.51 for MED, hence the mean precision given by the standard tests is (0.555 + 0.51) / 2 = 0.553. The AI^{2}R method was implemented in two applications whose precisions were evaluated using two series of experiments: with human subjects, and first five/ten precision. The mean of the results obtained for the first five/ten precision is (0.661 + 0.648 + 0.544 + 0.531) / 4 = 0.596. The results of user evaluations can be combined into the following numeric value: 0 × 0 + 0.333 × 0.12 + 0.666 × 0.75 + 1 × 0.13 = 0.669, which, averaged with the mean precision value for i2rMeta, gives (0.669 + 0.66) / 2 = 0.665.

The evaluations based on the standard tests and the first five/ten precision method are practically close to each other: both operate under controlled conditions with well defined parameters. Surprisingly enough, the users' perception of precision was higher, and this might not have been totally expected based solely on the results of the other two series of experiments. These values reflect that there is not a direct and necessary logical implication or equivalence between the results of standard evaluations and those based on real users.

7.4 Memory and Running Time in Applications

7.4.1 Data Values at Present

At present there are 210 objects (titles, authors' names, abstracts) stored in the object base of i2rApplication. The average number of index terms/object is 17 (15-20 keywords/object), the average disk space is 26 Kbytes/object (25 Kbytes for text +1 Kbyte for keywords), thus the object base takes up 26 × 200 = 5200 Kbytes = 5 Mbytes. The amount of (main) memory needed to answer a query at present is 38383 bytes (=0.04 Mbytes), and running time is practically instantaneous on the server.

The implementations of AI^{2}R were preceded by extensive and careful experimentation as regards memory and running time. An account of these follows.

7.4.2 Representation of the Object Base

The comparison of integers can be more than ten times faster than that of strings. Our simulations showed that the average comparison times varied between: (i) equal integers: 0-13,507 μs; (ii) not equal integers: 0-13,315 μs; (iii) equal strings: 0-183,794 μs; (iv) not equal strings: 0-182,162 μs. Due to this, in the applications, distinct index terms are assigned unique numeric codes (sequence numbers, sorted for binary search), thus every object can be represented as a vector of the codes of its terms (also sorted for binary search). As well-known, the document-term matrix is typically a rare matrix, at present the average proportion of non-zero weights is 23%. The object base is updated once a year, and is thus practically static during most of the year. This means that instead of using the whole document-term matrix, the object base is practically represented for retrieval purposes in the application as a block of non-zero numbers (requiring much less memory): for every term j its code and the corresponding df_j, and for every document i the codes of its terms k together with f_{kj}. These values only need be computed once after the yearly update. In retrieval, when the query is issued, this block is uploaded into the main memory, and updated according to query terms. For activation spreading it is enough to compute the weights between the object under focus and all the other nodes (notice that there is no need to calculate all the weights for all nodes) using a recursive routine. These techniques ensure very few memory and very fast computation.

The above idea to use a numeric block is being used in i2rMeta, too. However, the weights must be calculated online every time a query is issued. But server running time does not suffer from this (low number of objects, terms and weights), and this time is not felt by the user. What is felt is the time necessary to download Web pages from the four search engines, which depends on several factors (server, network time).

Comparing Web meta-search engines is a very relative task (Gudivada, 1997). However, the following results were obtained. Typically, i2rMeta returns between 4 and 10 hits, and average response time can take between 5 s and 30 s, which compares fairly well other meta-search engines. Using the query

"information retrieval" the following results were obtained with several Web meta-search tools (name of meta-searcher engine: number of hits returned, time):

C4: 50 hits, 10 s;

Dogpile: 10 hits in 5 s;

IxQuick: 46 hits, 15 s;

MetaEureka: 67 hits, 6 s;

Profusion: 25 hits, 10 s;

RedESearch: 100 hits, 5 s;

i2rMeta: 5 hits, 7 s;

Centrapoint and InfoZoid did not return any hit in 60 s (after which they were stopped).

