

Fast and Effective Query Refinement

Bienvenido Véllez¹, Ron Weiss¹, Mark A. Sheldon², David K. Gifford¹

¹ Programming Systems Research Group
MIT Laboratory for Computer Science

² Lotus Development Corporation

Abstract

Query Refinement is an essential information retrieval tool that interactively recommends new terms related to a particular query. This paper introduces *concept recall*, an experimental measure of an algorithm's ability to suggest terms humans have judged to be semantically related to an information need. This study uses *precision improvement* experiments to measure the ability of an algorithm to produce single term query modifications that predict a user's information need as partially encoded by the query. An *oracle* algorithm produces ideal query modifications, providing a meaningful context for interpreting precision improvement results.

This study also introduces RMAP, a fast and practical query refinement algorithm that refines multiple term queries by dynamically combining precomputed suggestions for single term queries. RMAP achieves accuracy comparable to a much slower algorithm, although both RMAP and the slower algorithm lag behind the best possible term suggestions offered by the oracle. We believe RMAP is fast enough to be integrated into present day Internet search engines: RMAP computes 100 term suggestions for a 160,000 document collection in 15 ms on a low-end PC.

1 Introduction

Formulating precise and effective queries in information retrieval systems has always been a difficult task, even for experienced users. Several factors contribute to this problem. The task of formulating an effective query requires the user to predict which terms appear in documents relevant to the information need. In addition, users want to avoid retrieving irrelevant documents due to a query that is underspecified or contains ambiguous terms. As a result, users of an information retrieval system with a large corpus are often faced with the task of manually sifting through very large and often inappropriate result sets. For example, Internet search engines such as AltaVista often return tens of thousands of hits for a query.

Query Refinement is the incremental process of transforming a query into a new query that more accurately re-

This work was supported by the Defense Advanced Research Projects Agency and the Department of the Army under contract DABT63-92-C-0012.

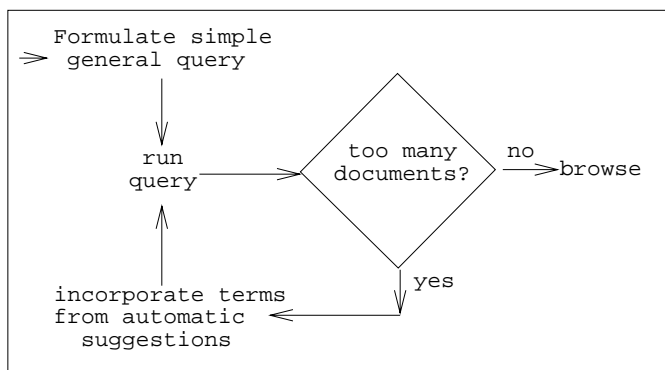


Figure 1 The query refinement paradigm

fects the user's information need. Figure 1 illustrates the general query refinement paradigm. The paradigm involves an interactive sequence of query transformations that require minimal user effort. Users searching very large information systems, such as those indexing the World Wide Web, often start with naive queries that return too many documents to browse exhaustively. It is therefore essential that these systems provide tools for helping users focus their queries. A query refinement facility recommends automatically generated term suggestions that the user can employ as a guide for improving the query. The user can add suggested terms to the original query conjunctively to reduce the result set size, disjunctively to broaden the scope of the query, by negation to focus the query away from a particular topic, or in lieu of existing query terms to modify the query. Figure 2 shows refinement suggestions for the query *scheme* (a programming language developed at MIT) generated by the HyPursuit prototype search tool[26].

This paper introduces *concept recall*, an experimental measure of an algorithm's ability to suggest terms that are semantically related to the user's information need. The experimental results reported in this study are based on the proportion of human selected keywords appearing in Tipster topics [14] that were suggested by the algorithms under investigation. An algorithm that automatically generates many of these concepts is more likely to be accepted by users because it is able to produce semantically related terms.

This study also uses *precision improvement* experiments to measure the ability of an algorithm to produce single term query modifications that predict a user's information need as partially encoded by the query. The algorithms focus on single term query modifications in order to minimize the user's effort. Our data suggests that even short query transforma-

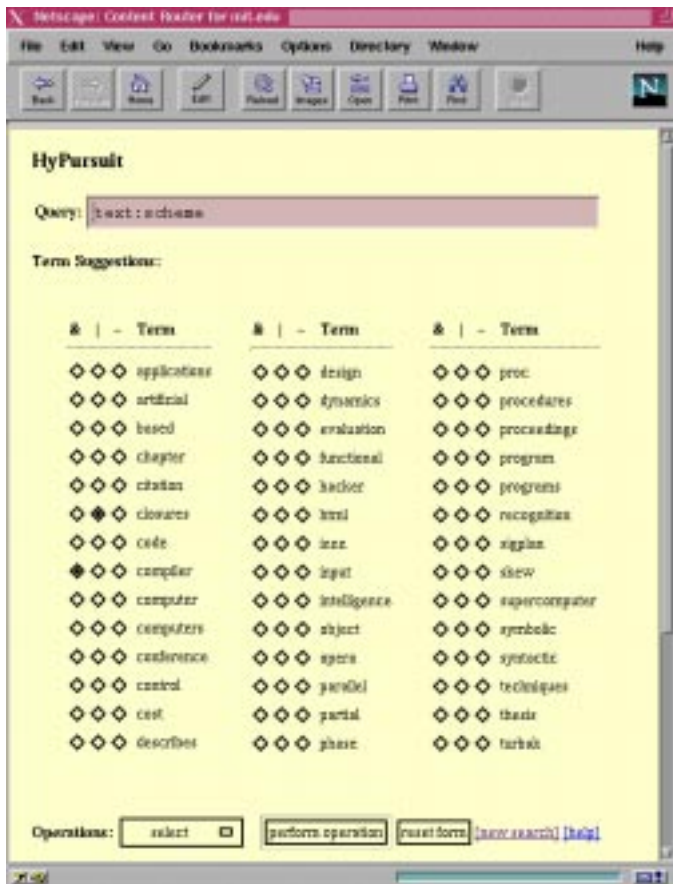


Figure 2 Sample query refinement suggestions

tions are useful in focusing both naïve, underspecified queries as well as more advanced queries. An *oracle* algorithm serves as an upper bound, providing a meaningful context for interpreting precision improvement results. The oracle produces the best single term query modifications given a set of relevance judgments. An interesting result is that, in our data set, more than half of the human generated terms that are semantically related to the information need actually decrease precision.

