IRIS at TREC-7

Kiduk Yang, Kelly Maglaughlin, Lokman Meho, and Robert G. Sumner, Jr.
School of Information and Library Science
University of North Carolina
Chapel Hill, NC 27599-3360 USA
{yangk, maglk, mehol, sumnr}@ils.unc.edu

0 Submitted Runs

unc7aal1, unc7aal2 – Category A, automatic ad-hoc task runs (long query)
unc7ias, unc7iap – interactive track runs

1 Introduction

In our TREC-5 ad-hoc experiment, we tested two relevance feedback models, an adaptive linear model and a probabilistic model, using massive feedback query expansion (Sumner & Shaw, 1997). For our TREC-6 interactive experiment, we developed an interactive retrieval system called IRIS (Information Retrieval Interactive System[1]), which implemented modified versions of the feedback models with a three-valued scale of relevance and reduced feedback query expansion (Sumner, Yang, Akers & Shaw, 1998). The goal of the IRIS design was to provide users with ample opportunities to interact with the system throughout the search process. For example, users could supplement the initial query by choosing from a list of statistically significant, two-word collocations, or add and delete query terms as well as change their weights at each search iteration. Unfortunately, it was difficult to tell how much effect each IRIS feature had on the retrieval outcome due to such factors as strong searcher effect and major differences between the experimental and control systems.

In our TREC-7 interactive experiment, we attempted to isolate the effect of a given system feature by making the experimental and control systems identical, save for the feature we were studying. In one interactive experiment, the difference between the experimental and control systems was the display and modification capability of term weights. In another experiment, the difference was relevance feedback by passage versus document.

For the TREC-7 ad-hoc task, we wanted to examine the effectiveness of relevance feedback using a subcollection in order to lay the groundwork for future participation in the Very Large Corpus experiment. Though the pre-test results showed the retrieval effectiveness of a subcollection approach to be competitive with a whole collection approach, we were not able to execute the subcollection retrieval in the actual ad-hoc experiment due to hardware problems. Instead, our ad-hoc experiment consisted of a simple initial retrieval run and a pseudo-relevance feedback run using the top 5 documents as relevant and the 100th document as non-relevant.

Though the precision was high in the top few documents, the ad-hoc results were below average by TREC measures as expected. In the interactive experiment, the passage feedback results were better than the document feedback results, and the results of the simple interface system that did not display query term weights were better than that of the more complex interface system that displayed query term weights and allowed users to change these weights. Overall interactive results were about average among participants.

2 Key Components of IRIS

2.1 Text Processing

IRIS processes the text first by removing punctuation, and then excluding the 390 high-frequency terms listed in the WAIS default stopwords list as well as "IRIS stopwords," which are defined as all numeric words, words that start with a special character, words consisting of more than 25 non-special characters, and words with embedded special characters other than a period, apostrophe, hyphen, underline, or forward or backward slash. The IRIS stopwords definition was arrived at by examining the inverted index and identifying low frequency terms that appeared meaningless. The removal of IRIS stopwords reduced the number of unique terms by over 25% (401,423 to 295,257 in the Financial Times collection), which can effect considerable savings in machine resources. Such savings can be a significant factor when dealing with massive collections.

After the initial processing step described above, IRIS conflates each word by applying one of the four stemmers implemented in the IRIS Nice Stemmer module,[2] which consists of a simple plural remover (Frakes & Baeza-Yates, 1992, chap. 8), the Porter stemmer (Porter, 1980), the modified Krovetz inflectional stemmer, and the Combo stemmer. The modified Krovetz inflectional stemmer implements a modified version of Krovetz's inflectional stemmer algorithm (Krovetz, 1993) and restores the root form of plural ("-s," "-es," "-ies"), past tense ("-ed"), and present participle ("-ing") words, provided this root form is in our online dictionary. Though this

[1] A prior version of IRIS was developed by Kiduk Yang, Kristin Chaffin, Sean Semone, and Lisa Wilcox at the School of Information and Library Science (SILS) at the University of North Carolina. They worked under the supervision of William Shaw and Robert Losee.
[2] Nice stemmer was implemented by Kiduk Yang, Danqi Song, Woo-Seob Jeong, and Rong Tang at SILS at UNC.

stemmer's conservative conflation approach can be advantageous over suffix-removal stemmers that can adversely affect precision by overstemming, it can also cause lower recall by understemming, since the morphological variations targeted for conflation are few. The Combo stemmer attempts to minimize the disadvantages of both understemming and overstemming by taking as the final result the shortest whole word (i.e., word that appears in a dictionary) returned by the three stemmers. For example, the Krovetz stemmer does not conflate "disappointment" and "goodness," and the Porter Stemmer overconflates "ponies," "agreed" and "troubling" to "poni," "agre," and "troubl," but the Combo stemmer correctly stems these words to "disappoint," "good," "pony," "agree," and "trouble."[3] Unfortunately, the Combo stemmer's computational cost is very high due to its multiple dictionary lookup per word. Given the resource limitations at SILS relative to the size of the TREC-7 collection and the fact that the effectiveness of the Combo stemmer has not yet been fully tested, we opted for the modified Krovetz stemmer as the default stemmer for the TREC-7 experiments.

2.2 Phrase Construction

In our TREC-6 experiments, we constructed a statistically significant, two-word collocation index by extracting co-occurring word pairs within a window of 4 words (Haas & Losee, 1994; Losee, 1994) and selecting those that co-occur with statistically significant frequency (Berry-Rogghe, 1974). Though this collocation index worked very well in some cases, its overall effect on retrieval effectiveness did not appear to be significant (Sumner et. al., 1998). Furthermore, the computational cost of constructing the collocation index was quite high.

Consequently, we tried another approach to constructing a phrase index in TREC-7. Using the online dictionary and the clause recognition algorithm built into the Nice Stemmer, we constructed a two-word noun-noun phrase index by first extracting adjacent word pairs of noun and proper noun combinations within a clause,[4] and then discarding the phrases occurring 20 or less times in the collection to reduce indexing time and to conserve computer resources. The phrase occurrence frequency threshold of 20 was arrived at by selecting the number that produced the phrase index whose size was most comparable to that of the collocation index. To augment the proper nouns in the online dictionary, all capitalized words not occurring at the beginning of a sentence were considered to be proper nouns. Since the Krovetz stemmer does not conflate hyphenated words, hyphenated words were broken up and stemmed by the simple plural remover before the noun-noun phrase construction module was applied. Hyphenated words in their raw form (i.e. as they appear in documents sans punctuation) were added to the index as well.

