Authors:	Pirkola Ari, Järvelin Kalervo
Name of article:	Employing the Resolution Power of Search Keys
Year of publication:	2001
Name of journal:	Journal of the American Association for Information Science and Technology
Volume:	52
Number of issue:	7
Pages:	575-583
ISSN:	1532-2882
Discipline:	Natural sciences / Computer and information sciences
Language:	en
School/Other Unit:	School of Information Sciences

URN: http://urn.fi/urn:nbn:uta-3-761
DOI: http://dx.doi.org/10.1002/asi.1106

[This is a preprint of an article accepted for publication in JASIST © 2001 Wiley Periodicals, Inc. Please cite the published version: Pirkola, A. & Järvelin, K. (2001). Employing the resolution power of search keys. Journal of the American Society for Information Science and Technology, 52 (7): 575 -583.]

Employing the Resolution Power of Search Keys

Ari Pirkola and Kalervo Järvelin

Department of Information Studies, University of Tampere, Finland

Email: pirkola@uta.fi; likaja@uta.fi

Abstract

Search key resolution power is analyzed in the context of a request, i.e., among the set of search keys for the request. Methods of characterizing the resolution power of keys automatically are studied and the effects search keys of varying resolution power have on retrieval effectiveness are analyzed. It is shown that it often is possible to identify the best key of a query while the discrimination between the remaining keys presents problems. It is also shown that query performance is improved by suitably using the best key in a structured query. The tests were run with InQuery[1] in a subcollection of the TREC collection, which contained some 515.000 documents.

1. Introduction

The aim of *text retrieval* is to retrieve relevant documents from a text database. Text database users, in particular in the Web context, tend to supply short queries. They may be able to point out, which among their search keys are the *logically* most important. However, which keys *in practice* are the

[1] The InQuery software was provided by the Center for Intelligent Information Retrieval, University of Massachusetts Computer Science Department, Amherst, MA, USA.

best ones in discriminating the relevant documents from other documents *for their request,* also depends on the database and the statistical properties of the keys in it. The users cannot be assumed to know such facts. Therefore it would be valuable to automatically identify the best discriminators.

The resolution power of keys has been studied widely in text retrieval. Salton and McGill (1984; Salton, 1989) summarize work on identifying good discriminators *in database indexing.* The *tf x idf* indexing is widely used in various versions in IR. Peat and Willett (1991) pointed out that the user's search keys often have a relatively high frequency in the database and thus are bad discriminators like their nearest neighbors. These findings relate the resolution power of keys to the context of the database. In this paper we analyze search key resolution power *in the context of a request* in addition to the database, i.e., among the set of search keys for the request.

Pirkola and others (1999) have shown that, in probabilistic retrieval, search keys are not equally beneficial for queries. They can be arranged by *a posteriori* analysis into *good keys*, which in combination increase effectiveness, and into *bad keys*, which lower effectiveness. In fact, the keys can be ordered into descending utility order by the *a posteriori* analysis. The first key in the order is the *best key* (among the set considered) for the query, the following ones, which increase performance, are good keys and the remaining ones bad. By executing queries of varying exhaustivity (i.e., varying number of keys) following the descending utility order, the best possible performance (e.g., average precision over ten recall points) is achieved. By following the reverse order, the worst possible performance for the set of keys is achieved. All other orderings yield intermediate performance.

The *resolution power* of a key is its ability, *in the context of other keys for a given request*, to increase query performance. If the resolution power of keys can be analyzed automatically, the keys

given by a user may be ordered and selected for the best performance. This offers important possibilities for automatic query processing. In this paper we study the statistical properties of search keys, the methods of characterizing the resolution power of keys automatically, as well as the effects keys of varying resolution power have on retrieval effectiveness.

Following the methods by Pirkola and others (1999), we shall first analyze the query tuning space by the *a posteriori* method in terms of the best and worst cases (orderings of keys). We shall then analyze the statistical properties of search keys and develop two algorithms for the identification of key types. Finally we develop a query structuring method that utilizes the automatically identified key categories, and evaluate the performance of these queries. We shall show that it often is possible to identify the best key of a query while the discrimination between the good and the bad keys presents problems. We also show that query performance is improved by suitably using the best key in a structured query.

The tests were performed in a subcollection of the TREC collection, which contained some 515.000 documents. We selected as training data 47 such TREC Topics for which at least five substantive keys could be identified. Therefore we were able to analyze all queries at exhaustivity levels 1 - 5, *exhaustivity* meaning the number of keys in the query. We shall present the average query tuning space for optimal key orderings and reverse orderings in terms of average precision over ten recall points by query exhaustivity. The agreement of automatically identified best keys and the *a posteriori* identified best keys is analyzed.

The rest of this paper is organized as follows. Section 2 presents the methodology and Section 3 the main findings. Sections 4 and 5 contain the discussion and conclusions.

2. Problems, Methods and Data

2.1. Research problems in brief

We shall investigate the following three main research problems: (1) What are the characteristic statistical properties of the best, good and bad keys of a request? (2) Can we automatically and reliably identify the best, good and bad keys based on their statistical properties? (3) How can we automatically utilize this identification in queries and what is the effectiveness of such queries?
In Pirkola et al. (1999) we found that one request key often is far more important than the other keys in terms of retrieval performance. The finding suggests that it is possible to identify the best keys of requests based on word statistics in a collection. The identification of the best keys might allow automatic reformulation of more effective queries. In this study, we will explore whether these suppositions hold, and will elaborate on the issue by differentiating between the best, good and bad keys. The characteristic statistical properties of different key types are identified. We then utilize in algorithms statistical information suggesting key goodness, construct structured queries using the algorithms, and test their effectiveness.