This paper compares the accuracy and performance of two algorithms using concept recall and precision improvement experiments. DM dynamically extracts and suggests terms from documents in the query result set. RMAP dynamically combines precomputed suggestions for single-term queries to refine multiple term queries. RMAP is shown to be fast and effective, generating 100 refinement suggestions in under 15 ms in a collection of 164,000 documents using a low-end PC. RMAP achieves accuracy comparable to the much slower DM algorithm, which requires over 2 seconds to generate its suggestions for the same corpus and queries. The performance of a query refinement algorithms is especially important for low-precision, underspecified queries. For queries with an initial precision of 0–20%, the percentage of DM term suggestions that enhance precision is 25, while for RMAP it is 22. For these queries, the precision-enhancing terms suggested by DM improve precision by 61% of the best possible average improvement for the same num-

ber of terms, and for RMAP’s terms it is 58%. Both algorithms fall short of the best possible term suggestions offered by the oracle: DM is able to suggest 43% of all terms that improve precision by 10% or more, while RMAP suggests 37% of these terms.

In the remainder of this paper we describe related work (Section 2), present the two query refinement algorithms that are the object of this study (Section 3), describe our experimental results (Section 4) and offer conclusions and possible avenues for future work (Section 5).

2 Related Work

Our interest in query refinement is based on past experience with the Community Information System [10, 9] as well as with the Discover [22, 6, 21, 24] and HyPursuit [26] network search engines. In the Community Information System, a simple theorem prover assists the user in choosing query terms that guarantee the query can be satisfied at available news wire databases. Discover and HyPursuit use a variant of the DM query refinement algorithm discussed in Section 3.

Related work can be broken down into two broad categories: mechanisms for *query modifications* and *query evaluation techniques*.

Query Modifications

Most prior work on query modifications has focused on *automatic query expansion* [3, 7, 17, 18], the addition of terms to a query to enhance recall. Query expansion has been done using global (i.e., thesauri [20, 8]) and local document analysis (i.e., result sets [27]) as well as relevance feedback [19, 12, 7]. These techniques represent points on a spectrum: fully automatic query expansion adds related and subsuming terms to the query according to a thesaurus or terms in highly ranked documents from the result set, with no intervention on the part of the user. The thesaurus itself may be automatically or manually generated. Relevance feedback, on the other hand, requires the user to weed through the result set and manually identify relevant documents (and in some cases documents to avoid) whose terms are added to (or removed from) the query.

Both query expansion and relevance feedback have traditionally focused on recall enhancement, while our work focuses on precision. More importantly, query refinement takes a position between these two levels of automation. Namely, query refinement automatically (and quickly) peruses the documents in a result set and selects terms to be added to the query. However, the term suggestions are added only explicitly by the user. This allows the user to refine the information need, to learn about the data in the collection by browsing the list of suggestions, to decide what suggestions are consistent with the actual information need, and even to take charge of whether terms increase recall or precision. We have elsewhere described our use of automatically generated thesauri for suggesting both broader and narrower search terms [26].

Harman [11] introduces a system that offers *interactive query expansion* by generating term suggestions using relevance feedback, nearest neighbors, and term variants of original query terms. The study evaluates the effectiveness of different algorithms in suggesting terms by expanding the original queries with the top 20 suggestions, and then measuring the resulting improvement in precision. This evalua-

tion mechanism does not consider the effects of single term (or short) query modifications, which should be the focus of query refinement. Smeaton and van Rijsbergen [25] also use relevance feedback and nearest neighbor to generate term suggestions.

The Open Information Locator project [15] has more recently taken up query refinement. They use two techniques: non-monotonic reasoning which combines user-directed relevance feedback and term co-occurrence analysis, and query by navigation which allows the user to browse a hierarchy of more general and more specific terms. The notion of browsing a hierarchical representation of a query result set is common to Scatter/Gather [5], the Content Routing System and HyPursuit. HyPursuit also offers a browsable set of more general and more specific terms, though the browsing tools were limited and the hierarchy was a simple two-level thesaurus.

A group at IBM has also focused on term suggestions based on relevance feedback [2]. This system allows users to annotate result documents as relevant or ‘trash’ and then tries to form a boolean query that distinguishes the two sets. The selection algorithm uses a unique scoring method for determining how much terms improve a query. This scheme also requires the user to look through the result set and find positive and negative example documents.

Query Evaluation Techniques

Recall and precision are standard evaluation metrics in information retrieval [20]. They have consequently been used in evaluating relevance feedback [13] and thesaurus-based query expansion techniques [27]. We are not aware of any work using concept recall to evaluate query refinement algorithms. The techniques here are also unique in their focus on single-term query modifications and on measurements that take into account initial query precision.

Aalbersberg [1] proposes a new user interface for incremental relevance feedback and measures the effectiveness of competing techniques in several standard collections. This study focuses on precision. Qiu and Frei [17] measure recall-precision and usefulness of query expansions based on a similarity thesaurus constructed from the corpus. This work uses fully automatic query expansion. Efthimiadis [7] describes a user study in which six algorithms were used to generate query expansion terms. Users evaluated which terms might be useful for augmenting the query.

3 Query Refinement Algorithms

This section introduces two query refinement algorithms that we have implemented and evaluated. Section 3.1 presents a generic query refinement algorithm, DM, based on our prior work [23, 6, 26] that can be instantiated by selecting a term weighting scheme. Section 3.2 describes and analyzes RMAP, a new and faster query refinement algorithm. Section 3.3 analyzes the time and space complexity of DM and RMAP.

3.1 The DM Algorithms

Figure 3 illustrates the operational flow of the DM algorithms. Given a query and a corpus, DM first finds all documents matching³ the user query. It then combines and ranks

³This paper is purposely vague about the definition of ‘‘matching’’. The algorithms described are independent of the specific matching

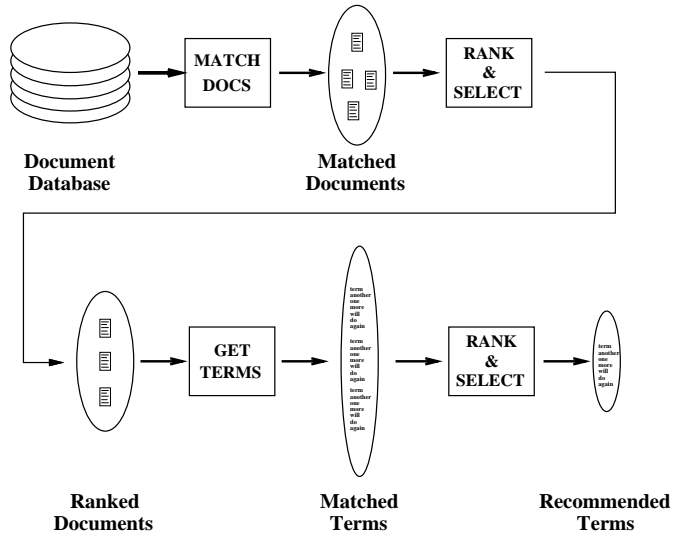


Figure 3 Generating DM Query Refinement Suggestions

the terms in these documents, and finally displays the highest ranked terms as query refinement suggestions. The distinguishing parameter for the DM algorithms is the weight function used to select the particular subset of terms. Thus DM_{df} , DM_{tf} , and DM_{nfx} refer to DM instantiated with the W_{df} , W_{tf} , and W_{nfx} term weighting functions respectively. These functions are described below.

