Automated Query Generation For Embedded Information Retrieval

Vladimir A. Kulyukin
School of Computer Science, Telecommunications,
and Information Systems
DePaul University
243 South Wabash Avenue,
Chicago, IL 60604
vkulyukin@cs.depaul.edu

Abstract

The integration of query generation and user feedback continues to challenge information retrieval technologies. Relevance feedback, while useful to professionals, is frequently inappropriate for lay users, because the initial query generation is manual and the subsequent feedback solicitation is intrusive or inconsistent with many lay users' information needs. To provide lay users with automated query generation and nonintrusive feedback solicitation, the concept of a personal information retrieval assistant is developed. A personal information retrieval assistant is a software agent embedded in an application. The assistant allows the application in which it is embedded to become part of an infrastructure of distributed information sources. Background samples of the application's usage are collected and used in retrievals from the sources. Feedback is never solicited explicitly, and is utilized only when volunteered. Retrieval is adjusted through background sampling, anydata indexing, and dual space feedback.

1 Introduction

The integration of query generation and user feedback continues to challenge information retrieval (IR) technologies. Relevance feedback [Aalbersberg 1992; Robertson and Walker 1997], the de facto standard for generating queries from user feedback, is useful to information science professionals [Spink and Saracevic 1997], but is frequently inappropriate for lay users. Many users are not able to adequately state their information needs in queries [Burke et al. 1996]. The constant and explicit solicitation of relevance judgments interferes with many users' information-seeking behaviors [Kulyukin 1999; Kulyukin 1998]. Feedback utilization in the query space alone prohibits retrieval adjustments over multiple interactions [Kulyukin 1998; Kulyukin et al. 1998]. The focus on maximizing the number of relevant documents in every retrieval is inconsistent with many users' information needs [Jansen et al. 1998]. Since many users want to find a small subset of relevant references quickly, they are not interested in finding all relevant references, most of which are placed in the middle or at the end of a long retrieval list.

The approach to the integration of query generation and user feedback henceforth presented is called *personalized application-embedded distributed IR*. The approach extends and builds on the application-embedded distributed IR framework [Kulyukin 1999]. An IR agent is embedded in an application, e.g., a text editor. The system collects background samples of the application usage, e.g., pieces of typed text, and uses them in retrievals from distributed network resources. Feedback is never solicited explicitly, and is utilized only when the user volunteers it during routine retrieval examinations. The retrieval procedure is adjusted through background sampling, dual relevance feedback, and anydata indexing. The approach is implemented in MetaCenter, a personal information retrieval assistant embedded in Microsoft Word^{TM}[1].

The paper is organized as follows. Section 2 describes the main problems and objectives of personalized application-embedded distributed IR. Section 3 outlines the main techniques used in the approach. Section 4 gives implementational details. Section 5 presents several experiments with the system. The results of the experiments should be construed as indicative rather than definitive. Section 6 presents conclusions.

2 The Problem

The problems addressed in MetaCenter are illustrated in the following example:

```
X is a CS researcher working
on a grant proposal on
intelligent network protocols.
Due to intense competition, it
is vital that X keep abreast
of the most recent developments
in the field. While X knows
many relevant online sources,
X cannot take full advantage
of them because of their size,
dynamic nature, and lack of
adequate search tools.
```

The primary objective of the MetaCenter project is a technology for building nonintrusive, feedback-sensitive IR

[1] MetaCenter can be freely downloaded from http://www.bitmobiletechnologies.com

agents that users, such as X, embed into their applications to tap into and monitor multiple information sources, while engaged in routine usages of those applications. Another objective of the project is the development of distributed IR protocols that allow information seekers and providers to share not only content but also means to manipulate that content. The technology is motivated by a growing number of distributed information infrastructures that offer users a wealth of data, but few tools to timely put it to use.

3 A Solution

A personal application-embedded IR assistant, such as MetaCenter, is deployed through *resource subscription* and *mode selection*. During resource subscription, the user specifies distributed resources from which information will be retrieved. For example, X selects several online collections of documents on network protocols. During mode selection, the user specifies how the application will be used. For example, X may select the text mode, thus telling the system that the application, i.e., MS-Word, will be used for composing documents. X may also select the HTML mode to edit HTML pages. As X composes documents, MetaCenter takes background text samples and uses them as queries to the distributed collections. The retrievals are examined and critiqued by X at X's convenience. The agent's operation relies on the following techniques: *distributed indexing*, *background sampling*, and *dual space feedback*.