2.2. Test collection and requests

The test database was a large text database containing 514,825 documents, and consisted of AP Newswire, Federal Register, and DOE abstracts subsets of the TREC collection. The size of the basic file was 1.46 GB.

Two request sets were used in the study. *The training request set* consisted of TREC Topics 101-150, and it was used to determine key status (the best, good and bad keys), to examine the statistical properties of the best, good and bad keys, and to determine statistical parameters and threshold values for automatic identification of the most important keys. *The test request set* consisted of TREC Topics 51-100, and it was used for the evaluation of structured queries.

TREC Topics 101-150 are narrower in scope and have fewer relevant documents than Topics 51-100 (Harman, 1994). Therefore the average performance level of the queries 51-100 is higher than that of the queries 101-150.

2.3. The retrieval system and query operators

As a test system, the study used the InQuery retrieval system (Broglio et al., 1994). Search keys and the words of documents were normalized using the morphological analyzer *Kstem*, which is part of InQuery. InQuery is based on Bayesian inference networks (Allan et al., 1997). All keys are attached with a *belief value*, which is approximated by the following tf.idf modification (Kekäläinen and Järvelin, 1998):

$$0.4+0.6*\left(\frac{tf_{ij}}{tf_{ij}+0.5+1.5*\left(\frac{dl_j}{adl}\right)}\right)*\left(\frac{\log\left(\frac{N+0.5}{df_i}\right)}{\log(N+1.0)}\right)$$

where tf_{ij} = the frequency of the key *i* in the document *j*

dl_j = the length of document *j* (as a number of keys)

adl = average document length in the collection

N = collection size (as a number of documents)

df_i = number of documents containing key *i*.

The InQuery query language provides a set of operators to specify relations between search keys. As with Boolean operators it is possible to construct facets, and mark relationships between concepts. The probability for an *and*-node is the product of the probabilities of its operands:

$$P_{and}(Q_1, Q_2, ..., Q_n) = p_1 * p_2 * ... * p_n$$

where P denotes probability, Q_i is either a key or an InQuery subquery, p_i, i = 1...n, is the belief value of Q_i. Thus, the system computes the product of key (or subquery) weights for the keys of the *and*-operator.

The *band*-operator is a Boolean and-operator. All the argument keys of the *band*-operator must occur in a document in order for the operator to contribute to the weight computed for that document. Otherwise *band* contributes to the document score like *and*.

2.4. Classification of key goodness by the *a posteriori* method

The key status was determined by the *a posteriori* method developed by Pirkola and others (1999; Pirkola, 1999). The method allows setting keys in an order of their relative importance to a request without human intervention.

The queries used in the experiments were formed on the basis of the *single words* of the Description

fields of TREC Topics 51-100 and 101-150. In the first phase, the words that were defined as stop-words in the retrieval system, as well as the performative words in the beginning of the Description fields, were removed. All Topics 51-100 and 101-150 begin with the performative expression "Document <verb(s)> ", for instance, "Document will report". These initial performative expressions (but not all performative words) were excluded. The purpose of the removal was to make it easier to perform query ranking and other steps associated with the runs of single word queries (see below), which required a lot of human effort. However, the separation between performative words and other words was not essential, because we examined both good and bad keys.

We examined *word* frequencies. Therefore proper name phrases (the names of organizations and countries) were searched as single words. The acronym *U.S.* was turned into the full expression *United States*.

After the removal of stop-words and performative words there were 47 out of 50 Topics in the Topic set 101-150 and 39 out of 50 Topics in the Topic set 51-100 that had at least five keys. The Topics that had less than five keys were removed, because query performance was analyzed at all *exhaustivity* (the number of keys in a query) *levels* 1-5.

Next, each word was used as a single word query. The five best keys {a, b, c, d, e} were then selected for further processing. The selection criteria were, in this order: average precision, 10%-R precision, and the lowest rank of the first relevant document in the result list.

For each request of the *training request set*, all possible nonempty combinations in the power set of the selected key set {a, b, c, d, e} were generated:{{a}, {b}, …,{e}, {a, b}, …, {d, e}, {a ,b ,c},

…, {c, d, e}, {a, b, c, d}, …, {b, c, d, e}, {a, b, c, d , e}}. Each combination was then run as a query, e.g., {a, b, c, d} was run as #and(a b c d).[2] The total number of queries run in this phase was 47 x (2^5 - 1) = 1457.

The 1-5 word queries were then arranged in series by taking each single word query in turn as the first query, by appending one of the remaining keys to it to form a 2-word query, etc. up to a complete series, e.g., #and(a), #and(a d), #and(a b d), #and(a b c d), #and(a b c d e). The principle is demonstrated in the Appendix for 4-word topics.

Next, the series were ranked by computing average precision for each series, i.e., for each series the average precision of 1-5 word queries was computed. For example, let 0.2, 0.3, 0.35, 0.33 and 0.27 be the average (ten-point) precision of #and(a), #and(a d), #and(a b d), #and(a b c d), #and(a b c d e), respectively. The average series precision is now 0.29.