More precisely, DM_{fcn} operates according to the following steps:

Let

- C = document corpus
- q = user query
- r = number of matching documents to consider
- n = number of suggestions to display

$W_{fcn}(S)$ = algorithm specific weight of term set S

1. Compute the set of documents $D(q) \in C$ that match the query q .
2. Select a subset $D_r(q)$ of top r matching documents
3. Compute the set of terms $T(q)$ from the documents $D_r(q)$ such that $T(q) = \{t | \exists d \in D_r(q) : t \in d\}$ where d is a document and t is a term.
4. Compute the subset S of n terms from $T(q)$ with the highest weight $W_{fcn}(S)$.
5. Present S to the user as the set of term suggestions.

The goal of a weight function $W_{fcn}(S)$ is to approximate the effectiveness of the terms in the selected suggestion set S in helping the user focus the query. This paper studies weight functions where the weight $W_{fcn}(S)$ is the sum of the individual weights of each term in S . In this case, the selected set S of term suggestions consists of the top n ranked terms in $T(q)$. For each algorithm fcn below, the function $w_{fcn}(q, t)$ computes the individual weight of a term t in refining query q . We are currently investigating other weight functions that consider the weight of a term set as a whole. For example, an alternate algorithm can weight a set of suggestions based on the number of matching documents that contain at least one of the terms from the set.

mechanism made by different query models.

weight	function
$w_{df}(q, t)$	$\sum_{d \in (D(q) \cap D(t))} df_{t,d}$
$w_{tf}(q, t)$	$\sum_{d \in (D(q) \cap D(t))} tf_{t,d}$
$w_{nfx}(q, t)$	$\sum_{d \in (D(q) \cap D(t))} 0.5 + \frac{0.5(tf_{t,d})}{\max_{i \in d} tf_{i,d}} * \log\left(\frac{N}{ D(t) }\right)$

Figure 4 Term weighting functions

Figure 4 shows the three term weighting functions studied in this paper. The first function, w_{df} assigns to each term t a weight equal to the document frequency of the term in the set of documents matching the query. The Discover system [6] uses w_{df} and provided anecdotal evidence that this approach generates useful query suggestions based on a collection of WAIS [16] document headlines.

The second weight function, w_{tf} , is based on term frequencies in the matching documents. The third weight function, w_{nfx} , adjusts term frequency weights with inverse document frequencies to reduce the weights of terms that appear frequently in the corpus. w_{nfx} also normalizes term weights by document sizes to counteract the increase in the weights of terms that appear in large documents. Ongoing research is also considering term weight functions that assign more weight to terms that appear in documents ranking higher with respect to the query.

3.2 RMAP: Fast Query Refinement

RMAP achieves greater speed than the DM algorithms by statically precomputing a substantial amount of work that is performed dynamically by DM. RMAP thus uses more storage space to avoid online computation. Specifically, RMAP operates in two phases: an offline corpus preprocessing phase, and a dynamic phase in response to a user request for term suggestions. During the offline phase, the algorithm generates a data structure that maps each term t in the corpus to an RSET, which is the set of m terms that the DM algorithm would suggest given the single term query t . The online computation only consists of term lookup and ranking. Because the RMAP data structure can be updated incrementally in response to changes in the composition of the corpus, there is no need to rerun the entire offline process when changes occur.

The hypothesis underlying RMAP is that relatively small values of m will not result in significant decreases in accuracy versus the purely dynamic algorithms. Preliminary results suggest that reducing m to values close to, or even below, the number of required term suggestions does not degrade the performance of RMAP significantly. Section 4 presents experimental and anecdotal evidence that RMAP with a small m of 100 achieves accuracy comparable to DM with r of 100. The length m of each RSET can be adjusted to meet different storage requirements. We are currently investigating how various factors affect the optimal choice of m , including corpus size, disk space, performance constraints, and term distribution.

Figure 5 illustrates the dynamic phase of RMAP that is performed in response to a request for query refinement suggestions. For each term in the user's query, the algorithm looks up the corresponding RSET containing both terms and

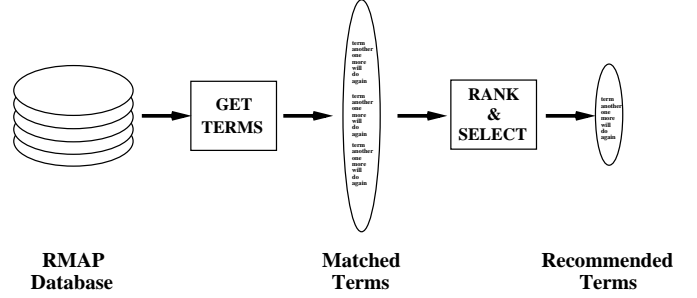


Figure 5 The Dynamic Phase of RMAP

the weights assigned to them during the static phase. All the terms are merged into a single set of terms. The new weight for each term is the sum of its weights from the RSETs. The highest weighted n terms are displayed to the user as suggestions.

3.3 Time/Space Complexity

This section compares the time and space complexity of RMAP and DM. The average running time for DM is the sum of the average running times of its four phases:

Let
 n_q = number of terms in query q
 \overline{df} = average number of documents matching a term
 r = number of matching documents to consider
 \overline{dt} = average number of terms per document
 n = number of suggestions to display

- MATCH DOCS takes on average $n_q \cdot \overline{df}$
- RANK & SELECT takes on average $(n_q \cdot \overline{df}) \cdot \log(n_q \cdot \overline{df}) + r$
- GET TERMS takes on average $r \cdot \overline{dt}$
- RANK & SELECT takes on average $r \cdot \overline{dt} \cdot \log(r \cdot \overline{dt}) + n$

The average running time for RMAP is the sum of the average running times of its two online phases:

- GET TERMS takes on average $n_q \cdot m$
- RANK & SELECT takes on average $(n_q \cdot m) \cdot \log(n_q \cdot m) + n$

Any optimization in query processing technology that can be applied to the MATCH DOCS and GET TERMS phases of DM can also be applied to the dynamic lookup phase of RMAP. For example, a search engine may extract and rank only a portion of the results. Furthermore, a system may substantially improve running times by keeping index and RMAP data structures in main memory rather than on disk. Our implementations of RMAP and DM use exactly the same code for database lookup and term ranking/selection tasks.