2.3 Ranking Function and Term Weights

IRIS ranks the retrieved documents in decreasing order of the inner product of document and query vectors,

$$\mathbf{q}^{\mathrm{T}}\mathbf{d}_i = \sum_{k=1}^{t} q_k d_{ik} \,, \qquad (1)$$

where q_k is the weight of term k in the query, d_{ik} is the weight of term k in document i, and t is the number of terms in the index. We used SMART *Lnu* weights for document terms (Buckley, Singhal, Mitra, & Salton, 1996; Buckley, Singhal, & Mitra, 1997), and SMART *ltc* weights (Buckley, C., Salton, G., Allan, J., & Singhal, A., 1995) for query terms. *Lnu* weights attempt to match the probability of retrieval given a document length with the probability of relevance given that length (Singhal, Buckley, & Mitra, 1996). Our implementation of *Lnu* weights was the same as that of Buckley et al. (1996, 1997) except for the value of the *slope* in the formula, which is an adjustable parameter whose optimal value may depend, in part, on the properties of the document collection.

According to the pre-test experiments, an *Lnu* slope of 0.5 performed best with feedback, especially when using both single term and phrase indexes. In initial retrieval without any feedback, however, a slope of 0.2 or 0.3 showed best results. Based on these findings, we used a slope of 0.3 in the ad-hoc experiment to optimize the initial retrieval results, but used a slope of 0.5 in the interactive experiment to optimize performance with feedback.

2.4 Feedback Models

2.4.1 Adaptive Linear Model

Currently, the default relevance feedback model of IRIS is the adaptive linear model (Wong & Yao, 1990; Wong, Yao, Salton, & Buckley, 1991). The basic approach of the adaptive linear model, which is based on the concept of the preference relation from decision theory (Fishburn, 1970), is to find a *solution vector* that, given any two documents in the collection, will rank a more-preferred document before a less-preferred one (Wong et al., 1988).

The goal of the adaptive linear model, in essence, is to construct a query vector that ranks the entire document collection according to the user's preferences. Since the user's preferences are not usually known for the whole collection, however, we can only create a solution vector for the *training set T*, which is the cumulative set of documents retrieved and evaluated by the user. As knowledge of the user's preferences accumulates with relevance feedback iterations, one can expect the solution vector for *T* to more accurately rank the entire collection (Wong & Yao, 1990).

---

[3] For interactive comparison of these stemmers, please visit http://ils.unc.edu/iris/nstem.htm.

[4] IRIS identifies a clause boundary by the presence of appropriate punctuation marks such as a comma, period, semicolon, question mark, or exclamation mark.

The *error-correction procedure* we used to find a solution vector for T in our TREC experiments is based on the procedure used by Wong et al. (1991). The error-correction procedure begins with a *starting vector* $\mathbf{q}_{(0)}$ and repeats the cycle of "error-correction" until a solution vector is found. The error-correction cycle i is defined by

$$\mathbf{q}_{(i+1)} = \mathbf{q}_{(i)} + \alpha\mathbf{b}\ , \qquad (2)$$

where α is a constant, and $\mathbf{b}$ is the *difference vector* resulting from subtracting a less-preferred document vector from a more preferred one. (For details about how this difference vector is chosen, see Sumner et al., 1998.) The choices for the constant α and the *starting vector* $\mathbf{q}_{(0)}$ are very important since they can influence not only the composition of the solution vector but also the number of error-correction cycles needed to arrive at it Different choices have been made for α and $\mathbf{q}_{(0)}$ in our TREC-5, TREC-6, and TREC-7 experiments (Sumner & Shaw, 1997, Sumner et al., 1998).

In the relevance feedback interface of IRIS, users can evaluate documents as "relevant," "marginally relevant," or "nonrelevant." By adapting the concept of the user preference relation to extend the relevance scale from a binary to a three-valued scale, we constructed the following formula for the starting vector. Note that this formula can be adjusted for any multivalued relevance scale:

$$\mathbf{q}_{(0)} = c_0\mathbf{q}_{rk} + \frac{c_1}{N_{new\,rel}}\sum_{new\,rel}\mathbf{d} + \frac{c_2}{N_{new\,mrel}}\sum_{new\,mrel}\mathbf{d} - \frac{c_3}{N_{new\,nonrel}}\sum_{new\,nonrel}\mathbf{d}\ , \qquad (3)$$

where $\mathbf{q}_{rk}$ is the query vector that produced the current ranking of documents; c_0, c_1, c_2, and c_3 are constants; $N_{new\ rel}$, $N_{new\ mrel}$, and $N_{new\ nonrel}$ are the number of new relevant, new marginally relevant, and new nonrelevant documents respectively in the current iteration; and the summations are over the appropriate new documents. This formula is similar to the relevance feedback formulas used by Rocchio (1971) and Salton and Buckley (1990). A "new" document during a given search iteration is one that was not retrieved and evaluated during a previous iteration. Alternatively, it may also be a document that was retrieved and evaluated in a previous iteration, but whose relevance judgement was changed in the current iteration.

Because every new document vector already contributes to the starting vector (Equation 3), we used a value of $\alpha = 0.5$ (Equation 2) to reduce the influence of any one new document. The value of $c_2 = 0.6$ was chosen so that a marginally relevant document could still contribute to the final query vector even after being subtracted in the error correction procedure ($c_2 - \alpha = 0.1$). We set $c_1 = 1.2$ so that the influence of relevant documents would be twice that of marginally relevant ones and set $c_3 = 0.6$ for internal consistency. Though we used $c_0 = 1.0$ in TREC-6 experiments, we adjusted it to the value of 0.2 in the TREC-7 interactive experiment to reduce the influence of the initial query. We noticed in our post-TREC6 experiments that the influence of the initial query tended to overshadow the user feedback, and consequently set $c_0 = 0.2$, which seemed to make the system more responsive to the user feedback. In the ad-hoc experiment, however, we used the value of $c_0 = 1.0$ since the importance of pseudo-feedback for that task was viewed as minimal.

2.4.2 Probabilistic Model

In addition to the adaptive linear model, a variation of the binary probabilistic feedback model that accommodates three levels of relevance judgments is implemented in IRIS. As is the case with the adaptive linear model, this probabilistic model with the graded relevance formula (Yang & Yang, 1997) can be adjusted for any multivalued relevance scale including the binary relevance scale.

According to the TREC-7 pre-tests as well as our past TREC results, our implementation of the adaptive linear model performed consistently better than that of the probabilistic model when using binary relevance feedback. The findings of the TREC-6 interactive experiment regarding the comparative performances of the two feedback models using the three-valued relevance scale is inconclusive due to other factors such as searcher effect. Given these considerations and our resource limitations, we decided to exclude the probabilistic model from the actual TREC-7 experiments. The detailed description of the probabilistic model can be found in Sumner et. al. (1998).

2.4.3 Passage Feedback Model

The conventional relevance feedback models assume the user's relevance judgement, whether binary or multi-valued, to be about an entire document. The unit of a document, however, can sometimes be arbitrary, as in the case of web documents whose boundaries are often determined for reasons of convenience or efficiency rather than content, or can contain subsections of various information content as in Congressional Record and Federal Register documents. The findings from passage feedback research (Melucci, 1998) as well as from comments made by IRIS users at large indicate that determination of relevance is sometimes based on certain portions of a document rather than the entirety of it.

