

Information Retrieval Advances using Relevance Feedback

Erik W. Selberg
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
selberg@cs.washington.edu
<http://www.cs.washington.edu/homes/selberg>

August 12, 1997

Abstract

Relevance Feedback (RF) is a powerful technique whereby a user can instruct an Information Retrieval (IR) system to find additional relevant documents by providing relevance information on certain documents or query terms. There are several major design decisions that can affect how a RF system can help users. In this paper, I present a brief literature survey covering two emphases of RF: term selection and term weighting in the two prevalent IR models: probabilistic and vector-based. I then summarize how both the techniques and the underlying models can be combined and to what ends. Finally, I hypothesize why Relevance Feedback is not prevalent on World Wide Web search services, and give suggestions on which methods may be appropriate and what, if anything, should be done to add feedback techniques to Web searching.

1 Introduction & Motivation

The field of Information Retrieval (IR) has a rich history of developing technology designed to help users find relevant documents. One of the more important issues in IR literature is handling “novice” users. Novice users are characterized as either completely inexperienced with the IR system at hand or are just unsure of the contents of a particular collection. A technology that has often been exploited to help novice users is Relevance Feedback (RF).

In a typical scenario, a user will engage in a cyclic pattern while looking for information. The user will first formulate and submit the query, and subsequently analyze the documents returned. Then, if the user is not satisfied, he or she will modify the query and begin the cycle again. Relevance Feedback is a technique by which the system is able to reformulate the query by taking relevance information on particular documents or terms from the user. For example, after the user has submitted the initial query, the user may decide that the first, third, and eighth documents are relevant. A RF system may automatically reformulate and resubmit the query, thus simplifying the potentially difficult task of selecting which terms and which ordering to use for the subsequent query.

Nowhere is the need to properly handle novice users more pronounced than with the World Wide Web. The average user on the Web is certainly not a professional searcher, and often assumes that “everything” is on the Web. When users search the Web, currently composed of well over 50 million documents¹, they often find that their queries can match several thousand documents. It would seem obvious that some type of Relevance Feedback could be used to help users refine their query so as to return documents that are indeed germane. Curiously, this is not yet the case.

This paper explores why Relevance Feedback is conspicuously absent from the Web. It begins by presenting a high level overview of information retrieval followed by a formal mathematical background. It then surveys four Relevance Feedback techniques, and examines how those techniques could be combined. Finally, it concludes with some speculation as to why Relevance Feedback has not yet been widely adopted on the Web, and what could be done to add it.

2 High Level Overview of IR

An Information Retrieval system, commonly known as a search engine, is a tool that takes as input some query and returns as output some set of information. This paper will focus on IR systems that take textual queries as input and returns a set of documents as output.

2.1 User-level View

The high-level user view I will be using is adapted from the INQUERY perspective [51]. A user has some *Information Need*. The user begins a session with an IR system by formulating a query Q_0 , and submitting it to the IR system. The system returns a set of documents $D_0 = \{d_0, d_1, \dots, d_n\}$. If D_0 satisfies the Information Need, then the session is over. Otherwise, the user reformulates Q_0 to Q_1 and the cycle continues.

An important factor affecting how the user will interact with the IR system is the type of Information Need the user has. Typically, there are three types of Information Needs that an IR system can handle:

Pinpoint A user needs *an* answer to a question; typically, there is one “right” answer which may be in one or more documents. Returning any document containing the answer satisfies the Information Need.

Exhaustive A user needs *all* relevant information regarding the query. Typically, the user’s Information Need is met once a certain number of relevant documents are returned.

Future A user needs all relevant information now and *in the future* regarding the query. Typically, the Information Need is satisfied when the system returns an optimized query to the user, as opposed to a set of documents. The methods that satisfy this type of Need are often called “Document Routing” or “Routing” in the literature.

¹HotBot and InfoSeek both claim to have indexed over 55 million

This paper will be concerned with Relevance Feedback as it applies to the *Exhaustive* types of Information Need. Relevance Feedback is typically not very helpful with *Pinpoint* Needs; by definition, the Need is satisfied once any relevant document is found, which is also a necessary precondition for any RF technique. The techniques used for *Exhaustive* and *Future* tasks are often similar; however, the additional detail necessary to fully explore *Future* Needs is beyond the scope of this paper.

2.2 Query Expansion and Relevance Feedback

Query Expansion describes the set of techniques for modifying a query in order to satisfy an Information Need. In most cases, terms are added to an existing query, although Query Expansion also encompasses techniques for the reweighting of terms as well as the deletion of terms. Query Expansion techniques can be partitioned into three sets:

Manual *Manual Query Expansion* (MQE) refers to techniques that a user may employ to modify the query. The system does not aid the user at all.

Automatic *Automatic Query Expansion* (AQE) refers to techniques that modify a query *without* user control. For example, a system that always adds thesaurus terms to a query would be considered an AQE system.

Interactive *Interactive Query Expansion* (IQE) refers to techniques where the user has some interaction with the system in the query expansion process. This set of techniques encompasses Relevance Feedback and is the focus of this paper.

Manual and *Automatic Query Expansion* techniques are beyond the scope of this paper, and since the literature concerning those topics is quite large and diverse, interested readers are referred to a 1996 survey paper by Efthimiadis [12].

2.3 Term Weighting and Term Selection

Relevance Feedback techniques tend to focus on one of two emphases:

Term Weighting is the process by which terms in the query are weighted or re-weighted.

Term Selection is the process by which terms are added, or in some case deleted, from the query.

In this paper, term weighting will refer to techniques that weight one term differently from another. A related concept is *relative term weighting*. Relative term weighting is a technique whereby older terms as a whole are weighted differently than newer terms.

Term selection will refer to techniques that select which candidate terms, all of which have a non-zero weight associated with them, are added or deleted from the query. Typically, term selection is implemented by using a term ranking formula to sort all candidate terms, and then selecting a set number of top candidates. In the literature, this implementation is often referred to as using alternate sorts for term selection.

2.4 Common IR Techniques

There are two automatic query expansion techniques that have become nearly universal and have been shown to increase performance [41]. The first involves the use of *stemming*, which is the process of removing suffixes from the base words. For example, the words “helping,” “helped,” and “helps” would all be stemmed to “help.” In a system that uses stemming, all words are stemmed before indexing. In addition, all query terms are stemmed before retrieval. Thus, a query for the word “helping” would match a document that contains the word “helps.”

The second technique is the use of *stop lists*, which are lists of words that are commonly found in documents and not indexed. Articles and conjunctions, such as “the,” “a,” “and,” and “but” typically comprise the majority of a stop list.

Most modern systems use both of these techniques, and their use has become so commonplace that they are often omitted from publications describing systems that use them. While there is some considerable literature regarding stemming algorithms and construction of stop lists, their inclusion or exclusion in any system is orthogonal to the issues addressed in this paper.

2.5 Evaluation and Comparison of IR systems

When comparing techniques, there needs to be some objective method of evaluation in order to determine which is better under some set of conditions. IR systems are typically evaluated using *precision* and *recall*. For any set of documents retrieved, *precision* is the percentage of those documents that are relevant, and *recall* is the percentage of relevant documents retrieved out of all relevant documents. In this paper, I will measure *performance* as the average increase in precision at a fixed recall. Typically, an improvement in recall comes at the expense of a degradation in precision, and vice versa. Relevance Feedback can be thought of as a method of improving performance by increasing precision, or slowing its degradation, when recall is increased.

Unfortunately, while precision and recall afford a reasonable method of comparing two systems, in practice published results using precision and recall are generally uncomparable, most often due to modifications made to standard test collections or in alternative methods of choosing documents used in calculations. For systems using Relevance Feedback, it is also difficult to compare results because there are a variety of methods of ascertaining the improvement caused by feedback [7, 43]. To that end, I will be conservative in stating that one system is generally better than another, as well as in predicting the potential gain in creating a system using techniques not actually combined.

A final note regarding comparison, which comes into play when discussing deployment of an IR system, is in regards to the evaluation metric. In IR, typically the evaluation question being asked is “How well does the system satisfy the query?” However, there are other metrics that may be important. For instance:

- How quickly does the system satisfy the query?
- How many resources does the system require to satisfy the query?
- How easy is it for the user to obtain an answer?

While these metrics are normally the domain of systems and user interface researchers, as IR systems are evaluated for deployment in particular settings they become increasingly more important. Unfortunately, since most published articles do not contain experiments that answer these metrics, it is often unclear how well any system will work under various real world constraints. Thus, this paper will not address these questions further.

3 Formal Background

While the user perspective of Relevance Feedback may be the same from system to system, the underlying mechanics are largely dependent on the mathematical model used by the IR system. I present a brief summary of the major mathematical models used by IR systems and introduce some definitions that will be used throughout the rest of this paper. For a more complete discussion on the underlying mathematical models, please see Salton's 1979 *Journal of Documentation* article [40].