The improvement in time complexity of RMAP's dynamic phase over DM comes at a price in storage requirements. Specifically, in a corpus with T unique terms RMAP requires space of size $O(T \cdot m)$ in addition to the space required by

# of Documents	\bar{df}	\bar{dt}	Unique Terms
8,466	28.87	138.76	40,684
84,660	91.31	141.64	131,314
164,597	180.92	203.98	185,524

Table 1 Collection Statistics

# of Queries	Avg Query Length	Relevant Docs/Query		
		8,466	84,660	164,597
150	2.08	17.30	187.64	364.83
200	2.84	18.73	203.36	390.36

Table 2 Query Statistics

DM. Both RMAP and DM require space to store an inverted file and additional storage to hold each document's terms. Section 4 will present space requirements for both RMAP and DM for a sample of collections of different sizes.

4 Evaluation of Algorithms

This section compares the performance of the algorithms presented in Section 3. It presents two experimental designs, accompanied with results, for evaluating the accuracy of query refinement algorithms. Both methods for accuracy evaluation measure the effectiveness of the suggested terms within the context of a subset of the Tipster collection [14]. The first evaluation method, *concept recall*, measures the proportion of suggested terms that appear in a list of human generated relevant terms. This measures an algorithm's ability to suggest terms related to a given query. The second approach, *precision improvement* measures the effectiveness of individual terms in modifying an underspecified query to better approximate the user's information need. The discussion proceeds by comparing the run-time performance of the different algorithms and offering an analysis of their storage requirements. It concludes by examining sample suggestions from our prototype. The experiments were conducted using an internally developed information retrieval system that supports boolean queries and document ranking. Our ranking algorithm differs from classical ranking algorithms such as [12] by ignoring document size and by giving primary consideration to the number of query terms a document contains.

4.1 Collections and Query Sets

The experimental framework uses a standard collection that consists of documents, sample queries, and human generated relevance judgments. The relevance judgments encode the information need corresponding to a query by enumerating the documents in the corpus relevant to the information need. Tables 1 and 2 show relevant statistics about the collections used. Each experimental collection contains AP press articles published during 1988 and 1989 appearing in volumes 1 and 2 of the Tipster CDRoms. The 164k collection contains all the articles published during these two years. The 84k collection contains all articles published in 1989. Finally, the 8k collection contains the first 8,466 articles published in 1989.

Number 008
Domain International Economics
Topic Economic Projections
Description Document will contain quantitative projections of the future value of some economic indicator for countries other than the U.S.
Narrative To be relevant, a document must include a projection of the value of an economic indicator (e.g., gross national product (GNP), stock market, rate of inflation, balance of payments, currency exchange rate, stock market value, per capita income, etc.) for the coming year, for a country other than the United States.
Concepts
<ol style="list-style-type: none"> 1. inflation, stagflation, indicators, index 2. signs, projection, forecast, future, 3. rise, shift, fall, growth, drop, expansion, slow-down, recovery, 4. billions, 5. Not U.S.

Figure 6 An Example Tipster Topic

Figure 6 shows a Tipster topic that posted typical performance with respect to the measures of accuracy in this section. A Tipster topic includes the following fields: a numeric identifier, a domain description, a topic field, a description of the query, a narrative to further elaborate the information need, and concepts related to the query.

The experiments below construct queries using the terms from the *topic* field. Many researchers use the description and narrative fields as the basis of the query because it more accurately reflects the information need specified in the detailed narrative. However, we chose to use the topic field because it better represents queries made by naive users in large information systems. For example, [4] determined that searches on the World-Wide Web have on average only two terms. Using the topic field results in an average query length of 2.08 for the set of 150 queries and 2.84 for the set of 200 queries. Query refinement attempts to bridge the gap between the short, underspecified queries users initially submit and the more detailed information needs that users have.

A query can be an encoding of many different information needs. This is especially true of the short queries frequently posed by naive users, which we model using the Tipster topic field. The accuracy results reported below are measured with respect to Tipster relevance judgments based on specific information needs not fully encoded in our queries. If a query refinement algorithm performs well over a range of queries and information needs, then it is likely that this refinement algorithm generally suggests useful query modifications, and thus achieves its goal in practice.

4.2 Concept Recall

Concept recall is a measure of the ability of an algorithm to recommend terms that are semantically related to the user's information need. The terms in a Tipster concept field are

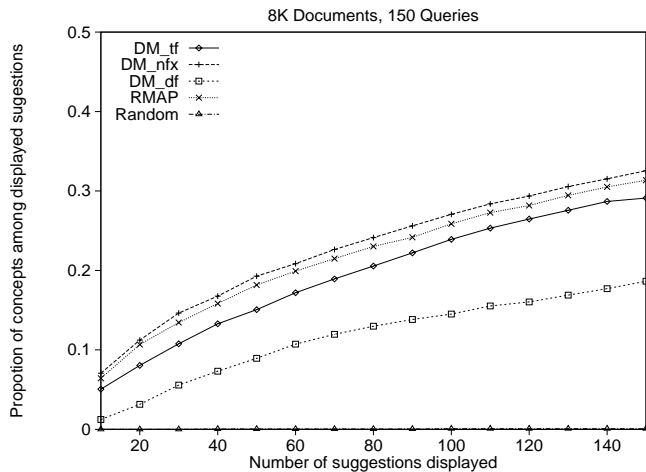


Figure 7 Proportion of Concepts in Suggestions (8K Docs)

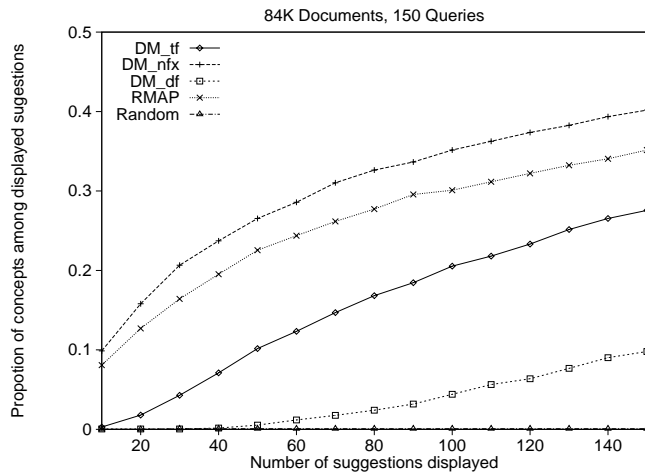


Figure 8 Proportion of Concepts in Suggestions (84K Docs)

keywords judged by humans to be semantically related to the information need of the topic. An important goal of query refinement is also to suggest terms related to the user’s information need. Thus, an algorithm that automatically generates many of these concepts is more likely to be accepted by users. The experiment described here measures this particular aspect of an algorithm’s performance by treating Tipster concept terms as human generated suggestions; it computes the number of topic concepts displayed by an algorithm divided by the total number of concepts associated with the topic.