Distributed indexing is a protocol implemented on top of the Common Object Request Broker Architecture (CORBA). CORBA allows objects executing on different systems to invoke methods on each other [Vogel and Duddy 1998]. All interobject communication is managed by Object Request Brokers (ORB's) through the Internet Inter-ORB Protocol (IIOP) implemented on top of TCP/IP. The protocol connects the embedded agent to distributed network resources. Every resource to which the user subscribes is a server offering a set of indexing and retrieval services. The agent can query the server about the services it offers. For example, if the server offers several indexing methods, the agent may select one of them. Unlike with other approaches to Internet IR [Burke et al. 1996; Jansen et al. 1998; Kulyukin 1998], there is no central index repository for every retrievable document. Collection indexing and maintenance are delegated to the servers. The agent communicates with the servers on an as needed basis.

During background sampling, as the user interacts with the application, background samples of the application's usage are collected by the system and used to generate queries. Three types of sampling are distinguished: *random*, *structured*, and *selective*. In random sampling, all parts of a sample are chosen randomly, e.g., a set of terms randomly chosen from a free text. In structured sampling, samples are components of a known template, e.g., abstracts and bibliographies of a document. When X selects the document mode, the templates for technical reports and papers are activated. Structured sampling is based on the assumption that some components of a document structure carry more information than others [Kulyukin, et al. 1996]. Random sampling is used when the mode has no templates, or when none of the active templates matches the user's input. In selective sampling, the document is parsed into sentences. For each sentence a content index is computed, and a finite number of the most content-bearing sentences is selected for query generation. The selective sampling is one way in which MetaCenter extends the EMIR agent [Kulyukin 1999], an IR agent embedded in Emacs. Since queries are automatically generated in the background, query generation does not disrupt the user's activities.

MetaCenter uses the vector space retrieval model [Salton and McGill 1983]. A document is a vector of its terms' weights. A document collection is a vector space whose dimensions are the documents' terms. Since many online collections are dynamic, i.e., documents are added, deleted, and modified, the computation of term weights reflects this dynamism. In smaller collections, a term's rarity is used as its most important distinguishing property. In larger collections, the term's pattern of occurrences in the documents becomes more important. Since indexing is adjusted to the currently available data, it is called *anydata*.

When applied in the document space, standard relevance feedback modifies term weights in the document vectors from relevance judgments [Brauen 1971]. Over long periods, some terms acquire negative weights, which adversely affect performance [Robertson and Walker 1997]. Since only the dimensions' weights are modified, query terms that are not dimensions have no impact on future retrievals [Kulyukin 1998]. Dual space feedback addresses both problems with two vector spaces, *positive* and *negative*, both of which represent the same document collection. When a document is relevant to a query, the weights are adjusted in the positive space (p-space). When a document is not relevant, the same adjustments are made in the negative space (n-space).

4 Implementation

A vector space retrieval model must take a stand on two issues: the computation of terms and the assignment of term weights.

4.1 Terms

The terms are computed from queries, i.e., background free-text samples of documents, and the documents in each collection. A semantic network with spreading activation [Quillian 1969; Norvig 1986; Riesbeck 1983; Fitzgerald 1995;] is used for concept detection. Each sentence in a sample is propagated through a semantic network. When a sentence activates a node, the phrase associated with the node, e.g., "distributed networks," is considered recognized. After the phrases are recognized, the remaining text is stoplisted, i.e., terms, such as "to," "any," and "or," are removed, with an extended version of the Brown Corpus stoplist [Francis and Kucera 1982]. A recognized phrase is considered as a term. No morphological analysis, e.g., stemming, is attempted.

4.2 Term weights

A term's weight unifies two approaches: *inverse document frequency* (IDF) [Salton and McGill 1983] and *condensation*

clustering (CC) [Bookstein et al. 2001; Kulyukin 1999; Kulyukin et al. 1998]. IDF values a term's rarity in the collection; CC values terms' nonrandom distribution patterns over the collection's documents.