The key order of *the best case*, i.e., the series with best average precision was used to rank the keys into the three categories of the best, good, and bad keys. The first key was defined as *the best key*. The keys that were in a key combination that gave better precision than the best key query or a key combination on the preceding exhaustivity level, were defined as *good keys*. The remaining keys were defined as *bad keys*. Due to bad keys in a combination, precision was decreasing (or remained the same). For example, if the sample series above was the best case, then the key *a* would be the best key, *b* and *d* good keys, and *c* and *e* bad keys.

To test the validity and reliability of the best case method to find the best keys, the best keys were also defined using the *worst case*, i.e., the series with worst average precision. In this case, the last

[2] We use here the InQuery system's query language.

key, i.e., the key that was last added to the query combination of the worst average precision, was defined as the best key. A high overlap percentage between the two methods suggested that the best case method is valid.

2.5. Automatic analysis of key goodness

For all keys of the queries of the training and test data, *collection* and *document frequencies* were counted. The proportion of collection frequency to document frequency was calculated as cf_i/df_i, where cf_i (collection frequency) is the number of occurrences of the word *i* in the collection, and df_i (document frequency) is the number of documents in which the word *i* occurs. For each key type, i.e., the best, good, and bad keys, average df and cf/df (macro-average) as well as standard deviation of df and cf/df (calculated on the basis of cf_i/df_i values of single keys) were computed[3].

All keys were then used as a single word query. For each key, the total number of the occurrences of the key (a morphologically normalized form) at documents 1-3 and 98-100 of the ranked output was counted. The number of the occurrences was divided by the total number of words in documents 1-3 and 98-100. This gave the average *within-document frequency* (wdf) for this sample of top ranked documents (note that most keys occurred in far more than 1000 documents). For each key type, the average *wdf* values and their standard deviation were calculated.

To get a representative sample, key occurrences were counted from six documents, which represented different ranked positions. To get wdf figures for all keys (i.e., also for keys of low document frequency) wdf was defined on the basis of the top ranked documents. Thus, practical factors affected the definition of wdf. Document rank level is a variable that may affect our findings

on the differences between the best, good, and bad keys in wdf, and the effectiveness of the HRP algorithm 1 (see below). Further research is needed to elaborate on the effects of different wdf definitions.

For each key, *the proportion of within-document frequency to document frequency* (wdf/df) was calculated as wdf_i/df_i , where

wdf_i = the total number of occurrences of the word *i* in documents 1-3 and 98-100 divided by the total number of words in the documents 1-3 and 98-100 (multiplied by 100)

df_i = the number of documents in which the word *i* occurs

For each key type, average wdf/df (macro-average) and standard deviation of wdf/df (calculated on the basis of cf_i/df_i values of single keys) were computed.

Based on the key statistics of our experiments, two HRP algorithms were developed. The first one was based on the cf/df and wdf/df statistics and the second one on the cf/df and df statistics. We found that two parameter values, α for wdf/df and β for cf/df, were two collection dependent parameters for automatically determining key goodness.

<u>*Algorithm 1*</u>. If a key's *cf/df* is greater than β and its *wdf/df* is greater than α times *wdf/df* of any other key of a query then select this key as a HRP key for the request.

<u>*Algorithm 2.*</u> (A) Among the keys whose *cf/df* is greater than β select the key whose *df* is at most the *df* of any other key divided by α as a HRP key for the request. If a HRP key is not found in (A),

---

[3] Average df, cf/df, wdf, and wdf/df values were computed only in the training data.

try the case (B). (B) Among the keys whose *cf/df* is greater than β select those two keys whose *df*s are at most the *df* of any other key divided by α as HRP keys for the request.

Formally, the algorithm 1 can be defined as follows:

Definition 1. Let K be the set of search keys for a request and α and β two database dependent coefficients. The function HRP1(K) gives the HRP key $k \in K$ or the empty set {} if there is no HRP key in K:

$$\text{HRP1(K)} = \begin{cases} \{k\}, \text{ when } k \in K : \forall l \in K \wedge l \neq k : \\ \qquad wdf_k/df_k \geq \alpha * wdf_l/df_l \wedge \\ \qquad cf_k/df_k \geq \beta \\ \{\}, \text{ otherwise} \end{cases}$$

The function HRP1 identifies the most important key for a query, when there is one dominating key, otherwise it yields the empty set to denote no dominating key. The latter happens if a query has several equally good keys or only bad keys. The performance of queries containing only good or bad keys cannot be improved by weighting one of them over the others. In both cases full exhaustivity queries often deliver the best performance.

Formally, the algorithm 2 can be defined as follows:

Definition 2. Let K be the set of search keys for a request and α and β two database dependent coefficients. The function HRP2(K) gives the one or two HRP key subset $H \subset K$ or the empty set {} if there are no identifiable HRP keys in K:

$$\mathrm{HRP2(K)} = \begin{cases} \{k\}, \text{when } k \in K : \forall l \in K \wedge l \neq k : \\ \qquad \alpha * df_k \leq df_l \wedge \\ \qquad cf_k/df_k \geq \beta \\ \{k_1, k_2\}, \text{when } k_1, k_2 \in K : \forall l \in K \wedge l \neq k_1, k_2 : \\ \qquad \alpha * df_{k1} \leq df_l \wedge \alpha * df_{k2} \leq df_l \wedge \\ \qquad cf_{k1}/df_{k1} \geq \beta \wedge cf_{k2}/df_{k2} \geq \beta \\ \{\}, \text{ otherwise} \end{cases}$$

The function HRP2 identifies the most important key for a query, when there is one dominating key, the two most important keys, when there are two dominating keys, and otherwise it yields {} to denote no dominating key. The latter happens if a query has several equally good keys or only bad keys.