Figures 7 and 8 illustrate the concept recall results for the different algorithms on corpora of 8466 and 84660 documents, respectively. The graphs show the average concept recall (over 150 queries) versus the number of term suggestions displayed. Concept recall ratios ignore concepts that do not appear in the corresponding collection. The average number of concepts per topic is 18.30, and on average 15.7 (2,350 total) of these appear in the smaller corpus while 16.3 (2,449 total) appear in the larger corpus. For example, for a corpus of 84,660 documents, the DM_{nfx} algorithm was able on average to automatically suggest 35% of the human generated concepts when displaying only 100 term suggestions. The faster RMAP algorithm performs better than DM_{tf} and DM_{df} and does not significantly lag behind DM_{nfx} . The graphs also include a control that suggests random terms from the corpus. The two figures illustrate an improvement in concept recall that results from increasing the size of the corpus from 8,466 documents to 84,660 documents and confirm the well known positive impact of increasing collection size on the utility of inverse document frequency (*idf*) factors.

4.3 Precision Improvement

Another approach, *precision improvement*, measures the performance of a query refinement algorithm by examining the effect of adding suggested terms to the original query. First, we compute the precision of the result set for each query in the standard collection. Then, each algorithm produces a set of term suggestions for every query. For each query, we construct a set of modified queries by adding each suggested term in turn to the original query. Finally, we measure the

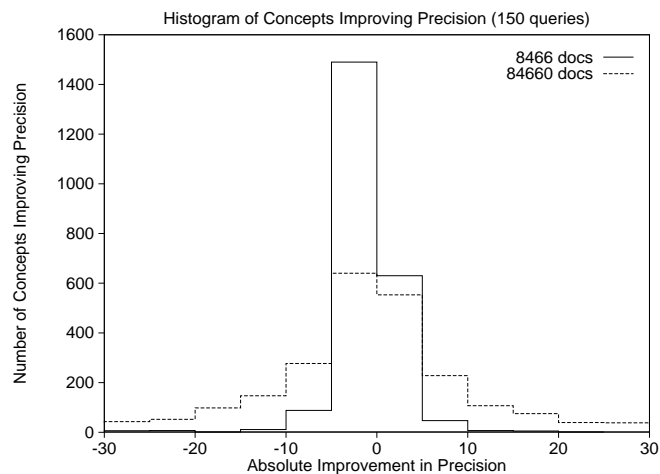


Figure 9 Effect of Concepts on Precision

precision of the result set for each of the modified queries. In computing precision, we consider only the top 100 documents matching a query. This section presents several views of this data for the DM_{nfx} and RMAP algorithms. It does not present results for DM_{df} and DM_{tf} because these algorithms have slightly lower performance than DM_{nfx} and the same running times.

The performance data for an algorithm underestimates the actual utility of its suggested terms because a suggestion may be useful even if it does not improve precision. For example, for the Tipster topic **economic projections** (presented in Section 4.1), the query suggestions **budget**, **deficit**, **loan**, and **debt** all decrease precision. Yet these terms are clearly related to the query, and moreover would be useful for information needs other than the one specified in the Tipster topic. See Section 4.4 for more example term suggestions along with a detailed discussion of this issue.

The above examples are indicative of the general pattern that semantically related terms may not necessarily improve precision. Figure 9 shows that more than half of the user generated concepts in the Tipster topics actually reduce pre-

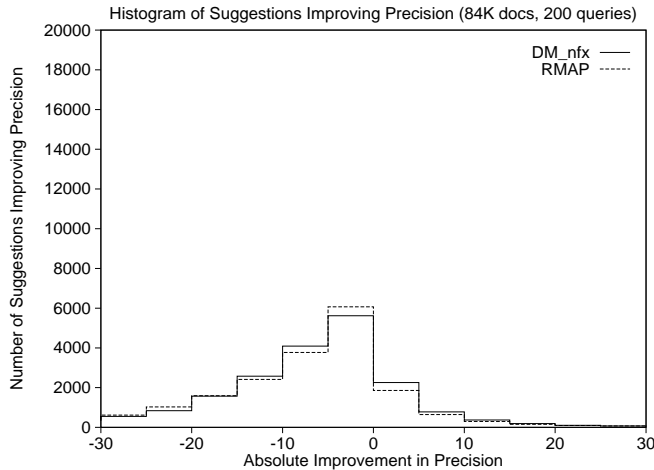


Figure 10 Distribution of Changes in Precision

cision. The graph displays the distribution of changes in precision resulting from adding individual concepts to 150 original queries. The x axis denotes the change in the number of relevant documents in the top 100 retrieved after adding a term to the original query. The y axis bars represent the number of terms that change the precision over the interval (exclusive of the lower x value and inclusive of the higher x value).

The remainder of this section presents an analysis of the distribution of the effects of single-term query modifications both in the aggregate and in relation to initial query precision.

Distribution of Changes in Precision

The histogram in Figure 10 shows the absolute changes in precision resulting from modifying each of 200 queries from the standard collection. For each original query we form 100 modified queries by adding the refinement terms suggested by RMAP and DM_{nfx} for that query. For example, out of 20,000 suggestions for all queries, DM_{nfx} suggests 2,256 terms that improve precision by adding more than 0 documents and less than or equal to 5 documents. Though RMAP is substantially faster, it performs comparably to DM_{nfx} in recommending terms that improve precision.

For comparison, Figure 11 illustrates the performance of two control algorithms, *Random* and *Oracle*. *Random* simply suggests terms from the collection at random. Intuitively, most terms in the collection are orthogonal to the query. *Random* shows that orthogonal terms are likely to have little effect on precision.

Oracle knows all documents relevant to each query and suggests the terms that improve precision the most. In this experiment, oracle extracted the terms from all documents relevant to a query, and measured the effect of each term on precision. It then suggests the top 100 terms. Oracle's performance is therefore the best that any algorithm can hope to attain. Note that even in oracle's case, 10% of the terms do not improve precision (these 1,962 terms have a value of 0).