Let D be the total number of documents in the collection. Define $f(i,j)$ to be t_i's frequency in the j-th document d_j. Put $\tilde{n}_j = 1$ if $f(i,j) > 0$, and 0, otherwise. Put $D_i = \sum_{i=1}^{D} \tilde{n}_i$. For t_i's IDF-weight, put $\omega_{idf}(i) = A_{idf} + log(D/D_i)$, with A_{idf} a constant. For t_i's $tfidf$ weight in d_j, put $\omega_{tfidf}(i,j) = f(i,j)(A_{idf} + log(D/D_i))$. The CC-weight of t_i is the ratio of the actual number of documents containing at least one occurrence of t_i over the expected number of such documents: $\omega_{cc}(i) = A_{cc} + log(E(\tilde{D}_i)/D_i)$, where A_{cc} is a constant and $\tilde{D}_i$ is a random variable assuming D_i's values. Put $T_i = \sum_{j=1}^{D} f(i,j)$. Since $\tilde{n}_i$ assumes 1 and 0 with the respective probabilities of p_i and $q_i = 1 - p_i$, $E(\tilde{n}_i) = p_i = 1 - (1 - 1/D)^{T_i}$. Since $\tilde{D}_i = \sum_{i=1}^{D} \tilde{n}_i$, $E(\tilde{D}_i) = Dp_i$. For t_i's *tfcc* weight in d_j, put $\omega_{tfcc}(i,j) = f(i,j)\omega_{cc}(i)$.

Let $A_{cc} = A_{idf}$. By definition, $\omega_{cc}(i) = A_{cc} + log(E(\tilde{D}_i)/D_i) = A_{cc} + log(Dp_i/D_i) = \omega_{idf}(i) + logp_i$. Hence, the following lemma: If $A_{cc} = A_{idf}$, $\omega_{cc}(i) = \omega_{idf}(i) + \log p_i$. Or, if the constants are equal, t_i's IDF-weight is its CC-weight plus the $logp_i$ correction. The lemma brings a closure to the discussion of the relationship between IDF and CC [Bookstein, Kulyukin, and Raita 2001; Bookstein, Klein and Raita 1998; Kulyukin, Hammond, and Burke 1998]. The lemma also yields a class of metrics unifying IDF and CC: $\omega_{idfcc}(i) = A + B\omega_{idf}(i) + Clogp_i$, where A, B, and C are constants. If $A = A_{idf}$, $B = 1$, and $C = 0$, $\omega_{idfcc} = \omega_{idf}$; if $A = A_{cc}$, $B = 1$, and $C = 1$, $\omega_{idfcc} = \omega_{cc}$. Since $f(i,j)$ approximates the importance of t_i in d_j, t_i's weight in d_j is given by $\omega_{tfidfcc}(i,j) = f(i,j)\omega_{idfcc}(i)$. Combining t_i's semantic and statistical weights gives t_i's weight in d_j: $\omega_{cw}(i,j) = \omega_{sm}(i)^{\alpha_1}\omega_{tfidfcc}(i,j)^{\alpha_2}$, where α's control the importance of the component weights.

4.3 Feedback

Each distributed collection is described by two vectors: the p-space centroid (p-centroid) and the n-space centroid (n-centroid). The centroid's components are the average weights of the space's dimensions. The magnitude of the negative space centroid is initially zero, i.e., in the absence of negative feedback, each n-space vector has a zero magnitude.

After a collection is subscribed to, MetaCenter requests both centroids from the collection's server. Thus, two vector spaces are stored locally: one of p-centroids and one of n-centroids. Since all collections are dynamic, the local and remote centroids are synchronized. Each local centroid has a time interval during which it is considered up-to-date. The time interval is inverse to the collection's rate of change: the smaller the interval, the faster the change, the more frequent the centroid recomputations. Each collection has a rate of change interval. The interval is periodically obtained from the collection's server by the embedded agent. When the interval expires, the system requests the collection's server to recompute its centroids. When a term is dropped from the p-centroid, e.g., all documents containing it were removed, it is dropped from the n-centroid. Otherwise, its weight in the n-centroid remains the same. Thus, negative evidence persists through changes in the collection.