2.6. Automatic structuring of queries

We tested several query operators of InQuery (*and*, *band*, *syn*, *sum* and *filreq*) and their combinations, i.e., various structured query types, in the training data. The most effective query type turned out to be that based on a combination of *and*- and *band*-operators (see Section 2.3.). Therefore, the structured queries (1) and their baseline queries (2) used in the tests were of the following type:

(1a) #and(#band(key_1 key_2) #band(key_1 key_3) #band(key_1 key_4) #band(key_1 key_5) key_1)

(1b) #and(#band(key_1 key_2) #band(key_1 key_3) #band(key_1 key_4) #band(key_1 key_5)
#band(key_2 key_1) #band(key_2 key_3) #band(key_2 key_4) #band(key_2 key_5) key_1 key_2)

(1c) #and(#band(key_1 key_2) #band(key_1 key_3) #band(key_1 key_4) #band(key_1 key_5) key_1 key_2 key_3 key_4 key_5)

(1d) #and(#band(key_1 key_2) #band(key_1 key_3) #band(key_1 key_4) #band(key_1 key_5)
#band(key_2 key_1) #band(key_2 key_3) #band(key_2 key_4) #band(key_2 key_5) key_1 key_2 key_3 key_4 key_5)

(2) #and(key_1 key_2 key_3 key_4 key_5)

In (1a) and (1c) the key_1 was *the best key* (*a posteriori* determination) or *a HRP key* (automatic identification), and the keys 2-5 were either good or bad keys. For some requests the *HRP algorithm 2* recognized two HRP keys. For these requests the structured queries were of the type (1b) or (1d). In (1b) and (1d) both the key_1 and key_2 were HRP keys.

The test queries of the types above were constructed by the following algorithms:

Definition 3. Let K = {k_1, k_2, k_3, k_4, k_5} be the set of search keys for a request. The functions HRPquery1(K), HRPquery2(K) and HRPquery3(K) construct a structured query of the type (1a) , (1b) , (1c) or (1d) for the HRP key(s) of K, if any, or the baseline query of type (2) if there are no HRP keys:

$$\text{HRPquery1(K)} = \begin{cases} \#\text{and}(\#\text{band}(k_1\ k_2)\ \#\text{band}(k_1\ k_3)\ \#\text{band}(k_1\ k_4)\ \#\text{band}(k_1\ k_5)\ k_1), \\ \qquad \text{if HRP1(K)} = \{k_1\} \\ \#\text{and}(k_1\ k_2\ k_3\ k_4\ k_5), \\ \qquad \text{otherwise HRP1(K)} = \{\} \end{cases}$$

$$\text{HRPquery2(K)} = \begin{cases} \#\text{and}(\#\text{band}(k_1\ k_2)\ \#\text{band}(k_1\ k_3)\ \#\text{band}(k_1\ k_4)\ \#\text{band}(k_1\ k_5)\ k_1), \\ \qquad \text{if HRP2(K)} = \{k_1\} \\ \#\text{and}(\#\text{band}(k_1\ k_2)\ \#\text{band}(k_1\ k_3)\ \#\text{band}(k_1\ k_4)\ \#\text{band}(k_1\ k_5) \\ \qquad \#\text{band}(k_1\ k_2)\ \#\text{band}(k_2\ k_3)\ \#\text{band}(k_2\ k_4)\ \#\text{band}(k_2\ k_5)\ k_1\ k_2), \\ \qquad \text{if HRP2(K)} = \{k_1, k_2\} \\ \#\text{and}(k_1\ k_2\ k_3\ k_4\ k_5), \\ \qquad \text{otherwise HRP2(K)} = \{\} \end{cases}$$

$$\text{HRPquery3(K)} = \begin{cases} \#\text{and}(\#\text{band}(k_1\ k_2)\ \#\text{band}(k_1\ k_3)\ \#\text{band}(k_1\ k_4)\ \#\text{band}(k_1\ k_5)\ k_1\ k_2\ k_3\ k_4\ k_5), \\ \qquad \text{if HRP2(K)} = \{k_1\} \\ \#\text{and}(\#\text{band}(k_1\ k_2)\ \#\text{band}(k_1\ k_3)\ \#\text{band}(k_1\ k_4)\ \#\text{band}(k_1\ k_5) \\ \qquad \#\text{band}(k_1\ k_2)\ \#\text{band}(k_2\ k_3)\ \#\text{band}(k_2\ k_4)\ \#\text{band}(k_2\ k_5)\ k_1\ k_2\ k_3\ k_4\ k_5), \\ \qquad \text{if HRP2(K)} = \{k_1, k_2\} \\ \#\text{and}(k_1\ k_2\ k_3\ k_4\ k_5), \\ \qquad \text{otherwise HRP2(K)} = \{\} \end{cases}$$

The difference between the functions HRPquery2 (for structured query types 1a-b) and HRPquery3 (for structured query types 1c-d) is that the latter repeats all keys, instead of the HRP keys only, in the query.

3. Findings

The effectiveness of the test queries was evaluated as precision at 10% recall (10%-R precision) and average precision over 10%-100% recall levels (avg. precision). The former is a user-oriented measure for high-precision queries while the latter is a system-oriented average performance measure. Statistical significance between the performance of the structured queries and that of baseline queries was tested using *Wilcoxon signed ranks test.* The test uses both the direction and the relative magnitude of the difference of comparable samples. The statistical program that was used is based on Conover (1980). The case of statistical insignificance of $p > 0.1$ is indicated by a hyphen (in

Table 5).