To test out this theory in TREC-7 experiments, we implemented in IRIS a third relevance feedback model called the "passage feedback model". The formula for feedback vector creation in the passage feedback model looks almost identical to the "Ide regular" formula (Ide, 1971; Salton & Buckley, 1990), except where the document vector $\mathbf{d}$ is replaced by $\mathbf{p}$, the passage vector.

$$\mathbf{q}_{\mathbf{new}} = \mathbf{q}_{\mathbf{old}} + \sum_{rel}\mathbf{p} - \sum_{nonrel}\mathbf{p} \qquad (4)$$

Since the normalization factor of the *Lnu* weight is based on document length, an inverse document frequency weight was used for the passage vector $\mathbf{p}$.

In the interactive setting of the IRIS passage feedback interface, the unit of passage is determined by the user, who can simply highlight the relevant and nonrelevant portions of documents with a mouse. Passage feedback can also be implemented by *automatically* selecting passages with high frequencies of matching query terms, though the automatic determination of nonrelevant passages is not possible with this approach. Such automatic passage feedback approach may be useful if activated after the initial feedback so as to expand the initial query with related terms.

The passage feedback approach differs fundamentally from the philosophy of the adaptive linear model and the probabilistic model. Regardless of whether a document or passage is used as the unit of feedback, the passage feedback model does not attempt, in principle, to rank a document collection in the preference or relevance order defined by a training set. Instead, the passage feedback model, similar to conventional vector space models, simply expands the query vector to make it more "similar" to relevant passages and "dissimilar" to nonrelevant passages.

3 Pre-test Experiments

In our TREC-7 pre-test experiments, we chose to examine the effects of 4 main system components of representation (single term vs. phrases), term weighting (normalization slope), feedback model (adaptive linear vs. probabilistic), and feedback query expansion size (full-expansion, 300 terms, 30 terms). As a preparation for potential future efforts to scale up to massive document collections, we also examined the effectivenss of relevance feedback using a subcollection. The FT collection with TREC-6 queries and relevance judgements was used in these experiments, and many system design decisions in both ad-hoc and interactive experiments were based on the findings from the pre-test results.

3.1. System Component Tests

3.1.1 Experiment Design

Prior experiments, both in and outside of TREC, have shown the use of syntactic phrases to be only marginally effective (Salton, 1968; Lewis, Croft & Bhandaru, 1989). However, most of the findings were based on the performance of initial retrieval only and did not investigate the effect of automatically expanding the feedback query with phrase index terms.[5]

Though *Lnu* weights with a slope of 0.2 proved effective in both TREC-4 and TREC-5 (Buckley et. al., 1996; Buckley, Singhal, & Mitra, 1997), we found a slope of 0.3 to be more effective with respect to initial retrieval in our TREC-6 experiments (Sumner et. al., 1998). As is the case with phrases, *Lnu* weight experiments did not investigate its effects on retrieval beyond the first feedback iteration.

In our past TREC experiments, we compared the performances of the adaptive linear model (ALM) and probabilistic model (PM) in relevance feedback, and though ALM generally outperformed PM, we noticed distinctly different retrieval patterns between the two models, which warranted further investigation (Sumner & Shaw, 1997; Sumner et. al., 1998).

In TREC-6, we also compared the performance of a fully expanded feedback vector with that of a shorter feedback vector, namely one with the top 250 positive-weighted terms and the lowest 50 negative-weighted terms. Previous routing and adhoc pseudo-feedback experiments in TREC have shown that effectiveness improves linearly with the log of the number of added terms, with the point of diminishing improvement at 300 terms (Buckley et. al., 1995). This was in direct contrast to our results in TREC-6, which, though somewhat suspect due to a system bug, indicated superior performance of the fully expanded feedback vector over the 300 term feedback vector. However, the advantage gained by full expansion of the feedback vector was marginal and the shorter feedback vector performed reasonably well given its size, which was about one tenth that of the full feedback vector. In addition to reconfirming our previous findings regarding feedback query size, we wanted to investigate the retrieval performance level of an even shorter feedback query that consisted of the top 25 positive-weighted terms and the lowest 5 negative-weighted terms—just to see how much gain in efficiency can be achieved without sacrificing too much in effectiveness.

In order to identify the optimum retrieval component combinations of representation, normalization weight, feedback model, and feedback query size, one of us (Yang) conducted an experiment using 5 retrieval iterations (4 feedback iterations with a feedback window of 20 documents) with 48 retrieval model combinations as outlined below. A feedback window of 20 documents means that the top 20 *previously unretrieved* documents of the current ranking are added to the training set. A feedback window of 20 documents and 5 retrieval iterations were chosen to simulate the capacity of a human searcher based on the data from TREC-6 experiments.

Representation	*Lnu* slope	Feedback model	Feedback expansion
Single term Single & phrase term	0.1 0.2 0.3 0.5	Adaptive Linear Probabilistic	Full expansion 250p + 50n 25p + 5n

2 * 4 * 2 * 3 = 48 Retrieval Model combinations

The retrieval results of all the model combinations were then compared using optimum effectiveness (F) in the top 20 documents retrieved as well as using standard TREC evaluation metrics. We chose these evaluation measures because optimum F, which represents the optimum performance level of the top 20 retrieved documents, and TREC metrics, which signify the overall performance level of the top 1000 retrieved documents, tend to complement each other.

[5] Here, routing and filtering experiments are not considered due to the different nature of those tasks and the ad hoc task.

TREC evaluation measures used were average precision across all relevant documents, R-precision, and the total number of relevant documents retrieved in the top 1000 documents. Optimum F is the highest F value in all retrieval iterations, where F is computed from recall and precision (Shaw, 1986) by the formula,

$$F = \frac{2}{\frac{1}{R} + \frac{1}{P}} . \tag{5}$$

3.1.2 Results

The analysis of retrieval results by all evaluation measures used showed a consistent pattern of improved retrieval performance with the larger feedback query. The difference in performance between the 30 term feedback vector and the 300 term feedback vector, however, was much greater than that between the 300 term feedback vector and the full feedback vector. As a matter of fact, the reduction in performance by limiting the feedback vector to 300 terms was almost negligible, whereas significant loss of performance occurred by reducing the feedback vector to 30 terms.

Though there were slight variations across evaluation methods, an *Lnu* slope of 0.5 seemed to be most advantageous for ALM and an *Lnu* slope of 0.2 seemed to perform best with PM. Using phrases in feedback as well as single terms resulted in slightly improved retrieval performances by both feedback models, which suggested that using phrases in feedback provides some utility though less than one might hope for.

As was the case in our previous TREC experiments, ALM consistently outperformed PM across all evaluation measures. The difference between the two models was most prominent in the number of relevant documents retrieved in the top 1000 documents, where ALM retrieved hundreds more relevant documents than PM. Upon closer inspection of ALM and PM, we discovered a pattern of "failure" by PM, where PM's feedback query formulation strategy of selecting terms from relevant documents would stagnate the performance of feedback when no more relevant documents could be found. ALM, on the other hand, would continue expanding the feedback vector in its attempt to find the solution vector (Wong et al., 1988; Wong et. al. 1991).