3.1 Definitions

For the purposes of this paper, I will use the following definitions:

$D = \{d_0, d_1, \dots, d_N\}$ is a set of N documents in the collection. d will refer to an arbitrary document $d_i \in D$.

$T = \{t_0, t_1, \dots, t_m\}$ is a set of m terms indexed in D . In this paper t_i will generally be a single word, but richer definitions may be used without impacting the underlying theory. t will refer to an arbitrary term $t \in T$.

w_i is the *weight* of term t_i . Weight is simply a numeric quantity that indicates the importance of a term.

d_i is an individual document, represented by the vector of length m

$$d_i = [w_{i1}, w_{i2}, \dots, w_{im}]$$

where w_{ij} is the weight of term t_j in document d_i . In the simplest case, each t_j is a term indexed from one or more documents, and w_{ij} is either 1 or 0, depending if the term t_j is present or absent from document d_i .

$D^t = \{d : t \in d \wedge d \in D\}$ is a set of documents restricted to those documents that contain the term t . Note that $|D^t|$ represents the *Document Frequency* of term t .

Q is a Query containing terms $t \in T$. The specific structure of Q is model dependent and will be further defined.

$|X|$ is the size of set X . $|V|$ will also be used to denote the length of a vector V .

tf_{ik} is the number of times term t_k appears in document d_i . This is the *Term Frequency* of t_k .

3.2 Vector Model of IR

In the vector model, a query Q is represented as an attribute vector in a similar fashion to documents:

$$Q = [w_1, w_2, \dots, w_m]$$

where w_i represents the weight of attribute t_i in Q .

In order to determine which documents satisfy the query, some similarity measure, or *document ranking function*, r is needed. Common values for r are the simple dot product

$$r'(Q, d) = Q \cdot d \tag{1}$$

the cosine function

$$r''(Q, d) = \frac{Q \cdot d}{|Q||d|} = \cos(Q, d) \tag{2}$$

and the similarity function

$$r'''(Q, d) = \frac{Q \cdot d}{Q^2 + d^2 - Q \cdot d} \tag{3}$$

Of note is that when the entries of Q and d are restricted to either 0 or 1, the dot product returns the number of terms contained in both Q and d . The denominators of r'' and r''' are two different methods to normalize for document length.

In order to satisfy a given query Q , $r(Q, d)$ is computed for all $d \in D$, and those documents with sufficiently high values of r are returned, usually in sorted order of decreasing value or in some kind of clustered display [54].

3.3 Probabilistic Model of IR

Another method of computing the similarity function r is probabilistically. There are two popular probabilistic models that we will discuss: the traditional Robertson / Sparck Jones formalization and the more structural inference network model.

3.3.1 Robertson / Sparck Jones Formalization

Robertson and Sparck Jones describe a probabilistic system as a system that determines the weights of terms in document and query vectors probabilistically [23]. Using log likelihoods, they derive the following expression for the ranking of a document based on sum of probabilistically-derived weights:

$$r(d) = \sum_{i=1}^m \left(\log \frac{p_i}{1 - p_i} + \log \frac{1 - q_i}{q_i} \right) \tag{4}$$

where p_i is the probability that term t_i is in a relevant document and q_i is the probability that t_i is in a non-relevant document. m is the number of terms in the collection. The question now comes to determining how to calculate p_i and q_i .

For a given term t_i and a query Q , Robertson and Sparck Jones define:

N The number of documents in the collection, i.e. $|D|$

R The number of relevant documents, i.e. $|D_R|$

n_i The number of documents containing term t_i , i.e. $|D^{t_i}|$

r_i The number of relevant documents containing t_i , i.e. $|D_R^{t_i}|$

and create the following contingency table used to determine the number of relevant and irrelevant documents when a given term is present or absent:

Occurrence of Term	Rel Docs	Non-Rel Docs	Total Docs
t_i is present	r_i	$n_i - r_i$	n_i
t_i is absent	$R - r_i$	$N - R - n_i + r_i$	$N - n_i$
Total Docs	R	$N - R$	N

Using this table, p_i and q_i , the probabilities of a document being relevant when a term is or is not present, are defined by Robertson and Sparck Jones as

$$\begin{aligned} p_i &= r_i/R \\ q_i &= (n_i - r_i)/(N - R) \end{aligned}$$

and by substituting back into Equation (4) they arrive at

$$\begin{aligned} r(d) &= \sum_{i=1}^m \left(\log \frac{r_i/R}{1 - r_i/R} + \log \frac{1 - (n_i - r_i)/(N - R)}{(n_i - r_i)/(N - R)} \right) \\ &= \sum_{i=1}^m \left(\log \frac{r_i}{R - r_i} + \log \frac{N - n_i - R + r_i}{n_i - r_i} \right) \end{aligned} \quad (5)$$

also referred to as the f_4 formula. In order to avoid undefined weights when n_i is 0, a common case in operational systems, a factor of 0.5 is added to each term, resulting in the f_4 *point 5* formula:

$$r(d) = \sum_{i=1}^m \left(\log \frac{r_i + 0.5}{R - r_i + 0.5} + \log \frac{N - n_i - R + r_i + 0.5}{n_i - r_i + 0.5} \right) \quad (6)$$

N and n_i are easily obtainable. However, since D_R is unknown to the system, it is necessary to estimate R and r_i for the initial query. There are a variety of estimation methods available [20, 53] for initial queries, although the common technique is to use the f_4 *point 5* or similar [35] formula and set R and r_i to 0 resulting in an initial uniform weight for all terms in a query.

Relevance Feedback is implemented in this system by successively updating R and r_i after each iteration. The theory is that after enough iterations, the values for p_i and q_i will converge upon the true weights, and thus the ranking formula defined by Equation (4) will produce the full set of relevant documents D_R .

3.3.2 Inference Networks

Inference networks use a slightly different probabilistic model based on reasoning under uncertainty [29]. An inference network is an inverted tree structure comprised of multiple parents culminating to a single child. The tree contains four levels, with each level consisting of a different type of node:

Document nodes, which represent a particular document;

Representation nodes, which represent a particular concept. Typically, this is a term contained within one or more documents;

Query nodes, which represent the concepts used to represent the information need of the user;

I node, which represents the information need of the user.

A sample network is shown in Figure 1. Here, the nodes labeled $D_1, D_i,$ and D_n correspond to three of the n Document nodes. The nodes labeled “Richard,” “Nixon,” and “Watergate” are Representation nodes, the “and” and “not” nodes are the Query nodes, and the “information need” node is the I node. The horizontal line separates the Document and Representation nodes, which are created at index time rather than at query time, from the Query and I nodes, which are created dynamically for each query.

Probabilities are calculated starting from the Document nodes and propagated down to the I node. Each child node contains a link matrix [51] that details how to combine the probabilities of the child’s parents into the probabilities for the child. Eventually, all probabilities are propagated down to the I node. The I node thus contains the probabilities of each parent node satisfying the information need. It is a simple matter then to sort the probabilities and display the most relevant ones, in accordance with Robertson’s Probability Ranking Principle [34].

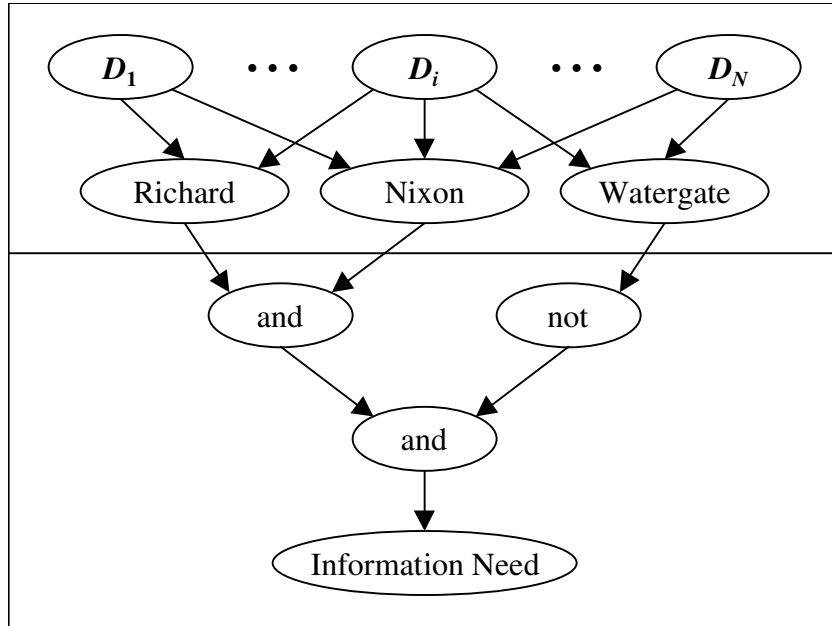
3.4 Boolean Model of IR

IR systems using the Boolean model have become less common in recent years, due in part to the popularity and increased ease of use of vector and probabilistic systems that accept keywords or natural language queries. Furthermore, front-end interfaces which allow Boolean or modified Boolean expressions, often called *structured queries*, have been implemented in both vector-based and probabilistic systems [44, 51], in attempts to obviate the need for a strictly Boolean model.