3.1. A posteriori experiments and frequency tests

Table 1 shows the precision of the best case and the worst case queries by exhaustivity level for the 47 requests of the training request set. Figure 1 shows the best and worst case curves drawn on the basis of the findings in Table 1. As can be seen, *in the best case* precision first improves as good keys are added to the best key queries, and then declines due to bad keys. The best precision figures are obtained at the second (10%-R precision) and third (avg. precision) exhaustivity levels. The values are, respectively, 40.5% and 16.6%. For 10%-R precision, the difference between the second and third level is small.

In the worst case, queries get increasingly higher precision values as exhaustivity increases from one to five (Table 1 and Figure 1). A prominent feature in the worst case is that precision improves substantially from the fourth to the fifth exhaustivity level. Improvement in 10%-R precision is 19.3% units (from 12.7% to 32.0%) and improvement in average precision 9.1% units (from 4.3% to 13.4%). A substantial precision improvement from the fourth to the fifth exhaustivity level shows that for most test requests one key was far more important than the other keys. For some requests there were two (or more) important keys. Therefore, precision improvement was also fairly good from the exhaustivity level three to four.

For 42 of the 47 (= 89.4%) requests, the best and worst case methods gave the same best keys. In other words, the first key of the best case and the last key of the worst case were the same keys for almost 90% of requests.

Table 2 presents document frequencies (df) and collection/document frequencies (cf/df) for the best, good and bad keys. For each key type, the second column shows the percentage of keys in three df categories, i.e., df < 5,000, df = 5,000-50,000, and df > 50,000. As shown, most best keys (= 53.2%) are in the category of df < 5,000. The average document frequency of the best keys is 8,182, which is much lower than that of good (26,646) and bad (25,586) keys. Bad keys give the highest and the best keys the lowest figure in standard deviation of df (column 4). The difference between good and bad keys is small both in average df (column 3) and average cf/df (column 5). For the best keys, average cf/df is 2.26. This figure is higher than the cf/df figures of good (1.86) and bad (1.79) keys. The best keys differ from bad keys in particular in the df category of df = 5,000-50,000. The difference is 0.69. The difference in average cf/df (= 0.47) is smaller, because many bad keys (n=10) belong to the category of df > 50,000, and the average cf/df is increased due to these keys.

The wdf of the best keys is lower than that of good and bad keys (Table 3, column 2). However, *wdf/df* is much higher for the best keys (0.383) than for good (0.144) and bad keys (0.138). The standard deviation of wdf/df is very high for the best and bad keys (the second last column). This is in particular due to keys (n=3) whose document frequencies were very low (df < 100). The last column (modified standard deviation) presents standard deviation of wdf/df for the best and bad keys when the three keys of df < 100 were not taken into account in calculations. The order of the key types is still the same. The keys of very low document frequencies (df < 500, around) still have strong influence on the figures (1.47 and 0.96).

3.2. Structured query tests

In this study the parameter values α (for wdf/df) and β (for cf/df) for the HRP algorithms (Section 2.5.) were 2 and 1.4, respectively, on the basis of our findings in Section 3.1. The keys that were

identified as HRP keys by the HRP algorithms were weighted structurally in structured queries by the HRPquery algorithms.

For the queries with *a posteriori* identified best keys, the structuring effect is clear both in 10%-R precision and average precision (Table 4). The relative improvement percentages obtained by structuring are 18.8% (10%-R precision) and 23.1% (avg. precision) with respect to the baseline queries.

The functions HRP1 and HRP2 were first applied in the training data (Topics 101-150) to test the agreement of automatically identified best keys and the *a posteriori* identified best keys. The function *HRP1* identified 36 of 47 queries (= 76.6%) as queries having a single HRP key. Altogether 31 out of the 36 HRP keys (= 86.1%) matched the keys that were recognized as the best keys *a posteriori* either by the best or the worst case method. The function *HRP2* identified 40 of 47 queries (= 85.1%) as queries having HRP keys. Of the 40 HRP keys 35 (= 87.5%) matched the *a posteriori* best keys.

The evaluation of the effectiveness of the functions for HRP identification and query structuring was done by means of the test data (Topics 51-100). The query type (1a) (or 1b) was first used. As can be seen in Table 5, the function *HRP1* identified a HRP key for 20 of the 39 requests (= 51.3%) while the function *HRP2* gave 28 HRP keys (= 71.8%). For the queries based on the function *HRPquery1,* precision improvement figures due to structuring are 7.3% (10%-R precision) and 16.0% (avg. precision). The results are statistically insignificant. For the queries based on the function *HRPquery2,* the relative improvement percentages are 10.5% (10%-R precision) and 15.2% (avg. precision). The results are statistically significant only at the significance levels 0.05 (10%-R precision) and 0.1 (avg. precision). The latter does not count as significant.

The function HRP2 found more effectively HRP keys than the function HRP1. In the final experiment we tested the structured query types (1c) and (1d), constructed by the function *HRPquery3*. As shown in Table 6, the structured queries perform clearly better than the baseline queries. In the test data, the relative improvement percentages yielded by the structural weighting of HRP keys are 11.1% (10%-R precision) and 20.5% (avg. precision). The results are statistically significant at the levels $p = 0.02$ and $p = 0.002$ for 10%-R precision and average precision, respectively. The findings should be confirmed with further TREC Topic sets because of variation in characteristics between TREC Topic sets (taken in chunks of 50 topics; Harman, 1994; Section 2.2.).