At this point, we devised the "Adaptive Probabilistic Model" (APM), which will keep adding terms from top-ranked non-relevant documents until finding the solution vector that will rank a more-preferred document before a less-preferred one (Wong et al., 1988). Due to time constraints, however, we did not engage in full-scale retrieval experiments with APM. Instead we tested APM in a limited fashion, which resulted in only a marginal improvement of retrieval performance.

3.2. Subcollection Tests

3.2.1 Experiment Design

One of the immediate challenges in the field of Information Retrieval is effective and efficient handling of massive document collections. When dealing with massive document collections, the conventional IR approach of ranking the entire document collection by document-query similarity scores becomes extremely resource-intensive, especially with relevance feedback, where retrieval cycles have to be repeated with expanded query vectors.

One way to deal with massive data may be to create a subcollection, which is small enough to be efficient and yet large enough to contain most of the relevant documents. Once such a subcollection has been created, we can not only refine the search with relevance feedback at relatively small cost, but can also continue to refine and/or update the subcollection by periodically resubmitting to the entire collection the reformulated query created using the subcollection. The main question in subcollection IR is twofold; First, how do we create a subcollection with high enough recall and small enough size? Second, is the retrieval performance of an optimum subcollection competitive to that of the whole collection? In order to investigate these questions, we explored various subcollection creation methods to identify the optimum subcollection creation method, after which we compared its retrieval performance with that of using the whole collection.

The first objective of subcollection creation is to maximize recall at some optimum document rank N, so that the subcollection is small enough to be efficient while containing enough relevant data to be effective. In addition to applying the optimum retrieval component combinations identified in the system component tests, we implemented combinations of document-reranking methods to retrieve relevant documents that may not necessarily contain any initial query terms. Initial retrieval, being essentially a Boolean OR retrieval, will only retrieve documents that contain at least one query term. Thus, a poorly formulated initial query will tend to not retrieve many relevant documents that may include only synonyms or related concept terms.

One way to overcome this problem is to expand the query vector with synonyms or related concept terms as well as using word-stems to conflate the morphological variations. Relevance feedback also expands the query vector indirectly with synonyms and related concept terms often contained in the body of relevant documents, though the effect may not be as precise as using a thesaurus or other such natural language processing methods. Consequently, we experimented with expanding the initial query with noun-noun phrases as well as expanding it by applying the "pseudo-relevance feedback" method of assuming that the top 5 documents are relevant and the 100^{th} document is non-relevant. Variations on this method of expanding the initial query by pseudo-relevance feedback (using terms from the top *n* documents) have been used by top performing participants in past TREC ad-hoc experiments (Buckley et. al., 1995; Voorhees & Harman, 1997).

In addition to query expansion by phrases and automatic feedback, we also tested query expansion methods by using passages[6] with matching initial query terms. Three variations of query expansion by passage feedback were tested by selecting terms from only

[6] IRIS identifies a passage boundary by SGML tags or a clause break followed by a carriage return.

the "relevant" passages, terms from relevant passages and top-ranked "non-relevant" passages, and terms from all passages (both relevant and non-relevant) in the top 100 documents retrieved. The "relevance" of a passage is determined by an arbitrary threshold of matching query term numbers in a given passage. These subcollection creation methods along with the baseline method of initial retrieval with the unexpanded original query were then investigated by comparing recall at various document ranks up to the rank of 21,000 (10 % of FT collection).

After identifying the optimum subcollection creation method and cutoff, we created 47 subcollections, one for each query[7], and recomputed their collection statistics, namely *Lnu* weights for documents, and *ltc* weights for queries. We then performed 5 retrieval iterations with a feedback window of 20 documents using selected retrieval models from the system component tests, and compared the performance of subcollection retrieval with that of whole collection retrieval. The same evaluation metrics used in the analysis of system component test results were applied to evaluate the performance of subcollection retrieval.

3.2.2 Results

According to recall values at fixed document ranks, the top-performing subcollection creation method was pseudo-relevance feedback by ALM, though average recall (recall averaged over queries) at document rank 5000 was the same for the top 4 methods. Interestingly enough, the baseline method was one of the top 4 methods, performing only slightly below the methods of initial query expansion by phrases and feedback by ALM. The results of passage feedback methods (PFM) were disappointing. However, poor performance of PFM could be due to an improper threshold of "relevant" passage identification (e.g. passages with *n* or more query terms). The hypothesis that a relevant passage would include related terms and concepts is critically dependent on the correct identification of relevant passages. Since the top 4 subcollection creation methods all achieved average recall of 0.87 at 5000 documents, which is only 2.4% of the total FT collection, we chose the optimum cutoff at 5000 and decided on the simplest method (i.e. baseline initial retrieval with single term queries) to create the subcollections.

The comparison of the subcollection retrieval results with the whole collection retrieval results showed an interesting difference between ALM and PM. The performance of PM using a subcollection was better than that of PM using the whole collection, whereas ALM's performance deteriorated slightly with subcollection retrieval. Overall performance of ALM, however, was again superior to that of PM, though the gap in performance between ALM and PM was much narrower in subcollection retrieval than whole collection retrieval.

Different behaviors of ALM and PM may be attributed to fundamental differences in feedback query formulation between the two models. ALM starts out with the initial query vector and keeps adding and subtracting terms to find the solution vector, which is a radically different approach from PM's strategy of estimating the probability of term occurrence in all relevant/nonrelevant documents from its occurrence characteristics in a training set. ALM's feedback vector is firmly anchored with the initial query terms and is more resilient to improper and/or insufficient feedback evaluations, whereas PM's feedback vector can be affected severely by bad relevance judgements and small training sets. It is therefore reasonable to assume that PM will perform better as the ratio of the training set size to the document collection size increases, as in the case of the subcollection retrieval.

The overall results of relevance feedback using subcollections has shown it to be almost as effective as using the whole collection while being much more efficient. However, by virtue of the fact that only the top 100 documents of various TREC runs are evaluated for a given set of topics, the TREC test collection may be inherently put together to show high recall for the top *N* documents, given *N* is sufficiently large enough. Accordingly, it is difficult to tell how much of these good results using subcollections are due to TREC bias, and whether the optimum subcollection creation method of using simple initial retrieval or ALM pseudo-relevance feedback at 2 or 3% of total collection cutoff will still be applicable in other instances. Though high recall value at such a low rank (under 3% of the total document collection) is somewhat suspect due to the potential bias introduced by the TREC pooling method of relevant document identification (Voorhees & Harman, 1997), it is still reasonable to think that subcollection IR can be an effective as well as efficient way to deal with the problem of massive document collections.

4 Ad-hoc Experiment

4.1 Research Question

As a natural consequence of our belief that the user is an integral component of a truly effective information retrieval system, our approach to information retrieval centers on various ways to involve the user and then to effectively incorporate the user contribution into the search process. Thus, our main goal of the ad-hoc experiment was to explore methods of preparing the system for such an eventuality. More specifically, we wanted to examine a strategy for creating a subset of a document collection to be used in relevance feedback.