7.4.3 Memory and Running Time in Future

Using the Validation and Statistics Module the amount of memory needed by the Search Module of i2rApplication for more objects can be computed. At present, the average proportion of distinct index terms is 60%; if we increase this pessimistically to, say, 90% then 1,000 objects require 228 Kbytes, 5,000 objects 1.1 Mbytes, 10,000 objects 2.22 Mbytes, whilst 10,000 objects 22 Mbytes. The expected growth rate at present is of about 65 objects/year, hence after 12 years there will 1,000 objects, after 73 years 5,000 objects, and after 150 years 10,000 objects (time spans hardly foreseeable, even at double growth rate). The average disk space for an object is 26 (25+1) Kbytes, so 5,000 objects, for example, will require 127 Mbytes of disk space, 10,000 objects the double. Simulations were carried on a 1GHz Pentium processor to estimate running time T for larger number N of documents: N = 5,000 , T < 1 s; N = 10,000, T < 1 s; N = 100,000, T < 9 s. This means that the running time of i2rApplication will practically be instantaneous on server in the future.

Carrying out the simulation for a typical TREC-like case (number of documents = 500,000, number of tokens/document = 200) the amount of memory needed would be 2.2 GBytes and running time 180 s. This running time might be too large, especially if it adds to Internet connection and transfer times. However, using a faster processor it would yield acceptable time performance.

7.5 AI^{2}R and AIR

AIR stands for Adaptive Information Retrieval system (Belew, 1987). AI^{2}R contains one type of node (document), AIR three types (terms, documents and authors). AI^{2}R uses term frequency and inverse document frequency in combination, AIR just the latter. In both models retrieval is conceived as activation spreading, and the answer set is that containing most active nodes. Learning in AI^{2}R means adjusting the weights to interconnect the query (interaction), in AIR it means changing the connection weights according user's relevance-feedback. It can be seen that AI^{2}R uses less node types and its weights are more articulate.

7.6 AI^{2}R and SCALIR

SCALIR (Rose, 1994) stands for Symbolic and Connectionist Approach to Legal Information Retrieval. In both AI^{2}R and SCALIR the nodes correspond to documents. In AI^{2}R there are two types of weights, whilst SCALIR has three: connectionist (inverse document frequency), symbolic (formal relations between cases and statutes, and are fixed), hybrid (assigned by an expert indexer). In both models retrieval is represented as activation spreading, and the result set contains those documents whose nodes have the highest activation. In both systems the number of retrieved documents remains within manageable limits: SCALIR retrieves no more than about a dozen, AI^{2}R no more than several dozens from a few thousands of documents [Dominich, 1998]. Learning capabilities of SCALIR were tested only in a small experiment, and there are no relevant data on this. In SCALIR the number of terms per document is limited to about 10 and only a small subset of a node's neighbors are visited. It can be seen that AI^{2}R is more generally applicable in that SCALIR needs domain specific (law) weights, too, to operate. Also, neither the number of terms per document nor the radius of activation spreading are limited in AI^{2}R.

7.7 AI^{2}R and Sir-Web

Sir-Web is a connectionist Web search system (Niki, 1997). It is based on a client (Java)-server (Horb) architecture, indexes 32,760 files with 463,616 keywords, and operates with AND-ed words queries. It adjusts weights during operation based on a user feedback by clicking on words offered on a list.

Unfortunately, there is not much information available as regards exact operation of Sir-Web, further the Sir-Web server broke the connection many times with time out error. Also, the files Sir-Web indexes seem to come from mainly Japanese sites (based on the hit lists returned). Due to such circumstances, apart from comparable response times and lengths of hit lists, other comparison between Sir-Web and AI^2R was not possible.

8 Conclusions

A connectionist activation spreading-based Information Retrieval model, called Associative Interaction Information Retrieval (AI^2R), was suggested using the Interaction Information Retrieval (I^2R) method.

It was shown that the complexity of the computations involved is polynomial, hence the AI^2R method is tractable in implementation. An estimation was given as to the probability to have multiple maximal activities, and this was found to be between 0.07-0.08. Because this value is not large, the number of reverberative circles (and thus of retrieved objects) is kept under a few dozens in present applications. But it also suggests that the number of reverberative circles (and hence of retrieved objects, too) can be enhanced without considerable effect on computation time by considering not only maximal activities but also the pre-maximal ones.