In summary, remarkable performance improvement percentages were obtained in relation to the unstructured baseline queries by applying the function *HRPquery3* yielding structured queries of types (1c) and (1d). Function HRPquery3 is based on simple *df* and *cf/df* values of database index terms. Therefore, it can be implemented readily in operational retrieval systems. In the case of the structured query type (1c), the strict conjunction restrictions of the structured query type (1a) are relaxed. This seems to be important because the identification of the correct HRP keys is sometimes uncertain. The structured query type (1c) performs well even when the identification of a correct HRP key fails.

Table 1. The precision of the best and worst case queries

Query Type	10%-R Pr.	Avg. Pr.
The Best Case (n=47)		
Exhaustivity 1	19,6	9,0
Exhaustivity 2	40,5	15,7
Exhaustivity 3	40,1	16,6
Exhaustivity 4	36,8	15,8
Exhaustivity 5	32,0	13,4
The Worst Case (n=47)		
Exhaustivity 1	0,3	0,1
Exhaustivity 2	1,3	0,3
Exhaustivity 3	3,2	1,1
Exhaustivity 4	12,7	4,3
Exhaustivity 5	32,0	13,4

Figure 1. The best and worst case curves

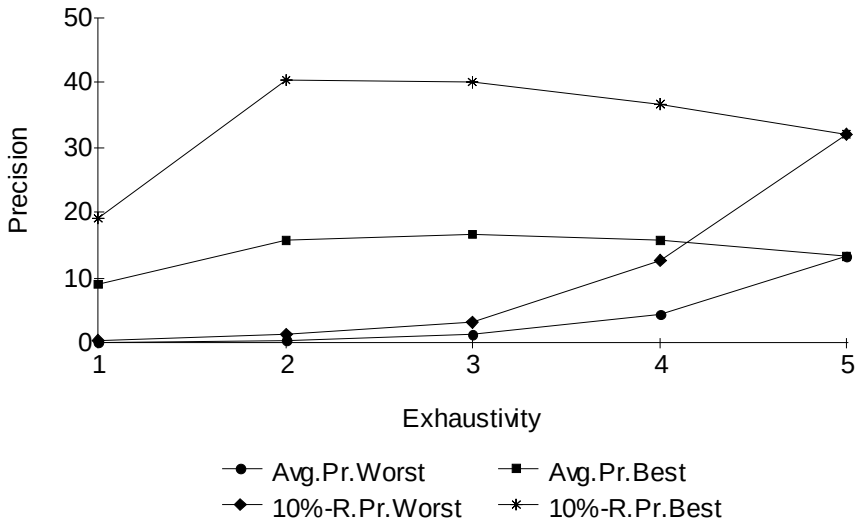

Table 2. Document frequencies and collection/document frequencies for the best, good, and bad keys

Key Type	% n	df	df st.dev.	cf/df	cf/df st.dev	cf/df BEST-GOOD	cf/df BEST-BAD
BEST							
Avg, n=47		8.182	11.735	2,26	0,76	-	-
df < 5000, n=25	53,2	2.082		1,91		-	-
df = 5000-50000, n=21	44,7	12.824		2,42		-	-
df > 50000, n=1	2,1	63.172		-		-	-
GOOD							
Avg, n=107		26.646	21.152	1,86	0,66	0,40	-
df < 5000, n=13	12,1	2.787		1,32		0,59	-
df = 5000-50000, n=81	75,7	23.701		1,86		0,46	-
df > 50000, n=13	12,1	68.629		1,87		-	-
BAD							
Avg, n=81		25.586	26.005	1,79	0,67	-	0,47
df < 5000, n=16	19,8	1.729		1,52		-	0,39
df = 5000-50000, n=55	67,9	22.510		1,73		-	0,69
df > 50000, n=10	12,3	80.681		1,89		-	-

Table 3. Within-document frequencies and within-document/document frequencies for the best, good, and bad keys

Key Type	wdf	wdf st.dev.	wdf/df x 1000	wdf/df st.dev.	wdf/df st.dev., mod.
BEST					
Avg, n=47	3,135	1,3	0,383	5,40	1,47
GOOD					
Avg, n=107	3,838	1,7	0,144	0,68	-
BAD					
Avg, n=81	3,581	1,4	0,138	4,77	0,96

Table 4. The effects of query structuring for the queries with *a posteriori* determined best keys

Query Type	10%-R Pr.	Avg. Pr.
A posteriori determination of the best keys (n=47)		
Structured type 1a	38,0	16,5
Unstructured (baseline)	32,0	13,4
Change %	18,8	23,1
Statistical significance level	0,001	0,001

Table 5. The effects of query structuring in the test data, structured query type (1a)

Query Type	10%-R Pr.	Avg. Pr.
Automatic identification of HRP keys		
Function HRP1 (n = 20 of 39)		
Function HRPquery1	53,2	27,5
Unstructured (baseline)	49,6	23,7
Change %	7,3	16,0
Statistical significance level	-	-
Function HRP2 (n = 28 of 39)		
Function HRPquery2	54,8	25,8
Unstructured (baseline)	49,6	22,4
Change %	10,5	15,2
Statistical significance level	0,05	0,1

Table 6. The effects of query structuring in the test data, structured query type (1c)

Query Type	10%-R Pr.	Avg. Pr.
Automatic identification of HRP keys		
Function HRP2 (n = 28 of 39)		
Function HRPquery3	55,1	27,0
Unstructured (baseline)	49,6	22,4
Change %	11,1	20,5
Statistical significance level	0,02	0,002

4. Discussion

The results of this study showed that in most requests one key was much more important than the other keys. This finding is consistent with the findings of Pirkola et al. (1999) which showed that one request concept or conceptual aspect was more important than the other concepts or aspects from the viewpoint of retrieval effectiveness. The finding also is consistent with the discourse model of de Beaugrande (1980) in which *a topic* is represented as a conceptual network where one node (concept) is *a primary node*. De Beaugrande's text-world model also involves the separation between primary and secondary concepts.

