RC 21057 (94253) 15, December 1997 **Information Retrieval**

Research Report

An Improved Ranking Formula For Free Text Search Engines

Herb Chong
IBM Research Division
Thomas J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

An Improved Ranking Formula for Free Text Search Engines

Herb Chong

herbie@watson.ibm.com

December 11, 2001

Contents

Figures

Summary

The new family of ranking formulae, Formula 55-57:

1. perform much better than the current GURU formula and performs at least as well as the best one pass formulae we can find

2. are far less biased to short document lengths when returning ranking results

3. are simple and easy to calculate

Other participants at TREC-5 performed better than GURU because of more advanced query processing and by using one iteration of an automatic relevance feedback procedure. In many cases, the other participants were much better than GURU. Even in applications where most queries are very short, automatic relevance feedback still shows worthwhile gains.

The current parser for GURU creates many "junk" entries in the dictionaries. Many of these entries are for words that are built from hyphenating two complete English words or contain numeric quantities. Processing to reduce these entries into a form more easily specified and searched and doing the same at query time would both reduce dictionary size and improve term matching during search.

The current implementation of GURU uses lexical affinities in its ranking formula. However, the original papers on GURU and the use of lexical affinities specifically excludes them from the ranking process and instead tries to identify conceptual affinities. The difference between them is that lexical affinities are word co-occurrences because of language's structural rules while conceptual affinities are word co-occurrences because of conceptual relations between the words. In many senses, the original formula was trying to find multiword terms participating in what we today call named and unnamed relations. We need to revisit this problem and evaluate the effect of being more selective in using term pairs when calculating a document's rank.

Results from the experiments in this report suggest several new avenues of research based on different classes of probabilistic ranking formulae and different sets of assumptions from the GURU ones. However, comparison of our results with the initial pass of other search engines suggest that we may be reaching the limits of what can be achieved by formula differences alone. We probably need to combine the best formula with advanced query processing and automatic relevance feedback to achieve substantially better results.

Conclusions

This report studies five ranking formula, the original GURU one and four new ones based on the same probabilistic model. GURU has been using Formula 1 for several years. It contains LA terms and ordinary word terms. Formula 5 contains only single word terms. Formula 55 extends it to include LA terms with an *ad hoc* change to the formula. Formulae 56 and 57 modify Formula 55 to conform more with theoretical formulations.

Formula 55 with an LA weight of 0.1 is the best on the TREC-5 *a posteriori* test. It performs better than any other formula we have implemented and evaluated. Formula 57 is the most correct mathematically as it can be derived from first principles of probability theory. It performs slightly worse but has a much better theoretical grounding. Both Formula 55 and Formula 57 and their slight variant, Formula 56, are very close to each in TREC-5 precision-recall plots. In turn, these with an LA weight of 0.1 perform better than any variation where the LA weight is 0.0, that is, those that ignore all LA term contribution. As a group, Formula 5 and 55-57 perform better than the other ranking formula candidates evaluated. Figure 1 shows a summary of these formulae's performance. (See “Mathematical Background” for details on the GURU Ranking formulae.)

Formula	Precision at 0 Recall	Average Precision
1	0.5027	0.1119
5	0.5503	0.1476
55	0.5856	0.1588
56	0.5278	0.1411
57	0.5667	0.1462

Figure 1 – Best Performance of Different GURU Ranking Formula

Automatic Relevance Feedback

Of all the participants in TREC-5, only Dublin City University [KELLEDY97] and UC Berkeley [GEY97] besides GURU uses a one pass ranking of documents to arrive at a final document list. All but one of the remaining participants explicitly state or describe their algorithm in sufficient detail to show that they are using automatic relevance feedback as a means of improving recall and precision. Indeed, researchers such as [KWOK96] consider an automatic relevance feedback procedure a standard feature of a modern search engine. In most cases, the submitted papers do not give raw performance of the first pass of the search. This means that just because GURU performs worse than the top search engines does not mean that it is a worse search engine,

just that GURU is not doing any of the advanced processing that the other search engines do to improve their TREC performance.

[ALLAN97] shows the raw performance of the first pass of the INQUERY search engine. It gives a precision recall curve that is significantly worse than the best Formula 55 results except at the very tail of the curve. Since traditionally the full INQUERY system has been one of the top performing search engines in prior TREC conferences, this is strong evidence that Formula 55 is at least as good as other ranking formula used in the first pass of an automatic relevance feedback system. Formulae 56 and 57 are close to enough to Formula 55 that they also are likely to be better. Apple's V-Twin search engine, even though it is optimized for low-end hardware and very short queries, performed noticeably better with automatic relevance feedback than without [ROSE97].

Junk Words

Manually examining the dictionaries produced by indexing the TREC collections showed many "junk" entries in them. Most of these entries were from two categories: hyphens joining two complete English words, or words containing numbers or numerical quantities. Removing the hyphens at indexing and query time would both reduce dictionary size and increase the likelihood of matching terms containing either component word. Transforming many of the words containing numbers to generic forms such as a date, a currency value, or a number would both reduce dictionary size and allow better matching at query time.

Lexical and Conceptual Affinities

Several experiments with random scoring where each term found was assigned a randomly generated score and then the totals aggregated showed that using lexical affinities improved precision and recall even with terrible ranking functions. Without weighting lexical affinities any different from singleton terms, we observed improvements of approximately 0.15 precision at 0 interpolated recall. (See "Random Ranking" for details.)

There is a fundamental difference in how the current Guru search engine and ranking formula work from how Yoëlle Maarek [MAAREK89, MAAREK91] designed it. She was not looking for lexical affinities but conceptual affinities. Lexical affinities are co-occurrences of words in text because of a language's syntactic rules. Conceptual affinities are co-occurrences of words because they reflect conceptual relations. Thus mere co-occurrence is not a sufficient criterion for inclusion in the ranking calculation. [MAAREK91] tried to

empirically identify conceptual affinity terms by filtering out low information content co-occurrences and retaining and ranking only terms that met an empirical threshold of information contribution.

Recommendations

We should use Formula 57 to rank documents in GURU and derived free text search engines. Although not the absolute best performing formula, it is close to the best, Formula 55. The advantage of using Formula 57 is that it is much better constructed theoretically and can be normalized to allow across query and across corpus document score comparisons. It has an absolute maximum value independent of any corpus or query specifics and allows a threshold to be chosen for determining the "significance" of ranked documents.

To improve search engine performance beyond what we can achieve with Formulae 55-57, we need to concentrate on four research areas:

1. Improve index quality so that indexed terms are of more regular form. We have new TALENT tools for word patterns and dehyphenation that will substantially reduce junk words in the index. After these have been removed, there may be other word patterns we can identify and also regularize.

2. Filter noise LA terms by correctly implementing treatment of pairs of terms. We can have much of this effect if we recognize and index multiword terms. It then becomes easier to detect noise terms. Implementing multiword terms will affect the current value of LA distance. We must correct for this by a series of experiments to determine the new value.

3. Find better ways to do query modification, augmentation and refinement, either automatically or manually. Many of the TALENT tools used elsewhere can help at two distinct phases, pre-search query augmentation, and post-processing of the first pass hit list to further augment the query.

4. Do more theoretical work on ranking formulae based on removing several of the most unwarranted assumptions, particularly the ones asserting the independence of terms. There also is a class of ranking formula that do not use a weighted average but confidence measures to compensate for sampling effects. They would, by their nature, include the lexical affinity versus conceptual affinity difference property of the original GURU formula. We need to study their usefulness as ranking formulae.

Introduction

The current NetQuestion product and its immediate predecessor, SearchManager, incorporate portions of a free text search engine named GURU developed at IBM Research. A free text search engine's ranking formula is the most crucial component in determining its effectiveness in retrieving good documents in the first few documents in a hit list. Yoëlle Maarak and Mark Wegman developed the GURU ranking formula in 1989. Since then, except for minor bug fixes and algorithmic changes in the C implementation embodied by the GURU research testbed search engine, no work had been done to substantially change the formula.

The original GURU formula was created by a modification of a probabilistic ranking formula. All of GURU's formulae differ from standard probabilistic ranking formulae in that they try to measure how closely a given document matches a query without any assumptions about how well the query matches any hypothetical concept or topic. They also incorporate an innovation in ranking based on associations of word pairs.

Word pairs play a strong role in every important GURU formula. Incorporating word pairs is a result of the observation that there are conceptual co-occurrence correlations between words within a certain distance. [MAAREK89] and [MAAREK91] concluded that about 98% of conceptually related words occur no more than five words apart. Prior work described in [SALTON89] concluded that multiword terms or co-occurrences were too expensive and unreliable to use. Finding and using a simpler mechanism to take advantage of word co-occurrences distinguishes Guru from most of other available technology. Today's available computing power also expands the feasible multiword term processing options even without the simplification.

Words that co-occur frequently within a five word window, however, may do so for a variety of reasons beyond conceptual relationship. The simplest is *lexical* affinities (LAs). A language's structure may require that certain word pairs occur together frequently. Such pairings are semantically similar to singleton terms as stop words, i.e., they may be ignored for the query. Only *conceptual* affinities (CAs) contain meaning and help to characterize a document. These are words that co-occur because they express a concept or relationship. Moreover, conceptual affinities involve only nouns, verbs, adjectives, and adverbs. Thus, identifying lexical affinities from the query that also occur in documents is only the first step toward finding conceptual affinities.

Although no formal measurements were made on the GURU formula's quality of results, empirically, we had noticed anomalies and deficiencies in the hit list quality. One major problem was the dominance of lexical affinities in a document's score. A single LA could account for most of the scoring value of a highly ranked

document even for a long query. Another is a strong bias toward returning short documents as the best ranked. As we developed more sophisticated applications built on free text search, knowing the source of the deficiencies and their correction became more and more important.

Early in 1995, Mark Wegman along with Vincent Della Pietra began working on a series of new ranking formulae [PIETRA95] based on work done by the Speech Recognition Group at IBM Research. These formulae are still based on probabilistic techniques as before, but pay much more attention to the distribution of words in a document and in a corpus and sometimes in their correlations. As a result, by late 1995, GURU incorporated seven different ranking formulas, but we had no objective means to measure performance. Several informal tests existed, but only one major one was recognized industry-wide, TREC.

Partly as a result of creating new ranking formula, the Digital Libraries Group decided to participate in TREC. The Text REtrieval Conference is sponsored by the National Institute of Standards and Technology (NIST). Part of TREC's mission is to provide a standard set of benchmarks for comparing text search engines. To do this, they provide a standard set of documents, queries, tools, and a procedure to evaluate the quality of results returned by a particular engine. Participating gave us an opportunity to compare our ranking formulae to other major industry and research search engines. (See "Experimental Technique" for the details of participating in TREC.)

The performance number most quoted when comparing search engines is the precision. Although explained in detail later, precision can be though of as the portion of the retrieved fixed length hit list that is relevant to a query. Another frequently quoted number is recall. This is the portion of the total relevant material actually retrieved in the hit list. When precision and recall are plotted against one another, the resulting precision-recall curve characterizes the effectiveness of a search engine. Figure 2 shows two hypothetical precision-recall plots. The more the curve is to the right and to the top of the axes, the better the search engine. (See "Experimental Technique" for a more precise definition of precision and recall and a description of how NIST calculates the curve.) Thus, the blue line represents a much better result than the red line.

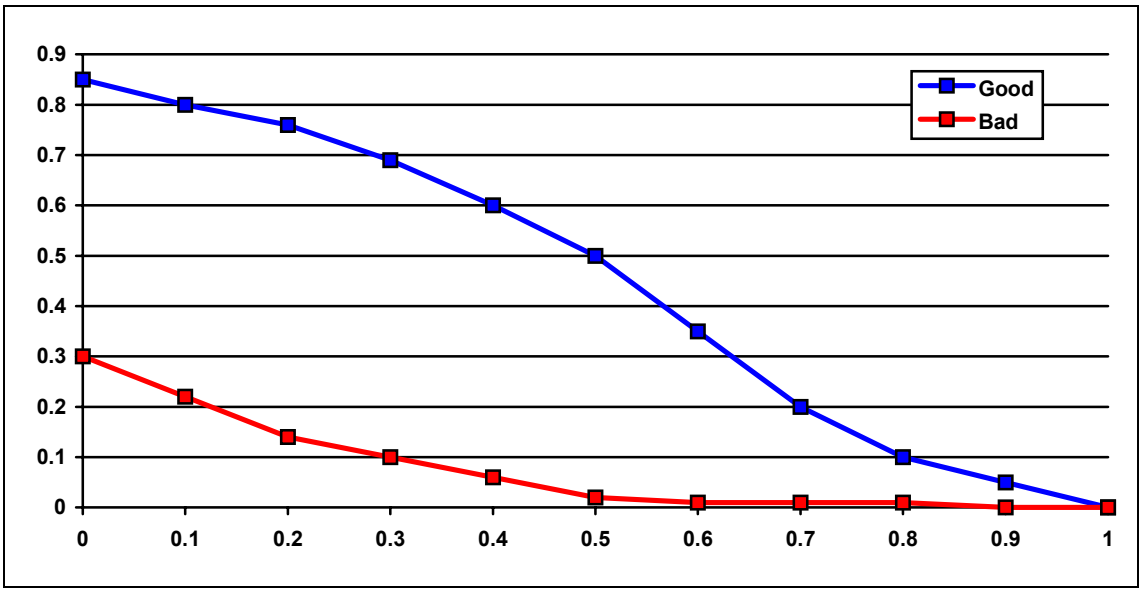

Figure 2 – Good vs. Bad Precision-Recall Performance

For TREC-5, we submitted the results of two formulae, 4 and 5. At the time, these were the most developed. (See "Mathematical Background" for details about the formulae.) Both had not been tested except in empirical tests by Mark Wegman. They were chosen solely based on theoretical grounds. The main difference between Formula 4 and Formula 5 is that Formula 5 ignores LAs completely. We ran evaluations of the other formulae too, but they were not submitted to TREC.

The results we received were surprising, to say the least. Figure 3 shows the official precision-recall curve for the submissions. Formula 5 significantly outperformed Formula 4 on precision despite ignoring LA terms and making many simplifying assumptions. Afterwards we were able to determine that there was a bug in the Formula 4 implementation that made the results useless. However, fixing the bug never made Formula 4 perform as well as Formula 5.

Overall, GURU ranked as merely average in TREC-5. See Figure 4 for a comparison with other Automatic Adhoc Query participants. IBM Digital Libraries ranked 13th and 23rd respectively for Formula 5 and Formula 4. The below-median showing, in addition to empirical observations on performance and other theoretical and practical reasons, made it very important to understand and improve GURU's ranking formula and build those into IBM's search products. Figure 5 shows the Precision Recall curves for the best entry by each organization. GURU Formula 5 is visibly below median performance.

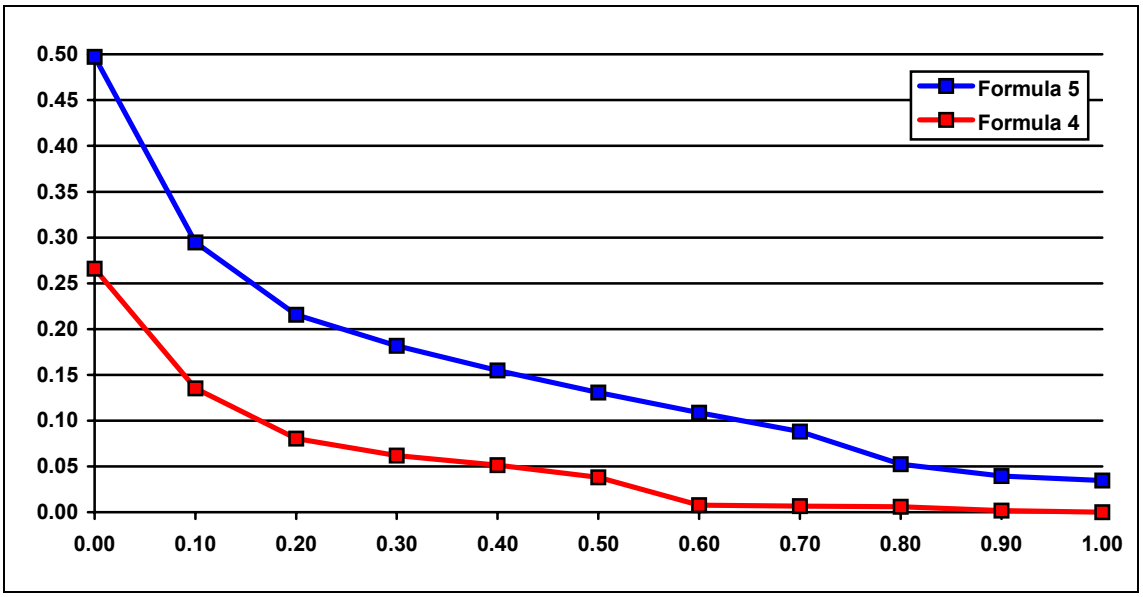

Figure 3 – Official TREC-5 Results for Guru

Entry	Precision at 0 Recall	Average Precision
Lexis-Nexis - 2	0.7288	0.2699
Lexis-Nexis - 1	0.7189	0.2661
University of Massachusetts	0.5919	0.2002
Cornell University - 2	0.5857	0.2109
Cornell University - 1	0.5841	0.2065
University of California, Berkeley	0.5716	0.1420
IBM Speech - 1	0.5472	0.1585
IBM Speech - 2	0.5463	0.1623
Apple Computer	0.5417	0.1894
General Electric	0.5405	0.1460
Queen's College, CUNY	0.5292	0.1809
City University of London	0.5139	0.1751
IBM Digital Libraries - Formula 5	0.4974	0.1451
Swiss Federal Institute of Technology(ETH)	0.4959	0.1726
Australia National University - 2	0.4858	0.1538
Royal Melbourne Institute of Technology	0.4685	0.1214
Australia National University - 1	0.4679	0.1537
Dublin City University - 1	0.4525	0.1340
Dublin City University - 2	0.4404	0.1334
George Mason University	0.4392	0.1079
University of Kansas - 1	0.3971	0.1073
University of Kansas - 2	0.3956	0.1056
IBM Digital Libraries - Formula 4	0.2660	0.0462

Figure 4 – Official TREC-5 Results for All Participants: Automatic Adhoc Short Queries