One obvious advantage of using a subcollection is the reduced computational cost. If it can be shown that the retrieval effectiveness of using a subcollection is competitive to that of using a whole collection, then subcollection retrieval may be a desirable strategy when iteratively querying (e.g. relevance feedback, query refinement) a document collection (Sumner & Shaw, 1997) that is massive or composed of distributed collections, where collection statistics for the whole collection are not known. A less obvious advantage of a subcollection might be its increased homogeneity. Being more densely populated with relevant documents that are likely to be topically similar, a subcollection may be more responsive to a refined query than a whole collection with diverse subject matter. For example, "court rulings on the use of peyote" queried against the entire Web document collection may retrieve documents

[7] Three TREC-6 queries that did not have any relevant FT documents were dropped from pre-test experiments.

about *either* courts *or* peyote, whereas the same query submitted to a subcollection of legal documents may boost those documents specifically about court rulings on the use of peyote to the top of the document ranking (Sumner, Yang & Dempsey, 1998).

For these potential advantages to be realized, a subcollection has to be small enough to incur savings in computational cost while at the same time contain enough relevant document to be effective. Thus, we are interested in finding answers to the following questions regarding subcollection retrieval strategy.

- What is the best way to create a subcollection?
- How effective is subcollection retrieval compared to whole collection retrieval?

The focus of our ad-hoc experiment, therefore, was on achieving high recall at some reasonable document rank in order to create an effective and efficient subcollection for relevance feedback.

4.2 Research Design

The constitution of the ad-hoc collection compounds the subcollection question. Since the ad-hoc collection is made up of four document collections, subcollection creation methods can be applied to the document collections separately or as a whole. If subcollection creation methods are applied to individual collections, then the question of how the results should be merged must be addressed. In previous research on this "collection fusion" problem, various strategies were employed to compensate for the potential incomparability of query-document similarity scores across collections (Voorhees, Gupta, & Johnson-Laird, 1995; Savoy, Calve, & Vrajitoru, 1997).

Though the "raw score" merging method can be problematic when collection-dependent term weights (i.e. *idf* weight) cause the retrieval scores of similar documents to vary in different collections (Dumais, 1993; Voorhees et. al., 1995), we thought longer queries and massive retrieval window used for subcollection creation might mute its adverse effects. Any advantages gained by more complex retrieval strategies are likely to have less impact on subcollection creation, whose goal is to retrieve the bulk of relevant documents at an acceptable document rank. Ideally, these assumptions should be empirically tested by experimenting with exhaustive combinations of subcollection creation, collection fusion, and various ad-hoc retrieval strategies, but we decided to test only a few subcollection creation methods for the reasons of practicality and simplicity. One of the overriding factors that influenced our research design in TREC-7 was the machine resource limitations that restricted a large scale experimentation. Besides, we figured if a simple method could create an effective enough subcollection, we could forgo complex methods in favor of a simple one.

Therefore, we chose the two most simple and yet effective subcollection creation methods from the pre-test and applied them to individual collections separately. Subcollection creation methods tested were:

- *unc7aal1*: Collection fusion by raw score merging of the simple initial retrieval results without any feedback.
- *unc7aal2*: Collection fusion by raw score merging of the pseudo-relevance feedback results with the adaptive linear model using the top 5 retrieved documents as relevant and the 100th document as non-relevant.

Both *unc7aal1* and *unc7aal2* were produced by first retrieving 10% of documents in each collection and merging the results by their raw query-document similarity scores.

The second phase of our ad-hoc experiment, which we planned to do in a post-study, involved performing relevance feedback on the subcollections by using the official TREC relevance judgments for the top 20 retrieved documents. The results of relevance feedback on a subcollection would then be compared with that on the whole collection to determine the relative effectiveness of subcollection retrieval.

The system construct for the ad-hoc experiment was based on the system component pre-test results. We used the document term weight of *Lnu* 0.3 to optimize the initial retrieval results and allowed for the full feedback query expansion to maximize the feedback effect. We also heavily weighted the initial query in the starting vector formulation of the adaptive linear model (i.e. $c_0 = 1.0$ in Equation 3) to reduce the adverse effect of the pseudo-relevance feedback. We did not create a phrase index for the ad-hoc experiment since we thought its creation cost in time and machine resources far outweighed any potential benefit gained by using it.

4.3 Results

The TREC-7 ad-hoc collection consists of 130,471 FBIS, 19,842[8] Federal Register, 210,158 Financial Times, and 131,896 LA Times documents. Each document collection was first processed individually to generate single-word indexes of 243,778 terms for FBIS, 117,743 terms for Federal Register, 295,257 terms for Financial Times, and 222,155 terms for LA Times collection.

Unfortunately, we experienced a hard disk problem that corrupted the entirety of TREC-7 data and disabled our main research computer soon after we completed the first phase of our ad-hoc experiment. We turned in the top 1000 retrieved documents produced by subcollection creation runs of initial retrieval (*unc7aal1*) and pseudo-feedback with ALM(*unc7aal2*). We are still in the process of restoring the data and consequently, we were not able to conduct the second phase of our experiment to test the effectiveness of relevance feedback using a subcollection.

According to TREC evaluation measures, which indicate the retrieval performance of the top 1000 documents only, the pseudo-relevance feedback with the adaptive linear model did slightly better than the initial retrieval without feedback, though both runs performed slightly below the median level of all the ad-hoc participants (Table 1). As for the subcollection creation results, the smaller and homogeneous FT collection results still held true for the larger and heterogeneous ad-hoc collection. As can be seen in Table 2,

[8] Using the corrected document tag <PARENT> instead of <DOCNO> reduced the number of Federal Register documents from 55,630 to 19,842.

there was very little difference in average recall between the two runs. In both runs, the subcollections consisting of only two percent (document rank 10,000) of the whole document collection contained over 80% of the relevant documents on the average.

Table 1. Performance Statistics of top 1000 documents

	unc7aal1	*unc7aal2*	*Best**	*Median**	*Worst**
Average Precision	0.1506	0.1618	0.4334	0.1822	0.0009
Number of Relevant Documents Retrieved	2188	2264	3890	2481	79

* Best, Median, Worst of all TREC7 ad-hoc participants

Table 2. Recall at Document Ranks averaged over 50 Queries

Document Rank	Average Recall by *unc7aal1*	Average Recall by *unc7aal2*
1,000	0.57	0.59
5,000	0.75	0.76
10,000	0.80	0.81
15,000	0.84	0.85
20,000	0.85	0.86
40,000	0.90	0.90

Closer examination of individual query results revealed some outlier queries with many relevant documents (e.g. queries 370, 389) or with possibly dissimilar relevant documents (e.g. query 373) that produced recall much below the average. Though it is very conceivable that more complex methods of collection fusion and/or ad-hoc retrieval methods may produce a subcollection with higher recall at a smaller size, whether those methods can push the results of outlier queries beyond the "recall block" remains to be seen.