The finding that one topic word often has higher resolution power than the other words of a topic can be explained as follows. Typically, one topic word (the HRP key) is a *primary topic word*, which represents the *primary topic concept*. Other words are either *secondary topic words* or words that have other than topical functions in a text. Specificity and generality are word properties. The tendencies of words to rather act as primary than secondary words are *word properties.* These seem to be associated with statistical properties of words in a collection. Primary words tend to occur in a smaller fraction of a collection than secondary words. Compared with the secondary words, the primary words also tend to have more occurrences in those documents where they appear. Therefore it is possible to use statistical properties of words as indicators of their semantic significance for a given topic. Thus, for example, scientists do several kinds of research, like *cancer research*, *electromagnetic research*, and *indexing research*. Therefore the word *research* has high document frequency in the document population on scientific research. More specific words (*cancer*, *electromagnetic*, and *indexing)* are concentrated in documents where they act as primary words, and they have low document frequencies in the whole population. Nevertheless, in a given document the secondary topic words may have more occurrences than the primary topic word (note that good and

bad keys had high wdf values).

Proper names (single words and the components of proper name phrases) were more common among the best keys than in the categories of good and bad keys. Nevertheless, proper names did not dominate within the set of 47 best keys nor in the sets of 40 and 28 HRP keys of the training and test requests. The percentages of proper names were as follows: the best keys (21.3%), good keys (10.3%), bad keys (12.3%), HRP keys of the training requests (22.5%), and HRP keys of the test requests (21.4%).

The frequency statistics of good and bad keys were quite similar and, therefore, we could not differentiate them automatically. The main difference between them was perhaps not that good keys would have been semantically significant and bad keys insignificant. The collected frequency statistics and the study of some sample documents revealed that the set of bad keys was heterogeneous, and there were several reasons for their negative effects. Bad keys had higher standard deviation of df and wdf/df than the good keys. Some bad keys were clearly semantically insignificant, e.g., the words *approach* and *consequence*. Their cf/df was low, i.e., 1.20 and 1.09 respectively. The average cf/df for bad keys was 1.79. A higher percentage of bad keys than good keys is in the category of df < 5,000 (Table 2). Low document frequency coupled with semantic insignificance can devastate a search. Some bad keys occurred in non-topical documents, e.g., including just lists (for example, the word *share*), and some appeared in documents that contained long lists with bad keys dominating. It seems that cf/df is not a good indicator of resolution power for very common words. On the other hand, wdf/df is not a good indicator of resolution power for very low-frequency words. Therefore the effects of very low and high df on key resolution powers need to be studied further.

The (*a posteriori*) best and worst case methods found the same best keys for 42 of 47 (= 89.4%) requests. The high overlap percentage, the consistency of the findings of the frequency tests, and the positive effects of the structural weighting of the best keys show that the best keys can be determined effectively by the best case method. The reason for the 5 controversial cases seems to be natural rather than methodological. Sometimes topic words do not differ clearly in semantic significance. Sometimes two (or more) topic words are highly important and other words less important. There may be two primary topic concepts or a primary concept may be expressed by a phrase whose components do not differ in semantic significance. For most requests, the best keys were parts of noun phrases (NP). In most cases, however, one phrase element was the core word of the phrase (the best key). An example of the case where two request concepts are equally important and one element of an NP is a core element of the NP is the Topic 102, *laser research related, or potentially related, to the U.S.'s Strategic Defense Initiative*. In this case, the words *laser* and *strategic* were the most important keys. An example of the case where one key dominates is the Topic 146, *the negotiating* (a good key) *process leading to an end* (a good key) *to the Nicaraguan* (the best key) *civil* (a bad key) *war* (a bad key).

The possible applications of the proposed method involve query expansion, phrase-based searching, and cross-language information retrieval (CLIR). The relative improvement figures yielded by the structural weighting of the best and HRP keys were 7.3-23.1%. The potential improvement gains using unexpanded structured queries are limited. Query structuring is useful in particular when applied to expanded queries (Kekäläinen and Järvelin, 1998; Pirkola, 1998). Therefore, the structural weighting of the most important keys is likely to be useful in particular when applied to expanded queries. Probably the main mechanism by which the structural weighting affects (expanded) queries is by lowering the effects of bad (expansion) keys.

In an additional test, we expanded phrase-based queries (with syntactic noun phrases as keys) with subphrases and phrase component words. For unstructured queries (baseline, n= 43) the 10%-R precision was 27.0%, and for structured queries with the #and-#band -structure and with the best keys (*a posteriori* determination) weighted structurally 37.1%. The relative improvement percentage is 37.4%. These figures are tentative but suggest that the method might be very useful in phrase-based searching.