Standard test collections (ADI, MEDLINE) were used to evaluate the classical effectiveness of the AI^2R method. The results (precision at seen relevant documents was 0.55) showed that it was useful when high precision was favored at low to middle recall values.

Two applications (i2rApplication, i2rMeta) based on the AI^2R method were also designed, implemented and presented. Their precisions were evaluated by experiments carried out with human subjects, and using a first five/ten precision method. The results of these series of experiments showed that both applications behaved as expected, and, more, met very well users' satisfaction (between 50%–70%).

Both i2rApplication and i2rMeta compared well with other connectionist systems (AIR, SCALIR, Sir-Web). Experiments also showed that i2rMeta compared well in terms of time and hit list size with usual

Web meta-search systems, and that its precision was high. This makes it a very good complementary web meta-search engine to start with.

The series of theoretical research, tests and experiments presented in this paper constitute (for the first time) a complex, exhaustive and methodical evaluation of a retrieval technique and system based on a connectionist approach. At the same time they perhaps constitute a line and methodology recommended for a complete research and analysis of any retrieval method and system.

9 Acknowledgements

The author would like to thank the anonymous reviewers for their comments and suggestions, the National Science Foundation research grant OTKA T 030194, and the Academic Research Foundation AKP 2001-140, Hungary, for financial support, and also E. Jeges, A. Nagy, and A. Skrop for helping implement the applications and carry out the tests, experiments and simulations.

10 References

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*, Addison Wesley.

Bartell, B., Cottrell, G.W. and Belew, R.K. (1995). Learning to retrieve information. *Proceedings of the Second Swedish Conference on Connectionism*, Skovde, Sweden, March.

Belew, R.K. (1987). A Connectionist Approach to Conceptual Information Retrieval. *Proceedings of the International Conference on Artificial Intelligence and Law* (pp. 116-126). Baltimore, ACM Press.

Belew, R.K. (1989). Adaptive information retrieval: Using a connectionist representation to retrieve and learn about documents. *Proceedings of the SIGIR 1989* (pp. 11-20). Cambridge, MA, ACM Press.

Bienner, F., Giuvarch, M. and Pinon, J.M. (1990). Browsing in hyperdocuments with the assistance of a neural network. *Proceedings of the European Conference on Hypertext* (pp. 288-297). Versailles, France.

Bodner, R., and Song, F. (1996). Knowledge-based approaches to query expansion in information retrieval. In McCalla, G. (Ed.) *Advances in Artificial Intelligence* (pp. 146-158). Springer.

Carrick, C. and Watters, C. (1997). Automatic Association of News Items. *Information Processing and Management*, 33(5), 615-632.

Chen, H. (1995). Machine Learning for information retrieval: Neural networks, symbolic learning and genetic algorithms. *Journal of the American Society for Information Science*, 46, 194-216.

Chen, H., Hsu, P., Orwig, R., Hoopes, L., and Nunamaker, J.F. (1994). Automatic concept classification of text from electronic meetings. *Communications of the ACM*, 37(10), 56-73.

Cohen, P., and Kjeldson, R. (1987). Information retrieval by constrained spreading activation in semantic networks. *Information Processing and Management*, 23, 255-268.

Crawford, S.L., Fung, R., Appelbaum, L.A., and Tong, R.M. (1991). Classification trees for information retrieval. *Proceedings of the 8th Workshop on Machine Learning* (pp. 245-249). Morgan Kaufmann.

Crestani, F. (1993). Learning Strategies for an Adaptive Information Retrieval System Using Neural Networks. *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, USA.

Cunningham S.J., Holmes G., Littin J., Beale R., and Witten I.H. (1997). Applying connectionist models to information retrieval. In Amari, S. and Kasobov, N. (Eds.*) Brain-Like Computing and Intelligent Information Systems* (pp 435-457). Springer-Verlag.

Deerwester, S., Dumais, S., Furnas, G., Landauer, T. and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41, 391-407.