A query is turned into a vector of terms (q-vector). The similarity is the cosine of the angle between the q-vector and a centroid. The similarity between the q-vector and a p-centroid is *positive*. The similarity between the q-vector and an n-centroid is *negative*. A collection is relevant if the difference between its positive and negative similarities exceeds a threshold. When the collection is relevant, MetaCenter requests its server to do the retrieval on the query. This request can be either to retrieve the top n documents or to retrieve all documents that clear a similarity threshold. The retrievals from each server are merge-sorted locally by the similarity to the query.

Each retrieved document is represented through its title, its author, and up to the first 100 words of its text. A relevance judgment volunteered by the user during a retrieval examination causes adjustments in the appropriate space. When a document is relevant, the adjustments are made in the p-space; otherwise, they are made in the n-space. The adjustment procedure is the same for both spaces: the similarities are rewarded, the differences are punished, and the new terms are added to document vectors to encourage or discourage future retrievals. Specifically, the weights of the terms common to the document and the query are increased by a small amount. The weights of the terms in the document, but not in the query are decreased by a small amount. If a term's weight becomes zero or negative, the term ceases to be a dimension. The weights of the terms in the query, but not in the document are added to the document vector with a small weight. This approach to feedback utilization is similar to negative evidence acquisition [Kulyukin et al. 1998], a technique for adjusting content representations in response to feedback. However, it is more principled, because it stays within one model, precludes negative weights, and avoids the proliferation of ad hoc representations.

5 Evaluation

The experimental data for MetaCenter were taken from the online collection of requests for comments (rfc's) published and maintained by the IETF, the Internet Engineering Task Force (ftp://ftp.isi.edu/in-notes/) and from the electronic proceedings of the AAAI-1999 conference. The complete rfc list is available in the site's `rfc-index.txt` file. 1423 rfc's were manually partitioned into three topics: data transport protocols, applications protocols, and data content. The first topic included 432 rfc's on TCP, UDP, and IP. The second topic included 481 rfc's on SMTP, POP, IMAP, NNTP, HTTP, NTP, SNMP, and Telnet. The third topic included 510 rfc's on Mail and Usenet message formats and MIME. One hundred papers on various AI topics were taken from the AAAI-99 electronic proceedings. All of the papers were put into the AI category.

Each topic constituted one document collection. Each collection was managed by a CORBA server on a separate CPU. MetaCenter was implemented as a COM/CORBA client. Each server published its services through the Inter-

face Definition Language (IDL) [Vogel and Duddy 1998]. The services included indexing a collection, retrieving top n matches under a given similarity metric, i.e., cosine or dot product, retrieving all matches above a threshold, computing centroids, adjusting term weights, adding new terms, retrieving rfc texts, and several other low level maintenance operations. MetaCenter's inputs were 20 papers randomly selected from a pool of 100 papers on network protocols and AI collected from the Web sites of several universities and research laboratories.

In each experiment, the following mode was used. The subject, a researcher with a background in network protocols and AI, was asked to type each paper as if it were his own. The subject would, at his convenience, inspect the retrieval folder and volunteer relevance judgments on retrieved documents. Two options, relevant and nonrelevant, were available on every retrieved document. To keep the feedback solicitation nonintrusive, the subject was not obligated to render judgment on every retrieved document. This mode was an attempt to imitate the way many researchers work on their papers: write, arrive at an impasse, look for relevant papers, read them, develop or borrow ideas, and write some more.

5.1 Metrics

The evaluation metric used in all experiments was the average search length, $avslen_n$, i.e., the number of nonrelevant documents before the n-th relevant one. For example, if $n = 1$, the numbers of nonrelevant documents before the 1st relevant one in all retrieved sets are added and divided by the number of submitted queries.