While there is some Relevance Feedback literature regarding Boolean systems [32, 45, 33, 8], they are mostly involved with adapting term selection formulas to append conjunctive terms to the expression and not generally germane to the purposes of this paper. Therefore, I will not address with the Boolean model further in this paper.

4 Two Vector-Based Systems

I now focus on two vector-based systems, one that emphasizes term weighting, and the other term selection. The term weighting system was originally described by J. J. Rocchio in 1965 as part of



In the above network, the nodes labeled D_1 , D_i , and D_n correspond to three of the n Document nodes. The nodes labeled “Richard,” “Nixon,” and “Watergate” are Representation nodes, the “and” and “not” nodes are the Query nodes, and the “information need” node is the I node. The horizontal line separates the Document and Representation nodes, which are created at index time rather than at query time, from the Query and I nodes, which are created dynamically for each query.

Figure 1: Sample inference network for query “(Richard and Nixon) and not Watergate”

the SMART system [38, 39]. The term selection system was originally described by Donna Harman in 1988 as part of the IRX project [16, 19].

4.1 Term Weighting in a Vector Model

Rocchio's technique of feedback is based on the approximation of an *optimal query*. In order to describe his technique, it is first necessary to recite his definition of an optimal query.

Given a document collection D and a query Q , Rocchio assumes the existence of $D_R \subseteq D$, which is the relevant set of documents in D to Q . He further defines an *ideal request* as a query that ranks all elements of D_R above all other elements in D . However, he is quick to point out that the ideal request is unlikely to exist for all queries. Thus, he defines an *optimal request* query Q_{opt} , which is the query that maximizes the similarity of the relevant documents and minimizes the similarity of the irrelevant documents. In mathematical terms, Rocchio defines C as

$$C = \frac{1}{|D_R|} \sum_{d \in D_R} r(Q, d) - \frac{1}{|D| - |D_R|} \sum_{d \notin D_R} r(Q, d)$$

Q_{opt} is the vector that corresponds to Q when C is maximal. To find this, Rocchio uses the cosine similarity function defined by Equation (2) to determine the correlation between a query and a given document. Substituting for $r(Q, d)$ he derives

$$\begin{aligned} C &= \frac{1}{|D_R|} \sum_{d \in D_R} \frac{Q}{|Q|} \cdot \frac{d}{|d|} - \frac{1}{|D| - |D_R|} \sum_{d \notin D_R} \frac{Q}{|Q|} \cdot \frac{d}{|d|} \\ &= \frac{Q}{|Q|} \cdot \left[\frac{1}{|D_R|} \sum_{d \in D_R} \frac{d}{|d|} - \frac{1}{|D| - |D_R|} \sum_{d \notin D_R} \frac{d}{|d|} \right] \\ &= \frac{Q}{|Q|} \cdot A \end{aligned}$$

Rocchio observes that C is maximized when $Q = kA$ for any arbitrary scalar k . Thus,

$$Q_{opt} = \frac{1}{|D_R|} \sum_{d \in D_R} \frac{d}{|d|} - \frac{1}{|D| - |D_R|} \sum_{d \notin D_R} \frac{d}{|d|} \quad (7)$$

While the formula for Q_{opt} shows that there exists an optimal query for a given set of relevant documents D_R , this formula does not help retrieve those documents, as knowledge of D_R obviates the need for retrieval and making Q_{opt} in the first place.

Rocchio then defines his Relevance Feedback technique as an iterative process whereby the user begins with query Q_0 that returns a set of documents D_0 , and continues to reformulate the query Q_1, Q_2, \dots, Q_n until the user is satisfied that D_n is D_R . The question that now arises is what is the process by which Q_i becomes Q_{i+1} ?

Rocchio describes the abstract mathematical model as

$$Q_{i+1} = f(Q_i, R, S)$$

where R is the set of documents the user has judged relevant, and S the set of irrelevant documents. Rocchio suggests that f should be

$$Q_{i+1} = \frac{Q_i}{|Q_i|} + \frac{1}{|R|} \sum_{j=1}^{|R|} \frac{R_j}{|R_j|} - \frac{1}{|S|} \sum_{j=1}^{|S|} \frac{S_j}{|S_j|}$$

or alternatively

$$\begin{aligned} Q_{i+1} &= |R||S| \frac{Q_i}{|Q_i|} + |S| \sum_{j=1}^{|R|} \frac{R_j}{|R_j|} - |R| \sum_{j=1}^{|S|} \frac{S_j}{|S_j|} \\ &= \alpha Q_i + \beta \sum_{j=1}^{|R|} \frac{R_j}{|R_j|} - \gamma \sum_{j=1}^{|S|} \frac{S_j}{|S_j|} \end{aligned} \quad (8)$$

where Equation (8) is the form commonly seen in subsequent literature.

Rocchio made the initial evaluation of his method using 17 requests on the SMART system. He made some modifications to the technique; in particular, since SMART was unable to handle negative weights, any negative weight was set to 0. His results were very promising, showing approximately a 10% increase in precision for every level of recall when using Relevance Feedback, although due to the extremely limited sample size the results could not be taken as statistically significant.

Ide conducted some experimental work on Rocchio's formula, as well as some modifications, again using the SMART system [21]. He verified that Rocchio's system did in fact result in an increase in performance. He also suggested a modification of Rocchio's formula which is often seen, commonly called *Ide Dec-Hi*:

$$Q_{i+1} = Q_i + \sum_{j=1}^{|R|} R_j - S_1 \quad (9)$$

where S_1 is the first non-relevant document retrieved. *Ide Dec-Hi* is commonly thought to be the best pure vector-based formula available [42].

The result of Rocchio's work is an elegant and simple way of incorporating relevance information to refine queries. The user interface necessary requires only that a user mark documents as either relevant or irrelevant, and the system infers exactly what components of those documents make it relevant or irrelevant. It is also computationally feasible, being linear in the number of documents retrieved and terms per document. All together, this technique combines a strong mathematical foundation with a useful practical method.

4.2 Term Selection in a Vector Model

One of the advantages of Rocchio's technique is that the user only has to judge whether an entire document is relevant or not, and the system will infer which terms or attributes corresponded to that relevance judgment. His particular implementation expands the query to include all potentially relevant terms, and then focuses on calculating the appropriate weight for each term. Unfortunately,

one of the side effects of his system is that reformulated queries tend to get large and include words that are arguably irrelevant, typically function words with no inherent meaning. The result of this is that while Relevance Feedback is able to increase recall, precision suffers as irrelevant documents are obtained.

As part of the IRX [16, 19] project, Donna Harman conducted a study to determine if a small set of highly relevant words could be found and if that set of words would produce better results than queries which incorporated all possibly relevant terms. Harman does this by looking at three different techniques for obtaining relevant terms: Relevance Feedback, term variation, and nearest neighbor routines. Of note in her work is that the IRX system does not use stemming, but indexes exact terms.

To determine if it is feasible to obtain a small set of relevant terms and generate better results than massive term expansion, Harman ran an experiment using the Cranfield 1400 test collection, which contains 1400 abstracts related to aerospace, and 225 related queries with relevance judgments. In conducting this experiment, no term weighting is done, and thus all terms have equal sway in determining relevance.

For her experiment, Harman defines the following:

tf_{ik} The frequency of term t_k in document d_i .

df_k The frequency of term t_k in D , e.g. $|D^{t_k}|$

ns_k The *noise* of term t_k , defined by $ns_k = \sum_{j=1}^{|D|} \frac{tf_{jk}}{df_k} \log \frac{df_k}{tf_{jk}}$

ns_{max} The maximal value of ns_k for all terms t_k

$r(Q, d)$ For this experiment, Harman sets r to be:

$$r(Q, d) = \sum_{k=1}^{|Q|} \frac{\log df_k (ns_{max} - ns_k)}{\log |d|} \quad (10)$$

R The relevant documents obtained after a query has been submitted.

postings Relevant documents that contain a given term, e.g. R^t .