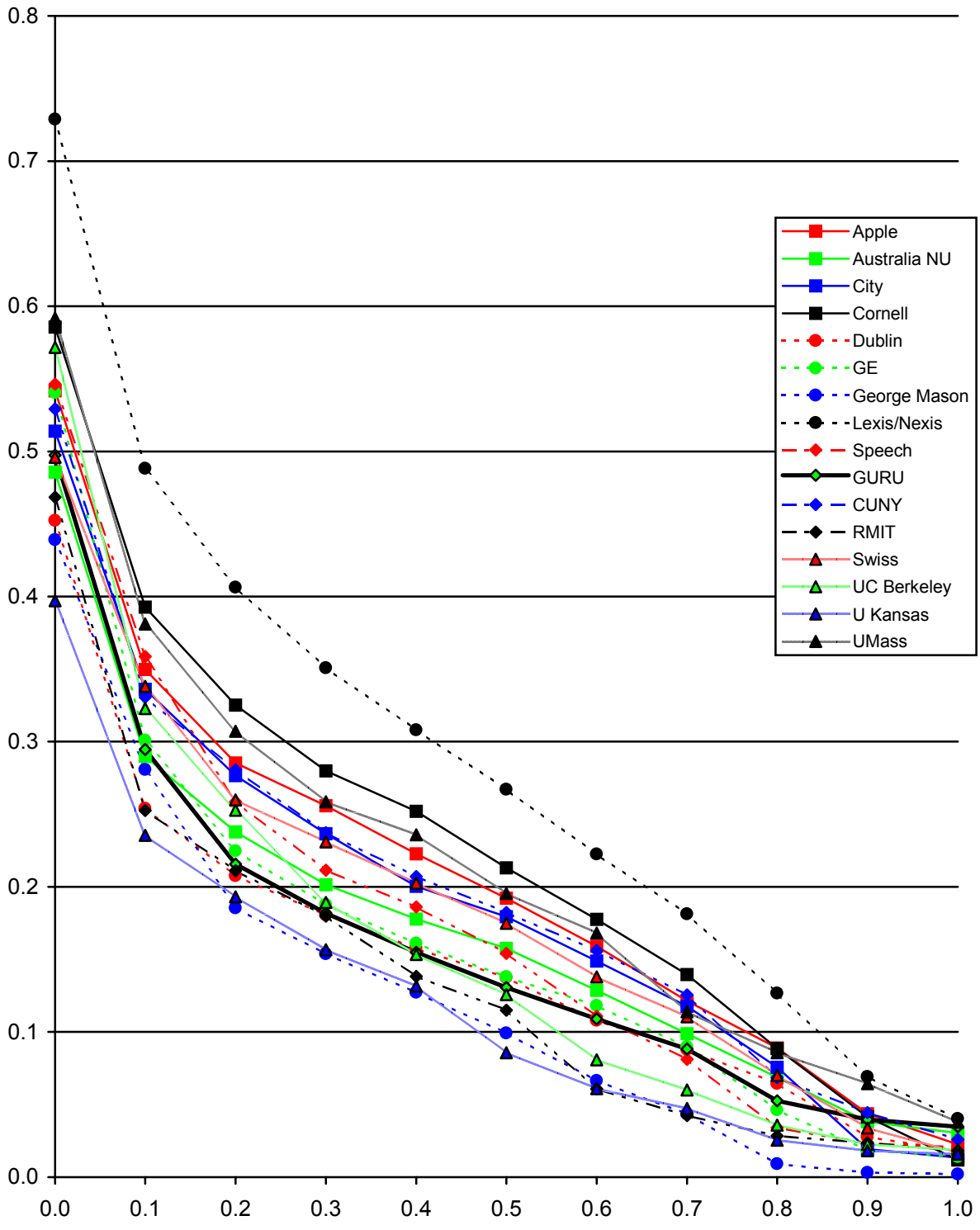

Figure 5 – Automatic Adhoc Short Query Precision Recall Curves by Organization

Mathematical Background

There are three main mathematical models used for ranking documents in information retrieval systems: Boolean models based on set theory, vector space models, and probabilistic models. Boolean models have fallen into disfavor because Boolean search systems are now considered outmoded technology. None of the TREC-5 Automatic Adhoc participants used Boolean searching, although there was one entry participating in the Manual Adhoc and Automatic Routing tracks. Several other participants used hybrid systems containing Boolean components in their search processing and ranking. Almost all of the remainder used vector space or probabilistic models or a hybrid of both, some with other techniques too. GURU uses a pure probabilistic formula for doing document ranking. Because of this, there will be only a general discussion of the other ranking techniques in this report. [SALTON89] contains good background material on all three techniques.

Extended Boolean Models

Boolean search models have their roots in early keyword and database-derived systems. A document was characterized by a tuple of terms (words or keywords). Boolean filters applied to a set of tuples result in a retrieval set of documents. No particular ordering as to relevance was practical given how much information was discarded to arrive at the keyword set.

As computing power increased, instead of indexing documents only by a carefully human-chosen abstraction of their contents, it became possible to index based on automatically extracted content. This allowed computing an occurrence count for all terms that appear in a document. With occurrence counts, it became possible to calculate rankings for documents.

IBM's SearchManager product does Boolean document ranking [IBM93]. Like most extended Boolean ranking schemes, it defines how to calculate the score for a particular term in a document. Each operator in the Boolean query language has a corresponding algebraic operation based on Boolean logic. After a suitable document has been selected, calculating its rank involves evaluating the expression associated with the Boolean query. For the basic Boolean operators such as AND, OR, and NOT, there are simple expressions that can be derived from set theory. However, many Boolean query languages allow NEAR and similar operators. These often have empirical expressions associated with them for calculating ranks.

The main difficulty with Boolean ranking is creating a way to calculate a term's score if it occurs in a document. Simply counting the number of occurrences or using some monotonically increasing function of it lets long documents dominate the results as longer documents are more likely to contain more occurrences of any given term. Scaling by some function of document length or total occurrences in a corpus approaches probabilistic techniques discussed later.

There is no general agreement on Boolean ranking functions and there is far less theoretical basis for these than for other types of document ranking [SALTON89]. Boolean search expressions are also harder to create and use. That is the main reason why they have fallen into disuse in advanced systems.

Vector Space Models

Vector space models take a similar approach to characterizing a document as Boolean systems. They derive a tuple representing the document from its content. However, instead of working with the tuple purely as a set of terms, the tuple defines a point in a vector space by assigning a numerical value to each term component. The component value frequently is a monotonically increasing function of its number of occurrences in the document, perhaps scaled by a function of document length. Commonly used values are simple [0, 1] set of binary values. Others are "importance" coefficients calculated from the corpus and document.

The beauty of the vector space representation is that queries are also represented as tuples in the same vector space. Using a suitable similarity measure based on vector operations assigns a document's rank with respect to a query tuple. There are a variety of similarity measures that have been studied. (See [SALTON89], pp. 313-319 for details.) Nothing in the model imposes restrictions on the nature of the similarity function except that it be a monotonically decreasing function of Euclidean distance.

Most vector space systems assume that the vector space's basis is orthogonal, that is, the terms in a corpus are uncorrelated. Doing otherwise greatly increases storage requirements for an indexed corpus and also greatly increases computational costs. Also implicit in the formulation is the assumption of a Euclidean vector space.

Major difficulties with the vector space model are the frequently assumed orthogonality of the vector space and the lack of a theoretical basis for choosing a similarity function. Other problems include how to compute the component values for the tuple. Despite these problems, vector space models are popular because of the ease of implementation and simple geometric interpretation of finding documents similar to a query.

Probabilistic Models

Probabilistic modeling of document ranking solves one of the most vexing problems in vector space modeling, the arbitrariness of the similarity function. From [SALTON89], many common formulations start with two parameters for document d_i, $P(d_i, R_i|Q)$, the probability of relevance given a query, and $P(d_i, \overline{R_i}|Q)$, the probability of nonrelevance given a query. If relevance is a binary property, then $P(d_i, \overline{R_i}|Q)$=1-$P(d_i, R_i|Q)$. If we associate a loss function for the cost of misclassifying a document as relevant when it is nonrelevant and one for nonrelevant it is relevant, the total loss caused by a retrieval process will be minimized whenever $l_1 \bullet P(d_i, R_i|Q) \geq l_2 \bullet P(d_i, \overline{R_i}|Q)$. From this definition, we can derive a ranking function of the form

$$r(d_i) = \frac{P(d_i, R_i|Q)}{P(d_i, R_i|Q)} - \frac{l_1}{l_2}.$$

Larger values imply a better match. If we assume that l_1 and l_2 are constant, applying Bayes' Theorem to $r(d_i)$ yields

$$r'(d_i) = \frac{P(d_i|R_i, Q)}{P(d_i|\overline{R_i}, Q)} \bullet \frac{P(R|Q)}{P(\overline{R}|Q)}.$$

where $P(R|Q)$ and $P(\overline{R}|Q)$ are the *a priori* probabilities of a document being relevant and nonrelevant respectively given a query. Most frequently, we take logarithms of both sides of this equation and arrive at the form

$$R'(d_i) = \log \frac{P(d_i|R_i, Q)}{P(d_i|\overline{R}_i, Q)} + \log \frac{P(R|Q)}{P(\overline{R}|Q)}$$

for ease of evaluation. Many of the probabilistic ranking formulae in the literature use this formula or something like it as a departure point for building their rankings. The exact differences usually come about in the distributional assumptions and in how to estimate the properties of a document given the properties of the individual terms that make up a document. The corresponding problems apply to the properties of a query given the properties of individual query terms.

TF•IDF

Suppose that there is a set of important terms, i.e., a query. One way to rank documents in a search is to order them such that the documents where a chosen query term is very important occur ahead of documents where the

same term has lesser importance. Assuming some appropriate means of combining the importance, or belief, of each term in the same document from the individual terms of a multiterm query, one has a general purpose ranking formula for a search engine. The trick is in what to use as an importance function. An entire category of search engine ranking functions based on this general approach have become known as tf•idf techniques because their importance or belief formulae contain document term frequency (tf) and corpus inverse term document frequency (idf) components.

The notion in a tf•idf ranking formula is that if a term appears frequently in a document, the document must be more about the term than in a document where it appears less frequently. However, this is not very useful when there are many documents where this term appears. If the overall frequency of a term in a corpus is low or the number of documents that it occurs in is low, a term's frequency contributes more to the belief that a document is about a term.

Let w_{ij} be the importance (belief or weight) of a term j in document i. Also, let t_{ij} be the frequency of occurrence of term j in document i, f_i be the number of documents where term j occurs, and N be the number of documents in the corpus. One commonly used early tf•idf ranking function, according to [SALTON89], is

$$w_{ij} = t_{ij} \bullet \log \frac{N}{f_i}.$$

Note that for a given corpus, this version of w_{ij} is a monotonically decreasing function of f_i.

Tf•idf ranking functions can also be couched in information theoretic terms. A term's importance can be thought of as the information contribution of a term to a document, or its entropy. The entropy of a term with occurrence probability p is given by $-\log p$ and the average information value per word for n_Q distinct terms occurring with probabilities $p_1, p_2, \ldots, p_{n_Q}$ is

$$\overline{H} = -\sum_{j=1}^{n_Q} p_j \log p_j .$$

This is another type of ranking function that can be used in a tf•idf search engine. From these two basic types of importance functions arise most of the tf•idf ranking formula. Empirical modifications as attempts to correct for a given formula's inadequacies constitute most of the changes seen today.

INQUERY

INQUERY from the University of Massachusetts at Amherst is traditionally one of the better performing search engines in the TREC conference. It has so far used tf•idf formulae of various types through the different conferences. Without making any attempts to justify the formula, we present it to show a sample of the various kinds of modifications made in a modern tf•idf formula. The basic description is included in [ALLAN97]. In summary, it is an amalgamation of prior INQUERY weighting formula and an OKAPI variation.

Let

$$F_j = \frac{\log\left(\frac{C+0.5}{f_j}\right)}{\log(C+1.0)}$$

$$T_i = \frac{\log(t_j + 0.5)}{\log(\tilde{t} + 1.0)}$$

$$T_0 = \frac{t_j}{t_j + 0.5 + \frac{1.5 \bullet l}{\bar{l}}}$$

$$\hat{t}_j = \bar{t} + (1 - \bar{t}) \bullet (F_j \bullet T_i + (1 - F_j) \bullet T_0)$$

$$\bar{t} = \frac{\log \bar{l}}{24} e^{-5\frac{f_j}{C}}$$

$$w_j = 0.4 + 0.6 \bullet \hat{t}_j \bullet F_j$$

where C is the number of documents in the corpus, f_j is the number of documents where the term appears, t_j is the number of times the term appears in the document, $\tilde{t}$ is the number of times the most frequent term in the document appears, l is the number of terms in the document, $\bar{l}$ is the average number of terms in a document, and $\bar{t}$ is an empirical threshold for a term weight that appears to be collection dependent. Despite appearances this is still considered a tf•idf formula. One thing quite obvious is the number of arbitrary numerical constants such as 24, 0.5, and 1.5.

OKAPI

OKAPI by the City University of London is another well known ranking formula in the IR community. Its basis is a tf•idf scheme with a strong probabilistic foundation. Another characteristic is that the ranking formula itself depends on relevance terms that can be determined only by an automatic relevance feedback procedure. Although OKAPI is referred to as if it were one ranking formula, it is actually a family for formulae based on the same theoretical foundations, but with different empirical adjustments for counteracting various unwanted effects.

Let n_D be the number of documents in a corpus, f_j be the number of a document containing the query term q_j, R be the number of documents known to be relevant to a specific topic, and r be the number of relevant documents containing the term. Then $w^{(1)}(q_j)$, the Robertson/Sparck Jones weight or belief of a query term q_j (and the idf component of the ranking formula) is given by

$$w^{(1)}(q_j) = \log\left(\frac{(r+0.5)/(R-r+0.5)}{(f_j-r+0.5)/(N-n-R+r+0.5)}\right).$$

The terms R and r must be evaluated by some automatic relevance feedback procedure.

Let

$$K = k_1 \bullet \left((1-b) + b \bullet \frac{l_i}{\bar{l}}\right)$$

be an empirical weighting constant where k_1 and b are corpus-specific free constants, l_i be document d_i's length, and $\bar{l}$ be the average length of all documents in the corpus.

Given $w^{(1)}(q_j)$, the weight of a query term for document d_i is given by

$$w(q_j, d_i) = w^{(1)}(q_j) \bullet \frac{(k_1+1)n_{q_j \in d_i}}{K + n_{q_j \in d_i}} \bullet \frac{(k_3+1)n_{q_j}}{k_3 + n_{q_j}} + k_2 \bullet n_Q \bullet \frac{\bar{l} - l_i}{\bar{l} + l_i}$$

where $n_{q_j \in d_i}$ is the number of times the term q_j appears in document d_i, n_{q_j} is the number of times the term appears in the query, n_Q is the total number of unique terms in the query, and k_2 and k_3 are more corpus-specific free constants. A document's weight is simply the sum over all unique query terms of $w(q_j, d_i)$. This is the OKAPI formula used in their TREC-5 entry [BEAULIEU97], and in turn is a slight variation of those used

in TREC-3 [ROBERTSON95] and TREC-4 [ROBERTSON96]. Each of the three terms modifying $w^{(1)}(q_j)$ accounts for a different effect.

The first effect is simply the tf component of the ranking formula. Since OKAPI is a tf•idf formula, there is no theoretical basis for choosing a particular tf formula. OKAPI uses a 2-Poisson model of a term's occurrence frequency in documents to derive its contribution to a document's weight [ROBERTSON94]. The 2-Poisson model postulates that documents in a corpus can be divided into two sets, one where the term is "important" and where the term is not "important". The term's within-document frequency in each of the sets is Poisson distributed with different means. The one with the larger mean is where the term is more important. The exact formula for a term's weight given this model is a large and complex exponential term. OKAPI does not evaluate this function. Instead, it uses an empirical function that has similar characteristics to it and uses it to modify $w^{(1)}(q_j)$. Thus

$$w(q_j, d_i) = w^{(1)}(q_j) \bullet \frac{n_{q_j \in d_i}}{k_1 + n_{q_j \in d_i}}$$

where k_1 is an unknown and possibly corpus-specific constant.

The second effect that an empirical term accounts for is document length. OKAPI assumes that long documents about a topic have the same term distribution as short documents. This is commonly known as the verbosity hypothesis. The net effect is that k_1 is scaled by a function of document length and average document length. Thus, the formula can be modified to be

$$w(q_j, d_i) = w^{(1)}(q_j) \bullet \frac{n_{q_j \in d_i}}{\frac{k_1 \bullet l}{\bar{l}} + n_{q_j \in d_i}}$$

However, this is not all there is. A detailed analysis of the 2-Poisson model shows that an additive correction term needs to be applied that is a function of document length and query length. This term can be modeled as

$$c_1 = k_2 \bullet n_Q \bullet \frac{\bar{l} - l_i}{\bar{l} + l_i}.$$

The final effect is that of query term frequency and query length. OKAPI postulates that, just as for documents, there is a 2-Poisson model of terms in a query. This leads to a query-based multiplicative correction factor

$$c_2 = \frac{n_{q_j}}{k_3 + n_{q_j}}.$$

Nothing in the derivation of the correction factors suggests what are appropriate values for any of the free constants. Each of the OKAPI participations in TREC only suggest that these are corpus dependent and that in some cases, the best values to use are 0. Like most tf•idf formulae, what probabilistic techniques used to justify various components of a term's weight have many empirical adjustments or approximations to correct for various effects found during evaluation of the formula.

GURU

GURU's ranking formulae take a different approach to ranking than the others mentioned so far. It matches documents to the query given no other information about the query or its relevance to a hypothetical or real topic. As used today, GURU is a one pass ranking formula, although there is nothing in principle to prevent it from being used in the second pass of an automatic relevance feedback search too. The earliest published versions of this basic class of formulae in GURU appeared in [MAAREK89]. The first part of derivation that follows is detailed in [PIETRA95].

Let $P(d_i|Q)$ be the probability that a document d_i is relevant given a query Q. From Bayes' Theorem

$$P(d_i|Q) = \frac{P(d_i)}{P(Q)} \bullet P(Q|d_i)$$

Since the goal is to arrive at a formula suitable for ranking documents in order of highest to lowest $P(d_i|Q)$, not all terms of this relation need be computed to arrive at a suitable ranking formula.

$P(d_i)$ is the *a priori* probability that a document is relevant to a query given no knowledge about the query itself. Without prior knowledge about the corpus or the queries submitted against it, the only reasonable assumption is that all documents are equiprobable. In this case, $P(d_i)$ is a constant and can be omitted without affecting the document rankings. If there is additional knowledge about the corpus and its query history and we can identify the relevant documents for each prior query, $P(d_i)$ need not be constant. GURU, however, does assume that there is no *a priori* information.

$P(Q)$ is a normalization constant required to make $P(d_i|Q)$ a valid probability, i.e.,

$$\sum_{d_i \in D} P(d_i|Q) = 1 .$$

and also assumed constant. It is the probability of any arbitrary query being issued given no knowledge about the corpus. In the absence of other information, any query can be issued against the corpus with equal likelihood, so all queries have the same probability. Since

$$P(Q) = \sum_{d_i \in D} P(d_i) P(Q|d_i)$$

is not a function of D, omitting this term does not affect the relative document rank values. Thus, if we assume that all documents are equiprobable

$$P(d_i|Q) = c \bullet P(Q|d_i)$$

where c is a constant of proportionality. For use as a ranking formula, c can be omitted, leading us to the general result

$$r(d_i) = P(Q|d_i) .$$

The function $r(d_i)$ can be interpreted as the probability of a query matching a document given the document. Another name for this type of function is a similarity function. If we ask a slightly different question, we get a simpler formula useful for document ranking that is both easier to compute and makes no reference to the actual properties of the set of relevant documents for a query. The important property is that the ranking function has a higher value for a greater degree of similarity.

Given $r(d_i)$, how does one compute the probability in a practical and useful manner? Alternately, is there a simple way to compute a similarity function that behaves like $r(d_i)$? Two main techniques have been used in the past to derive such functions: distance measures in some vector space, and likelihood functions.

Although there are many different ways to derive $P(Q|d_i)$, the technique used in the main GURU formulae are based on likelihood functions, but with some empirical adjustments to account for real-world effects. They all assume that the probability of a query and a document matching is some function of the terms that appear in each and only of those terms.

Stopwords are treated as they are in many other search engines. Any stopwords or LA terms containing stopwords do not contribute to a document's ranking score. A single term query that is also a stopword always matches no documents. Stopwords do, however, contribute to determining whether two arbitrary query terms participate in the LA detection process and they also contribute to total document and corpus size.