The two standard evaluation metrics, *precision* and *recall* [Harman 1996; Salton and McGill 1983], were not used. Recall is the ratio of retrieved relevant documents to all relevant documents; precision is the ratio of retrieved relevant documents to all retrieved documents. Recall was not used, because its assumption that *all* relevant documents are known for *every* query is not realistic for large, dynamic collections. Such relevance counts, while useful in theory, are not feasible in practice. A growing body of experimental evidence acquired from deployed systems [Fitzgerald 1995; Jansen et al. 1998; Kulyukin 1998] suggests that most users (80-90%) examine only the top 2-4% of retrievals. Consequently, little benefit is gained from retrieving all relevant documents if none of them makes it to the top. Precision was not used, because, while it does consider relevant documents with respect to the actual retrievals, it fails to consider *where* in the retrieved set the relevant documents are. For example, a retrieved set of 20 documents with the relevant documents occurring in the last two positions has the same precision as a set of 20 documents with the relevant documents occurring in the first two positions. Since MetaCenter queries are computed from samples that can come from anywhere in the paper, empty retrieved sets are probable. But, if precision is used, such sets must be treated as special cases, because precision is undefined if nothing is retrieved.

It should be noted that precision and $avslen_n$ are not unrelated. If the retrieved set's cardinality is fixed, by varying n one approximates the percentage of retrieved relevant documents. For example, $avslen_3$ measures the system's ability to retrieve at least 3 relevant documents, assuming that every retrieved set either has at least 3 relevant documents or has its search length set to its cardinality.

5.2 Results

The first experiment tested the differences between random and structured sampling. In random sampling, queries were computed from random term samples of sizes 10, 20, and 30. The system would randomly select the required number of terms from the text of the document typed so far. In structured sampling, queries were computed from samples taken from specific parts of the paper. The parts were the abstract, introduction, and references. To compute a sample of size n from a part P, the system would parse the text for P, compute the total number of terms (T) in it, randomly choose a position i, $0 \leq i \leq T - n - 1$, take n consecutive terms beginning at i, and apply the greedy morphological analysis to the selection. The sizes for structured samples were the same as for random samples. When sampling from references, the terms were taken from titles. The semantic network processing was disabled in all experiments to reduce the number of independent factors.

The top 10 documents were retrieved in response to every query. Three metrics $avslen_1, avslen_2$, and $avlsen_3$ were computed. A retrieved set with at least one relevant document had a minimum search length of 0, i.e., the first relevant document was at the top, and a maximum search length of 9, i.e., the first relevant document was at the bottom. Sets with no relevant documents had the search length of 10. For each of the 20 test papers, 20 queries (10 random and 10 structured) were computed for each query size (qsize), yielding a total of approximately 1200 random and 1200 structured queries. The term weight metric was $\omega_{cw}(i,j) = \omega_{sm}(i)^{\alpha_1}\omega_{tfidfcc}(i,j)^{\alpha_2}$, where $\alpha_1 = \alpha_2 = 1$, $A = B = 1$, and $C = 0$.

The effects of query selection were also tested. To test selective querying, the system would partition each document into sentences, assign to each sentence each content bearing index, and select the top n content-bearing sentences. The content bearing index was computed as a normalized sum of the weights of the terms found in it.

The first experiment's results are presented in Figure 1. The top table gives the random queries' avslen's. The avslen's are given for each qsize. The bottom table gives the structured queries' avslen's for the three document parts. Thus, the values in the $avslen_1$ row under $abst$, i.e., 8.5, 5.1, and 3.2, are the $avslen_1$'s for the qsizes 10, 20, and 30, respectively, computed from the abstracts. The top table suggests that the avslen improves as the random queries' sizes increase. Thus, random queries are a viable backup when no structure is available because of the nature of the data or parsing failures. The middle table presents the results of testing selective querying. The column numbers represent the number of top content-bearing sentences taken from the document. The table suggests that selective querying does better than either structured or random. The tradeoff is that it is more expensive. The bottom table suggests that the introduction of structure into sampling makes a difference. The avslen's for the queries computed from abstracts and introductions are lower

slen vs. qsize	10	20	30
$avslen_1$	10.1	7.5	6.4
$avslen_2$	12.4	8.9	7.7
$avslen_3$	15.7	11.3	8.3

slen vs. qselect	5	10	15
$avslen_1$	5.4	7.9	8.3
$avslen_2$	6.1	6.4	8.8
$avslen_3$	6.5	6.7	9.1

slen vs. part	abst			intro			refs		
$avslen_1$	8.5	5.1	3.2	8.1	4.7	3.4	14.4	11.1	8.7
$avslen_2$	8.9	7.3	3.7	9.2	7.5	7.1	15.7	14.5	12.3
$avslen_3$	11.1	8.1	4.7	11.9	7.9	8.5	18.2	18.1	15.4