5 Interactive Experiment

5.1 Research Question

Feedback from IRIS users at large as well as those in TREC-6 include mixed response regarding the complexity of its user interface. Some like the interactive nature of its interface throughout the search process, while others are taken back by the complexity of it. One of the most often mentioned IRIS interface components is its ability to display and modify term weights. Most novice searchers are confused by it, though some like "seeing how the system works" and the opportunity to intervene in the system process.

In addition to the users' ambivalence, there is also the question of how the users' term weight modifications will affect the retrieval result. Certainly, if the user does not understand the significance of term weights and modifies them inappropriately, the search result will be adversely affected. If the system's search direction is amiss and needs to adjusted, however, user intervention by term weight modification might be beneficial.

Another often discussed IRIS component is the relevance feedback interface, especially its three levels of relevance (i.e. "yes," "maybe," and "no," representing relevant, marginally relevant, and nonrelevant). Users like the option of judging a document beyond the dichotomous "relevant' or "nonrelevant," but they are not quite sure what a "marginally relevant" document should be. The question of what makes a document relevant is a fertile ground for research (Schamber, 1991; Barry, 1994). In a prior research, we investigated the relationship between the proportionality and the degree of relevance and found that the number of relevant passages in a document corresponded directly with the degree of relevance awarded to the document by the user (Maglaughlin, Meho, Yang, & Tang, 1998). If the proportionality of relevance is an important factor in determining the relevance of a document, then the relevance levels used by the system should be finely graded to better reflect the user's evaluation of relevance. One method of addressing this aspect of relevance may be to use "passage feedback," where passages instead of documents are used as the unit of relevance feedback.

Based on these observations, we asked the following questions in our TREC-7 interactive experiment.

- Does the display and modification option of term weights in an interactive retrieval system help or hinder the retrieval result?
- Is passage feedback an effective alternative to conventional "document" feedback?

5.2 Methodology

We learned from our TREC-6 interactive experience that it is difficult enough to gauge the effects of various contributing factors in an interactive retrieval experiment without compounding the analysis by introducing numerous system features. Consequently, we attempted to isolate the effect of system features by keeping the experimental and the control system identical except in one aspect.

In one interactive experiment (*unc7iap*), the only difference between the experimental and the control system was the display and modification capability of term weights. In another experiment (*unc7ias*), the difference was relevance feedback by passage versus document. Both experiments used the same control system called "*iriss*" which did not have the term weight display. The experimental system in *unc7iap*, "*irisa*", is essentially the standard IRIS with term weight display used since TREC-6, whereas the experimental system in *unc7ias*, "*irisp*", implements the passage feedback based on the "simple" interface (i.e. without term weight display) [9].

All three systems use the same initial interface, but the initial query modification interface differs in the display of query terms. Though all three systems offer the "suggested phrases" with which the user can supplement the initial query, *iriss* and *irisp* (Figure 1.1) do not have the term weight display where the user can change the term weights as in *irisa* (Figure 1.2). Instead, the term display of the simple interface offers check boxes users can click to include or exclude terms. The feedback query modification interface is structured in the same fashion (Figures 3.1 and 3.2). In addition to the modification of existing terms, the feedback query modification interface allows the user to add terms with emphasis, indicated by the plus or minus symbol (simple interface, Figure 3.1) or by term weights (advanced interface, Figure 3.2).

Another major difference between systems occurs in the relevance feedback interface. Both *iriss* and *irisa* employ the conventional feedback mechanism of judging the relevance of a document as a whole (Figure 2.1), but this document feedback mechanism is replaced by the passage feedback in *irisp* (Figures 2.2). Instead of checking each document as yes, maybe, or no for relevance, the user can simply copy and paste relevant and nonrelevant portions of documents into the appropriate passage feedback box in *irisp*.

The system construct for all three systems was essentially the same except for the mechanism of passage feedback. This is described in the section 2.4.3. Based on the system component pre-test results, we used document term weights of *Lnu* 0.5 to maximize the relevance feedback influence and restricted the feedback query expansion to the 250 terms with highest positive weights and the 50 terms with lowest negative weights in order to optimize the system for efficiency. We also reduced the contribution of the initial query in the starting vector formulation of the adaptive linear model (i.e. $c_0 = 0.2$ in Equation 3) to allow the actions taken during the feedback process to have a stronger influence on the direction of the search. We also created a phrase index of adjacent noun-noun pairs to use in suggesting potentially useful phrases for the initial query as well as in expanding the feedback vectors.

5.3 Results

The performance of IRIS measured by the mean instance recall (MIR) measure was slightly below the median of all interactive track runs (Table 3). Though the term weight display system of *irisa* had the highest MIR of all three IRIS systems tested, the passage feedback system (*irisp*) showed more improvement when performance is compared pairwise within each experimental run. The superior performance of both *irisa* and *irisp* over *iriss* seems to indicate that searcher interventions help rather than hinder the retrieval process. Also, the high relative MIR of *irisp* suggests passage feedback is an effective alternative to conventional document feedback.

Table 3. Interactive Experiment Result Statistics

	iriss vs. irisp		*iriss vs. irisa*		*Best*	*Median*	*Worst*
Mean Instance Recall	0.281	0.314	0.340	0.350	0.420	0.363	0.221

Tables 4.1 and 4.2 show the information about each searcher's background and search experience gathered by pre-study questionnaires. All searchers had received a bachelor's degree and were enrolled in the School of Information and Library Science. Three searchers had previous graduate degrees. The searchers had been searching between 1 and 15 years, with 5 being the average. Four of the 16 searchers were male.

In addition to the pre-search questionnaire, the searchers also completed a psychometric evaluation in an attempt to assess their query formulation skills. The psychometric evaluation scores, which ranged from 11 to 70, were computed by comparing the searcher's synonyms to a list of "correct" synonyms and scoring a point for each correct synonym they recorded. When the searchers' psychometric scores were compared to their average precision and recall values, little correlation was found between search results and psychometric results (Table 4.3). Additionally, there was almost no relationship found between the searchers' performance and their perceived knowledge of the topics, satisfaction with the search, search confidence, the ease of system use, or understanding of the task. Also, there was almost no relationship found between the searchers' performance and their interactions with the systems (Table 4.4). It was however interesting to note that the searchers using the simple and advanced systems, on average, evaluated and saved more documents than those using the simple and passage systems. This may be due to the difficulty of the passage feedback interface.

[9] Due to an oversight, the system names we submitted to NIST were not consistent across experiments. For the sake of clarity and consistency, we have changed the system name mappings in this paper and submitted the corrected mappings to NIST. Eight subjects searched on *irisa* and *iriss*, and eight searched on *irisp* and *iriss*.