Suppose there is a single query term and the document is long enough to not have sampling errors when estimating document term probabilities. Let q_j be a term that appears in Q. Since there is only one query term, $P(Q|d_i) = P(q_j|d_i)$ is the probability that the query term appears in the document. Thus

$$P(q_j|d_i) = \frac{n_{q_j \in d_i}}{n_{d_i}},$$

where $n_{q_j \in d_i}$ is the number of times query term q_j appears in document d_i and n_{d_i} is the total number of terms in document d_i, is an unbiased estimate of the probability. Implicit in this calculation is not only that the document is long enough but there are also enough appearances of q_j in d_i to provide a good estimate of the query term's probability over the universe of documents of which d_i is only a sample. If $n_{q_j \in d_i}$ or n_{d_i} are small, sampling and quantization effects complicate the goodness of the estimate.

Since a user specifies a query, it is usually not practical to automatically transform a query and make $n_Q \equiv |Q|$, the number of query terms, larger simply to reflect the user's view of a term's importance. However, it is possible to empirically account for sampling effects caused by short documents and rare terms. Suppose that document d_i is a sample from the corpus D and that Q is known to appear in D. Let $n_{q_j \in D}$ be the number of times the query term q_j appears in the corpus (nonzero by definition) and $n_D \equiv |D|$ be the number of terms in the corpus. One way to account for sampling effects when $n_{q_j \in d_i}$ is small is to use a weighted average to estimate $P(q_j|d_i)$. Thus

$$P'(q_j|d_i) = \frac{\alpha P(q_j|d_i) + \beta P(q_j|D)}{\alpha + \beta},$$

where α and β are free nonnegative constants that sum to 1, is useful when n_{d_i} or $n_{q_j \in d_i}$ are small. This change takes a weighted average of a term's estimated document probability and its corpus probability. Rare terms where there may be significant sampling effects at the document level have some of their variation dampened by their corpus properties.

If we use the fact that the ultimate goal is a ranking function and not a probability, we can define

$$r(q_j, d_i) = \alpha P(q_j | d_i) + \beta P(q_j | D)$$

$$= \alpha \frac{n_{q_j \in d_i}}{n_{d_i}} + \beta \frac{n_{q_j \in D}}{n_D}$$

where α and β no longer need to sum to 1. The actual values of α and β are not important, only their ratio. (Note: this is a departure from the derivation in [PIETRA95].) This formula calculates the ranking value of single document when there is a single query term. Note that when the term does not appear in a document, the document receives a default ranking value of $\beta \bullet n_{q_j \in D} / n_D$. However, since all counts are nonnegative, all documents where the term appears at least once will have a larger ranking value than all documents where the term does not appear. Without loss of generality, we can compute $r(q_j, d_j)$ only for documents where q_j appears and still correctly rank all documents in a corpus with respect to a single term query.

Generalizing $r(q_j, d_j)$ to multiterm queries is straightforward. Let $q_i \in Q$ be a term in the query and n_Q be the number of terms in the query. Then

$$P(Q|d_i) = P(q_1|d_i)P(q_2|d_i, q_1)P(q_3|d_i, q_1, q_2)\cdots P(q_{n_Q}|d_i, q_1, q_2, \cdots, q_{n_Q-1}) .$$

If we assume that the query terms are independent,

$$P(Q|d_i) = \prod_{j=1}^{n_Q} P(q_j|d_i) .$$

Using the same arguments as before, we can transform this probability into a ranking function that empirically accounts for sampling effects. Thus

$$r(Q, d_i) = \prod_{j=1}^{n_Q} (\alpha P(q_j|d_i) + \beta P(q_j|D))$$

is an appropriate ranking function for a document d_i . Note that, as before, documents where no query terms occur receive a default rank determined by the corpus statistics. Also note that, without loss of generality, we can compute the ranking function only for documents where at least one query term q_j appears and still correctly rank all documents in a corpus with respect to a multiterm query. Any document where at least one query term appears will receive a higher ranking value than any document where no query terms appear. (Note:

the derivation in [PIETRA95] stops at this point and proceeds in a different direction. Here, our goal is to derive the actual formulae used in GURU.)

At this point, we still have not defined what we mean by a query term, only that a user can specify one. Nor have we shown what role lexical affinities play in a document's ranking. Define $S \subseteq Q$ as the query terms made up from singleton terms in the query. Also, define $L \subseteq Q$ as the lexical affinities in the query, i.e. pairs of singleton terms. Note that a singleton term can be a single word or multiple words and a word can participate in multiple singleton terms. (We defer the exact definition of how we detect LA terms until later.) From the definition of lexical affinities, $S \cup L = Q$ and $S \cap L = \phi$. We can partition the ranking calculation into two parts based on whether a term is a member of S or L. This assumes that S and L are independent. Define n_S as the number of terms in S and n_L as the number of terms in L. Thus

$$r(Q,d_i) = \prod_{j=1}^{n_S} (\alpha P(q_j|d_i) + \beta P(q_j|D)) \bullet \prod_{j=1}^{n_L} (\alpha P(q_j|d_i) + \beta P(q_j|D)).$$

For ease of interpretation, GURU follows many other probabilistic ranking formulas and computes $\log r(Q|d_i)$ as the actual ranking function. This leads to

$$R(Q,d_i) = \sum_{j=1}^{n_S} \log(\alpha P(q_j|d_i) + \beta P(q_j|D)) + \sum_{j=1}^{n_L} \log(\alpha P(q_j|d_i) + \beta P(q_j|D)).$$

From an implementation point of view, an algorithm that efficiently computes $R(Q,d_i)$ will never see query terms that do not appear in the document. However, since they have a non-zero contribution to a document's ranking value, their absence must be explicitly accounted for. Define s_i as a term in S and l_i as a term in L. Then

$$R(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log(\alpha P(s_j|d_i) + \beta P(s_j|D)) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\alpha P(s_j|d_i) + \beta P(s_j|D))$$
$$+ \sum_{j=1}^{n_{l_j \in d_i}} \log(\alpha P(l_j|d_i) + \beta P(l_j|D)) + \sum_{j=1}^{n_{l_j \notin d_i}} \log(\alpha P(l_j|d_i) + \beta P(l_j|D)).$$

Suppose furthermore that we introduce an empirical weighting constant w to compensate for the effects of lexical affinities versus conceptual affinities. The ranking formula then becomes

$$R'(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log(\alpha P(s_j|d_i) + \beta P(s_j|D)) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\alpha P(s_j|d_i) + \beta P(s_j|D))$$
$$+ w \sum_{j=1}^{n_{l_j \in d_i}} \log(\alpha P(l_j|d_i) + \beta P(l_j|D)) + w \sum_{j=1}^{n_{l_j \notin d_i}} \log(\alpha P(l_j|d_i) + \beta P(l_j|D))$$

Since $P(s_j|d_i) = 0$ and $P(l_j|d_i) = 0$ for terms that don't appear in the document, this simplifies to

$$R'(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log(\alpha P(s_j|d_i) + \beta P(s_j|D)) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta P(s_j|D))$$
$$+ w \sum_{j=1}^{n_{l_j \in d_i}} \log(\alpha P(l_j|d_i) + \beta P(l_j|D)) + w \sum_{j=1}^{n_{l_j \notin d_i}} \log(\beta P(l_j|D))$$

This is the starting place for the GURU ranking formula we describe in this report. There are several variations depending on what assumptions are made about query terms and their contributions to the final rank.

Formula 1

The current implementation of Formula 1 can be derived from the basic starting formula by adding and subtracting terms to keep the formulae equivalent. First, we note that $n_{s_j} \in d_i \cup n_{s_j} \notin d_i \equiv n_{s_j} \in Q$ and $n_{l_j} \in d_i \cup n_{l_j} \notin d_i \equiv n_{l_j} \in Q$, i.e., the set of query terms that appear in the document added to the terms that do not appear in the document is identical to the terms that appear in the query. Thus

$$R'(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log(\alpha P(s_j|d_i) + \beta P(s_j|D)) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta P(s_j|D)) - \sum_{j=1}^{n_{s_j \in Q}} \log(P(s_j|D)) + \sum_{j=1}^{n_{s_j \in Q}} \log(P(s_j|D))$$
$$+ w \sum_{j=1}^{n_{l_j \in d_i}} \log(\alpha P(l_j|d_i) + \beta P(l_j|D)) + w \sum_{j=1}^{n_{l_j \notin d_i}} \log(\beta P(l_j|D)) - w \sum_{j=1}^{n_{l_j \in Q}} \log(P(l_j|D)) + w \sum_{j=1}^{n_{l_j \in Q}} \log(P(l_j|D))$$

Using the aforementioned identities, we can rearrange the terms to arrive at

$$R'(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log(\alpha P(s_j|d_i) + \beta P(s_j|D)) - \log(P(s_j|D)) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta P(s_j|D)) - \log(P(s_j|D)) + \sum_{j=1}^{n_{s_j \in Q}} \log(P(s_j|D))$$
$$+ w \sum_{j=1}^{n_{l_j \in d_i}} \log(\alpha P(l_j|d_i) + \beta P(l_j|D)) - \log(P(l_j|D)) + w \sum_{j=1}^{n_{l_j \notin d_i}} \log(\beta P(l_j|D)) - \log(P(l_j|D)) + w \sum_{j=1}^{n_{l_j \in Q}} \log(P(l_j|D))$$

Combining subtracted log terms as division operations yields

$$R'(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log\left(\frac{\alpha P(s_j|d_i) + \beta P(s_j|D)}{P(s_j|D)}\right) + \sum_{j=1}^{n_{s_j \notin d_i}} \log\left(\frac{\beta P(s_j|D)}{P(s_j|D)}\right) + \sum_{j=1}^{n_{s_j} \in Q} \log(P(s_j|D))$$
$$+ w \sum_{j=1}^{n_{l_j \in d_i}} \log\left(\frac{\alpha P(l_j|d_i) + \beta P(l_j|D)}{P(l_j|D)}\right) + w \sum_{j=1}^{n_{l_j \notin d_i}} \log\left(\frac{\beta P(l_j|D)}{P(l_j|D)}\right) + w \sum_{j=1}^{n_{l_j} \in Q} \log(P(l_j|D))$$

This can be simplified to

$$R'(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log\left(\alpha \frac{P(s_j|d_i)}{P(s_j|D)} + \beta\right) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta) + \sum_{j=1}^{n_{s_j} \in Q} \log(P(s_j|D))$$
$$+ w \sum_{j=1}^{n_{l_j \in d_i}} \log\left(\alpha \frac{P(l_j|d_i)}{P(l_j|D)} + \beta\right) + w \sum_{j=1}^{n_{l_j \notin d_i}} \log(\beta) + w \sum_{j=1}^{n_{l_j} \in Q} \log(P(l_j|D))$$

Since the objective of the derivation is a ranking formula, we note that the two summations over all types of terms in a query are each constant for any given query and corpus. This allows us to economize the calculation without affecting the ranking results if we ignore these terms and create a new ranking function. This leads us to the general form of the ranking function implemented as Formula 1.

$$R''(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log\left(\alpha \frac{P(s_j|d_i)}{P(s_j|D)} + \beta\right) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta) + w \sum_{j=1}^{n_{l_j \in d_i}} \log\left(\alpha \frac{P(l_j|d_i)}{P(l_j|D)} + \beta\right) + w \sum_{j=1}^{n_{l_j \notin d_i}} \log(\beta).$$

It can be shown that the best estimates of the various term probabilities in the preceding equation are given by their occurrence rates. Thus

$$P(s_j|d_i) = \frac{n_{s_j}}{n_{d_i}}$$

$$P(s_j|D) = \frac{n_{s_j,D}}{n_D}$$

$$P(l_j|d_i) = \frac{n_{l_j}}{n_{d_i}}$$

$$P(l_j|D) = \frac{n_{l_j,D}}{n_D}$$

Substituting these into the ranking formula yields

$$R''(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log\left(\alpha \frac{n_{s_j} / n_{d_i}}{n_{s_j,D} / n_D} + \beta \right) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta) + w \sum_{j=1}^{n_{l_j \in d_i}} \log\left(\alpha \frac{n_{l_j} / n_{d_i}}{n_{l_j,D} / n_D} + \beta \right) + w \sum_{j=1}^{n_{l_j \notin d_i}} \log(\beta)$$

which can be rearranged to give

$$R''(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log\left(\alpha \frac{n_{s_j} \bullet n_D}{n_{s_j,D} \bullet n_{d_i}} + \beta \right) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta) + w \sum_{j=1}^{n_{l_j \in d_i}} \log\left(\alpha \frac{n_{l_j} \bullet n_D}{n_{l_j,D} \bullet n_{d_i}} + \beta \right) + w \sum_{j=1}^{n_{l_j \notin d_i}} \log(\beta) .$$

This is the ranking formula that Formula 1 implements, although it rearranges the order of evaluation of the terms in the summations for computational efficiency. In the implementation, $w = 1$.

There are several shortcuts taken to simplify the corpus indexing task. The first is that the total number of LA terms in a document and a corpus is estimated by taking the total number of non-LA terms and multiplying by the empirical constant 3. Both n_{d_i} and n_D are the total number of terms in the document and corpus respectively, so their values are equal to 4 times the number of non-LA terms in the document and corpus respectively. (We have not yet done experiments to determine whether 3 is an appropriate number or not.) However, since the numerical constant of 4 appears in both the numerator and denominator of the ratios, their effect cancels out. Using this knowledge, we can redefine both n_{d_i} and n_D as the size of the document and the corpus in their number of non-LA terms. The GURU algorithm counts each of n_{s_j}, $n_{s_j,D}$, n_{l_j}, and $n_{l_j,D}$ during data accumulation, so their counts are known exactly. The final equation is

$$R''(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log\left(\alpha \frac{n_{s_j} \bullet n_D}{n_{s_j,D} \bullet n_{d_i}} + \beta \right) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta) + w \sum_{j=1}^{n_{l_j \in d_i}} \log\left(\alpha \frac{n_{l_j} \bullet n_D}{n_{s_j,D} \bullet n_{d_i}} + \beta \right) + w \sum_{j=1}^{n_{l_j \notin d_i}} \log(\beta)$$

Formula 5

Formula 5 starts from the same starting place that Formula 1 does. This formula uses no information from LAs. Despite this, Formula 5 is one of the better performing formula on the TREC-5 tests. The net difference caused by an accidental change in the ratio of α and β likely account for the improvement. Here again is that starting place for deriving Formula 1.

$$R'(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log(\alpha P(s_j|d_i) + \beta P(s_j|D)) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta P(s_j|D))$$
$$+ w \sum_{j=1}^{n_{l_j \in d_i}} \log(\alpha P(l_j|d_i) + \beta P(l_j|D)) + w \sum_{j=1}^{n_{l_j \notin d_i}} \log(\beta P(l_j|D))$$

First, since the formula ignores all LA terms, we can delete them from the basic equation, resulting in

$$R'(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log(\alpha P(s_j|d_i) + \beta P(s_j|D)) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta P(s_j|D)) .$$

Just as in Formula 1, we add and subtract terms to maintain the equality to arrive at

$$R'(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log(\alpha P(s_j|d_i) + \beta P(s_j|D)) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta P(s_j|D))$$
$$- \sum_{j=1}^{n_{s_j \in D}} \log(\beta P(s_j|D)) + \sum_{j=1}^{n_{s_j \in D}} \log(\beta P(s_j|D))$$

The last term is constant for any given query and so can be ignored for a ranking formula. We can split the subtracted term and obtain

$$R''(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log(\alpha P(s_j|d_i) + \beta P(s_j|D)) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta P(s_j|D))$$
$$- \sum_{j=1}^{n_{s_j \in d_i}} \log(\beta P(s_j|D)) - \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta P(s_j|D))$$

The terms summed over $n_{s_j \notin d_i}$ cancel and the remaining term can be moved inside the summation, yielding

$$R''(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} (\log(\alpha P(s_j|d_i) + \beta P(s_j|D)) - \log(\beta P(s_j|D)) .$$

At this point Formula 5 scales by $P(s_j|D)$. This causes problems with assumptions about independence. It basically assumes that $P(s_j|D)$ is the same for all s_j, but other parts of the formula later do not. This problem is not corrected until Formula 57. However, this is how Formula 5 does its calculation. The result of scaling by $P(s_j|D)$ is

$$R'''(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \left(\log\left(\frac{\alpha P(s_j|d_i) + \beta P(s_j|D)}{P(s_j|D)} \right) - \log\beta \right)$$

$$= \sum_{j=1}^{n_{s_j \in d_i}} \left(\log\left(\alpha \frac{P(s_j|d_i)}{P(s_j|D)} + \beta \right) - \log\beta \right)$$

$$= \sum_{j=1}^{n_{s_j \in d_i}} \left(\log\left(\alpha \frac{n_{s_j} / n_{d_i}}{n_{s_j,D} / n_D} + \beta \right) - \log\beta \right)$$

$$= \sum_{j=1}^{n_{s_j \in d_i}} \left(\log\left(\alpha \frac{n_{s_j} \bullet n_D}{n_{s_j,D} \bullet n_{d_i}} + \beta \right) - \log\beta \right)$$

where $n_{s_j,D}$ is the number of occurrences of a query term in the corpus and n_D is the total number of all terms in the corpus. Because Formula 5 uses part of the code for Formula 1 in performing its partial sum calculation, n_{d_i} is actually calculated as the total number of non-LA terms plus LA terms in the document. This is exactly 4 times the number of non-LA terms. We keep n_D as only the number of non-LA terms. Both $n_{s_j,D}$ and n_{s_j} are counted from the actual occurrences.

If we define n_{d_i} to be only the number of non-LA terms, the equation transforms to

$$R'''(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \left(\log\left(\frac{\alpha}{4} \bullet \frac{n_{s_j} \bullet n_D}{n_{s_j,D} \bullet n_{d_i}} + \beta \right) - \log\beta \right).$$

Formula 55

This formula basically is Formula 5 expanded for LA terms. The formula is

$$R'''(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \left(\log\left(\frac{\alpha}{4} \bullet \frac{n_{s_j} \bullet n_D}{n_{s_j,D} \bullet n_{d_i}} + \beta \right) - \log\beta \right) + w \sum_{j=1}^{n_{l_j \in d_i}} \left(\log\left(\frac{3\alpha}{4} \bullet \frac{n_{l_j} \bullet n_D}{n_{s_j,D} \bullet n_{d_i}} + \beta \right) - \log\beta \right)$$

where the counts for total document and corpus size are as in Formula 5. Note that just as in Formula 5, both n_{d_i} and n_D are only the counts of the non-LA terms. The multiplying factors of 3 and 4 account for estimates of the total number of terms. Surprisingly, this is the best performing formula on the TREC-5 tests when w is

set to a value of approximately 0.1. The additional factor of 3 used for LA terms is the assumed total number of LA terms in the corpus as a function of the number of non-LA terms.

Formula 56

We modified Formula 55 to the form:

$$R'''(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log\left(\frac{\alpha}{4} \bullet \frac{n_{s_j} \bullet n_D}{n_{s_j,D} \bullet n_{d_i}} + \beta\right) + w \sum_{j=1}^{n_{l_j \in d_i}} \log\left(\frac{3\alpha}{4} \bullet \frac{n_{l_j} \bullet n_D}{n_{s_j,D} \bullet n_{d_i}} + \beta\right).$$

Although this formula performed worse than Formula 55 and Formula 5 on the TREC-5 test, the difference was much smaller than between Formula 55 and Formula 1. Again, a value of w near 0.1 performed best.