Most of the terms suggested by RMAP and DM_{nfx} are correlated to the query because they have a noticeable positive or negative effect on precision. Oracle shows that there

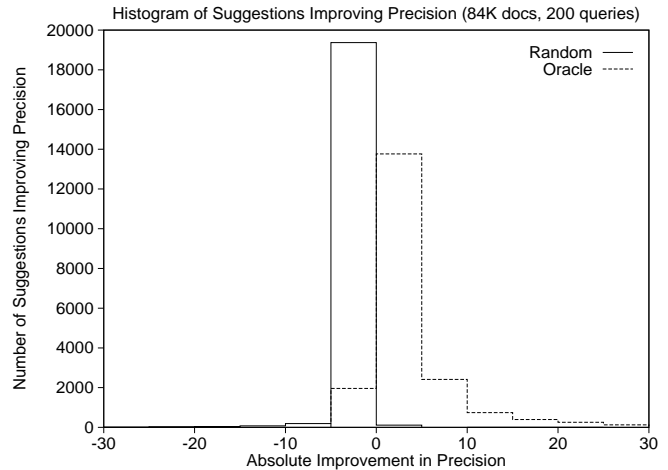


Figure 11 Distribution of Changes in Precision

are only a few terms that substantially improve precision, and RMAP and DM_{nfx} suggest a significant portion of them. For example, DM_{nfx} is able to suggest 43% of all terms that improve precision by 10% or more, while RMAP suggests 37% of these terms. RMAP and DM_{nfx} are less able to suggest a significant proportion of the terms that improve precision modestly.

Improvement versus Initial Query Precision

Queries with low initial precision have a lot of room to improve but do not carry much information. Conversely, queries with high initial precision will be difficult to improve upon, but they have a high information content. This section investigates the combined effect of the two factors of information content and query specificity.

Figure 12 shows the percentage of suggested terms that improve precision versus the precision of the initial query. For example, DM_{nfx} on average suggests 29 (out of 100) terms that improve precision for queries with an initial precision of 10 – 20%. For queries with an initial precision of 0 – 20%, the percentage of DM_{nfx} term suggestions that enhance precision is 25%, while for RMAP it's 22%. The average performance of DM_{nfx} over all queries is 20%; the average for RMAP is 16%. We suspect that since RMAP is a global algorithm and DM is a local algorithm (in the sense of [27]), DM is better able to take advantage of information rich queries.

Figure 13, which considers only suggestions that improve precision, shows the number of new relevant documents out of 100 added to the result set versus initial query precision. For example, RMAP's suggestions that improve precision result in an average of 9 additional relevant documents among the top 100 for queries with an initial precision of 10 – 20%. The average across all initial query precisions for both algorithms is 7. This figure does not show the performance of *Random* because the average percentage of its suggestions that improve precision is low enough to render the algorithm useless.

The oracle values depicted in Figure 13 represent ideal performance. The oracles for DM_{nfx} and RMAP, denoted by $oracle(\#DM_{nfx})$ and $oracle(\#RMAP)$ respectively, generate as many precision-enhancing terms as the correspond-

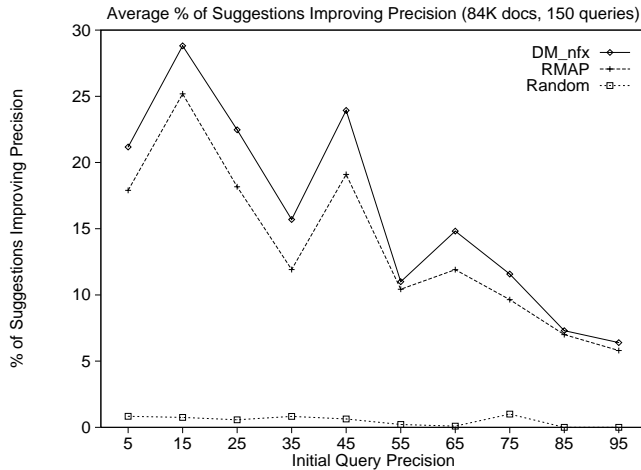


Figure 12 % of Suggestions Improving Precision

ing algorithm. Thus, if RMAP recommends n precision-enhancing terms for a particular query, the oracle for RMAP suggests its top n terms for the same query. The performance of RMAP and DM_{nfx} converges to the performance of their respective oracles as initial query precision increases. For queries with an initial precision of 0 – 20%, the average improvement in precision of terms suggested by DM_{nfx} is 49% of the best possible average improvement for the same number of terms, and for RMAP’s terms it’s 51%. For queries with an initial precision of 70 – 80%, DM_{nfx} achieves 70% of the best possible average improvement and RMAP achieves 71%. The overall average improvement for precision-enhancing terms suggested by DM_{nfx} is 7%, which is 61% of the best possible for the same number of terms, while the overall average for RMAP is also 7%, which is 58% of the best possible.

4.4 Example Term Suggestions

Figure 14 shows the actual terms suggested by the DM_{nfx} and RMAP algorithms in response to the Tipster topic from Figure 6, which has an initial precision of 17%. The column labeled “Oracle Rank” contains the position of each term among the set of oracle suggestions. For each term, the columns labeled “In DM_{nfx} ?” and “In RMAP?” indicate whether the algorithm suggests the corresponding term. The “+ Docs” column displays the total number of additional relevant documents retrieved by the modified query after adding the corresponding term. The last column titled “+% Docs” displays the same information as a percentage of the number of relevant documents retrieved by the original query. All the terms are shown stemmed. As was alluded in the discussion regarding Figures 10 and 11, both algorithms retrieve many of the top precision improving terms but are less able to retrieve terms that only modestly improve precision,

A term yielding a decrease in precision may not necessarily be a bad suggestion. Figure 15 displays terms suggested by the DM_{nfx} algorithm which yield a decrease in precision when added to the query from Figure 6 Terms **fiscal** and **budget** are semantically related to the query and still decrease precision. Moreover, notice that terms **china** and **japan** can focus the query among alternative search paths.