Dominich, S. (1994). Interaction Information Retrieval. *Journal of Documentation*, 50(3), 197-212.

Dominich, S. (1997). The Interaction-based Information Retrieval Paradigm. In Kent, A. (Ed.) *Encyclopedia of Library and Information Science*, Vol. 59, Suppl. 22. (pp. 218-236). Marcel Dekker, Inc., New York Basel Hong Kong.

Dominich, S. (1998). An Interaction Retrieval Pre-processor for Relevance Feedback. *Technology Letters*, 2(1), 5-18.

Dominich, S. (2001). *Mathematical Foundations of Information Retrieval*. Kluwer Academic Publishers, Dordrecht, Boston, London.

Doszkocs, T., Reggia, J., and Lin, X. (1990). Connectionist models and information retrieval. *Annual Review of Information Science & Technology*, 25, 209-260.

Feldman, J.A. and Ballard, D.H. (1982). Connectionist models and their properties. *Cognitive Science*, 6, 205-254.

Fisher, D.H., and McKusick, K.B. (1989). An empirical comparison of ID3 and back-propagation. *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI-89)* (pp. 788-793). Detroit, MI.

Fuhr, N. and Buckley, C. (1991). A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9(3), 223-248.

Garey, M.R. and Johnson, D.S. (1979). *Computers and Intractability*. Bell Telephone Laboratories.

Gudivada, V.N., Raghavan, V.V., Grosky, W.I. and Kasanagottu, R. (1997). Information Retrieval. *IEEE Internet Computing*, September-October, 58-68.

Hearst, M. A., and Pederson, J. O. (1996). Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. *Proceedings of the 19th Annual International ACM SIGIR Conference*, Zurich.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79, 2554-2558.

i2rApplication. http://dcs.vein.hu/CIR

i2rMeta. http://dcs.vein.hu/CIR

Johnson, A., Fotouhi, F., and Goel, N. (1994). Adaptive clustering of scientific data. *Proceedings of the 13th IEEE International Phoenix Conference on Computers and Communication* (pp. 241-247). Tempe, Arizona.

Johson, A., and Fotouhi, F. (1996). Adaptive clustering of hypermedia documents. *Information Systems*, 21, 549-473.

Kraft, D.H., Bordogna, P. and Pasi, G. (1998). Fuzzy Set Techniques in Information Retrieval. *In*: Didier, D. and Prade, H. (Eds.) *Handbook of Fuzzy Sets and Possibility Theory. Approximate Reasoning and Fuzzy Infomation Systems*, (Chp. 8). Kluwer Academic Publishers, AA Dordrecht, The Netherlands.

Kwok, K.L. (1989). A Neural Network for the Probabilistic Information Retrieval. In Belkin, N.J. and van Rijsbergen, C.J. (Eds.) *Proceedings of the 12th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 21-29). ACM Press, Cambridge, MA, USA.

Kwok, K.L. (1990). Application of Neural Networks to Information Retrieval. In Caudill, M. (Ed.) *Proceedings of the International Joint Conference on Neural Networks*, Vol. II (pp. 623-626). Hilldale, NJ, Lawrance Erlbaum Associates, Inc.

Kwok, K.L. (1995). A network approach to probabilistic information retrieval. *ACM Transactions on Information Systems*, 13(3), 243-253.

Layaida, R., Boughanem, M. and Caron, A. (1994). Constructing an Information Retrieval System with Neural Networks. *Lecture Notes in Computer Science*, 856, Springer.

Lebowitz, M. (1987). Concept learning in a rich input domain: Generalization-based memory. In Carbonell, J.G., Michalski, R.S., and Mitchell, T.M. (Eds.) *Machine Learning, An Artificial Intelligence Approach, Vol. II*. (pp. 193-214). Morgan Kaufmann.

Leighton, H. V., and Srivastava, J. (1999). First Twenty Precision among World Wide Web Search Services (Search Engines). *Journal of the American Society for Information Science,* 50(10), 870-881.