Formula 57

Finally, we examined the Formula 1 derivation and how it corrected for terms from the query appearing in the corpus and not in the document. This resulted in a change in Formula 55's correction term to be more in line with the theoretical version of Formula 1. This yields:

$$R'''(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log\left(\frac{\alpha}{4} \bullet \frac{n_{s_j} \bullet n_D}{n_{s_j,D} \bullet n_{d_i}} + \beta\right) + \sum_{j=1}^{n_{s_j \notin d_i}} \log\beta + w \sum_{j=1}^{n_{l_j \in d_i}} \log\left(\frac{3\alpha}{4} \bullet \frac{n_{l_j} \bullet n_D}{n_{s_j,D} \bullet n_{d_i}} + \beta\right) + w \sum_{j=1}^{n_{l_j \notin d_i}} \log\beta .$$

This formula performed worse than Formula 55 on the TREC-5 test, but better than Formula 56. Compare this formula with Formula 1. Note that the only differences between the two are the ¼ for non-LA query terms, the ¾ for LA query terms, and $w = 0.1$ while Formula 1 has 1 for each of these values. This change is enough to substantially improve performance. Using $w = 0.1$ instead of 1.0 does not make as large a difference as changing the ratio of α and β.

Rank Normalization

There are two goals to rank normalization. The first is to present ranking scores to an untrained user that gives them a good idea of how well documents matched their query. The second is to be able to compute ranking scores that allow comparison of values across queries in the same corpus.

Part way through the derivation of Formula 1, we arrived at the ranking formula

$$R'(Q,d_i) = \sum_{j=1}^{n_{s_j \in d_i}} \log\left(\alpha \frac{P(s_j|d_i)}{P(s_j|D)} + \beta\right) + \sum_{j=1}^{n_{s_j \notin d_i}} \log(\beta) + \sum_{j=1}^{n_{s_j \in Q}} \log(P(s_j|D))$$
$$+ w \sum_{j=1}^{n_{l_j \in d_i}} \log\left(\alpha \frac{P(l_j|d_i)}{P(l_j|D)} + \beta\right) + w \sum_{j=1}^{n_{l_j \notin d_i}} \log(\beta) + w \sum_{j=1}^{n_{l_j \in Q}} \log(P(l_j|D))$$

If α and β sum to 1 and $w = 1$, $R'(Q,d_i)$ is the log of a probability. This means that by including the two summations based only on the terms in the query and the corpus, using restricted values for the constants, and within the assumptions made earlier, we can trivially compute a theoretical maximum value of 0 when a query exactly matches a document's and a corpus' statistics. We can rewrite the equation in terms of Formula 1 as

$$R'(Q,d_i) = R''(Q,d_i) + \sum_{j=1}^{n_{s_j \in Q}} \log(P(s_j|D)) + w \sum_{j=1}^{n_{l_j \in Q}} \log(P(l_j|D))$$

so that it explicitly shows the normalization calculation in terms of the ranking function. When $w = 1$, $R'(Q,d_i)$ is less than or equal to 0 and its range is independent of query or corpus. The last two terms depend only on the query and the corpus and can be computed independently of the document rank computation.

One way to normalize to a range that people using search engines have come to expect is to add a constant to $R'(Q,d_i)$ such that it falls into a more usual range of 100 being a perfect match and decreasing with poorer matches. Adding a constant to $R'(Q,d_i)$ that shifts its numerical range does the job. However, we have found that, through experiment, $w = 0.1$ is a better value. This affects the normalization calculation in that the $R'(Q,d_i)$ values are less negative and no longer a valid probability, although still limited to be less than or equal to 0. If we assume that the ratio of LA to non-LA terms tends toward a constant over a sufficiently large body of text, we can still use a constant shift of the $R'(Q,d_i)$ value. The constant will be less than for when $w = 1$, but we need to empirically determine the constant.

Summary

All of the GURU formulae examined in this report are based on the same set of assumptions and the derivation that can be made because of these assumptions. They are:

1. the similarity of the document to the query is the best estimate of a document's relevance

2. the *a priori* probability of any document being relevant to all queries is the same for all documents

3. a weighted average of a term's document and corpus probability is an appropriate way to correct for sampling effects

4. query terms are independent within a query

5. non-LA terms are independent of LA terms in a document

6. the total number of all LA terms in a document and in a corpus are 3 times the number of non-LA terms in each

Several GURU formulas not mentioned in this report examine one or more of these assumptions and try to generalize based on not making these assumptions. Discussing these formulae are beyond the scope of this report but are the subject of continuing research.

Experimental Technique

Recall and precision are the most popular measures used for measuring the effectiveness of information retrieval systems. Recall is defined to be the proportion of all relevant documents retrieved by a query. Precision is the proportion of retrieved documents that are relevant. Systems that always return only a few documents may have high precision, but the recall will be low. As more and more documents are returned, the recall increases, but the precision decreases.

Performing real-world evaluations of a search engine require a specific test corpus, a set of queries against that corpus, and, for each query, a known set of relevant documents in the corpus. The first two are relatively easy to come by, but the latter is much harder. Only one organization has attempted to do a regularly updated evaluation system and procedure, the National Institute of Standards and Technology, a part of the US Department of Commerce. It sponsors the yearly Text REtrieval Conference (TREC) and associated infrastructure to maintain a test corpus, updated queries, and a body of people able to perform relevance judgments on documents.

TREC calculates and shows idealized precision-recall figures in their evaluations [HARMAN96]. In reality, a precision-recall curve is not smooth and monotonically decreasing. Each time a new relevant document is added as one progresses from best to worst ranked, the precision increases. The rest of the time, the precision decreases. To get around this, NIST uses something they call interpolation. The program scans the ranked list from worst to best rank. At each rank, it takes the maximum of the actual precision and the precision of the most recent prior relevant document. It then uses these values to assign precision values to specific values of recall used in the official reports. NIST calculates average non-interpolated precision by averaging all fifty queries' precision at the eleven standard recall points.

TREC Performance Benchmarks

[HARMAN97] details the work and results of several conferences. Automatic adhoc short queries had most relevance for us. This is most like GURU's target audience use of search engines. All of the adhoc sections, however, start from the same TREC topics. They vary in what topic sections they can use and how much manual formulation takes place. In the case of automatic short queries, no manual intervention is allowed, and the query generator can use only the topic description. This is the TREC-5 a posteriori test mentioned earlier.

A TREC topic has a number, a title, a description, and a narrative. The number is only for identification purposes. The title is a short summary of the topic. The description expands somewhat on the title, while the narrative explains the details of what makes a document relevant and not relevant. TREC participants in Automatic Adhoc Short Queries are allowed to use only the description portion of the topic to create their queries. Figure 6 shows a typical topic with the description emphasized.

```
<top>
<num> Number: 254
<title> Topic: Non-invasive procedures for persons with heart ailments
<desc> Description:
The document will discuss all those cases in which medications and
procedures were used instead of or prior to heart surgery.
<narr> Narrative:
A relevant document will report/discuss those cases in which
persons diagnosed with heart ailments were treated with
medications and/or techniques such as angioplasty, stents, lasers,
arthrectomy (roto router) etc., in place of surgery. Also advantages
of non-invasive procedures over surgery and comparative studies
which show any disparity in longevity when either procedure is used.
</top>
```

Figure 6 – A sample TREC-5 topic

Each year, NIST chooses a test collection of documents from its available pool. It also chooses 50 topics having relevant documents in the test collection. Each participant receives the collection and topics and runs their search engine on them according to the rules of each participation category. For adhoc query participation, each entry submits the results of the top 1000 documents from each topic.

NIST gathers the results from all the entries and, for each topic, uses the union of the set of the top 100 documents retrieved by each entry as the basis for relevance judgments. Someone goes through all the documents returned and assesses each document's relevance. After making the relevance judgments, NIST takes the results from each entry's queries and generates precision-recall evaluations. These are the results that are shown in the TREC conference proceedings. NIST explicitly states that the entries are presumed good enough that the union of all entries' top 100 document lists will always capture all relevant documents in a collection [HARMAN96].

Our evaluations of ranking formula use the same technique as would be done if we were participating in TREC. The main difference is that no dynamic modification of the relevant document list takes place. We use only the documents judged by NIST and their relevance determination to calculate the precision-recall curves. If, for

some reason, a new formula returns a document that should be relevant, but was never returned in the top 100 documents by any other search engine, it does not improve our evaluation of the formula's results.

General Observations on TREC

Participants in TREC know that there are several biases and assumptions in TREC that must be taken into account when designing new formula. Although these biases and assumptions are well known, they still need to be accounted for when evaluation the results from a TREC conference. Complicating the problem is that when performing *a posteriori* evaluations, such as done for this report, relevant documents from the tested search engine or formula do not influence the set of judged documents as they would if they were formally in the conference.

[ROSE97], [BROWN96] and others have shown that most queries submitted to general purpose search engines are very short, on the order of one or two words. The TREC topic descriptions tend to be full English sentences, sometimes as many as three or four sentences. This biases overall performance toward search engines that do some amount of natural language processing or other techniques that otherwise make attempts to analyze a query's intent. Sophisticated query analysis, as some TREC entrants do, is rarely applicable or productive with very short queries. There simply is not enough information present.

The wording of the TREC topics are what one would say to a human research assistant when giving them a search assignment. Many of the TREC topics are asking for inferences about documents instead of documents themselves (see Figure 7). Others require metaknowledge about the search topic to build a successful query or to evaluate a document's relevance (see Figure 8 and Figure 9). At least one topic (see Figure 10) requires an arithmetic operation on the result set to arrive at an answer. (Note: the figures show a topic's narrative and not the description, which is much shorter.) Human experts in the field have no trouble translating and inferring. It is a much larger task to ask an information retrieval system without "common knowledge" and inferencing capabilities to do the same job.

TREC uses 50 queries each year as the basis for its evaluations. This is a small number. Common wisdom of TREC participants is that this number as too small to make any definitive statements of search engine performance outside of the TREC context. However, there is no other established system at this point.

```
Illegal entry into sensitive computer networks is a
serious and potentially menacing problem. Both 'hackers'
and foreign agents have been known to acquire unauthorized
entry into various networks. Items relative this
subject would include but not be limited to instances of
illegally entering networks containing information of
a sensitive nature to specific countries, such as defense or
technology information, international banking, etc.

Items of a personal nature (e.g. credit card fraud, changing of
college test scores) should not be considered relevant.
```

Figure 7 – Topic 258: Inferencing about what constitutes "personal"

```
Documents must provide new theories about the assassination
of President Kennedy that are contrary to the findings of
the Warren Commission. This does not mean the new theory
must indicate more than one assassin but merely dispute
any part of the Warren commission findings.
```

Figure 8 – Topic 259: Metaknowledge required about the findings of the Warren Commission

```
To be relevant, a document must describe an effort
being made (other than routine border patrols) in any
country of the world to prevent the illegal penetration
of aliens across borders.
```

Figure 9 – Topic 252: Metaknowledge that "aliens" is a US government term not usually used elsewhere

```
We are looking for a count of operable submarines in
any country that currently has a navy with submarines.
To be relevant a document should give a specific number
of submarines, but not necessarily its entire fleet of
submarines (although, that is our ultimate goal).
A report of a French submarine suffering a mishap
in the North sea would not be relevant. However, a report
of a new submarine being built in Shanghai that contains
other valuable information, such as "this is the third
reported unit constructed at this base" would be relevant.
Any information that would be considered useful as an
intelligence tool in determining a country's submarine
order of battle would be relevant.
```

Figure 10 – Topic 285: Arithmetic operation on retrieved information

Results

When we started out trying to improve on the results from TREC-5, one of the first questions asked was how we were different from the other TREC participants. It turns out that even in the specialized part of TREC Automatic Adhoc where the descriptions alone form the basis for the queries, GURU differs significantly from most of the entries. Out of seventeen organizations entered in this section, only Dublin City University [KELLEDY97] explicitly states that they do not use an automatic relevance feedback procedure. Of the remaining fifteen besides GURU, all but one, UC Berkeley, use some form of automatic relevance feedback for query augmentation. Berkeley had used automatic relevance feedback in TREC-4 but worsened their results with it. For TREC-5, they felt it was safer to not use the procedure until they understood it better [GEY97].

Several organizations document their results before and after using their automatic relevance feedback procedures. Figure 11 shows before and after results of Average Noninterpolated Precision values for those participating in Automatic Adhoc with description only. Some of these values are derived from tests using TREC-4 queries and collections and their judged relevant documents. These will not be the same as their official TREC-5 results. Of the ones listed, only the MDS system from RMIT ranked worse than Formula 5 did in the official TREC-5 results shown in Figure 4. Included in the table is the result for the best GURU formula we have today using the TREC-5 *a posteriori* test. The results suggest that if we implemented a good automatic relevance feedback procedure or further improve GURU's term expansion capability, or combine the two, we could move further ahead in relative ranking among TREC participants. GURU would have ranked very highly among these search engines if they were not allowed to use automatic relevance feedback.

Entry [reference]	Base Precision	Final Precision	% Gain
V-TWIN [ROSE97]	0.210	0.267	27%
Guru Formula 55	0.159	????	???
OKAPI [BEAULIEU97]	0.152	0.185	21%
SMART [BUCKLEY97]	0.148	0.211	43%
INQUERY [ALLAN97]	0.144	0.200	39%
PIRCS [KWOK97]	0.140	0.181	29%
MDS [KASZKIEL97]	0.103	0.121	17%

Figure 11 – TREC-5 Improvements Because of Automatic Relevance Feedback

A further difference that distinguishes GURU from other participants is the amount of query processing it does. The base search engine does two forms of query processing: stopword detection, and algorithmic morphology

on non-stopwords. All other query processing takes place outside the search engine in a hand tweaked Perl script. All it does is eliminate boiler-plate words from a TREC topic description and punctuation that confuses GURU. Contrast that with INQUERY [ALLAN97], which does two main stages of query processing followed by one stage of automatic relevance feedback analysis. In stage 1, INQUERY does several different types of syntactic and semantic analysis of the query to identify parts of speech, deduce subject and object, and infer the most important subject and object of a TREC topic. Stage 2 clusters terms and tries to identify and rank the query terms in order of importance.

Many of the TREC topics contain both positive and negative attributes. For example, Topic 258 in Figure 7 contains a list of desirable concepts but also lists concepts specifically to be excluded. There are several such topics in TREC and, in general, many situations where such concepts need to be specified. GURU contains code and data structures suitable for several implementations of negating or decreasing the rank of documents containing unwanted terms, but the query parser itself does not support such operations. On several such TREC topics, the exclusion specification forms a concept of similar importance to the inclusion concept.

OKAPI

During the preparation for the TREC-5 submission we built a version of the OKAPI BM25 ranking formula [BEAULIEU97] and used it for some tests. The main difficulty we had during implementation was how to deal with the assumed automatic relevance feedback. At first, this led to confusion about where the relevance judgments were coming from since we did not understand that there was an automatic relevance feedback procedure taking place. Later, after research showed that OKAPI, along with most other TREC-5 participants, used the first pass to obtain relevance estimates, we understood how these factors worked. Still, there are some arbitrary, “collection-specific” constants whose values were take directly from their TREC-4 paper [ROBERTSON96].

Nonetheless, since GURU is not a search engine that automates relevance feedback, we could only implement a formula that does ranking with estimates for the relevance related values. We constructed three different formulae based on OKAPI. Formula 50 works only with singleton terms. Formula 51 uses an *ad hoc* addition to the formula to use LA terms. Formula 52 used weighting on the LA contribution to a document’s score.

Figure 12 shows the results on TREC-4 queries and collections using each of the OKAPI formula variations. Note: we could only process 48 queries for the evaluation. A query construction error caused a memory problem

during the search and crashed two of the queries. We do not know how much this biases our results. However, for the purposes of comparing OKAPI formulae to themselves, there is no problem.

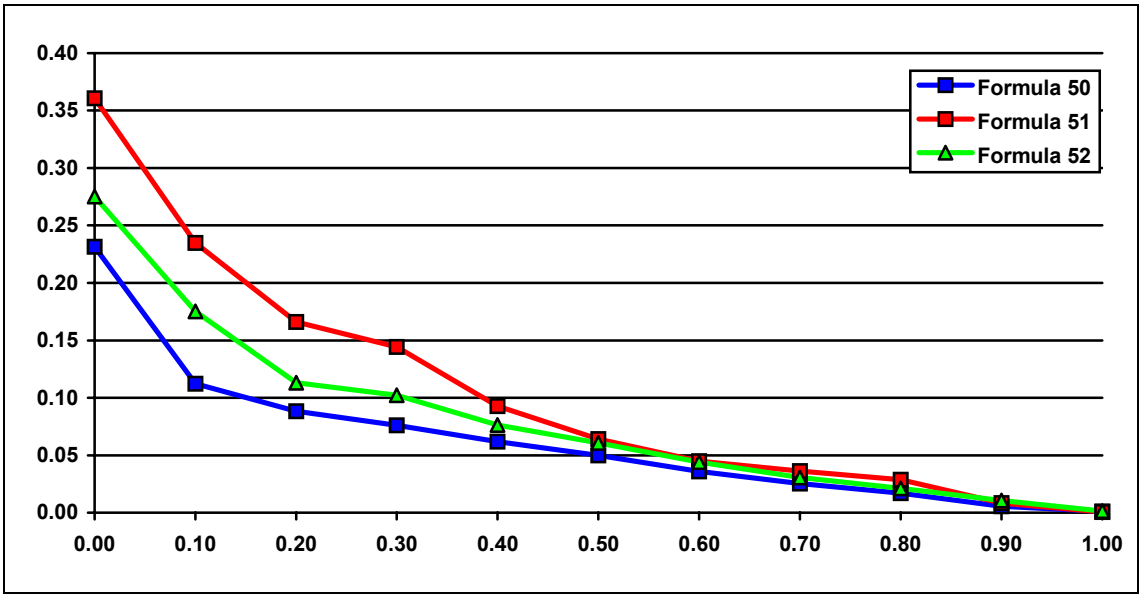

Figure 12 – Precision-Recall of Single Pass OKAPI

Overall performance is substantially worse than any of the other GURU formula we were comparing to. Because GURU at this time has no automatic relevance feedback capability, we were unable to experiment further, but the general conclusion is that our GURU formulae based on Formula 1 and 5 performed better. We also understood the foundations of our formulae better. GURU formulae also have far fewer arbitrary constants. Note, however, that OKAPI performed better with our arbitrary addition of LA terms into their formula than without. This supports the assertion that LA terms play a significant role in improving precision, even with a simplistic implementation.

INQUERY

INQUERY by the University of Massachusetts at Amherst is traditionally one of the better search engines participating in TREC. It is the only search engine whose TREC-5 paper [ALLAN97] shows the 11-point precision recall curve of the initial pass of their search engine before they apply their automatic relevance feedback procedure. Figure 13 compares Formula 55's best results with INQUERY's official TREC-5 results.