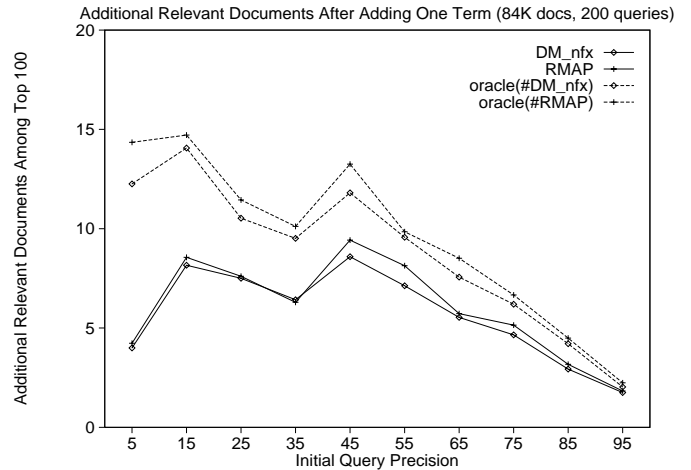


Figure 13 Average Improvement in Precision

Term	Oracle Rank	In DM_{nfx} ?	In RMAP?	+ Docs	+% Docs
forecast	1	x		22	129
growth	2	x	x	19	111
percent	"	x	x	19	111
rise	5	x		18	105
inflat	7	x	x	15	88
rate	"	x	x	15	88
slow	10	x		13	76
economi	13	x	x	11	64
economist	"	x		11	64
increas	19	x		9	52
gross	21	x		8	47
price	26		x	7	41
annual	36	x		6	35
busi	39	x		5	29
expect	46	x		4	23
expand	57	x		3	17
report	"	x		3	17
spend	"	x		3	17
feder	"		x	3	17
unemploy	90	x		2	11
interest	112	x		1	15
administr	"		x	1	15
next	"	x		1	15

Figure 14 Suggestions that add relevant documents

They happen not to improve the precision of the original query because they do not narrow the query in the direction of the particular information need as described in English in the topic’s description and narrative fields. These last two suggestions focus the query to specific countries of interest. However, the information need as stated in the Tipster query specifically requests documents including “a projection of the value of an economic indicator”. The documents containing the above three suggestions may not satisfy this requirement. These alternative search paths are not explicitly represented in the query, and thus it is unreasonable to penalize the refinement algorithm for showing suggestions that focus the query towards such paths. In addition, recall the discussion in Section 4.3 that displays suggestions that are clearly related to the query, but nonetheless decrease precision. For this reason, the experiment shown in Fig-

Term	+	+ %	Term	+	+ %
	Docs	Docs		Docs	Docs
nation	-1	-5	between	-12	-70
product	-2	-11	world	-13	-76
improv	-2	-11	provid	-13	-76
deficit	-2	-11	premier	-13	-76
should	-3	-17	foreign	-13	-76
industri	-3	-17	fiscal	-13	-76
howev	-3	-17	expert	-13	-76
third	-4	-23	develop	-13	-76
suppli	-4	-23	social	-14	-82
polici	-4	-23	problem	-14	-82
figur	-4	-23	power	-14	-82
estim	-4	-23	plan	-14	-82
budget	-4	-23	loan	-14	-82
futur	-5	-29	germani	-14	-82
bank	-5	-29	enterpris	-14	-82
term	-7	-41	debt	-14	-82
senior	-7	-41	relat	-15	-88
privat	-7	-41	reform	-15	-88
gener	-7	-41	meet	-15	-88
export	-7	-41	japan	-15	-88
construct	-7	-41	intern	-15	-88
will	-8	-47	communist	-15	-88
invest	-8	-47	assist	-15	-88
govern	-9	-52	western	-16	-94
dam	-9	-52	program	-16	-94
cost	-9	-52	polit	-16	-94
condition	-9	-52	offici	-16	-94
would	-10	-58	minist	-16	-94
presid	-10	-58	fund	-16	-94
countri	-10	-58	financ	-16	-94
capit	-10	-58	european	-16	-94
build	-10	-58	cooper	-16	-94
billion	-10	-58	chines	-16	-94
auster	-10	-58	china	-16	-94
work	-11	-64	agricultur	-16	-94
million	-11	-64	visit	-17	-100
told	-12	-70	east	-17	-100
shortag	-12	-70	aid	-17	-100
help	-12	-70			

Figure 15 DM_{nfx} Terms that Decrease Precision

ure 13 only considers precision-enhancing suggestions when computing the average improvement in precision.

4.5 Runtime Measurements

This section presents experimental results comparing the average total execution times of the DM_{nfx} and RMAP query refinement algorithms. Each query was refined by each algorithm a total of 5 times. The maximum and minimum runtimes were discarded, and the average of the remaining three measurements is reported below as average execution time. All the execution time data was generated on a virtually unloaded Intel PC with a 75 MHz Pentium and 96MB of memory running the Linux operating system. The data are stored on a 32 GB Level 5 RAID system connected via a Ultra Fast and Wide SCSI bus.

Figure 16 illustrates the dramatic reduction in execution time that RMAP with m of 100 achieves versus the DM algorithms with r of 100. As suggested by the runtime analysis presented in section 3, the experimental data confirms that the DM algorithm is much more sensitive to corpora size than RMAP. For example, for a corpus of 84660 documents, RMAP suggests 100 terms in an average time of 10ms over 150 queries, while the DM algorithm requires 577ms for the

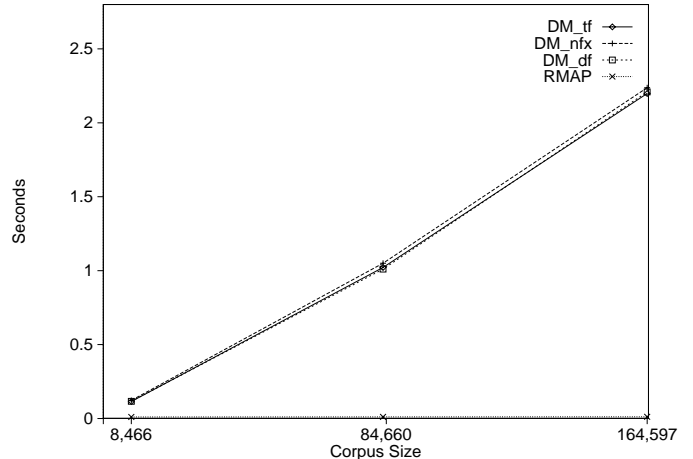


Figure 16 Timing Performance for Algorithms

Algorithm	# of Docs in Collection		
	8460	84660	164597
DM_{nfx}	20	210	607
RMAP	41	135	197

Figure 17 Size (in MB) of Databases

same task. This implies that on average, RMAP is capable of refining almost 58 queries in the amount of time it takes DM to refine one. For a corpus approximately twice as big, this ratio goes from 58:1 to 105:1.