Figure 1: Random, structured, and selected queries with dual space feedback

slen vs. qlen	10	20	30
$avslen_1$	10.4	7.5	6.4
$avslen_2$	12.4	8.9	7.7
$avslen_3$	15.7	11.3	8.3

slen vs. qselect	5	10	15
$avslen_1$	5.3	7.8	8.3
$avslen_2$	6.2	7.9	8.7
$avslen_3$	6.5	8.1	9.1

slen vs. part	abst			intro			refs		
$avslen_1$	10.1	6.2	4.2	8.9	5.2	4.2	14.7	11.3	8.7
$avslen_2$	10.7	7.9	5.8	10.1	8.4	7.8	15.9	14.7	12.5
$avslen_3$	14.3	9.4	7.7	13.0	10.7	9.3	18.6	18.4	15.7

Figure 2: Random, structured, and selected queries without dual space feedback

and, in several cases, statistically significantly [Moore and McCabe 1993] lower than their random counterparts. No significant difference was found between the abstract and introduction queries. A closer analysis revealed that many papers exhibited an overlap between the terms in the abstract and introduction. Ideas were stated in short sentences in the abstract and repeated or expanded with the same terms in the introduction. The reference queries performed worse than their structural and random counterparts.

The second experiment tested the dual space feedback. The queries and relevance judgments from the first experiments were taken, and the avslen's were recomputed with the dual space feedback disabled. As the tables in Figure 2 show, feedback disablement had a negative impact on avslen's, with the exception of the selective queries. The change was most pronounced with abstract and introduction queries. It was less noticeable with random and reference queries. A closer analysis of retrievals indicated that the dual space feedback prevented nonrelevant documents from moving ahead of the relevant ones.

The third experiment tested anydata indexing. The experiment's objective was to find evidence to support the hypothesis that in larger collections, both homogeneous and heterogeneous, a term's tendency to condense in documents is a better predictor of relevance than its rarity. Since many real world collections are dynamic, i.e., documents are constantly added or removed, it is useful to know how to adapt term weight metrics to the collection at hand to improve performance. Two document collections, homogeneous and heterogeneous, were used. The homogeneous collection included 432 rfc's on TCP, UDP, and IP. The heterogeneous collection included the homogeneous collection plus 510 rfc's on Mail and Usenet message formats. The similarity metric, queries, and relevance judgments were the same as in the first two experiments. The performance metric was $avslen_3$. For each collection, the values of A and C ran between -30 and 30 in steps of .25; B was set to 1. Fixing B at 1.0 and varying A and C was aimed at testing the effects of the constant and the correction on IDF. If the best performance were to be found at $C = 0$, it would indicate that the tendency to condense is not as good a content predictor as rarity.

In the first collection, the best $avslen_3$ was 9.21, when $A = -1$ and $C = -1.2$; the best value of $avslen_3$ for IDF, i.e., $A = B = 1$, and $C = 0$, was 12.74. In the second collection, the best $avslen_3$ was 8.2 when $A = -1.25$ and $C = -20$; the best value of $avslen_3$ for IDF was 15.25. Thus, in the larger heterogeneous collection, CC performed significantly better than IDF. In both collections, the correc-

tion's effect was greater than the effect of the constant. The effect was more pronounced in the larger heterogeneous collection. One explanation is that a larger collection is likely to cover different topics. If terms are valued by their tendency to condense in the documents pertinent to their topics, content-bearing terms have a better chance to deviate from being randomly distributed over the documents. The results of the third experiment led to the anydata retrieval heuristic currently used in MetaCenter. When the collection's size is less than 400 documents, IDF is used; otherwise, CC takes over.

6 Conclusion

To integrate automatic query generation with nonintrusive feedback solicitation, the concept of an application-embedded IR system was developed and illustrated with MetaCenter, a personal IR assistant embedded in MS-Word. MetaCenter allows the application in which it is embedded to become part of an infrastructure of distributed information sources. It generates queries from background samples of documents. The queries are used in retrievals from the distributed collections to which the user subscribes. Feedback is never solicited explicitly, and is utilized persistently only when it is volunteered by the user during retrieval examinations. A family of term weight metrics was presented that unifies IDF and CC. It was shown how retrieval can be adjusted through background sampling, anydata indexing, and dual space feedback.