In our current projects we will apply the findings of this study in a CLIR environment. The main problems of CLIR involve the pruning and weighting of translation equivalents (Grefenstette, 1998). In dictionary-based CLIR each source language key is replaced by all of its target language equivalents in a dictionary. Therefore the number of irrelevant keys in a target language query is usually high. A method**,** which would identify the good and bad keys seems to be very useful to balance target language queries in cross-language retrieval.

5. Conclusions

This study shows that the search keys in a limited set of keys typical of natural language information requests are not equally important for the request. When the keys are ranked (through *a posteriori* information) in an order of their relative importance, a query of 2-3 keys often gives the best results. On the other hand, when it is not easy to rank the keys, high exhaustivity queries yield reasonable results.

In most requests, the resolution power of one key (the HRP key) was much higher than that of the other keys. We devised a method, which automatically and with good reliability identified the HRP keys. However, we could not separate the good and the bad keys among the remaining keys through

statistics. We also were able to utilize the most important keys in automatically structured queries by *treating the keys in different ways according to their value for a query*. This improved query effectiveness remarkably. While the users may be able to point out, which among their search keys are the *logically* most important, they do not possess the skills, interest and knowledge to find out, which keys *in practice* discriminate the relevant documents from the other. Therefore it is valuable to automatically identify the most important keys. It was shown in this paper that automatically structured HRP queries improve retrieval effectiveness.

Acknowledgements. The InQuery software was provided by the Center for Intelligent Information Retrieval, University of Massachusetts Computer Science Department, Amherst, MA, USA. This research is part of the research project *Query structures and dictionaries as tools in concept-based and cross-lingual information retrieval* funded by the Academy of Finland (Research Projects 44704; 49157). This research was completed while the second author was on a leave at the Department of Information Studies, University of Sheffield, UK, fall 2000.

References

Allan, J., Callan, J., Croft, B., Ballesteros, L., Broglio, J., Xu, J., & Shu, H. (1997). INQUERY at TREC 5. In: The Fifth Text Retrieval Conference (TREC-5), Gaithersburg, MD. Available: http://trec.nist.gov/pubs/trec5/t5_proceedings.html

Beaugrande, R. de. (1980). Text, discourse, and process: toward a multidisciplinary science of texts. Norwood: Ablex Publishing Corp.

Broglio, J., Callan, J., & Croft, W.B. (1994). Inquery system overview. In: Proceedings of the TIPSTER Text Program, Phase I (pp. 47-67).

Conover, W.J. (1980). Practical non-parametric statistics. New York: John Wiley & Sons.

Grefenstette, G. (1998). The problem of cross-language information retrieval. In: G. Grefenstette (Ed.), Cross-Language Information Retrieval (pp. 1-9). Boston: Kluwer Academic Press.

Harman, D. (1994). Overview of the Second Text REtrieval Conference (TREC-2). In: The Second Text REtrieval Conference (TREC-2), Gaithesburg, MD.
Available: http://trec.nist.gov/pubs/trec2/papers/txt/01.txt

Kekäläinen, J., & Järvelin, K. (1998). The impact of query structure and query expansion on retrieval performance. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 130-137), Melbourne, Australia.

Peat, H. J., & Willett, P. (1991). The limitations of term co-occurrence data for query expansion in document retrieval systems. Journal of the American Society for Information Science, 42, 378–383.

Pirkola, A. (1998). The effects of query structure and dictionary setups in dictionary-based cross-language information retrieval. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 55-63), Melbourne, Australia

Pirkola, A. (1999). Studies on Linguistic Problems and Methods in Text Retrieval: The Effects of

Anaphor and Ellipsis Resolution in Proximity Searching, and Translation and Query Structuring Methods in Cross-Language Retrieval. PhD Dissertation. Acta Universitatis Tamperensis, 672, University of Tampere, Department of Information Studies.

Pirkola, A., Keskustalo, H., & Järvelin, K. (1999). The Effects of Conjunction, Facet Structure, and Dictionary Combinations in Concept-based Cross-language Retrieval. Information Retrieval, 1, 217-250.

Salton, G., & McGill, M.J. (1983). Introduction to Modern Information Retrieval. New York, NY: McGraw-Hill.

Salton, G. (1989). Automatic text processing: The transformation, analysis, and retrieval of information by computer. Reading, Mass.: Addison-Wesley.

Exh. 1		Exh. 2		Exh. 3		Exh. 4
a	→	ab	→	abc	→	abcd
a	→	ab	→	abd	→	abcd
a	→	ac	→	abc	→	abcd
a	→	ac	→	acd	→	abcd
a	→	ad	→	abd	→	abcd
a	→	ad	→	acd	→	abcd
b	→	ab	→	abc	→	abcd
b	→	ab	→	abd	→	abcd
b	→	bc	→	abc	→	abcd
b	→	bc	→	bcd	→	abcd
b	→	bd	→	abd	→	abcd
b	→	bd	→	bcd	→	abcd
c	→	ac	→	abc	→	abcd
c	→	ac	→	acd	→	abcd
c	→	bc	→	abc	→	abcd
c	→	bc	→	bcd	→	abcd
c	→	cd	→	acd	→	abcd
c	→	cd	→	bcd	→	abcd
d	→	ad	→	abd	→	abcd
d	→	ad	→	acd	→	abcd
d	→	bd	→	abd	→	abcd
d	→	bd	→	bcd	→	abcd
d	→	cd	→	acd	→	abcd
d	→	cd	→	bcd	→	abcd