Harman also uses the above definitions of term frequency, document frequency, and noise restricted to R and R^t :

tf'_{ik} The frequency of term t_k in document $d_i, d_i \in R$.

df'_k the frequency of term t_k in R , e.g. $|R^{t_k}|$. Referred to as *frequency within postings*.

ns'_k *Noise within postings*, as defined by $ns_k = \sum_{j=1}^{|R^t|} \frac{tf'_{jk}}{df'_k} \log \frac{df'_k}{tf'_{jk}}$

The primary goal of Harman's experiment is to determine if a small set of relevant terms could be obtained to create better queries. Harman hypothesizes that the following selection formulas will

satisfy that goal. Each formula below returns a numeric value for term t_k . All terms are sorted by this ranking from largest value to smallest, and then the appropriate number are added to the query.

noise The noise of term t_k . The sorting semantics of noise are from lowest to highest, thus the inverse is used.

$$s(t_k) = \frac{1}{ns_k}$$

number of postings The raw number of relevant documents containing the term t_k .

$$s(t_k) = |R^{t_k}|$$

noise within postings The noise of term t_k restricted to relevant documents containing term t_k .

$$s(t_k) = \frac{1}{ns'_k}$$

noise * tf within postings The noise of term t_k multiplied by the term frequency of t_k within relevant documents containing t_k .

$$s(t_k) = \frac{1}{ns_k} df'_k$$

noise * tf * postings The noise of term t_k multiplied by the term frequency of t_k within all documents multiplied by the number of relevant documents containing t_k .

$$s(t_k) = \frac{1}{ns_k} df_k |R^{t_k}|$$

noise * tf The noise of term t_k multiplied by the term frequency of t_k within all documents.

$$s(t_k) = \frac{1}{ns_k} df_k \tag{11}$$

Upon the submission of a query, the experiment runs as follows:

1. Obtain the set of top ten documents D' using a set ranking formula r ;
2. Simulate user interaction by marking all relevant documents, as determined by the Cranfield relevance judgments. Let R be the set of documents within D' that are relevant.
3. Create a list l of all non-common terms from relevant documents in R ;
4. For each selection function:
 - (a) Sort list l using the given selection function;
 - (b) Add 20 terms from the top of the sorted list to the query;

	no feedback	noise	post	noise w/ post	noise * tf w/ post	noise * tf * post	noise * tf
Ave. Prec. Imp. (%)	0	5.3	5.2	7.7	8.8	9.4	8.1
Rel at 10	2.55	2.55	2.55	2.55	2.55	2.55	2.55
Rel at 20	3.25	3.55	3.56	3.71	3.74	3.77	3.73
% improvement	0	41.7	43.9	63.9	68.9	73.3	67.2
Rel at 30	3.71	3.93	4.04	4.16	4.22	4.26	4.22
% improvement	0	18.9	28.7	38.5	43.9	47.3	43.9
Queries Imp.	0	70	82	94	94	91	92
Queries Deg.	0	36	24	25	25	24	22

Ave. Prec. Imp. shows the average precision improvement using recall-level averages [43]. This measures performance in Harman’s experiments. Rel at N counts the average number of relevant documents at N documents, with the % improvement being the percentage improvement between each adjacent count. Queries Imp. and Queries Deg. show the number of queries improved or degraded by the particular method. There were 255 queries total.

Table 1: Results from Harman Feedback Experiment

-
- (c) Re-submit the new query;
 - (d) Evaluate.

The results from this experiment are shown in Table 1. Of note in the table are the two rows labeled “% improvement” which measure the performance of each selection formula. They indicate that there is a significant difference in performance, depending on which sorting formula is used, with the best sorting formula being *noise * tf * postings*, at least for the Cranfield 1400.

In addition to sorting, the user can be consulted as an additional filter to determine true relevance. Harman describes an experiment with simulated user selection whereby a 31% improvement in performance can be obtained, with an average of 12 terms being added to the query instead of 20. She does admit that this is the result of a “perfect choice” by the user and unlikely in the real world setting, although a 1996 user study by Koenemann and Belkin [24] demonstrates that user filtration of the query terms increases performance about 30%.

Harman also describes a secondary experiment in varying the number of terms added to the query from 10 through 40 (in increments of 10). Her findings indicate that performance increases when 10 terms are added, and again when 20 are added. However, for 30 and 40 terms, she observes that the performance using the best sorting methods *decreased*, and while the performance of the worse methods still increased, it was never superior than the better sorts. This phenomenon appears to be a classic case of overfitting.

Harman carries out two further experiments to again find 20 terms to add to the query. The specifics of these experiments are not directly germane to this paper, and thus I will just summarize the results. The first is to find 20 term variants using stemmed words. The second is to find 20

best *nearest neighbors*, where the nearest neighbors of a term are the terms adjacent to it in the document. Harman finds that adding arbitrary stemmed words to the query decreases performance, and performance increases only when terms were filtered via the simulated user selection described above. Similarly, adding arbitrary nearest neighbor terms decreases performance, but performance increases by adding terms filtered with simulated user selection.

The results of Harman's experiments shows that one can achieve a similar level of performance by expanding a query by 20 feedback terms as one can by expanding the query using all feedback terms. Furthermore, she shows that by adding more than 20 terms, performance degrades, at least for the Cranfield collection. She also shows that performance can be further enhanced by giving user direct control over the specific terms to be added to a query. The tradeoff for this higher performance comes in the form of increased user interaction, as the user interface requires users to make judgment calls on both documents and subsequently terms. However, taken together, the results present a strong case for systems that attempt to select the best terms for feedback and filter those through a user.

5 Two Probabilistic Systems

The common alternative to the vector model is the probabilistic model. Probabilistic methods are those which attempt to find a set of documents based on the probability that it satisfies a user's "Information Need." I now look to see how Relevance Feedback is implemented using two probabilistic models. The first is Haines and Croft's implementation of Relevance Feedback in INQUERY using an inference network [15]. The second is a formula derived by Robertson that attempts to select the terms that will have the greatest affect in separating relevant from irrelevant documents [36].

5.1 Term Weighting in a Probabilistic Model

Haines and Croft implement Relevance Feedback using an inference network model in the INQUERY system [15]. They accomplish this by attaching new Query nodes to new terms and by altering existing nodes' link matrices, although it is not specified in their paper how they do this. Once the nodes are added and the matrices adjusted, they are able to recalculate and repropagate all probabilities to the I node. This cycle is repeated until the user is satisfied with the results.

In order to evaluate if Relevance Feedback can be implemented in an inference network model and what degree of improvement it could have, Haines and Croft carry out an extensive set of experiments. They evaluate their system using two test collections: the CACM collection comprised of 3,204 titles and abstracts with two different 50 query sets, the second being a custom set, and the WEST collection, containing 11,953 full text legal documents and the standard 34 query set. Their system is evaluated on several axes. They examine a variety of methods to select terms to add to the query. Furthermore, based on Harman's result described in Section 4.2 as well as subsequent results [17], they experiment with adding between 0 and 150 terms for the CACM collection and 0 to 100 terms for the WEST collection. These results will be discussed more extensively in Section 5.2.

In addition, they also carry out some experiments using structured queries. The results from these

Coll.	Weight	rdfidf	EMIM	idf	PMIM	P_4	rtf	rtfidf	Avg
CACM	rtf	82.9	94.9	94.4	84.6	78.1	76.3	84.1	83.6
	rtfidf	97.3	106.7	102.0	100.3	96.7	94.9	100.8	99.8
WEST	rtf	40.1	39.8	32.4	39.9	40.2	29.4	27.0	35.5
	rtfidf	36.2	31.3	27.2	29.6	33.0	32.1	24.5	30.5

Average percentage increase for precision reported by Haines and Croft for two test collections (CACM and WEST) and weighting functions (rtf and rtfidf) over seven different selection methods, detailed in Section 5.2. These values are summarized from Tables 1 and 2 in Haines and Croft’s original paper, using no relative weighting [15].

Table 2: Average percentage increase in performance in INQUERY.

experiments are beyond this paper’s scope to document fully, but they do confirm that Relevance Feedback can done on structured queries, but the performance increase isn’t as large as keyword or natural language queries.

Haines and Croft also examine methods of term weighting, which in their model correspond to re-estimating the probabilities given prior knowledge. They make use of tf'_{ik} and df_k as defined in Section 4.2. They choose to calculate the weight w_k of term t_k using one of the following two functions:

rtf Frequency of the term in relevant documents.

$$w_k = \frac{1}{|R|} \sum_{i=1}^{|R|} tf'_{ik} \quad (12)$$

rtf * idf rtf multiplied by the *Inverse Document Frequency* (*idf*). The *idf* factor attempts to correct for terms found in both relevant and irrelevant documents.

$$w_k = \frac{1}{|R|} \sum_{i=1}^{|R|} tf'_{ik} \log \frac{|D|}{|D^t|} \quad (13)$$

A selection of their results are summarized in Table 2, which measure the performance of each selection formula by the average increase in precision at a fixed level of recall. The conclusions they are able to reach from their experiments regarding term weighting are:

1. Term reweighting increases performance;
2. The increase in performance by term weighting is dependent on the collection used.

In particular, it is interesting that Haines and Croft are able to obtain an 83.6% increase in performance on average using the CACM collection of abstracts, compared with just a 30.5% increase using the full-text WEST collection.