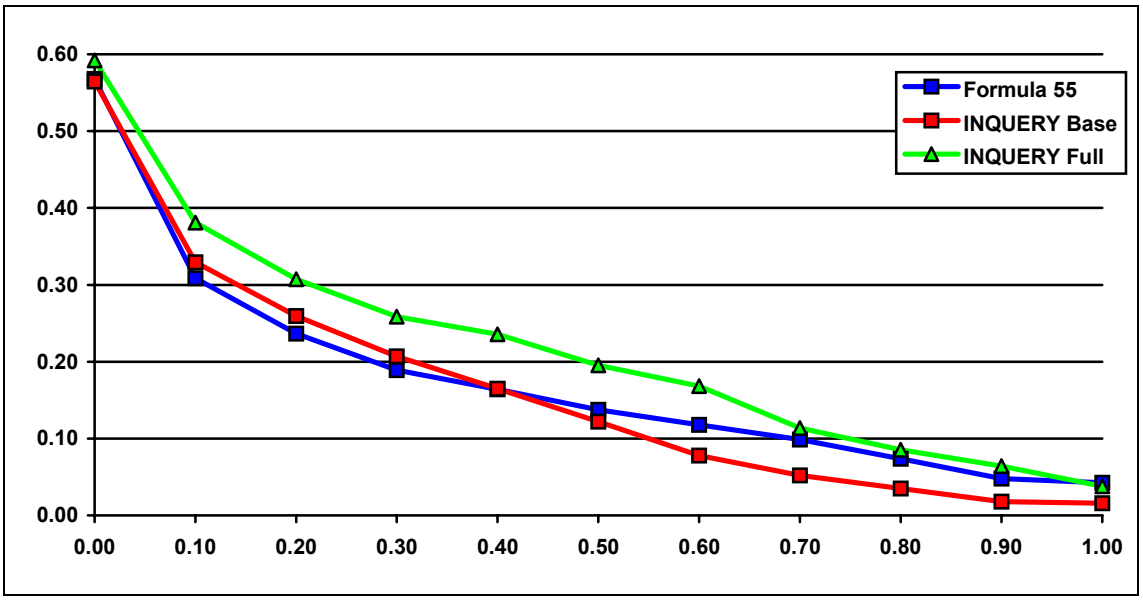

Figure 13 – Precision-Recall of INQUERY at TREC5 and Formula 55

Formula 55 and the first INQUERY pass are different in two important respects. The first is that initially, INQUERY's ranking formula shows less decline as recall increases. The second is that at approximately 0.40 recall, GURU becomes better and remains better at higher recall. At this time, we have no explanation for this, but we believe that LA terms are making the difference.

The full INQUERY result shows the effect of sophisticated query processing and automatic relevance feedback. Although recall at 0 precision improves only slightly, the remainder of the curve is substantially above that of the base INQUERY processing. Detailed examination of [ALLAN97] shows that most of the improvement is because of advanced query processing and a small amount because of automatic relevance feedback. Since TREC topics differ substantially from what we expect GURU users to submit, it is not clear based on these results whether the INQUERY automatic relevance feedback algorithm would help GURU.

GURU

Since GURU is our series of ranking formula, we spent the most time and effort analyzing its performance and understanding the effects of various parameters. This report shows only the Formula 1 and 5 family as these are the most developed and consistent in their performance. Also, as shown earlier, we have a complete understanding of their mathematical and statistical basis and their weaknesses. Note that we used the identical transformed TREC topics to queries for testing each of the GURU formulae.

FORMULA 1

Formula 1 is what is in the base Search Manager and NetQuestion products. We were able to perform some experiments with one of the parameters in the formula, the size of the window used to detect LA terms. Singleton terms must occur within the LA window in the query to be marked as an LA term. Only these terms are detected during search and accumulated into a document's score. Each singleton term must occur in the same sentence of a document even if they are otherwise within the window distance of each other in the text. Figure 14 shows the precision-recall of this formula on the TREC-5 *a posteriori* test with several reasonable values for the LA window.

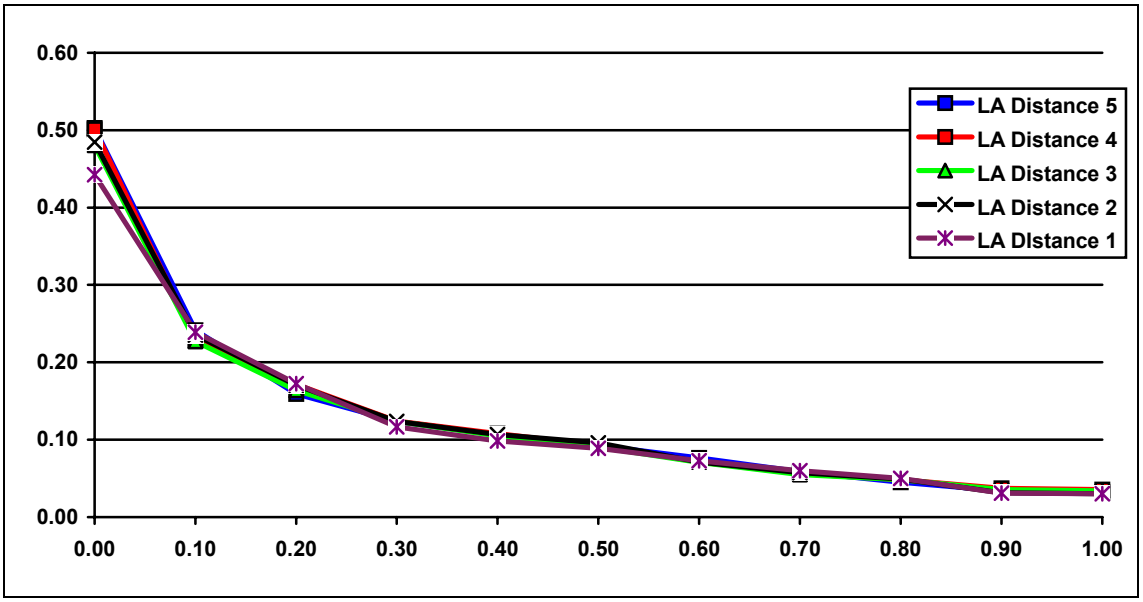

Figure 14 – Precision-Recall of Formula 1

The different formulae perform nearly identically except at 0 recall. At that point, larger LA distances performed noticeably better. Although not shown, using an LA distance of 50 performed very slightly better than the default value of 5 we have been using for years, but it substantially increased the processing time. Using such a large LA window value basically detects all singleton term co-occurrences within the same sentence since LA terms never cross sentence boundaries.

FORMULA 5

We were quite surprised at how well Formula 5 performed in TREC-5 considering how little theoretical work went into the formula before we submitted it. Only during the writing of this paper did we realize that Formula 5

was closely related to Formula 1 when we excluded LA terms from the ranking calculation. The only practical difference besides LA terms is a fortuitous change in the ratio of α and β caused by recycling code from Formula 1. Figure 15 shows its precision-recall curve on the TREC-5 *a posteriori* test. There is only one curve because aside from α and β, there are no adjustable parameters in the formula.

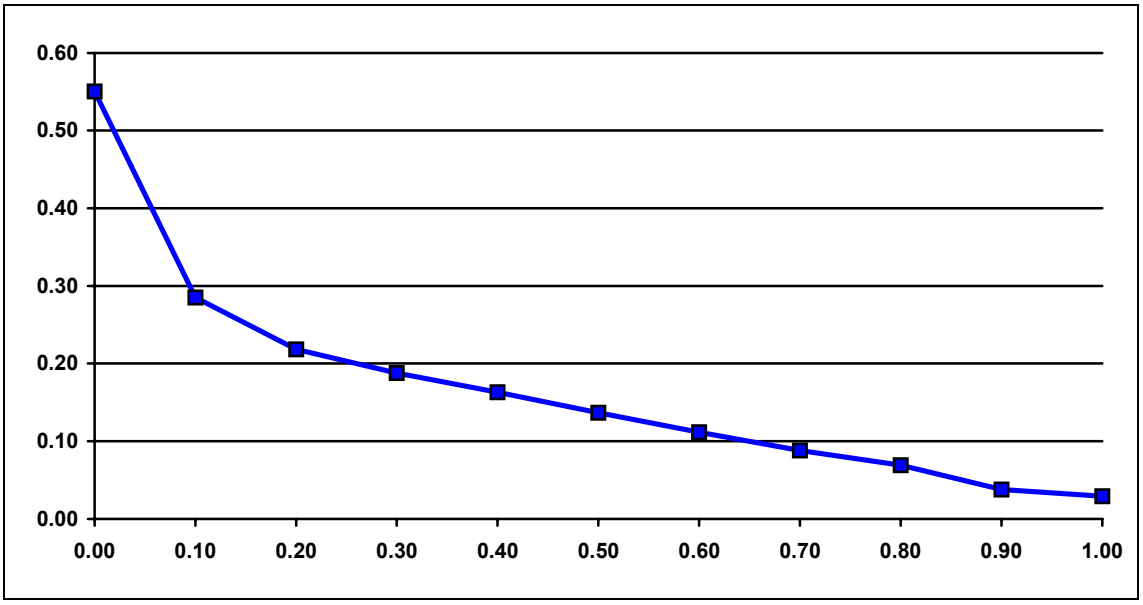

Figure 15 – Precision-Recall of Formula 5

The precision at 0 recall and average non-interpolated precision for this test are sufficient to place it slightly above median if these were its official TREC-5 results. This is slightly better than the actual submitted results. One of the reasons for the better performance in the *a posteriori* test is that we fixed some bugs in the document parser and improved index quality. We also made slight modifications to the automatic TREC topic to GURU query generator and removed some noise words.

GURU FORMULAE 55-57

Although we have shown that by making only moderately controversial assumptions of independence, we can derive Formula 55 from the definition of the GURU formula, this was not the case when we first created formula 55. At the time, it was a completely unjustified but symmetric extension of Formula 5, which in turn was only shakily understood. Since then, we have shown that it is possible to derive Formula 5 and Formula 55 and show their relationship to Formula 1.

Aside from α and β, Formula 55, 56, and 57 have two free parameters we can vary, LA window size, and LA weight, the arbitrary w factor shown in the derivation. Figure 16 shows the effect on precision at 0 recall by varying both LA distance and LA weight on formula 55. Note that the LA weight and distance scales are not linear.

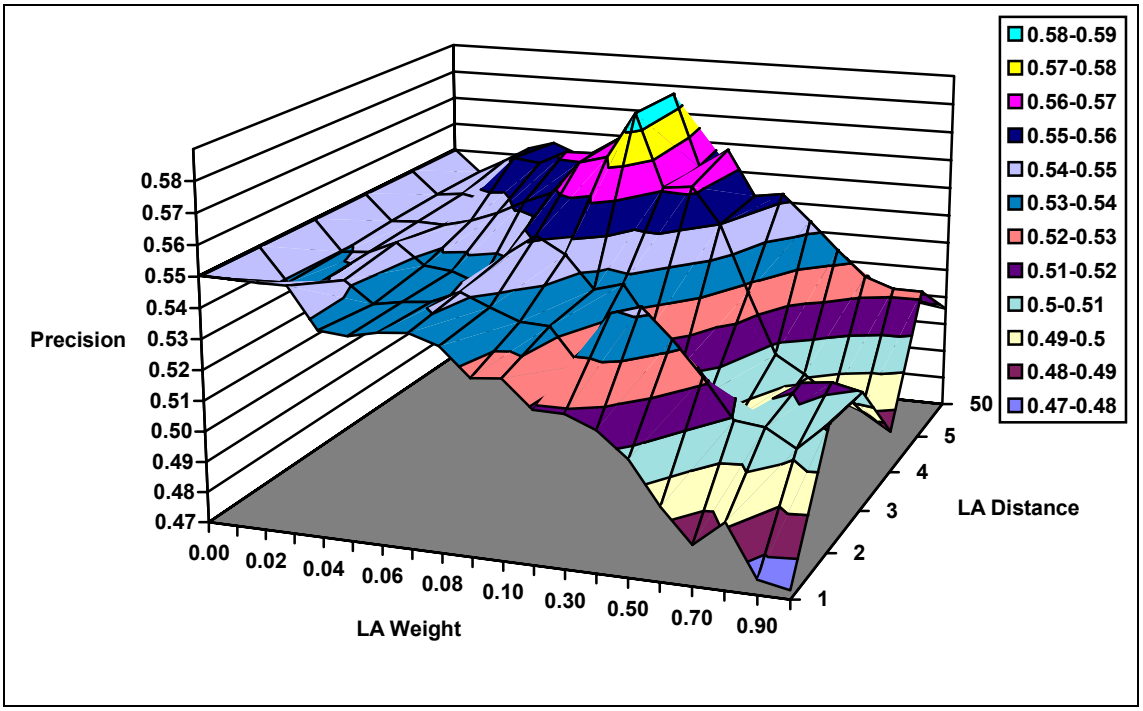

Figure 16 – Precision at 0.0 Recall with Varying LA Weight and LA Distance of Formula 55

As can be seen from the chart, the best precision occurs at higher LA distances and an LA weight of approximately 0.1. We always had some confidence in the LA distance value based on work done in [MAAREK89] and [MAARKEK91], but LA weight was a new concept we had no experience with. However, the results fit with the empirical observation that LA terms contributed too much to a document's score and that we needed to decrease their contribution in some systematic manner. Precision at 0 recall is an extrapolated number and is subject to noise in its estimates. This accounts for the bumpiness of the surface in Figure 15. Average non-interpolated precision is an average and much less subject to noise. Figure 17 shows a plot of average non-interpolated precision over the same range of LA distance and LA weight values for Formula 55. Again as for precision at 0 recall, the scale for LA weight and LA distance is nonlinear.

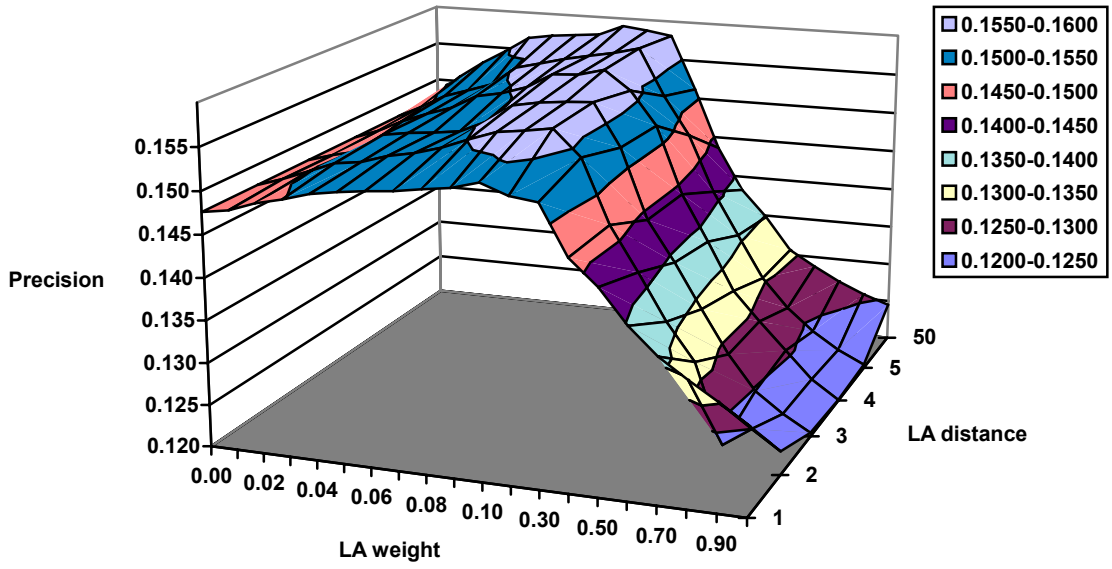

Figure 17 – Average Precision with Varying LA Weight and LA Distance of Formula 55

Figure 18 shows the same plot omitting the intermediate data between LA weight of 0.0 and 0.1 to make the scale linear. This plot emphasizes the sharpness of the ridge on the surface plot and average precision's sensitivity to small changes in the LA weight. Note that an LA weight of 0 corresponds to ignoring LA terms completely and makes Formula 55 identical to Formula 5.

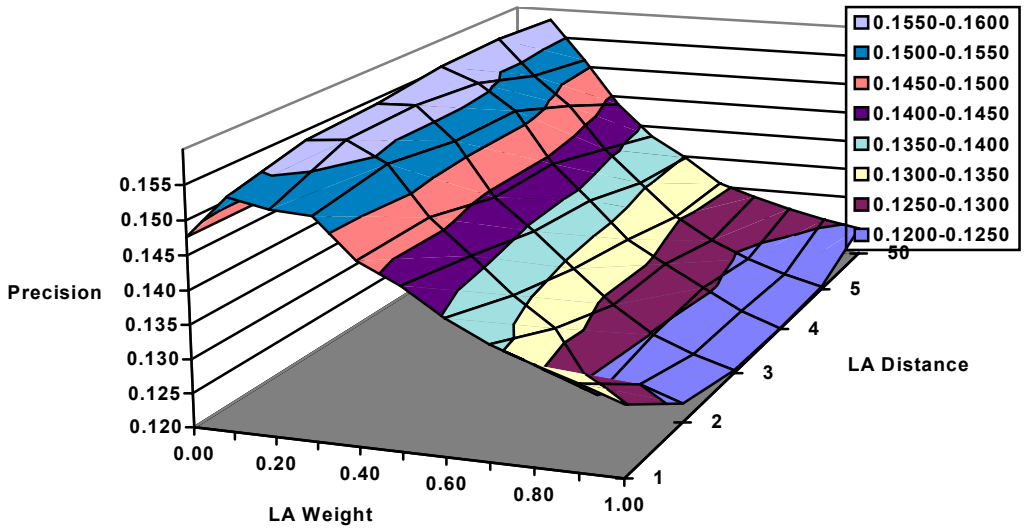

Figure 18 – Average Precision with Varying LA Weight and LA Distance of Formula 55 (linear scale)

One surprising finding from doing this study on Formula 55 is that once the LA weight increases beyond about 0.3, LA terms make the ranking formula perform worse than if it had ignored LA terms altogether. We have not determined the reason for the worsening performance, but one possible factor is the inclusion of low information-bearing singleton pairs into the ranking calculation when they should be excluded.

Both LA weight and LA distance are constants determined by the language and not by corpus characteristics. LA distance measures the word distance between conceptually related terms. LA weight compensates for the relative ratios of LA terms to singleton terms in a language and their effects in the ranking calculation. Thus, finding good values for both using the TREC collection should provide very good values for all remaining English language documents. The only variations should be caused by changes in the writing style of documents within a corpus.

Figure 19 shows the more traditional precision-recall curve of Formula 55 using an LA weight of 0.1 and varying LA distance. Except for small changes in precision at 0 recall, the formulae are extremely close to one another. Note that both an LA distance of 4 and 5 compute curves that overlap enough to make them indistinguishable in this plot.

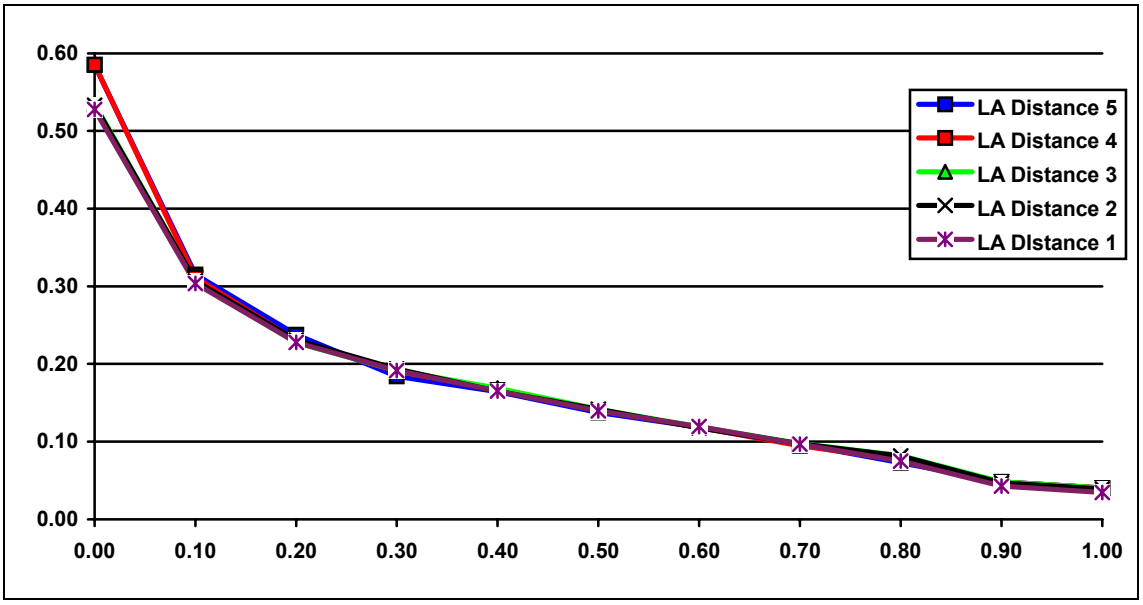

Figure 19 – Precision-Recall of Formula 55 with LA Weight 0.1

We performed some experiments with varying LA weight and LA distance for formulae 56 and 57, but the values where they performed best are very close to that for Formula 55. Thus, we used the same values for these

formulae in later experiments. Figure 20 and Figure 21 show the precision recall curves for these two formulae with two values of LA distance. More striking than their differences are their similarities. The plots clearly show that the family of formula evaluated are affected by the same parameters in the same ways.