Liu, G.Z. (1997). Semantic Vector Space Model: Implementation and Evaluation. *Journal of the American Society for Information Science*, 48(5), 395-417.

Mather, L. A. (2000). A Linear Algebra Measure of Cluster Quality. *Journal of the American Society for Information Science*, 51, 602-613.

Merkl, D. (1999). Document classification with self-organizing maps. In Oja, E. and Kaski, S. (Eds.) *Kohonen Maps*. Elsevier, Amsterdam, The Netherlands.

Miller, G. (1990). Wordnet: An online lexical database. *International Journal of Lexicography*, 3(4), Special Issue.

Mock, K.J. and Vemuri, V.R. (1997). Information Filtering via Hill Climbing, Wordnet and Index Patterns. *Information Processing and Management*, 33(5), 633-644.

Niki, K. (1994). Connectionist Self-organized Information Retrieval System: SIR. *Proceedings of International Conference on Neural Information Processing* (pp. 1803-1808). Vol 3., http://www.etl.go.jp/etl/ninchi/niki/welcome.html

Niki, K. (1997). Sel-organizing Information Retrieval System on the Web: Sir-Web. In Kasabov, N. et al. (Eds.) *Progress in Connectionist-based Information Systems. Proceedings of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems*, Vol. 2. (pp. 881-884). Springer, Singapore.

Pearce, C. and Nicolas, C. (1996). TELLTALE: Experiments in a Dynamic Hypertext Environment for Degraded and Multilingual Data. *Journal of the American Society for Information Science*, 47(4), 263-275.

Rose, D. E. (1994). *A symbolic and connectionist approach to legal information retrieval*. Hillsdale, NJ, Erlbaum.

Rose, D.E. and Belew, R.K. (1991). A connectionist and symbolic hybrid for improving legal research. *International Journal of Man-Machine Studies*, 35(1),1-33.

Salton, G., Allan, J. and Singhall, A. (1996). Automatic Text Decomposition and Structuring. *Information Processing and Management*, 32(2), 127-138.

Salton, G., and McGill, M. (1983). *Introduction to Modern Information Retrieval*. McGraw Hill, New York.

Salton, G., Singhall, A., Mitra, M. and Buckley, C. (1997). Automatic Text Structuring and Summarization. *Information Processing and Management*, 33(2), 193-207.

Sebastiani, F. (1994). A probabilistic terminological logic for information retrieval. Proceedings of the *ACM SIGIR 17th International Conference on Research and Development in Information Retrieval* (pp. 122-130). Dublin, Ireland, Springer, London.

Shaw, W., Burgiu, R., and Howell, P. (1997). Performance standards and evaluations in IR test collections: Cluster-based retrieval models. *Information Processing and Management*, 33, 1-14.

SirWeb. http://www.etl.go.jp/etl/ninchi/niki/welcome-e.html

Tanaka, H., Kumano, T., Uratani, N., and Ehara, T. (1999). An efficient document clustering algorithm and its application to a document browser. *Information Processing and Management*, 35(4), 541-557.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworth, London.

van Rijsbergen, C. J. (1996). Quantum Logic and Information Retrieval. *Proceedings of Workshop on Logical and Uncertainty Models in Information Retrieval* (pp. 1-2). University of Glasgow, Glasgow.

Vorhees, E. (1985). The cluster hypothesis revisited. *SIGIR*, 188-196.

Warner, D.R. (1993). A neural network-based law machine: The problem of legitimacy. *Law, Computers and Artificial Intelligence*, 2(2),135-147.

Wermter S. (2000). Neural Network Agents for Learning Semantic Text Classification. *Information Retrieval*, 3(2), 87-103.

Willett, P. (1988). Recent trends in hierarchic document clustering. *Information Processing and Management*, 24, 577-597.

Wong, S.K.M. and Yao, Y.Y. (1990). Query formulation in linear retrieval models. *Journal of the American Society for Information Science*, 41(5), 334-341.

Yu, C.T., Suen, C., Lam, K., and Siu, M.K. (1985). Adaptive record clustering. *ACM Transactions on Database Systems*, 10(2), 180-204.