In addition, they also examine whether it is better to change the relative weights of pre-existing terms and new terms, rather than to evaluate each term’s weight regardless of whether it is selected initially or subsequently. In vector model terms, they evaluate Equation (8) using different values for β and γ . They find that the best overall relative weighting is 65%/35% old/new. However, results from 60%/40% through 80%/20% were comparable. The conclusions reached from this experiment indicate that weighting older terms more than newer terms does increase performance, but it is unclear what those weightings should be.

The results obtained by Haines and Croft demonstrate that Relevance Feedback can be adapted to probabilistic models, including those that use a more structural form than the mathematical models originally suggested by Robertson and Sparck Jones [37]. Furthermore, they indicate that term weighting also affects probabilistic models, and that judicious use of term weighting can dramatically affect performance.

5.2 Term Selection in a Probabilistic Model

In addition to term weighting, term selection also appears to be equally germane to probabilistic models as to vector models. Similar to Harman’s work in Section 4.2, Haines and Croft conducted experiments attempting to ascertain if any term selection methods worked better than others, and if the number of terms added affected performance in any significant way.

Haines and Croft evaluate the performance increase by adding a variable number of terms to queries, from 0 to 150 for the CACM collection and 100 for the WEST collection. They find that for the CACM collection performance is increased by adding any number of terms, although this performance increase begins to plateau at about 40 terms. For the WEST collection, they find a similar result to Harman in that after 20 to 30 terms, performance began to slowly degrade.

Haines and Croft also experiment with a variety of term selection formulas. As in Section 4.2, each of the ranking formulae below return a ranking for a term t_k . All terms are then sorted by this ranking, and the appropriate number are added to the query.

EMIM The expected mutual information [52, 20], defined by:

$$s(t_k) = \sum_{i \in \{t_k, \bar{t}_k\}, j \in \{R, \bar{R}\}} P(i, j) \log \frac{P(i, j)}{P(i)P(j)} \quad (14)$$

where $P(i, j)$ is the probability of both i and j occurring in a document judged relevant, $i \in \{t_k, \bar{t}_k\}$ corresponds to term t_k appearing or not appearing in a given document, and $j \in \{R, \bar{R}\}$ corresponds to a given document being relevant or irrelevant. EMIM was originally derived to take into account term dependence.

PMIM This is EMIM when the term is present and the given document is relevant, e.g.

$$s(t_k) = P(t_k, R) \log \frac{P(t_k, R)}{P(t_k)P(R)} \quad (15)$$

P.4 This probability that the document will be relevant (or irrelevant) depending on the occurrence (or absence) of term t_k

$$s(t_k) = P(t_k, R)P(\bar{t}_k, \bar{R})(1 - P(\bar{t}_k, R))(1 - P(t_k, \bar{R}))$$

idf The *Inverse Document Frequency*, defined as

$$s(t_k) = \log \frac{|D|}{|D^{t_k}|}$$

rdfidf The product of the term's relevant document frequency and the inverse document frequency, defined as

$$s(t_k) = |R^{t_k}| * idf(t_k)$$

rtf as defined by Equation (12).

rtfidf as defined by Equation (13).

While Haines and Croft find that using a ranking formula improved performance, they do not find that any of the above formulas stood out from the rest. Two hypotheses come to mind:

1. The structure of INQUERY is robust enough to give similar results regardless of the term selection methods used;
2. What is important is selecting only a limited number of terms, and any reasonable ranking formula will do.

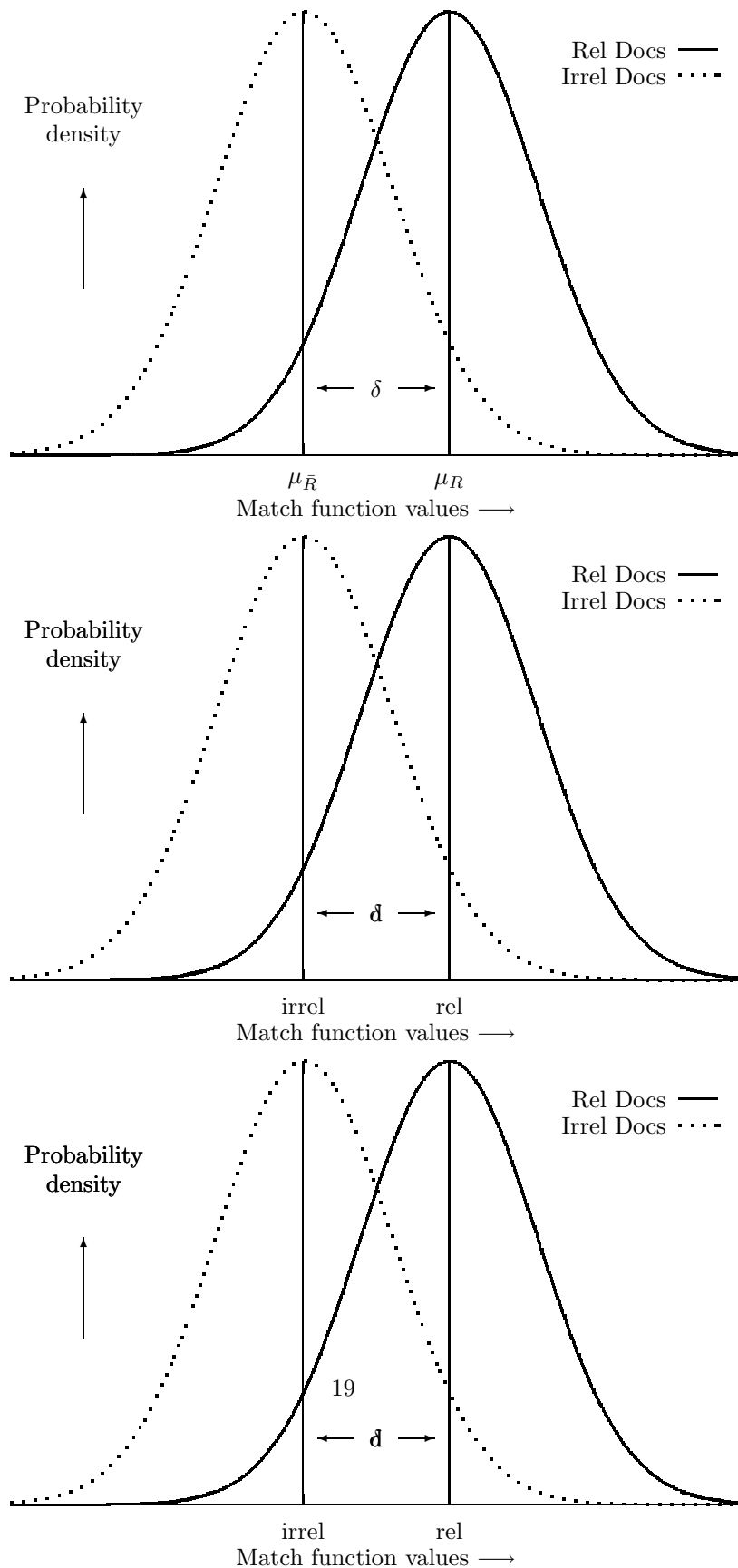
Fully investigating the first hypothesis is beyond the scope of this article. Currently, INQUERY uses *rtfidf* for term selection [4], which would seem to indicate that unpublished empirical evidence suggests that *rtfidf* does in fact perform better than others. As for the second, I will now examine an approach taken with the traditional probabilistic model.

5.3 Term Selection using Traditional Probabilistic Methods

Sparck Jones and Robertson show that the optimal weight for a given term t is defined by Equation (4). Robertson observes that while this is the optimal function for weighting a particular term, terms should not be added to a query solely on the basis of their weight [36]. Robertson suggests a term selection formula based on the Swets model [50] of Information Retrieval Performance. Before proceeding, it is necessary to describe the Swets model.

5.3.1 The Swets Model of IR System Performance

Assume an IR system uses a document ranking function $r(Q, d)$. The Swets model examines the distribution of the values of r ; in particular, it looks at two distributions: the distribution of values when d is the relevant set of documents D_R , and the distribution when d is in the irrelevant set $D_{\bar{R}}$. Intuitively, the ranking values when $d \in D_R$ should generally be higher than when $d \in D_{\bar{R}}$. The assumption is that the more separated the distributions are, the better the ranking function. Assume for a moment that these two distributions exist with means of μ_R and $\mu_{\bar{R}}$. Let $\delta = \mu_R - \mu_{\bar{R}}$ be the distance between the two means.



Shown are two distributions for values of a ranking function $r(Q, d)$, one for irrelevant documents, corresponding to $d \notin D_R$, and the other for relevant documents, where $d \in D_R$. $\mu_{\bar{R}}$ and μ_R correspond to the mean values of the distributions, and δ the distance between those two means.