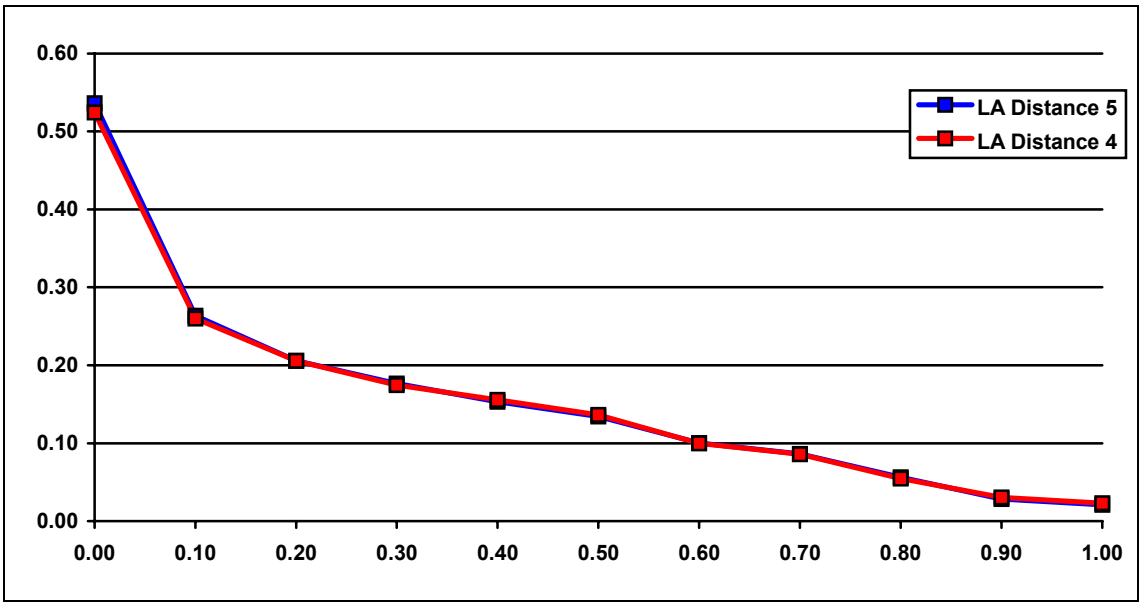

Figure 20 – Precision-Recall of Formula 56 with LA Weight 0.1

Note that Formula 56 performs worse than Formula 55 and 57 while Formula 57 is very close to that of Formula 55. In both cases, the differences in performance with changes in LA distance were tiny except at 0 recall. This value of recall is extrapolated and the least reliable of the measured values.

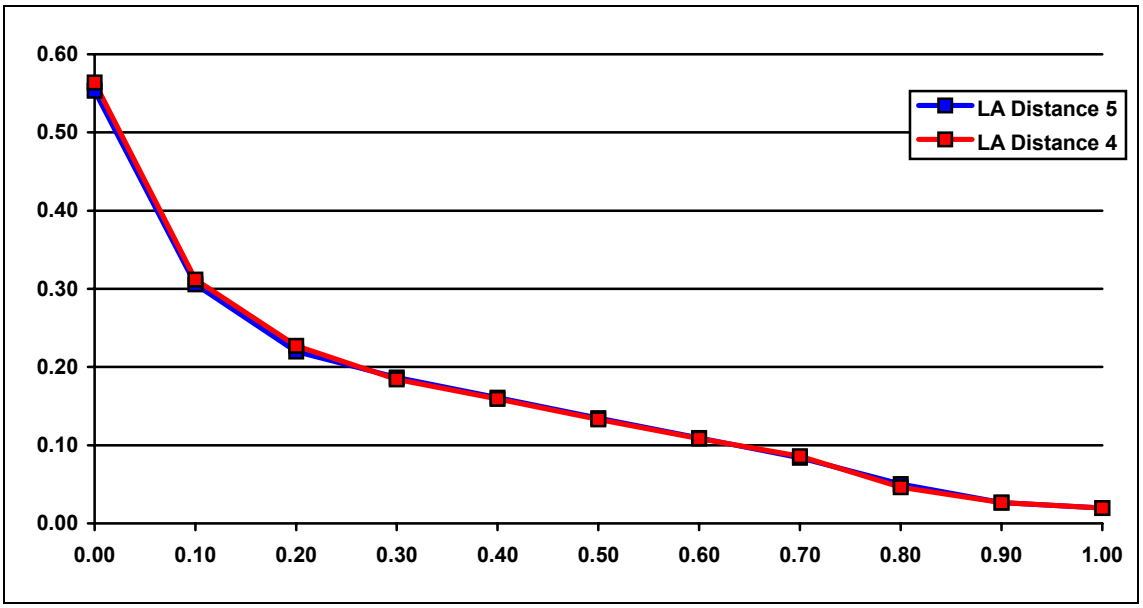

Figure 21 – Precision-Recall of Formula 57 with LA Weight 0.1

VARYING ALPHA AND BETA

After fixing values of LA weight and LA distance, we needed to study the effects of varying α and β. We chose to use Formula 57 because of its mathematical structure and because it performed close to best. LA distance was fixed at 5 while LA weight was fixed at 0.1. Figure 22 shows precision recall results for several values of α. We preserved the original formulation where $\alpha + \beta = 1$. The differences were not large, but a value for α of 0.5 showed a slight improvement over other values tried. This happens to be the default value for α that we have used for years. Note that the built-in values of ¼ for singleton terms and ¾ for LA terms as shown in the derivation of Formula 57 remained in effect for this study. Also, note that the effect of varying α and β over their allowable range is less than that of varying LA weight.

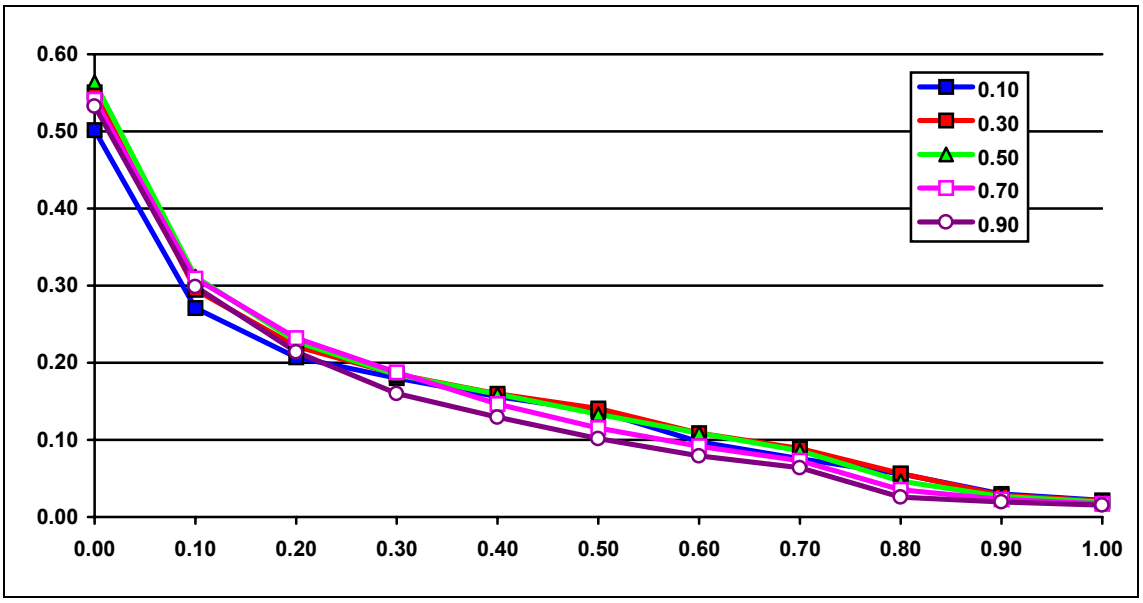

Figure 22 – Effects of Varying Alpha and Beta on Formula 57

RANDOM RANKING

Although we already had strong evidence about how much LA terms contributed to improving precision from the studies on Formula 55, we needed to examine their contribution independently from LA weight and LA distance effects. To do this, we constructed four new ranking formulae where LA terms were included and excluded, and tried two different distribution functions of a term's score to see what effect that had. In each case, we retrieved the same sets of documents as for either Formula 5 or Formula 55, as appropriate, but created different term scores from pseudo-random number generators with different distributions.

For each document retrieved, we generated a score for each query term appearing in the document. These terms were then summed. All the formula ignored how many times a given term occurred in a document. They only used the absence or presence of a term in their calculation. Terms not present in a document always received a score of 0. The differences only lay in what distribution was used for generating the term's random score.

Formula 90 only generated scores for singleton terms. All LA terms received a score of 0. It used a random uniform distribution for scores. Formula 91 added a random uniform distribution for LA terms equally weighted to singleton terms. This is equivalent to using an LA weight of 1.0. Formula 92 was like Formula 90 but used a random inverse exponential distribution. Formula 93 corresponds to Formula 91 with random inverse exponential for the LA terms. Formulae 90 and 92 are basically Boolean retrieval functions with scores

proportional to the number of matched singleton query terms. Formulas 91 and 93 are the same except they also include scores proportional to the number of matched LA terms. The random number generators merely added some noise to the results to reduce any systematic biases caused by how we sort the hit list before extracting the top 1000 documents for the TREC evaluation procedure. Figure 23 shows the results of this experiment.

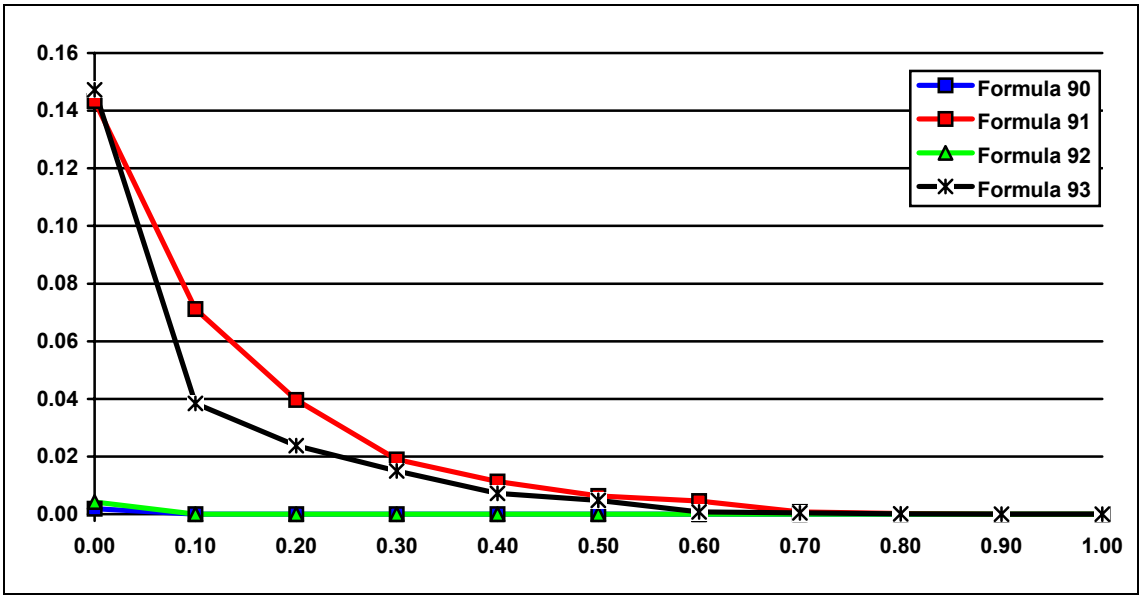

Figure 23 – Precision-Recall of Formulae with Randomly Assigned Term Scores

As expected the formulae without LA terms performed extremely badly. Without performing detailed calculations, it is hard to say how much better than pure chance these formulae performed. The formulae with LA terms did much better than the ones without. Formula 91's random ranking function is more strongly correlated with the number of LA terms in a document. This shows up as a better precision recall curve than for Formula 93. These results support the assertion that LA terms, when properly used, contribute significantly to a formula's precision. Note that this is a best case contribution since the singleton and LA term scores in these formulae are least correlated. In real ranking formula, the terms are more correlated, so the actual gain because of LA terms is less because singleton terms contribute more to a formula's precision. We never experimented with varying LA weight to see what effect that has.

Individual Query Results

The normal TREC-5 evaluation code produces average non-interpolated precision results for each of the 50 queries. Figure 24 summarizes how many queries where each of the GURU formula had better average non-

interpolation values than the median of all TREC-5 entries in the particular task category. If a formula always performed better than the TREC-5 median, it would have the number 50 next to it in the table. The closer to this number a formula achieves, the better. In each formula, we used the best possible values for LA weight.

Formula	Number of Queries Better than TREC-5 Median
1	11
55	19
56	14
57	17

Figure 24 – Average Non-interpolated Precision Compared to TREC-5 Median

This ordering of the ranking formulae exactly matches their rankings by average non-interpolated precision and precision at 0 recall over all queries and provides an additional confirmation of their relative performance.

Comparison with TREC-4 Results

We loaded TREC-4 data and other information needed to run a TREC-4 *a posteriori* Automatic Adhoc test for comparison with INQUERY. We could only compare with the full INQUERY results. Figure 25 compares the two. Given the limited query processing that we do in GURU, it performs surprisingly well. In TREC-4, INQUERY ranked 4th in average non-interpolated precision with a value of 0.2407. GURU's result of approximately 0.16 would have placed it near 10th overall. A more detailed comparison of individual TREC-4 participants' results is in [HARMAN96]. A more interesting comparison would be with the base results of query engines that do automatic relevance feedback. As with TREC-5, GURU would likely have ranked very highly in that type of comparison.

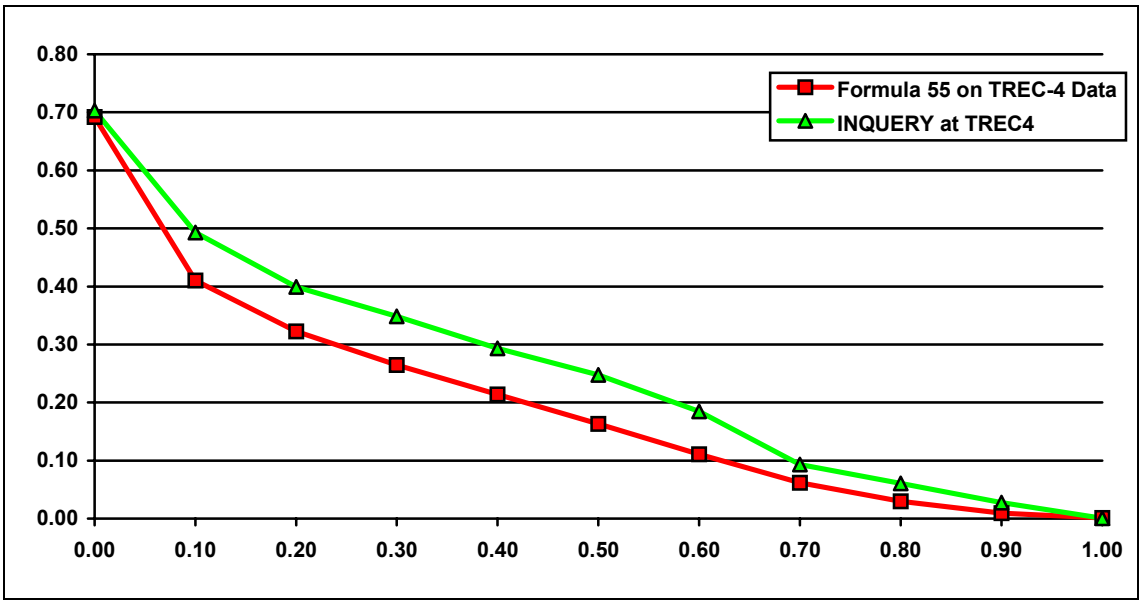

Figure 25 – Precision-Recall of Formula 55 on TREC-4 and INQUERY at TREC-4

GURU Document Length Effects

Length normalization in ranking formula has been recognized problem for a long time. Document length is usually not an explicit term in tf•idf formulae. Thus there is an implicit bias toward longer documents because terms occurring at a certain rate will occur more often in a longer document. [SINGHAL96] shows one method of addressing this problem, but it is only a way of compensating for an effect not explicitly part of a tf•idf ranking formula. Other ranking schemes also have this problem, as noted in [CLARKE97]. Probabilistic ranking schemes, if properly implemented, should not have any document length bias. This is because probabilities normally are independent of document length. Any effects visible in real implementations are more likely to be because of sampling effects than of flaws in the theoretical foundations.

One of the disconcerting effects empirically observed from Formula 1 is how often the top ranked documents are very short compared to documents further down the ranked hit list. An important goal in deriving the new ranking formula was to identify the causes and to correct for them, if possible. The first step was to create a measurement procedure to quantify the effects.

The measurement procedure we devised uses a document's rank in a query hit list to identify how well a document matches. We use this instead of score to make it easier to compartmentalize the data and accumulate results for averaging. We take all of the documents ranked first from each query and average their lengths in

words (singleton terms). We do the same for each succeeding rank until we reach the limit of 1000 documents in a hit list. This gives us a set of data that has been smoothed over ranking score and should show only systematic effects caused by moving through the rankings. We subject Formula 1 and 55 to this study.

FORMULA 1

Figure 26 shows a scatter plot of average document length against average document score for the top 1000 ranks. Visually, there is some correlation between document score and length, but with the overlapping points, it is hard to tell how significant is the correlation effect. Generally, though, there appears to be a systematic increase of score with document length. This happens to be the opposite of what we have empirically observed for Formula 1.