4.6 Storage Requirements

The dramatic reduction in response time achieved by RMAP comes at the cost of increased storage space. This section presents the actual sizes of the DM and RMAP data structures for the implementations analyzed above. For DM, the size includes the size of the inverted file that maps terms to matching documents plus the size of the inverted file that maps documents to the terms they contain. For RMAP, the size includes the size of the inverted file mapping terms to the precomputed suggestions. All inverted files are kept hashed.

Table 17 shows the amount of storage required by DM_{nfx} and RMAP with $m = 100$ for collections of different sizes. RMAP's fixed row size (m) allows a smaller rate of growth as the collection size increases.

5 Conclusions

Query Refinement is an essential information retrieval tool that interactively recommends new terms related to a particular query. This paper proposes an experimental framework for evaluating whether an algorithm suggests effective query refinements and describes RMAP, a new algorithm that computes suggestions of similar quality to computationally intensive approaches. Our accuracy and runtime measurements of RMAP suggest that it is an attractive query refinement algorithm in situations where processing time is at a premium.

Further evaluations of RMAP and DM are necessary. For example, it is important to establish the effects of different choices for parameters such as RMAP's row size and the number of documents considered in DM's MATCH DOC phase. These parameters allow an administrator to fine tune a system based on performance and space constraints.

We are also investigating enhancements to RMAP as well as new query refinement algorithms. For example, we are exploring new ways to combine and rank the suggestions generated by RMAP including taking into account term proximities. An alternate refinement algorithm can take into account the number of documents that contain terms from the set of suggestions. Finally, we are interested in embedding query refinement algorithms in scalable hierarchical information systems using lossy index compression techniques.

References

- [1] IJsbrand Jan Aalbersberg. Incremental relevance feedback. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 11–22, Copenhagen, Denmark, June 1992.
- [2] R. C. Barrett and E. J. Selker. Finding what i am looking for: An information retrieval agent. Technical Report RJ 9816, IBM Research Division, Almaden Research Center, San Jose, California, May 1994.
- [3] Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. Automatic query expansion using SMART:TREC3. In Donna Harman, editor, *Proceedings of the Third Text Retrieval Conference (TREC-3)*, pages 69–80, Gaithersburg, MD, November 1994. NIST and ARPA.
- [4] W. B. Croft, R. Cook, and D. Wilder. Providing government information on the internet: Experiences with THOMAS. In *Digital Libraries Conference DL95*, pages 19–24, Austin, Texas, June 1995.
- [5] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *15th Annual International SIGIR*, pages 318–329, Denmark, June 1992.
- [6] Andrzej Duda and Mark A. Sheldon. Content routing in networks of WAIS servers. In *Proceedings of the 14th International Conference on Distributed Computing Systems*, pages 124–132, Poznan, Poland, June 1994. IEEE.
- [7] Efthimis N. Efthimiadis. A user-centered evaluation of ranking algorithms for interactive query expansion. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 146–159, Pittsburgh, PA USA, June 1993.
- [8] William B. Frakes and Ricardo Baeza-Yates, editors. *Information Retrieval: Data Structures & Algorithms*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.
- [9] David K. Gifford. Polychannel systems for mass digital communication. *Comm. ACM*, 33(2), February 1990.
- [10] David K. Gifford, John M. Lucassen, and Stephen T. Berlin. An architecture for large scale information systems. In *10th Symposium on Operating System Principles*, pages 161–170. ACM, December 1985.
- [11] Donna Harman. Towards interactive query expansion. In *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 321–331, Grenoble, France, June 1988.
- [12] Donna Harman. Chapter 14: Ranking algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*, pages 363–392. Prentice Hall, Englewood Cliffs, New Jersey, 1992.
- [13] Donna Harman. Relevance feedback revisited. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1–10, Copenhagen, Denmark, June 1992.
- [14] Donna Harman. TIPSTER information retrieval text research collection. CDROM set, Linguistic Data Consortium, 1994.
- [15] Renato Iannella, Nigel Ward, Andrew Wood, Hoylen Sue, and Peter Bruza. Digital libraries and the open information locator project. Technical Report DSTC 34, Research Data Network Cooperative Research Centre, Resource Discover Unit, Queensland, Australia, 1996.
- [16] Brewster Kahle and Art Medlar. An information system for corporate users: Wide Area Information Servers. Technical Report TMC-199, Thinking Machines, Inc., April 1991. Version 3.
- [17] Yonggang Qiu and H. P. Frei. Concept based query expansion. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 160–169, Pittsburgh, PA USA, June 1993.
- [18] Salton, E. A. Fox, and E. Voorhees. Advanced feedback methods in information retrieval. *Journal of the American Society for Informaiton Science*, 36(3):200–210, 1985.
- [19] G. Salton, E. A. Fox, C. Buckley, and E. Voorhees. Boolean query formulation with relevance feedback. Technical Report TR 83-539, Department of Computer Science, Cornell University, Ithaca, NY, January 1983.
- [20] Gerard Salton. Another look at automatic text-retrieval systems. *Comm. ACM*, 29(7):648–656, July 1986.
- [21] Mark A. Sheldon. *Content Routing: A Scalable Architecture for Network-Based Information Discovery*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, October 1995.
- [22] Mark A. Sheldon, Andrzej Duda, Ron Weiss, and David K. Gifford. Discover: A resource discovery system based on content routing. In *Proceedings of The Third International World Wide Web Conference*, Darmstadt, Germany, April 1995. Also in *Computer Networks and ISDN Systems*, Elsevier North Holland, 27(1995), pp. 953–972.
- [23] Mark A. Sheldon, Andrzej Duda, Ron Weiss, James W. O'Toole, Jr., and David K. Gifford. A content routing system for distributed information servers. Technical Report MIT/LCS/TR-578, M.I.T. Laboratory for Computer Science, June 1993.
- [24] Mark A. Sheldon, Andrzej Duda, Ron Weiss, James W. O'Toole, Jr., and David K. Gifford. Content routing for distributed information servers. In *Fourth International Conference on Extending Database Technology*, pages 109–122, Cambridge, England, March 1994. Available as Springer-Verlag LNCS Number 779.
- [25] A. F. Smeaton and C. J. van Rijsbergen. The retrieval effects of query expansion on a feedback document retrieval system. *The Computer Journal*, 26(3):239–246, August 1983.
- [26] Ron Weiss, Bienvenido Véllez, Mark A. Sheldon, Chanathip Namprempre, Peter Szilagy, and David K. Gifford. Hypersuit: A hierarchical network search engine that exploits content-link hypertext clustering. In *Proceedings of the Seventh ACM Conference on Hypertext*, Washington, DC, March 1996.
- [27] Jinxi Xu and W. Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, Zurich, Switzerland, August 1996.