Table 4.1 Response Frequency of *irisa/iriss* Searchers on Pre-Study Questionnaire

	No Experience		Some Experience		Great Deal of Experience
1.Using a point-and-click interface				1	7
2.searching on computerized library catalogs			1	3	4
3.searching on CD ROM systems			3	4	1
4.searching commercial systems		3	4	1	
5.searching WWW search services			1	5	2
6.searching other systems	7			1	
	Never	**Once or twice a year**	**Once or twice a month**	**Once or twice a week**	**Once or twice a day**
7.Searching frequency				3	5
	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**
8.Enjoys carrying out information searches				5	3

Table 4.2 Response Frequency of *irisp/iriss* Searchers on Pre-Study Questionnaire

	No Experience		Some Experience		Great Deal of Experience
1.Using a point-and-click interface				1	7
2.searching on computerized library catalogs				6	2
3.searching on CD ROM systems		1	2	4	2
4.searching commercial systems	1	1	4	1	1
5.searching WWW search services				1	7
6.searching other systems	6		1**		1*
	Never	**Once or twice a year**	**Once or twice a month**	**Once or twice a week**	**Once or twice a day**
7.Searching frequency				4	4
	Strongly Disagree	**Disagree**	**Neutral**	**Agree**	**Strongly Agree**
8.Enjoys carrying out information searches			2	5	1

* OCLC ** Military

Table 4.3 Searchers' Average Psychometric Score, Precision and Recall

	irisa/iriss	*irisp/iriss*
Average Psychometric score	35.37	39.25
Average Precision	.7056	.7255
Average Recall	.3447	.2975
Correlation between Psychometric score and Precision	.442	.372
Correlation between Psychometric score and Recall	.308	.646

Table 4.4 Searchers' use of system features

Searchers	System	initial terms	Fbk Iteration	docs saved	docs evaluated	positive terms modified	negative terms modified	positive terms added	Negative terms added
unc7iap	*irisa*	4.857	3.600	15.829	30.629	0.143	0.057	1.543	1.143
unc7iap	*iriss*	4.118	3.118	15.882	30.618	1.353	0.147	1.176	1.176
unc7ias	*iriss*	4.079	2.947	2.921	13.342	0.105	0.000	2.211	1.632
unc7ias	*irisp*	3.898	2.510	2.571	9.245	0.000	0.020	1.102	0.224

References

Barry, C. L. (1994). User-defined relevance criteria: an exploratory study. *Journal of the American Society for Information Science*, 45(3), 149- 159.

Berry-Rogghe, G. (1974). The computation of collocations and their relevance in lexical studies. In A. J. Aitken, R. W. Bailey, & N. Hamilton-Smith (Eds.), *The Computer and Literary Studies* (pp. 103-112). Edinburgh: Edinburgh University Press.

Buckley, C., Salton, G., Allan, J., & Singhal, A. (1995). Automatic query expansion using SMART: TREC 3. In D. K. Harman (Ed.), *Overview of the Third Text REtrieval Conference (TREC-3)* (NIST Spec. Publ. 500-225, pp. 69-80). Washington, DC: U.S. Government Printing Office.

Buckley, C., Singhal, A., & Mitra, M. (1997). Using query zoning and correlation within SMART: TREC 5. In E. M. Voorhees & D. K. Harman (Eds.), *The Fifth Text REtrieval Conference (TREC-5)*.

Buckley, C., Singhal, A., Mitra, M., & Salton, G. (1996). New retrieval approaches using SMART: TREC 4. In D. K. Harman (Ed.), *The Fourth Text REtrieval Conference (TREC-4)* (NIST Spec. Publ. 500-236, pp. 25-48). Washington, DC: U.S. Government Printing Office.

Dumais, S. T. (1993). LSI meets TREC. In D. K. Harman (Ed.), *Proceedings of the First Text REtrieval Conference (TREC-1), 137-152*.

Fishburn, P. C. (1970). *Utility theory for decision making*. New York: John Wiley & Sons.

Frakes, W. B., & Baeza-Yates, R. (Eds.). (1992). *Information retrieval: Data structures & algorithms*. Englewood Cliffs, NJ: Prentice Hall.

Haas, S. W., & Losee, R. M. (1994). Looking into text windows: Their size and composition. *Information Processing and Management*, *30*, 619-629.

Ide, E. (1971). New experiments in relevance feedback. In G. Salton (Ed.), *The Smart System-- experments in automatic document processing* (pp. 337-354). Englewood Cliffs, NJ: Prentice-Hall, Inc.

Krovetz, R. (1993). Viewing morphology as an inference process. *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 191-203.

Losee, R. M. (1994). Term dependence: Truncating the Bahadur Lazarsfeld expansion. *Information Processing and Management, 30*, 293-303.

Lewis, D., Croft, W. B., & Bhandaru, N. (1989). Language-oriented information retrieval. *International Journal of Intelligent Systems, 4,* 285-318.

Maglaughlin, K.L., Meho L., Yang, K., & Tang, R. (1998). Utilizing Users' Relevance Criteria in Relevance Feedback. *Unpublished manuscript.*

Melucci M. (1998). Passage retrieval: A probabilistic technique. *Information Processing & Management*. *34*, 43-68.

Porter, M. (1980). An algorithm for suffix stripping. *Program, 14,* 130-137.

Rocchio, J. J., Jr. (1971). Relevance feedback in information retrieval. In G. Salton (Ed.), *The SMART Retrieval System: Experiments in Automatic Document Processing* (pp. 313-323). Englewood Cliffs, NJ: Prentice-Hall.

Salton, G. (1968). *Automatic Information Organization and Retrieval.* McGraw-Hill.

Salton, G., & Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, *41*, 288-297.

Savoy, J., Calve, A., & Vrajitoru, D. (1997). Report on the TREC-5 experiment: Data fusion and collection fusion. In E. M. Voorhees & D. K. Harman (Eds.), *The Fifth Text REtrieval Conference (TREC-5)*.

Schamber, L. (1991a). Users' criteria for evaluation in a multimedia environment. *Proceedings of the American Society for Information Science*, Washington, DC, (pp. 126-133). Medford. N.J.: Learned Information, Inc.

Shaw, W. M., Jr. (1986). On the foundatin of evaluation. *Journal of the American Society for Information Science, 37*, 346-348.

Singhal, A., Buckley, C., & Mitra, M. (1996). Pivoted document length normalization. *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 21-29.

Sumner, R. G., Jr., & Shaw, W. M., Jr. (1997). An investigation of relevance feedback using adaptive linear and probabilistic models. In E. M. Voorhees & D. K. Harman (Eds.), *The Fifth Text REtrieval Conference (TREC-5)*.

Sumner, R. G., Jr., Yang, K., Akers, R., & Shaw, W. M., Jr. (1998). Interactive retrieval using IRIS: TREC-6 experiments. In E. M. Voorhees & D. K. Harman (Eds.), *The Sixth Text REtrieval Conference (TREC-6)*.

Sumner, R. G., Yang, K., & Dempsey, B. (1998). Interactive WWW search engine for user-defined collections. *Digital 98 Libraries: The Third ACM Conference on Digital Libraries.* 307-308.

Voorhees, E., Gupta, N. K., & Johnson-Laird, B. (1995). The Collection fusion problem. In E. M. Voorhees & D. K. Harman (Eds.), *Overview of the Third Text REtrieval Conference (TREC-3)*.