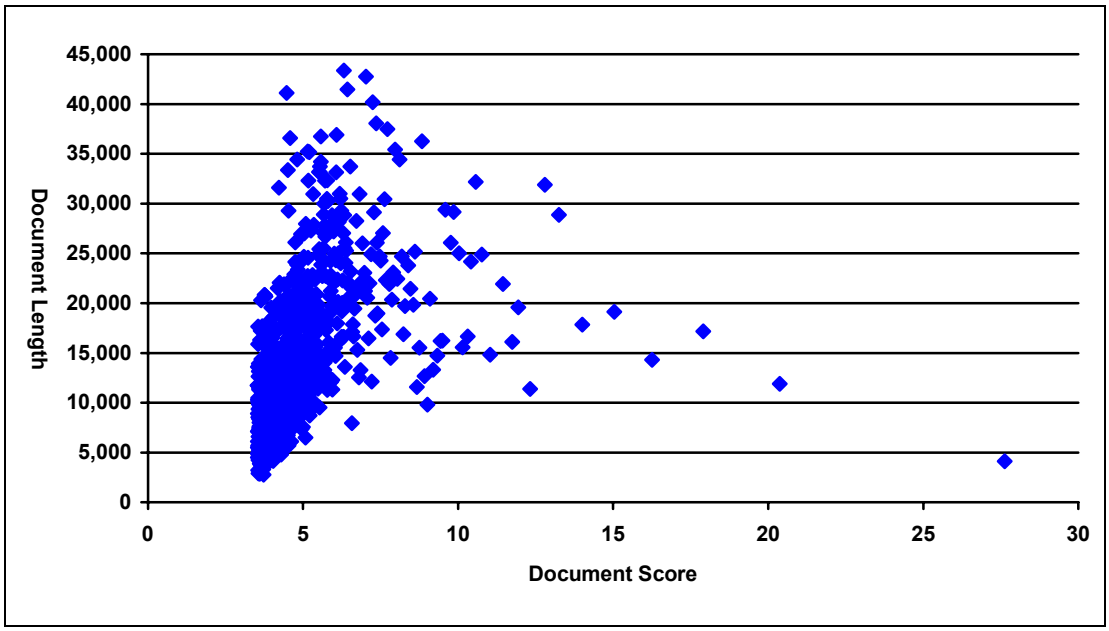

Figure 26 – Average Document Length versus Document Score of Formula 1 (Top 1000)

Plotting document length against document rank, as shown in Figure 27, clears up the effects somewhat. Most of the curve shows a decline of document length with lower ranking, but the head of the curve, above a rank of approximately 50 documents, average document length decreases with better ranking. The top ranked document tends to be shorter than documents of lower rank.

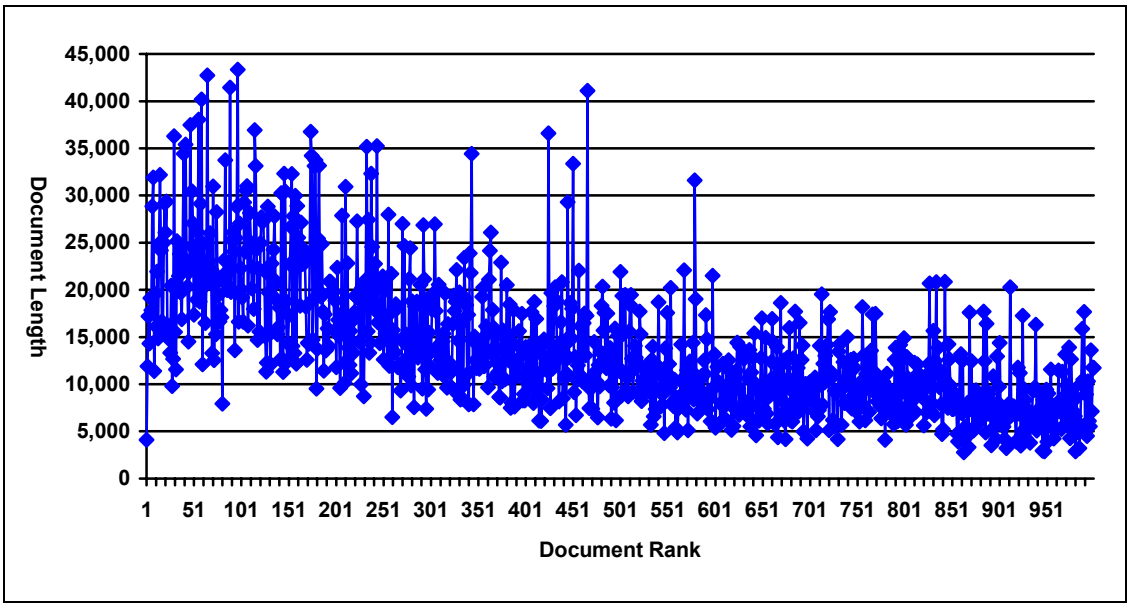

Figure 27 – Average Document Length versus Document Rank of Formula 1 (Top 1000)

Figure 28 shows only the top 40 document ranks. Here, it is easier to see the slope and in particular shows that the first 8 ranks have an even steeper slope than the general trend of this set of rankings.

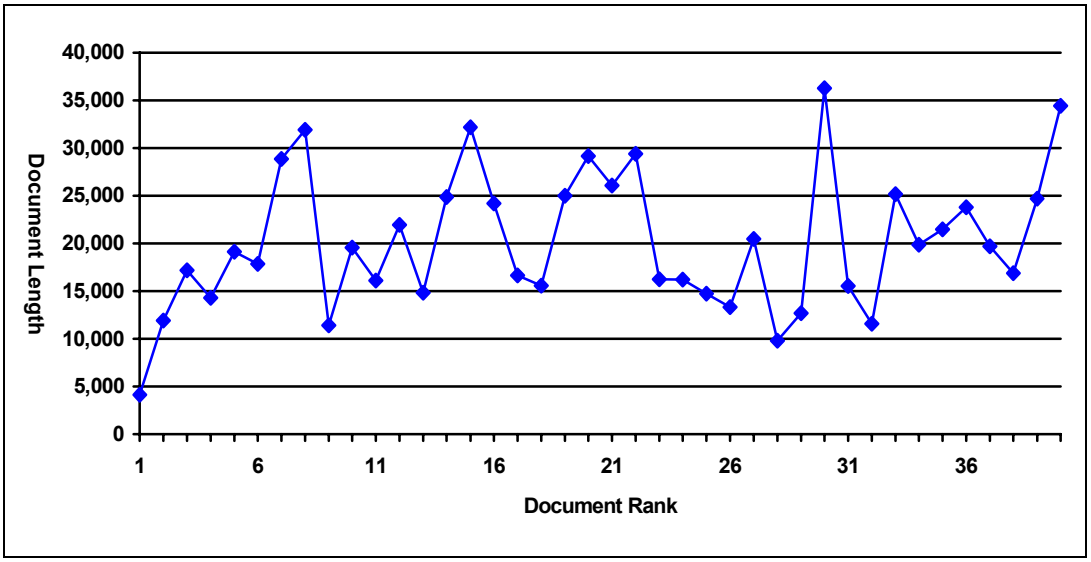

Figure 28 – Average Document Length versus Document Rank of Formula 1 (Top 40)

Averaging document lengths by ranks across queries is showing the effects we had hoped to see and provided some unexpected insights into the length effects we were trying to measure. Since for purely performance

reasons, we will not be continuing with Formula 1, the main value of doing this measurement was proving that we could measure what we had observed empirically.

FORMULA 55

Figure 29 shows a scatter plot of average document length against document score for Formula 55. Comparing this plot to Figure 26 shows major differences. The most important difference is that most of the variation is in document score and not in document length. Another important difference is how much lower the average document lengths are than retrieved by Formula 1. The final difference we are concerned with here is that there is no easily discernible systematic variation of document length with document score.

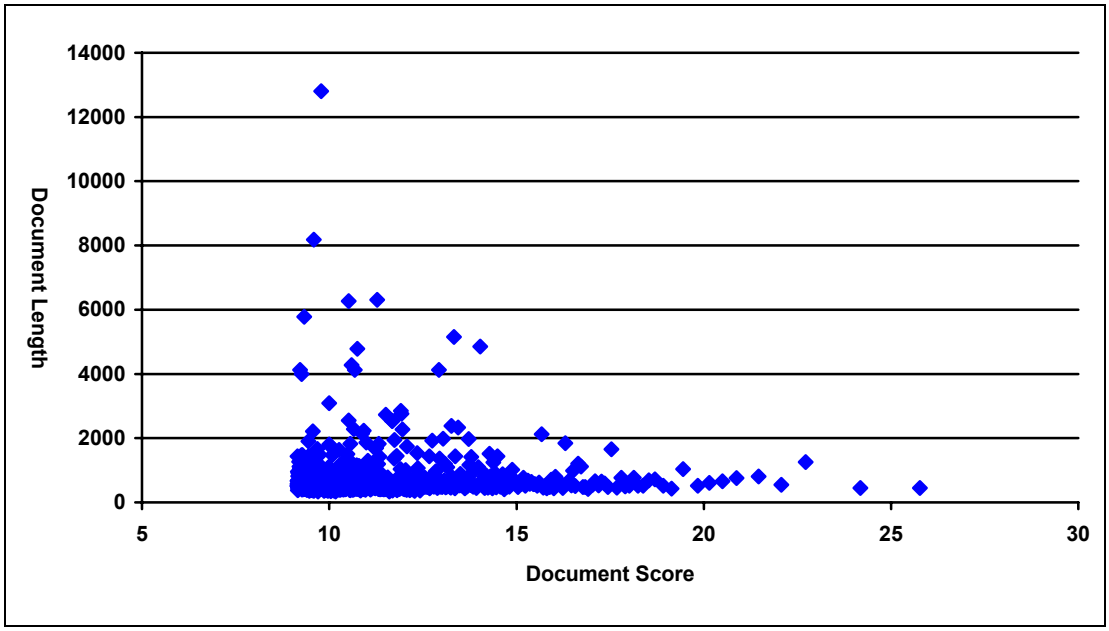

Figure 29 – Average Document Length versus Document Score of Formula 55 (Top 1000)

Figure 30 shows the plot of average document length against rank and corresponds to Figure 27 for Formula 1. There is a striking difference in the shape of the curve. Here, where a systematic variation of document length with document rank would be most obvious, there is no such obvious variation.

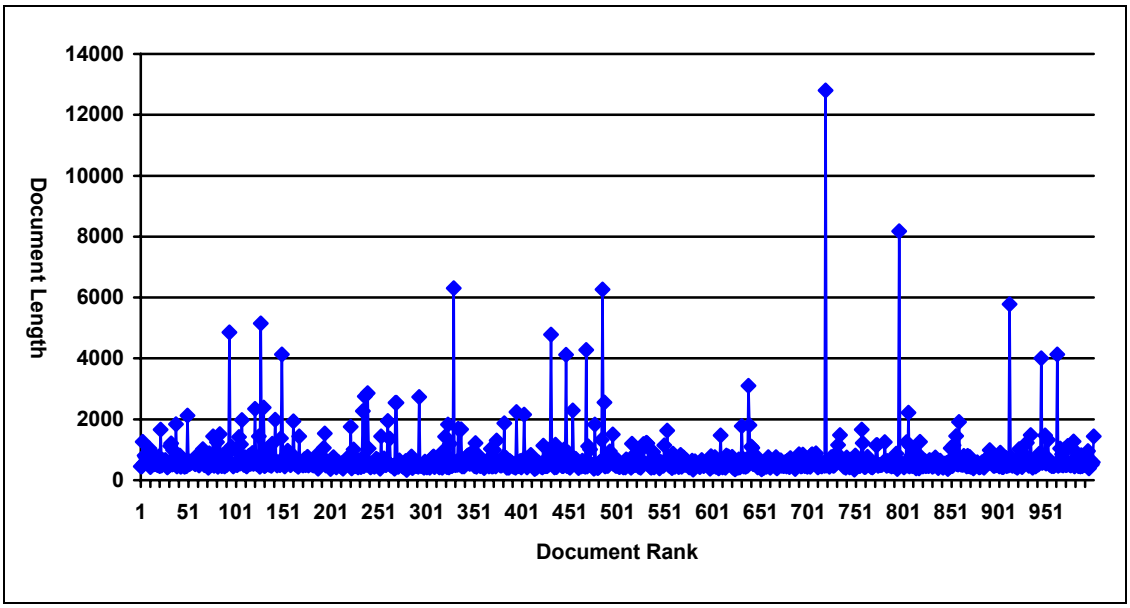

Figure 30 – Average Document Length versus Document Rank of Formula 55 (Top 1000)

Figure 31 shows an enlargement of the first 40 ranks only. Based on this plot and comparing it with Figure 28, we concluded that whatever systematic document length variations there are with rank, it was much smaller than for Formula 1 and that we had satisfactorily solved Formula 1's problem with preferentially returning shorter documents as highest ranked.

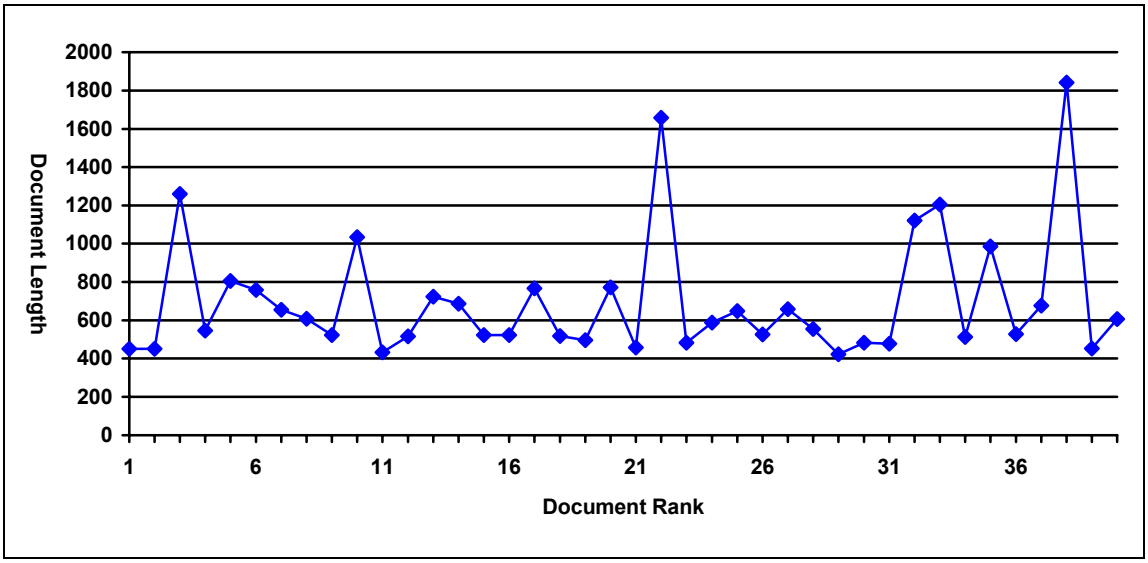

Figure 31 – Average Document Length versus Document Rank of Formula 55 (Top 40)

Although not shown here, we also performed the same analysis for Formulae 56 and 57 and came to the same conclusions. Neither of these formulae show a significant preference for shorter documents as Formula 1 does.

JUDGED DOCUMENTS

One of the immediate questions raised by the widely different average document lengths between Formula 1 and 55 is what were the average document lengths for the subcollections making up the TREC-5 test collection and was there a systematic variation in the judgments made by NIST. Figure 32 shows the data for the subcollections broken down by their relevance judgments. Viewed this way, there is no systematic difference between relevant and nonrelevant documents with regard to document length.

Collection	Relevant		Not Relevant	
	Documents	Length	Documents	Length
Associated Press	1672	491.31	15237	517.75
Congressional Record	844	12357.52	9670	31952.82
Federal Registry 88	38	51509.47	5362	19010.58
Federal Registry 94	509	572.69	31207	744.63
Financial Times	1583	644.41	18208	525.10
Wall Street Journal	1064	802.87	61869	684.88
Ziff-Davis	123	707.59	9581	604.10

Figure 32 – Average Document Size (Words) in TREC Document Collections

When we take average across only relevant and non-relevant, the means are 2650.7 and 3306.8 words respectively. This is a statistically significant difference between the two sets of documents. [KASZKIEL97] also has also found the same difference in lengths between the judged relevant and non-relevant documents. There may be several causes for the difference. One may be that the human judges preferentially choose shorter documents as relevant. Another may be that all of the search engines are biased toward shorter documents and the NIST relevance judgments reflect this bias. Finally, all the relevant documents may be shorter than average. Without manually judging every document in the test collection, we cannot be sure. Formula 55's average document lengths are well below the TREC document lengths, but we have no cause we can attribute yet.

Index Quality

One of the problems we have to deal with in GURU is how well our document parsers perform. This influences retrieval effectiveness because we do not currently use the same parser to analyze our queries. Thus, any mismatches will lead to not finding terms nor retrieving documents that contain them. For instance, out of the

approximately 180,000 unique symbols indexed from the Wall Street Journal subcollection, nearly 30,000 contain a leading numeric digit or punctuation character. These are disallowed by the query parser and were filtered out by the Perl script generating the GURU queries.

Another problem is in handling hyphenated words. The query parser disallows hyphens, while more than 34,000 symbols in the index dictionary contained a hyphen. Compounding this problem is that many of the hyphenated symbols join two complete English words ("part-time") or join word fragments that would normally not be hyphenated except across a line break ("abstrac-ted"). These are only two examples of the types of index quality problems that affect our retrieval effectiveness but are not controllable by GURU formula modifications.

Leading punctuation and digits and hyphenation is a general type of problem having to do with inconsistent technical implementation or just plain errors. We also have to deal with variations in language itself to handle the hyphenation problem. Some terms occur both in hyphenated and unhyphenated forms ("bug-free" and "bugfree"), depending on user preference. Since TREC topics are written in natural language form, they take no special account of such inconsistencies. The search engine must resolve these somehow. Since this is a realistic expectation for search engines, we must be able to handle these difficulties and quantify their effects on retrieval effectiveness, specifically precision and recall.

Further Work

Formula Experiments with Changed Assumptions

Derivation of the current GURU formula makes six important assumptions. Two of the most important ones that are the most questionable are the ones having to do with independence of terms within a query, and that of LA terms with non-LA terms. Although beyond the scope of this report, Formula 4 and the Formula 7 family explicitly address these two problems simultaneously. Currently, we have not worked out enough theory and mathematics to implement reliable and well performing ranking formula. Given Formula 55-57's performance and how they compare to the results in the literature, more advanced formula may only increase performance slightly. However, we have no way of telling without further research and testing of the independence assumptions.

Another avenue of research departs from the 5 and 55-57 series of ranking formula described in this paper by not using a weighted average of corpus and document term statistics to compensate for sampling effects. With Formula 57, shorter documents are more likely to have "unusual" document statistics and so are more likely to receive high ranks. One way to compensate is to use sampled confidence levels for evaluating a term's score. Doing so would take into account a term's distribution within documents that contain it as well as its rarity. Such formulae would also not have any length bias caused by sampling effects.

Improved Index Quality

Without specific proof, but by knowing how the GURU parser works and how we handle document parsing, we know we are creating situations where we simply cannot find terms in a corpus although they should be matched up at search time. Two problems stand out in particular: the large percentage of terms containing leading punctuation and numerics, and the uneven handling of hyphenated words. We have technology developed as part of the TALENT tools to handle these problems. However, none of the testing in this report made use of any of them. A slightly newer version of GURU using TALENT was developed for TREC-6, but the additional tools could not be made to work properly in time for the result submission. We need to experiment with the tools and their modifications to what eventually becomes indexed and searched to quantify their improvements. Following are some of the things handled by document parser enhancements.

AUTOMATIC NAME AND TECHNICAL TERM IDENTIFICATION AND CLASSIFICATION

TALENT contains tools to recognize proper nouns and technical terms. We have believed for several years that recognizing and indexing names and technical terms can help, but have never had a chance to prove it. GURU 1.6 will incorporate these and other components of TALENT into document and query parsing. With these in place, we will be able to quantify their benefits.

TERM EQUIVALENCE CLASSES

TALENT has the capability of recognizing numbers, dates, and currency values in text. These make up a disproportionate fraction of the unique terms indexed. Empirically, what people frequently search for are not specific numbers or dates but the fact that such a term occurs near to some other word term. We can reduce dictionary size and improve searches involving numbers, dates, and currency if we map specific strings of these types of terms to generic terms. The same mapping taking place at query time would enable at least approximate matching based on the other terms in the query.