References

[Aalbersberg, 1992] Aalbersberg, I. J. Incremental Relevance Feedback. *In Proceedings of the 15th Annual International SIGIR Conference*, 1992.

[Bookstein et al., 1998] Bookstein, A., Klein, S. T., and Raita, T. Clumping Properties of Content-Bearing Words. *Journal of the American Society for Information Science*, 49(2):102-114, 1998.

[Bookstein et al., 2001] Bookstein, A., Kulyukin, V., and Raita, T. Discovering Term Occurrence Structure in Text: Variance of `M_N`. *Journal of the American Society for Information Science and Technology*, 52(6):485-486, 2001.

[Brauen, 1971] Brauen, T. L. Document Vector Modification. *The SMART Retrieval System: Experiments in Automatic Document Processing*, pp. 456-484. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[Burke et al., 1996] Burke, R. D., Hammond, K. J., and Young, B. C. Knowledge-based Navigation of Complex Information Spaces. *Proceedings of AAAI-96*, AAAI Press, 1996.

[Fitzgerald., 1995] Fitzgerald, W. Building Embedded Conceptual Parsers. Ph.D. dissertation, Deptartment of Computer Science, Northwestern University, 1995.

[Francis and Kucera, 1982] Francis, W., and Kucera, *Frequency Analysis of English Usage*, New York: Houghton Mufflin, 1982.

[Harman, 1996] Harman, D. 1996. Overview of the Fourth Text REtrieval Conference. *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, ACM Press, 1996.

[Jansen et al., 1998] Jansen, B. J., Spink, A., Bateman, J., and Saracevic, T. Real Life Information Retrieval: A Study of User Queries on the Web. *SIGIR Forum*, 32(1):5-18, 1998.

[Kulyukin, 1999] Kulyukin, V. Application-Embedded Retrieval From Distributed Free-Text Collections. *Proceedings of AAAI-99*, 1999.

[Kulyukin, 1998] Kulyukin, V. FAQ Finder: A Gateway to Newsgroups' Expertise. *Proceedings of the 40th Conference of Lisp Users*, Franz Inc., 1998.

[Kulyukin et al., 1998] Kulyukin, V., Hammond, K., and Burke, R. Answering Questions for an Organization Online. *Proceedings of AAAI-98*, AAAI Press, 1998.

[Kulyukin et al., 1996] Kulyukin, V., Hammond, K., and Burke, R. Automated Analysis of Structured Online Documents. *Proceedings of the AAAI-96 Workshop on Internet-Based Information Systems*, AAAI Press, 1996.

[Moore and McCabe, 1993] Moore, D. S., and McCabe, G. P. *Introduction to the Practice of Statistics.* 2nd ed. New York: W. H. Freeman and Company, 1993.

[Norvig, 1986] Norvig, P. A unified theory of inference for text understanding. Ph.D. dissertation, Department of Computer Science, University of California at Berkeley, 1986.

[Quillian, 1969] Quillian, M. The teachable language comprehender. *Communications of the ACM*, 12:(8), 1986.

[Riesbeck, 1983] Riesbeck, C. Knowledge reorganization and reasoning style. Technical Report No. 270, Department of Computer Science, Yale University, 1983.

[Robertson and Walker, 1997] Robertson, S., and Walker, S. On Relevance Weights with Little Relevance Information. *Proceedings of ACM SIGIR*, 1997.

[Salton and McGill, 1983] Salton, G., and McGill, M. *Introduction to Modern Information Retrieval*, New York: McGraw-Hill, 1983.

[Spink and Saracevic, 1997] Spink, A., and Saracevic, T. Interactive Information Retrieval: Sources and Effectiveness of Search Terms During Mediated Online Searching. *Journal of the American Society for Information Science*, 48(8):741-761, 1997.

[Vogel and Duddy, 1998] Vogel, A., and Duddy, K. *Java^{TM} Programming with CORBA*, New York: John Wiley & Sons, Inc., 1998.