Figure 2: The Swots model of ranking distribution

Figure 2 shows a graphical representation of the Swets model. Pictured are two distributions of values of a ranking function r , the leftmost distribution corresponding to values of $r(Q, d)$ when d is not relevant, and the rightmost distribution corresponding to values of $r(Q, d)$ when d is relevant. The means of each distribution, $\mu_{\bar{R}}$ and μ_R are highlighted, as is δ , the distance between the means.

Swets uses δ as an approximation for the quality of the distributions, with higher values indicating better distributions. Intuitively, larger values of δ imply that the ranking function r is able to do a better job of differentiating relevant documents from irrelevant documents. While there are some issues in dealing with this model to compare practical systems [5, 40], the Swets model nevertheless offers an attractive theory on which to base formulas for document ranking.

5.3.2 The *wpq* formula

Robertson uses the weighting function

$$w_i = \log \frac{p_i}{1 - p_i} + \log \frac{1 - q_i}{q_i} \quad (16)$$

to arrive at the weight w_i for term t_i ; the sum of this weight for all terms can be used to rank documents and is the basis for Equation (4).

Using the Swets model, Robertson considers the effect of adding a term t which has weight w_t . Robertson shows that the means of the two distributions in the Swets model become:

$$\begin{aligned} \mu'_R &= (1 - p_t)\mu_R + p_t(\mu_R + w_t) \\ &= \mu_R + p_t w_t \\ \mu'_{\bar{R}} &= \mu_{\bar{R}} + q_t w_t \end{aligned}$$

and subsequently the new distance δ' between them becomes:

$$\begin{aligned} \delta' &= \mu_R + p_t w_t - \mu_{\bar{R}} - q_t w_t \\ &= \mu_R - \mu_{\bar{R}} + w_t(p_t - q_t) \\ &= \delta + w_t(p_t - q_t) \end{aligned}$$

Thus, Robertson shows that inclusion of term t should increase the effectiveness of the distribution by

$$a_t = w_t(p_t - q_t) \quad (17)$$

This equation is commonly called the *wpq* formula.

Robertson notes that this selection function is actually independent of the weighting function, and thus alternative weighting functions could be used instead of the one he originally picked.

Robertson points out that this function does rest on the assumption that the distributions of ranking values are normal. Since this is unlikely, he concedes that the above formula is perhaps more indicative than concrete. In a 1995 study, Efthimiadis evaluates a selection of formulas, including Robertson's *wpq* formula, *f4* and *f4point5*, and EMIM, among others [11]. Efthimiadis concludes that the *wpq* and EMIM formulas, along with Porter's [31] and *r-lohi* [10], outperformed all others and are all equally useful.

6 Compositions of Technologies

Having surveyed the major techniques, I turn now to questions regarding their synthesis. In particular, the questions that need to be asked are:

- What performance gain is there in combining term weighting and term selection techniques under the same model?
- Can term weighting and term selection techniques devised under different models be combined effectively?
- In constructing a system from scratch that would use Relevance Feedback, what would be the best way to build it?
- Rather than choosing the “best” weighting or selection technique, can better results be obtained by merging the output of multiple redundant techniques?
- What are the common assumptions from all these techniques? Do the answers for the above questions still hold if the underlying assumptions are relaxed?

6.1 Combining Weighting and Selection under Similar Models

Combining a probabilistic term weighting function and a probabilistic term selection function in one system is obviously possible, as evidenced by the Haines and Croft system. In fact, most modern systems now use both a weighting and selection formula; even the SMART system, known for its use of massive term expansion, is now selecting terms rather than adding all available candidates [6].

While most published results are not directly comparable, they do give a fair indication of the range of results. Haines and Croft are able to obtain a 99.8% increase on average in the CACM abstract collection, and a 30.5% increase on average in the full-text WEST collection. In a followup study to the one presented in this paper, Harman compares term selection, term weighting, and the combination of selection and weighting [17]. She shows that, using the selection formula defined by Equation (11) and the weighting formula defined by Equation (10) that the average improvement in precision by reweighting is 23.9%, the improvement using term selection is 91.0%, and the combined improvement is 112.2%. For this study, Harman again uses the Cranfield collection of 1400 aeronautical abstracts with 225 queries.

Thus, as a baseline indication, a reasonable system should be able to demonstrate at least a 100% performance increase, as measured by the average improvement in precision, using both term selection and term weighting for collections containing only abstracts. For full text collections, a reasonable system should be able to boast at least a 25% increase in performance.

6.2 Combining Weighting and Selection under Different Models

The difficulty in combining weighting and selection formulas based on different models is that the intermediate variables necessary to calculate something in the vector model are completely unusable

for calculating something in the probabilistic model. Thus, the real question becomes whether or not the performance increase outweighs the time and space tradeoff for the additional overhead.

In Harman's 1992 followup study, she looks at some probabilistic term selection formulas, in particular the *wpq* formula, using two different variations for vector-based term weighting. Her findings indicate that both variations of the *wpq* formula perform without significant difference from the other three top selection formulas studied. Efthimiadis' findings, summarized in Section 5.3.2, are similar, although he suggests that his *r-lohi* formula works well all around.

Many studies show that trimming the number of terms used for expansion results in an increase in performance, although no study has conclusively found a single selection formula that works well in all circumstances. In most cases, they find two to four formulas that tend to work equally well. With that in mind, it is likely that combining weighting and selection functions from different models will not result in any significant performance increase over using selection and weighting functions based on the same model, and that the added overhead will make the combination unattractive to operational systems.

6.3 Best Methods

Having surveyed the primary techniques and models, the obvious question of "Which is the best?" arises. Not surprisingly, others have investigated this question.

A 1990 study by Salton and Buckley compares a variety of vector and probabilistic models, using query expansion of all terms and a select number of relevant terms [42]. They determine that *Ide Dec-Hi* augmented with a term selection formula is the best performer of the bunch, and that, in general, vector-based systems outperform probabilistic.

Naturally, this brings several counterclaims and other studies. Partially in response to this, TREC, the Text REtrieval Conference, was created. The idea behind TREC is that all interested parties would be given some test collections. They would run their IR systems on those collections, and submit their results. Then, everyone's results would be evaluated using the exact same metrics, allowing for comparison between systems as a whole. Recently, SMART has been outperforming other systems at TREC, although it hasn't been winning by large margins [18].

Based on the Salton and Buckley study, along with recent TREC studies, it would seem that the best way to implement a Relevance Feedback system would be to use *Ide Dec-hi* with an appropriate term selection function, such as Equation (11). It is also much easier to implement, involving fewer calculations and manipulations than either the standard probabilistic or inference network models. However, while a vector-based model may currently yield the best performance, researchers have been unable to make large performance improvements to the top vector-based systems in recent years [27]. This implies that for a research system, the probabilistic model may still have some undiscovered techniques for greater performance.

6.4 Combining Redundant Techniques

A technique that is gaining popularity is the *meta-engine*. A meta-engine is simply an engine that submits a query in parallel to distinct sub-engines and collates the results somehow. This has been

done successfully using both traditional IR systems [2, 1, 14] as well as Web-based systems [46, 47]. The meta-engine technique is designed to return more relevant documents on the initial query. However, the technique may be applicable to term weighting and selection techniques as well. The assumption here is that terms weighted highly by multiple techniques are more important than those weighted highly by a few or one; likewise, terms selected by multiple techniques are likely more important than those selected by one.

Thus, the feedback system envisioned here would be a combination of selecting those terms that several term selection formulas used, and then weighting those terms using some combination of weighting formulas. Based on results from other fusion methods, I would hypothesize that this method would be an improvement over existing methods, but it is an open question as to how well this approach would actually perform.

6.5 Unstated Assumptions

The methods and techniques reviewed and suggested in this paper all have a common set of fundamental assumptions regarding how an IR system operates and how a Relevance Feedback interacts with it. In particular, some of these fundamental assumptions are now being re-evaluated as IR systems are used for a wider variety of tasks. I will now present a brief look at some of these assumptions, how they are being relaxed, and what affects that may have on the IR system.

6.5.1 Documents and abstracts are “well-formed”

Most IR systems, and subsequently RF systems, are designed for and evaluated with documents or abstracts that make a conscious effort to communicate something. In essence, all text that is indexed by an IR system is assumed relevant to some query.

As IR systems are deployed in new settings, the assumption regarding the quality of documents is being tested. Particularly with the World Wide Web, documents are often nothing more than a collection of single words and hyperlinks to other sites. Another difficulty with the Web is that authors, in attempts to attract people, often insert misleading words or phrases in efforts to appear relevant to searches. Thus, document ranking algorithms need to be re-evaluated in the face of very little information as well as intentionally misleading information.