HYPHENATION

The main document parser used for TREC experiments is based on SentSep without any special processing options. This tends to create many terms that contain fragments of words or words containing hyphens. We can address the problem of words broken across line boundaries, even though the TREC collections should not contain such terms, but hyphenation in general is a trickier problem to tackle. Word forms with and without hyphens frequently occur in the language. Hyphenated words can also occur as two separate words. We have, since the tests for this report were run, added to TALENT a tool for handling hyphens and producing more regular forms for indexing, but it is not a part of GURU yet.

Query Term Treatment

Right now, GURU treats all terms equally once they have been parsed and stop words removed. TREC queries and queries in general frequently do not want this equal treatment. The TREC queries can contain negative qualifications. Certain terms or concepts are undesirable and their occurrence should be a penalty to a document. We have the capability within the search engine to handle this with small coding changes, but the query parser does not have a way of expressing this. Other research has indicated that non-uniform treatment of term importance may bring gains in search engine performance.

VARIABLE AND USER-DEFINABLE TERM WEIGHTS

Sometimes, a user knows in advance that some query terms are more important than others. They may also know that they do not want specific terms. By means of a term weighting mechanism, we can incorporate these concepts into GURU and allow the user to specify them. Also, if we choose to, we can analyze a query and, by using corpus statistics, weight query terms by some function of their information content.

Currently, GURU has no capability of penalizing documents containing undesirable terms. TREC queries frequently make use of exclusion in their topic descriptions. If we hope to do better in TREC, at some point, we have to be able to handle these properly. Once we introduce the concept of term weights into GURU, we can empirically adjust for exclusions simply by using a negative weight for penalizing documents that contain them without excluding them completely from the ranking calculation. It will take further work to make a mathematically sound extension to the ranking formula.

ANTI-LA EFFECT

Several workers have done experiments with the converse of the concept of conceptual affinities. They have observed that words that are far apart in queries probably represent distinct concepts or relationships. Words close together are likely to represent parts of a single concept or relationship [LU97]. We have some capability to recognize these effects by being able to identify LA pairs. If a term occurs in a document but is not a part of an LA term, we can modify the term's weight. This is at least a way of detecting terms far apart in a query and accounting for them separately.

Complicating things is our planned incorporation of name and technical term recognition. These effectively shrink the LA window by changing inter-word-pair relationships to inter-multiword-term relationships. We may be able to account for this by indexing and querying only on canonical forms and changing the LA window size, but only experimentation can tell for sure. Also, the LA weight factor determined earlier may need changing.

NOISE LA TERMS

This idea is an extension of the stop word concept for single word terms. There are certain words required by the language to preserve syntactic structure and otherwise fit into its normal patterns of use. However, these words both occur so frequently and so uniformly in documents that they contribute little information about them. We, along with nearly all other search engines, prefilter these words and ignore them during query processing except

as placeholders for LA detection. We know that in a corpus there are the same types of word pairings that also are ubiquitous and not useful for searching.

Maarek's early work on GURU emphasized that she was looking for conceptual affinities and not lexical affinities. She used search time heuristics to determine these terms by calculating their information measures. Terms not meeting her empirical threshold for importance did not participate in the ranking calculation. Only terms that were sufficiently unusual in the document contributed to a document's score. We need to investigate this idea more fully because we know that certain LA terms in a query are not helpful while others are.

INFORMATION QUOTIENT

When trying to identify the relative importance of a term, many people use information measures such as entropy to identify candidates and rank them. Effectively, all rare terms become marked as highly important. From the point of view of search however, this may not be adequate. Rare terms may be uniformly distributed or may be concentrated in a few documents. Prager has defined a measure he calls "information quotient" (IQ) related to entropy that penalizes terms that occur too uniformly in documents, even if they are rare [PRAGER97]. The happens to match what a we want to happen in a probabilistic model for ranking. A rare query term that shows document concentration is more useful for document discrimination than a rare term that is more uniformly spread through a corpus. We can use IQ to screen terms at query time to select and use ones that provide better document discrimination.

Automatic Relevance Feedback

GURU was one of the few TREC-5 entries that did not use automatic relevance feedback to improve its search performance. The best formula we have today is Formula 55. The precision and recall averages we measured from this formula are nearing the upper third of all TREC entries even without such a procedure. However, since nearly all search engines in TREC use it, we need to incorporate something like it just to place ourselves at the same baseline performance as other search engines.

Automatic relevance feedback makes two assumptions about query formulation that we need to take into account. The first assumption is that someone specifying a query will not incorporate all the terms that cause relevant documents to rank highly. By automatically expanding the query to bring in these terms, the hope is that they will improve the ranking of relevant documents. The second assumption is that incorporating more terms will not add more noise by raising the rankings of irrelevant documents.

Since short queries predominate in situations where we want to use our search engine, searching for sets of documents can be improved by the term expansion done as part of automatic relevance feedback. If the goal of the search is not a set of documents but meta-information such as hyperdocument structure, perhaps automatic relevance feedback may not help. Also, some of the features of automatic relevance feedback can be captured by preprocessing a corpus and doing automatic query expansion using the preprocessed information in only one pass. We have technology in TALENT that does this type of information extraction and it is being incorporated into GURU 1.6.

There are six distinct steps to any automatic relevance feedback procedure: initial query submission, "relevant" document selection, "relevant" term extraction, query modification, query resubmission, and result ranking. Of the six, the first three are where most of the work can be concentrated. The last may or may not be important.

ALGORITHMIC QUERY AUGMENTATION

Many query systems do algorithmic query augmentation for the initial query. Systems that do stemming will stem the words in the query. More flexible systems like GURU that do automatic morphology will expand terms. The assumption here is that variant forms and close synonyms of terms are important. How important these new terms are compared to the terms actually specified can be debated. The tradeoffs being made here are how much automatic "understanding" of the searcher's intent can be made purely based on the entered query.

Nothing prevents a system from preprocessing a corpus and using the information extracted to augment the query. This can provide some of the effects of automatic relevance feedback since document and relationships outside that of the hit list contribute to an augmented query. However, traditional two pass techniques with an automatic relevance feedback procedure condition the extracted terms use for augmentation based on the documents in the hitlist. This difference may negate any query augmentation benefits not based on the top ranked documents of the hit list.

It is possible to overdetermine a query. [PRAGER96] has shown that from an information theoretic point of view, no more than twelve query terms are sufficient to return all relevant documents in the first 1000 ranked documents for TREC-5. These twelve terms would be independent. Finding enough real terms to be equivalent to twelve independent terms would constitute the maximum required size of a query. More than that and a ranking algorithm could be adding irrelevant documents into the ranked list.

SELECTING RELEVANT DOCUMENTS

Most of the systems participating in TREC-5 that do automatic relevance feedback choose some fixed number of the top ranked documents retrieved by the initial query and assume they are relevant. They examine these for terms to augment the query. Note that any automatic relevance feedback technique assumes that highly ranked documents contain a concentration of relevant documents and that there is a concentration of terms in them that that have different statistical properties from documents that are not relevant, i.e. the ranking function is actually accomplishing what it is supposed to.

Empirical studies of score versus rank studies with GURU suggest that the number of top ranked documents should not be fixed. Plots of score versus rank suggest that the change in slope of the values as a function of order in the hit list is much more usable across situations where there are many relevant documents and where there are very few relevant documents (see Figure 33). These plots show that there frequently is an initial steep dropoff in rank value at first and then a slower dropoff when there are definite set of relevant documents and a generally slow dropoff when there are no relevant documents. Somewhere on the change in slope is where a cutoff should occur. How sharp that cutoff is may indicate whether there are any relevant documents at all. We need further research to test these hypotheses.

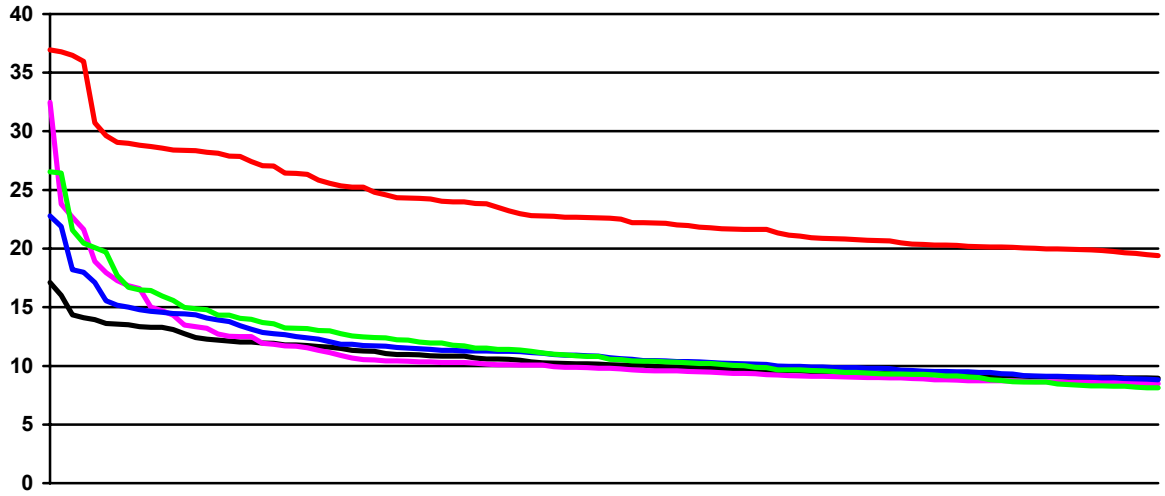

Figure 33 – Document Score versus Rank for 5 Queries

Another problem that researchers have tried to solve is whether all highly ranked documents are relevant. [LU97] proposes is to cluster the highly ranked documents in some metric space and find clusters. The number

and nature of clusters found indicates whether there are many, few, or no relevant documents. The metric used in [LU97] is a relative of the cosine measure. Their vector space model assumed lends itself to this approach.

RELEVANT TERM EXTRACTION

Presuming that one now has a set of relevant documents chosen from the hit list, one has to chose good terms from these documents to use. If, however, the query or the ranking function are poor, automatic relevance feedback will amplify their poor performance. Thus, it is important that before any further automatic relevance feedback processing takes place, some evaluation of a query's performance takes place to estimate the likelihood of improvement. What this evaluation might be is an open topic, but perhaps something based on the shape of the score versus rank curve may prove promising.

There are two basic ways of choosing the terms: conditioned on the relevant document list, or not conditioned on the relevant document list. The latter is the easiest to implement. The former is probably the better approach for higher precision and recall. However, research and experimentation are the only ways to determine the exact effects.

During some processing step related to indexing, it is possible to find a set of "important" or possibly "meta" terms for a document based on their properties in the document and relative to the corpus. These can be stored under a given document id and used to retrieve important terms at query time. These important terms for a document would be used as the pool of terms used to augment the query.

The problem with this approach is that the important terms that can best distinguish a given document in a corpus are not necessarily the best terms to find similar documents for a specific hit list. The "important" terms are in some fashion related to an "ideal" query for the document. This helps set the document apart from other documents in the corpus. It does not necessarily correlate with being more similar to the relevant documents. However, computing the set of terms is relatively easy using something such information measure to select terms. It may be possible with sufficient word statistics stored for a document to determine the measure relative to the "relevant" documents and arrive at a set of terms that is query and hit list specific.

The set of relevant documents in a query hit list represent documents that are in some way distinguishable from the rest of the documents in their corpus. If we assume that a query specifies only part of the set of terms that identify the relevant documents, some form of processing of the relevant documents should be able to find more important characteristics about the relevant document set and use them to find an expanded relevant set.

The cluster hypothesis is at work here. Apart from the fact that a single query selected the relevant documents, they must be similar (clustered) in some metric space. If an appropriate metric space can be found where these documents are clustered and is not just a simple transformation on the query's space, then other documents near or in the same cluster ought to be relevant too. In addition, the properties that define the cluster should be a superset of the properties used to originally locate the documents in the cluster if the query is not overspecified. The additional properties can be transformed into additional query terms to augment the original query and thus specify the cluster more precisely.

One technique for finding additional terms would be to identify all the high information measure terms in the relevant set of documents and then keeping only the ones that occur at a higher rate than expected in the relevant document set. This assumes that these terms are somehow related to the semantic content of the query and its initial set of relevant documents. The terms would be positive evidence that the documents are relevant. If we were to find that some high information measure terms occur less frequently in the relevant set than in the corpus, the absence of these terms would help identify more relevant documents. This would be negative evidence. Until such time as GURU supports such a term type, we cannot use this further information.

QUERY MODIFICATION

Given the original query as submitted to the first pass of a search engine and a new set of terms that identify the relevant documents from the original hit list, how should the query be modified to incorporate these terms? The most straightforward and most commonly implemented approach is augment the original query by adding all of the new terms. Presumably, one can do duplicate elimination and otherwise remove redundancies as a simple enhancement. If we use a clustering and feature extraction technique, we presumably have identified all the terms needed to locate a document cluster whether or not they occurred in the original query.

What if there are terms in the original query that do not contribute or add noise? Should extraneous terms be removed? Many of the TREC-5 participants using automatic relevance feedback algorithms augment a query by adding more terms. The assumption is that if a term is a noise term that does not contribute anything. Putting it into the query will not hurt the resulting query. However, we already know from an information theoretic point of view, a noise term does add to the noise and can degrade identification of relevant documents. Perhaps removing these terms can improve identification of relevant documents. We need to do further research in this area.

RESULT RANKING

That ranking comes up as an issue when doing automatic relevance feedback is because there is a difference of opinion on how important are the terms extracted automatically versus the terms specified by the user in their query. One can argue that because the user did not specify them, the extracted terms are less important. On the other hand, one can also argue that a user cannot be expert in all things and that there may be a highly useful term in relevant documents that was not specified in the original query.

In addition, if at the query modification step original terms are omitted, is there a need to modify or condition the relative weighting of terms? If we view the modified query as a brand new and refined query, then there is no need at all. If we take the view that the original terms are very important and must be treated special, then we must take that into account.

Acknowledgments

I would like to thank Roy Byrd, Mark Wegman, Yael Ravin, John Prager, and Eric Brown for their comments on this paper and discussions with them about the various fine points presented.

Ed Fox and his students at the Virginia Tech had prepared and run the TREC-5 tests for us. After completing the TREC-5 entries, he and his students did some more work with different formulae as we experimented with minor changes. The infrastructure that he and his students built provided us with the tools needed to objectively measure different ranking formulae and discriminate the effects of changes.

References

[ALLAN96] James Allan, Lisa Ballesteros, James P. Callan, W. Bruce Croft, and Zhihong Lu. Recent Experiments with INQUERY, In D. K. Harman, ed., *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pp. 49-63, NIST, 1996.

[ALLAN97] James Allan, Jamie Callan, Bruce Croft, Lisa Ballesteros, John Broglio, Jinxi Xu, and Hongmin Shu. INQUERY at TREC-5, In D. K. Harman, ed., *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, NIST (to appear), 1997.

[BEAULIEU97] M. M. Beaulieu, M. Gatford, Xiangji Huang, S. E. Robertson, S. Walker, and P. Williams. OKAPI at TREC-5, In D. K. Harman, ed., *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, NIST (to appear), 1997.

[BROWN96] Eric Brown, Personal correspondence, 1996.

[BUCKLEY97] Chris Buckley, Amit Singhal, and Mandar Mitra. Using Query Zoning and Correlation Within SMART: TREC 5, In D. K. Harman, ed., *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, NIST (to appear), 1997.

[CLARKE97] Charles L. A. Clarke and Gordon V. Cormack. Interactive Substring Retrieval (MultiText Experiments for TREC-5), In D. K. Harman, ed., *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, NIST (to appear), 1997.

[GEY97] Fredric C. Gey, Aitao Chen, Jianzhang He, Liangjie Xu, and Jason Meggs. Term importance, Boolean conjunct training, negative terms, and foreign language retrieval: probabilistic algorithms at TREC-5, In D. K. Harman, ed., *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, NIST (to appear), 1997.

[HARMAN96] Donna Harman. Appendix A, In D. K. Harman, ed., *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pp. A-7—A14, NIST, 1996.

[HARMAN97] Donna Harman. Overview of the Fourth Text REtrieval Conference (TREC-4), In D. K. Harman, ed., *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, NIST (to appear), 1997.

[IBM93] IBM SeachManager/2 Product Overview, Document Number GK10-2018.

[KASZKIEL97]. Marcin Kaszkiel, Phil Vines, Ross Wilkinson, and Justin Zobel, The MDS Experiments for TREC5, In D. K. Harman, ed., *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, NIST (to appear), 1997.

[KELLEDY97] Fergus Kelledy and Alan F. Smeaton. TREC-5 Experiments at Dublin City University: Query Space Reduction, Spanish, and Character Shape Encoding, In D. K. Harman, ed., *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, NIST (to appear), 1997.

[KWOK96] K. L. Kwok, A New Method of Weighting Query Terms for Ad-Hoc Retrieval, In Hans-Peter Frei, Donna Harman, Peter Schäuble and Ross Wilkinson, eds., *Proceedings of the Nineteenth Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 187-195, Zurich, 1996.

[KWOK97] K. L. Kwok and L. Grunfeld. TREC-5 English and Chinese Retrieval Experiments using PIRCS, In D. K. Harman, ed., *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, NIST (to appear), 1997.

[LU97]. Allan Lu, Maen Ayoub, JianHua Dong, Ad Hoc Experiments using EUREKA, In D. K. Harman, ed., *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, NIST (to appear), 1997.

[MAAREK89] Yoëlle Maarek and Frank Smadja, Full Text Indexing Based on Lexical Relations. An Application: Software Libraries, In N. J. Belkin and C. J. van Rijsbergen, eds. *Proceedings of the Twelfth Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 198-206, Cambridge, MA, 1989.

[MAAREK91] Yoëlle S. Maarek, Daniel M. Berry, and Gail E. Kaiser, An Information Retrieval Approach For Automatically Constructing Software Libraries, *IEEE Transactions on Software Engineering*, Vol. 17, No. 8, August, 1991, pp. 800-813.

[PIETRA95] Vincent Della Pietra, Y. S. Maarek, and M. Wegman. A New Probabilistic Measure for Incremental IR Systems, Unpublished paper, IBM Research, 1995.

[PRAGER96] John Prager, Personal correspondence, 1996.

[PRAGER97] John Prager, Information Quotient, unpublished paper, 1997.

[ROBERTSON94] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. OKAPI at TREC-2, In D. K. Harman, ed., *Proceedings of the Second Text REtrieval Conference (TREC-2)*, pp. 21-34, NIST, 1994.

[ROBERTSON95] S. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. OKAPI at TREC-3, In D. K. Harman, ed., *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pp. 109-126, NIST, 1995.

[ROBERTSON96] S. E. Robertson, S. Walker, M. M. Beaulieu, M. Gatford, and A. Payne. OKAPI at TREC-4, In D. K. Harman, ed., *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pp. 73-96, NIST, 1996.

[ROSE97] Daniel E. Rose and Curt Stevens. V-Twin: A Lightweight Engine for Interactive Use, In D. K. Harman, ed., *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, NIST (to appear), 1997.

[SALTON89] Gerard Salton, *Automatic Text Processing—the transformation, analysis, and retrieval of information by computer*. Addison Wesley, Reading, MA, 1989.

[SINGHAL96] Amit Singhal, Chris Buckley, and Manda Mitra. Pivoted Document Length Normalization, In Hans-Peter Frei, Donna Harman, Peter Schäuble and Ross Wilkinson, eds., *Proceedings of the Nineteenth Annual ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 21-29, Zurich, 1996

[TOU75] J. T. Tou and R. C. Gonzales, Pattern Recognition Principles, Addison Wesley, Reading, MA, 1974.