Voorhees, E., & Harman, D. (1997). Overview of the Fifth Text Retrieval Conference. In E. M. Voorhees & D. K. Harman (Eds.), *The Fifth Text REtrieval Conference (TREC-5)*.

Wong, S. K. M., & Yao, Y. Y. (1990). Query formulation in linear retrieval models. *Journal of the American Society for Information Science*, *41*, 334-341.

Wong, S. K. M., Yao, Y. Y., & Bollmann, P. (1988). Linear structure in information retrieval. *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 219-232.

Wong, S. K. M., Yao, Y. Y., Salton, G., & Buckley, C. (1991). Evaluation of an adaptive linear model. *Journal of the American Society for Information Science*, *42*, 723-730.

Yang, K., & Yang, K. (1997). Graded relevance in information retrieval. *Unpublished manuscript.*

Figure 1.1 Initial Query Modification Interface for *iriss* and *irisp*

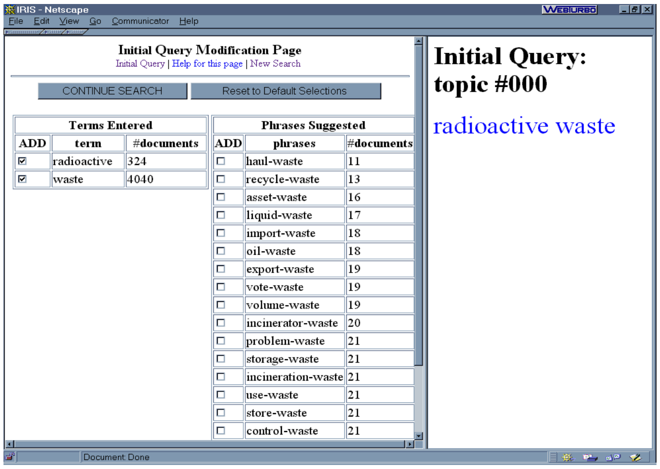

Figure 1.2 Initial Query Modification Interface for *irisa*

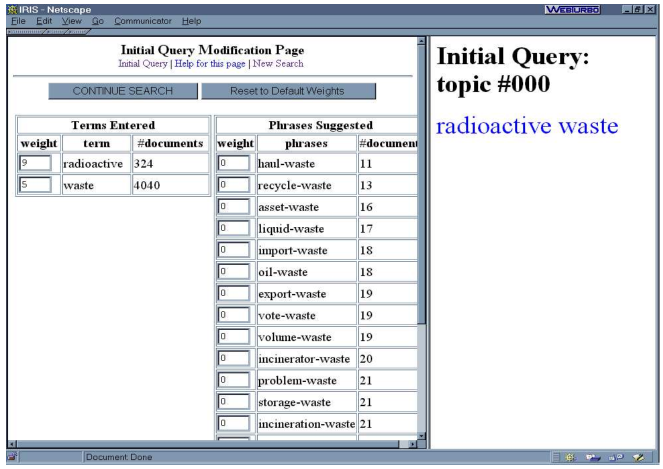

Figure 2.1 Relevance feedback interface for *iriss* and *irisa*

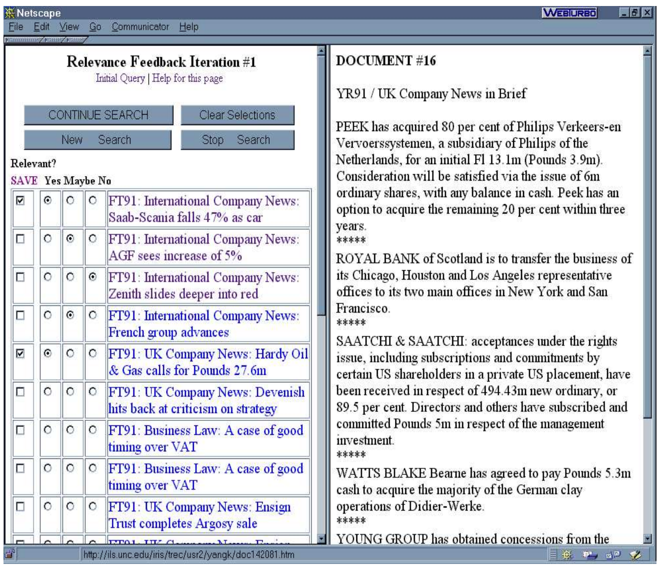

Figure 2.2 Relevance Feedback Interface for *irisp*

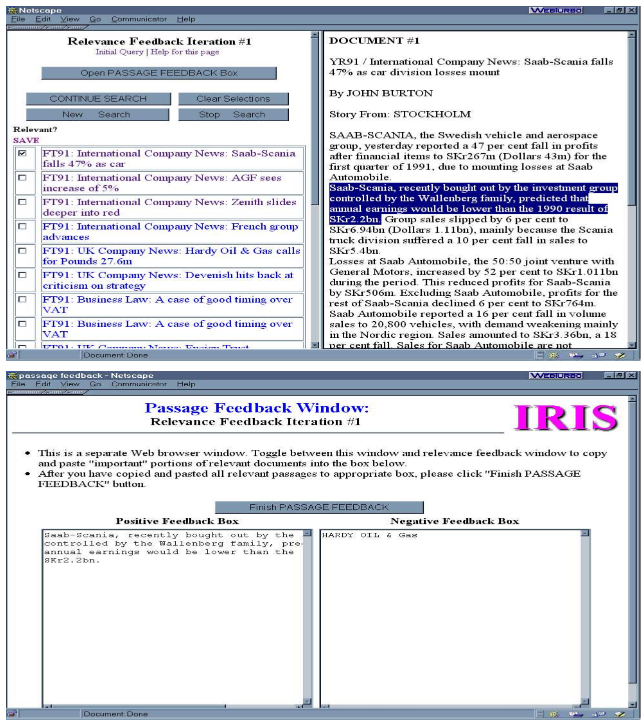

Figure 3.1 Feedback Query Modification Interface for *iriss* and *irisp*

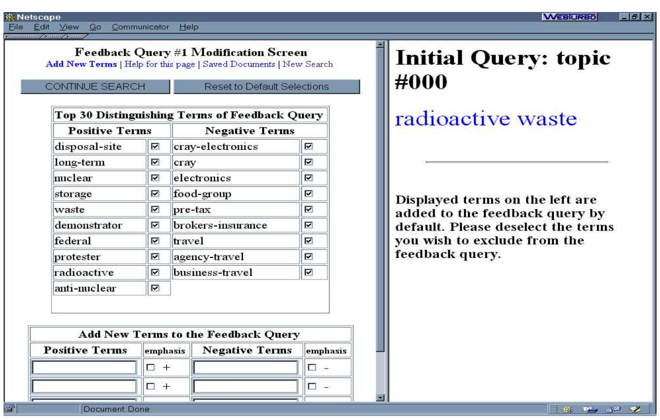

Figure 3.2 Feedback Query Modification Interface for *irisa*