6.5.2 Users queries are “well-formed”

In addition to assuming documents are well-formed, IR researchers often assume that queries are well-formed, meaning that they contain a large number of keywords, and a human can judge whether a document satisfies a query with a reasonably high rate of accuracy.

Various researchers have found that users actually enter between one and three keywords [30, 3]. The direct result of this is that the query is under specified, thereby inundating the user with irrelevant documents. Thus, there would seem to be a clear need for Relevance Feedback. However, it is also likely since the initial results are poor, there may be little relevance information initially available — perhaps only one document. There has been some earlier research with little relevance information

[22], however this needs to be revisited and applied to current practices.

6.5.3 Users make “perfect” choices

In the evaluation of RF systems, it is assumed that the user makes a “perfect choice” in that all documents that are relevant are marked as such. In operational settings, this is unlikely, and thus the empirical results only give an upper bound on performance. However, while a user may not select some relevant documents for feedback, it is typically assumed that the user will not mark an *irrelevant* document for feedback. In addition to revisiting feedback in the context of limited relevance information, it would seem that evaluating different feedback techniques in the context of noisy feedback information is warranted, especially as users interact with new, unfamiliar IR systems employing some form of RF.

6.5.4 Closed World Assumption

One assumption in an IR system is that all documents in a collection are indexed. Thus it is possible to find all relevant documents, and know that all documents have been found, given the proper query. In the field of Artificial Intelligence, this is known as the *Closed World Assumption*. Relevance Feedback techniques are designed with the assumption that all documents are indexed, and that their purpose is to find relevant documents that have been indexed, but have not yet been returned to the user.

In many cases, in particular the World Wide Web, the document index is often incomplete. Thus, it may be possible for an IR system to return all relevant documents that have been indexed, but not all relevant documents in the collection. In order for Relevance Feedback to work, it will likely need to engage in search through unindexed documents. This assumption alters how Relevance Feedback operates more than any I’ve stated. By adding a requirement that a RF technique search through unindexed documents, there are now questions as to which documents to search, in what order, and at what cost. There has been some work on the Web that implements a system to search for relevant documents on the web given some relevance information [25], however this needs to be evaluated in the context of Relevance Feedback.

7 Speculation

Having surveyed examples of the two major RF techniques in the two prominent mathematical models, as well as overviewing the results of combining those techniques into more sophisticated systems, I argue that Relevance Feedback does in fact help find more relevant documents for the user. Given that, why is Relevance Feedback so conspicuously absent from the major IR systems on the World Wide Web?

7.1 Relevance Feedback is Out There

One answer to the question of why Relevance Feedback isn't available is that it really is available, it's just not prominent. In truth, there are some systems available that use Relevance Feedback. The Fusion project [47] is a meta-search engine similar to MetaCrawler [46]. It forwards queries to six Web search services and collates the results. Currently it is prototyping a Relevance Feedback system. Euroferret [28] based on the Muscat system [31] uses Relevance Feedback to aid the user in expanding queries with appropriate terms. Alta Vista [9] LiveTopics is a Relevance Feedback technique where the system presents the user with a choice of words and allows the user to expand the query based on those words directly. Deviating from most systems, LiveTopics doesn't get relevance judgments from the user initially, and thus presents all candidate terms to the user. Excite [13] has a similar but more rudimentary feature as well.

A similar answer to the above is that while Relevance Feedback is not present on many systems, it is coming. The systems on Alta Vista and Excite would indicate that they are moving towards inclusion of Relevance Feedback. Since most of the popular Web search systems available were created from scratch and are only a few years old, this is a perfectly plausible explanation.

7.2 Feedback is Painful

A more psychological answer is that no one has come up with a Relevance Feedback interface that users are willing to use. In independent studies, Pinkerton and Bray observe that most users only enter one to three terms [30, 3]. This leads to a conjecture that users are impatient and unwilling to interact with a feedback interface, preferring instead to try a manual refinement or find the information a different way.

7.3 Resource Requirements

Something that has not been an issue in RF literature are the system requirements and performance of the systems. The techniques described in this paper all have a theoretical lower time bound that is linear in the size of the judged document set and number of terms indexed in those documents. It is entirely unclear how much time an implementation of any of these techniques takes. There are also questions regarding space. In an IR system that does not use Relevance Feedback, once a document has been inserted into its index, the IR system has no further need of it, and thus space can be saved by discarding the document. On the other hand, RF techniques require the document to be present, and thus additional space is necessary to store the document. In addition, there will be additional I/O overhead as the document is accessed.

Most of the popular Web search systems currently have document collections of between 30 and 55 million documents [49], receive about 5 million queries per day, and are able to satisfy each query within a few seconds. In order to accomplish this, these systems are running on top-of-the-line servers with multiple CPUs and gigabytes of memory. It is an open question as to the amount of additional resource requirements a Relevance Feedback system would consume, as well as if any such system could satisfy queries quickly enough to meet user expectations.

7.4 User Access Patterns

As stated in Section 2.1, Relevance Feedback is most useful with Exhaustive queries, when a user needs to find more, or often all, relevant documents. However, if a user is simply searching the Web for the answer to some question and any relevant document will do, then there is no real need for Relevance Feedback for Web IR systems. Thus, while Relevance Feedback does help performance for Exhaustive queries, it is an open question what percentage of users have queries that will benefit from feedback.

7.5 Hyperlinks obviate Relevance Feedback

The final possible answer to why Relevance Feedback isn't conspicuous on the Web deals with the very structure of the Web itself. One of the motivating reasons behind Relevance Feedback is that the only way to get to a document is by querying the retrieval system. On the Web, this is not the case. Users are able to travel from document to document via the hyperlinks contained within. Since documents connected via hyperlinks are generally related in some way, it is a valid conjecture that a relevant document will have hyperlinks to other relevant documents. Some studies have even proposed using the number of hyperlinks pointing to a particular page as a method of augmenting document ranking functions [26, 48]. Thus, another open question is whether users are better served by the traditional Relevance Feedback techniques surveyed in this paper versus exploiting the hyperlinks found in the initial set of relevant documents returned via a conventional IR system.

8 Conclusions

In this paper, I attempt to ascertain why Relevance Feedback techniques are currently absent from the Web. I survey four feedback techniques, spanning both probabilistic and vector models as well as emphases on term weighting and term selection. I further detail some of the results of combining these techniques and the gains that can be obtained. Finally, having laid the groundwork of how these systems work and what they require, I speculate on the possible reasons as to why Relevance Feedback is not available on the Web in any useful form. In doing so, I detail several open questions regarding Relevance Feedback. It is my belief that in investigating these questions both existing and new, innovative Relevance Feedback techniques will be implemented on the Web and thus improve everyone's ability to find good, useful, and relevant information.

9 Acknowledgments

I would like to thank Mary Kaye Rodgers, both for her assistance in proofreading this paper and for her patience with my late nights and short attention span. I would also like to thank Oren Etzioni for his help in guiding the initial paper selection, and Raya Fidel for helping get my initial bearings in the world of Information Retrieval.

References

- [1] N. J. Belkin, P. Kantor, C. Cool, and R. Quatrain. Combining Evidence for Information Retrieval. In Donna Harman, editor, *TREC-2, Proceedings of the Second Text REtrieval Conference*. NIST Special Publication 500-215, 1993.
- [2] N. J. Belkin, P. Kantor, E. A. Fox, and J. A. Shaw. Combining the Evidence of Multiple Query Representations for Information Retrieval. *Information Processing & Management*, 31(3):431–448, 1995.
- [3] Tim Bray. Measuring the Web. In *Proceedings of the 5th World Wide Web Conference*, 1996.
- [4] John Broglio, James P. Callan, and W. Bruce Croft. INQUERY System Overview. In *Proceedings of the TIPSTER Text Program (Phase I)*, pages 47–67. Morgan Kaufmann, 1994. URL: <http://ciir.cs.umass.edu/info/psfiles/irpubs/brogliocallancrofttipI.ps.gz>.
- [5] B. C. Brookes. The Measures of Information Retrieval Effectiveness Proposed by Swets. *Journal of Documentation*, 24(1):41–54, 1968.
- [6] C. Buckley, A. Singhal, M. Mitra, and (G. Salton). New Retrieval Approaches using SMART: TREC-4. In Donna Harman, editor, *TREC-4, Proceedings of the Second Text REtrieval Conference*. NIST Special Publication 500-236, 1995.
- [7] Y. K. Chang, C. Cirillo, and J. Razon. Evaluation of Feedback Retrieval using Modified Freezing, Residual Collection, and Test and Control Groups. In Gerard Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 17, pages 355–370. Prentice Hall, Englewood Cliffs, NJ, 1971. Originally published as Section X in Scientific Report ISR-16, Oct. 1969.
- [8] W. Bruce Croft, Howard R. Turtle, and David D. Lewis. The Use of Phrases and Structured Queries in Information Retrieval. In *Proceedings of the 1991 ACM SIGIR Conference*, pages 32–45, Chicago, IL, 1991. ACM Press.
- [9] Digital Equipment Corporation. Alta Vista Home Page, 1996. <http://www.altavista.digital.com>.
- [10] Efthimis N. Efthimiadis. A User Centered Evaluation of Ranking Algorithms for Interactive Query Expansion. In *Proceedings of the 1993 ACM SIGIR Conference*, pages 146–159, Pittsburgh, PA, June 1993. ACM Press.
- [11] Efthimis N. Efthimiadis. User Choices: A New Yardstick for the Evaluation of Ranking Algorithms for Interactive Query Expansion. *Information Processing & Management*, 31(4):605–620, 1995.
- [12] Efthimis N. Efthimiadis. Query Expansion. In Martha E. Williams, editor, *ARIST*, volume 31, chapter 4. Information Today, Inc., Medford, NJ, 1996.
- [13] Excite, Inc. Excite Home Page, 1995. <http://www.excite.com>.

- [14] E. A. Fox and J. A. Shaw. Combination of Multiple Searches. In Donna Harman, editor, *TREC-2, Proceedings of the Second Text REtrieval Conference*. NIST Special Publication 500-215, 1993.
- [15] David Haines and W. Bruce Croft. Relevance Feedback and Inference Networks. In *Proceedings of the 1993 ACM SIGIR Conference*. ACM Press, 1993.
- [16] Donna Harman. IRX: An Information Retrieval System for Experimentation and User Applications. In *RIAO '88*, pages 840–848, Cambridge, MA, March 1988. Center de Hautes Etudes Internationales d'Informatique Documentaires (CID).
- [17] Donna Harman. Relevance Feedback Revisited. In *Proceedings of the 1992 ACM SIGIR Conference*, pages 1–10, Copenhagen, Denmark, June 1992. ACM Press.
- [18] Donna Harman. Overview of the Fourth Text REtrieval Conference. In Donna Harman, editor, *TREC-4, Proceedings of the Second Text REtrieval Conference*. NIST Special Publication 500-236, 1995.
- [19] Donna Harman, D. Bensen, L. Fitzpatrick, R. Huntsinger, and C. Goldstein. IRX: An Information Retrieval System for Experimentation and User Applications. *SIGIR Forum*, 22(3-4):2–10, 1988.
- [20] D. J. Harper and C. J. van Rijsbergen. An Evaluation of Feedback in Document Retrieval Using Co-Occurrence Data. *Journal of Documentation*, 34(3):189–216, September 1978.
- [21] E. Ide. New Experiments in Relevance Feedback. In Gerard Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 16, pages 337–354. Prentice Hall, Englewood Cliffs, NJ, 1971. Originally published as Section VIII in Scientific Report ISR-14, Oct. 1968.
- [22] Karen Sparck Jones. Search Term Relevance Weighting Given Little Relevance Information. *Journal of Documentation*, 35(1):30–48, March 1979.
- [23] Karen Sparck Jones and C. J. van Rijsbergen. Information Retrieval Test Collections. *Journal of Documentation*, 32(1):59–75, 1976.
- [24] Jürgen Koenemann and Nicholas J. Belkin. A Case for Interaction: A Study of Interactive Information Retrieval Behavior and Effectiveness. In Ralf Bilger, Steve Guest, and Michael J. Tauber, editors, *Proceedings of the 1996 ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 205–212, Vancouver, BC, 1996. ACM Press.
- [25] Brian A. LaMacchia. The Internet Fish Construction Kit. In *Proceedings of the 6th World Wide Web Conference*, pages 277–288, Santa Clara, CA, April 1997.
- [26] Massimo Marchiori. The Quest for Correct Information on the Web: Hyper Search Engines. In *Proceedings of the 6th World Wide Web Conference*, pages 265–276, Santa Clara, CA, April 1997.
- [27] Mandar Mitra, Chris Buckley, Amit Singhal, and Claire Cardie. An Analysis of Statistical and Syntactic Phrases. In *RIAO '97: Computer-Assisted Information Searching on the Internet*, pages 200–214, Montreal, CA, June 1997.

- [28] Muscat Ltd. Muscat EuroFerret Home Page.
<http://www.muscat.co.uk/euroferret>.
- [29] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988.
- [30] Brian Pinkerton. Finding What People Want: Experiences with the WebCrawler. In *Proceedings of the 2nd World Wide Web Conference*, Chicago, IL USA, October 1994.
<http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/Searching/pinkerton/WebCrawler.html>.
- [31] Martin Porter and Valerie Galpin. Relevance Feedback in a Public Access Catalogue for a Research Library: MUSCAT at the Scott Polar Research Institute. *Program*, 22(1):1–20, 1988.
- [32] Tadeusz Radecki. Incorporation of Relevance Feedback into Boolean Retrieval Systems. In *Proceedings of the 1983 ACM SIGIR Conference*, pages 133–150. ACM Press, 1983.
- [33] Tadeusz Radecki. Probabilistic Methods for Ranking Output Documents in Conventional Boolean Retrieval Systems. *Information Processing & Management*, 24(3):281–302, 1988.
- [34] S. E. Robertson. The Probability Ranking Principle in IR. *Journal of Documentation*, 33(4):294–304, December 1977.
- [35] S. E. Robertson. On Relevance Weight Estimation and Query Expansion. *Journal of Documentation*, 42(3):182–188, September 1986.
- [36] S. E. Robertson. On Term Selection for Query Expansion. *Journal of Documentation*, 46(4):359–364, December 1990.
- [37] S. E. Robertson and K. Sparck Jones. Relevance Weighting of Search Terms. *Information Processing & Management*, 27(3):129–146, 1976.
- [38] J. J. Rocchio, Jr. Relevance Feedback in Information Retrieval. In Gerard Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 14, pages 313–323. Prentice Hall, Englewood Cliffs, NJ, 1971. Originally published as Section XXIII in Scientific Report ISR-9, Aug. 1965.
- [39] Gerard Salton, editor. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice Hall, Englewood Cliffs, NJ, 1971.
- [40] Gerard Salton. Mathematics and Information Retrieval. *Journal of Documentation*, 35(1):1–29, March 1979.
- [41] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, Reading, MA, 1989.
- [42] Gerard Salton and Chris Buckley. Improving Retrieval Performance by Relevance Feedback. *J. American Society of Information Science*, 41(4):288–297, 1990.
- [43] Gerard Salton, Chris Buckley, and E. A. Fox. Automatic Query Formulations in Information Retrieval. *J. American Society of Information Science*, 34(4):262–280, 1983.

- [44] Gerard Salton, E. A. Fox, and Ellen Voorhees. Advanced Feedback Methods in Information Retrieval. *J. American Society of Information Science*, 36(3):200–210, 1985.
- [45] Gerard Salton, Ellen Voorhees, and Edward A. Fox. A Comparison of Two Methods for Boolean Query Relevance Feedback. *Information Processing & Management*, 20(5/6):637–651, 1984.
- [46] Erik Selberg and Oren Etzioni. Multi-Service Search and Comparison Using the MetaCrawler. In *Proceedings of the 4th World Wide Web Conference*, pages 195–208, Boston, MA USA, December 1995.
<http://huskysearch.cs.washington.edu/papers/www4/html/Overview.html>.
- [47] Alan F. Smeaton and Francis Crimmins. Using a Data Fusion Agent for Searching the WWW. In *Proceedings of the 6th World Wide Web Conference*, Santa Clara, CA, April 1997. Presented as a poster. URL: <http://lorca.compapp.dcu.ie/fusion/papers/fusion-www6.html>.
- [48] Ellen Spertus. ParaSite: Mining Structural Information on the Web. In *Proceedings of the 6th World Wide Web Conference*, pages 201–214, Santa Clara, CA, April 1997.
- [49] Danny Sullivan. Search Engine Watch, 1999.
<http://www.searchenginewatch.com>.
- [50] J. A. Swets. Information Retrieval Systems. *Science*, 141:245–50, July 1963.
- [51] Howard R. Turtle and W. Bruce Croft. Inference Networks for Document Retrieval. In *Proceedings of the 1990 ACM SIGIR Conference*, pages 1–24, Brussels, Belgium, September 1990. ACM Press.
- [52] C. J. van Rijsbergen. A Theoretical Basis for the Use of Co-Occurrence Data in Information Retrieval. *Journal of Documentation*, 33(2):106–119, June 1977.
- [53] Harry Wu and Gerard Salton. The Estimation of Term Relevance Weights using Relevance Feedback. *Journal of Documentation*, 37(4):194–214, December 1981.
- [54] Oren Zamir, Oren Etzioni, Omid Madani, and Richard M. Karp. Fast and Intuitive Clustering of Web Documents. In *KDD-97*, 1997.