TREC2 Document Retrieval Experiments using PIRCS

K.L. Kwok & L. Grunfeld

Computer Science Dept., Queens College, CUNY
Flushing, NY 11367. email: kklqc@cunyvm.cuny.edu

Abstract

We performed the full experiments, using our network implementation of component probabilistic indexing and retrieval model. Documents were enhanced with a list of semi-automatically generated two-word phrases, and queries with automatic Boolean expressions. An item self-learning procedure was used to initiate network edge weights for retrieval. Initial results submitted were above median for ad hoc, and below median for routing. They were not up to expectation because of a bad choice of high-frequency cut-off for terms, and no query expansion for routing. Later experiments showed that our system does return very good results after correcting the earlier problems and adjusting some parameters. We also re-design our system to handle virtually any number of large files in an incremental fashion, and to do retrieval and learning by initiating our network on demand, without first creating a full inverted file.

1. Introduction

In TREC1 our system called PIRCS (acronym for Probabilistic Indexing and Retrieval -Components- System) took part as Category B participant, handling only the 0.5 GB Wall Street Journal collection because both our software and hardware were not sufficient for the full set of text files. In TREC2, we participated in category A. However, during a large portion of the time period we have to face fairly uncertain and sometimes difficult conditions. Plans to install a dedicated SPARC10 workstation and associated large memory and disk drives did not materialize until about three weeks from the deadline. Before this period, the SPARC2 workstation that we have been using was also shared with other users during the semester. Certain things that we wished to do were not done, and corners were cut to fit programs and data in the existing system. Much of our time was spent revamping our software to be more efficient in space and time utilization. Our focus remains as in TREC1, namely, to improve representations of documents and queries, to test different learning methods and to combine different retrieval methods to improve final ranked retrieval output. Section 2 summarizes our retrieval network; Section 3 discusses our improved system design; Section 4 is on item representation; Sections 5&6 are about our learning and retrieval procedures; Section 7 discusses the results we submitted and Section 8 contains results of our later experiments. Section 9 follows with the conclusion.

2. A Retrieval Network in PIRCS

Our retrieval process is based on a three layer Q-T-D (Query-Term-Document) network, details of which are given in [KwPK93,Kwok9x]. Here we give a review. From Fig.1, DTQ query-focused retrieval means spreading an initial activation of 1 (one) from a document d_i towards query q_a and gated by intervening edges w_{ki} and w_{ak}. The resultant activation received at q_a is $W_{i/q} = \Sigma_k\ w_{ak}*w_{ki}$, and is the retrieval status value (RSV) of d_i with respect to q_a. When activation initiated at q_a spreads towards d_i, we obtain activation received at d_i equals to $W_{i/d} = \Sigma_k\ w_{ik}*w_{ka}$, and is our QTD document-focused RSV for d_i . Combining the two additively: $W_i = W_{i/q} + W_{i/d}$ gives our basic retrieval ranking function. Edge weights w_{ka}, w_{ki} represent items (q_a or d_i) acting on terms t_k and reflect usage of terms within items. Edge weights w_{ak}, w_{ik} (representing t_k acting on q_a or d_i) embed Bayesian inference and are initialized based on a component consideration of probabilistic indexing and retrieval. These weights can improve via a learning process when relevant documents are known to queries and vice versa: DTQ query-focused training when we know the set of documents and their components relevant to a query, and QTD document-focused training when we know the set of queries and their components relevant to a document. Query-focused training prepares queries to match new similar documents better, while document-focused training helps documents to match new similar queries better. With learning capability, the net behaves and can be viewed as a superposition of two 2-layer direct-connect artificial neural networks, one in each direction. If a boolean expression for query q_a is known, it can also be represented as a tree and hung onto the net as shown in Fig.2. Edge weights from

the net are used to initialize the tree leaf nodes, from which activation spreads to the query root node. Processing at the nodes implements soft-Boolean evaluation [SaFW83] in the DTQ direction resulting in a third RSV: S_i. The RSVs are latter combined for ranking retrieval outputs.

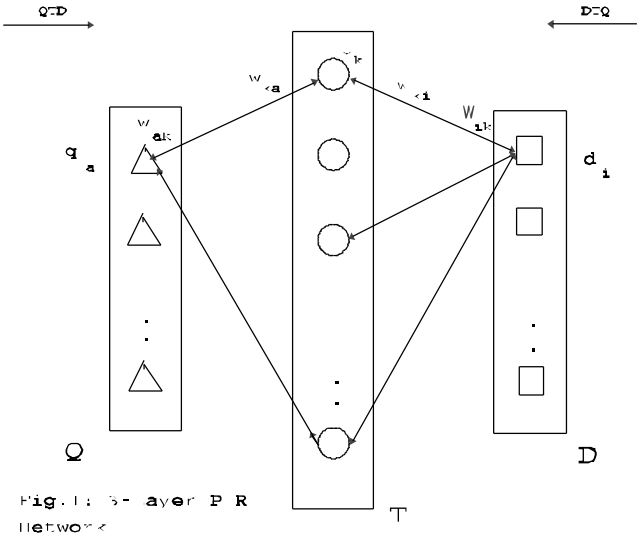


Fig.1: 3- ayer P R Network

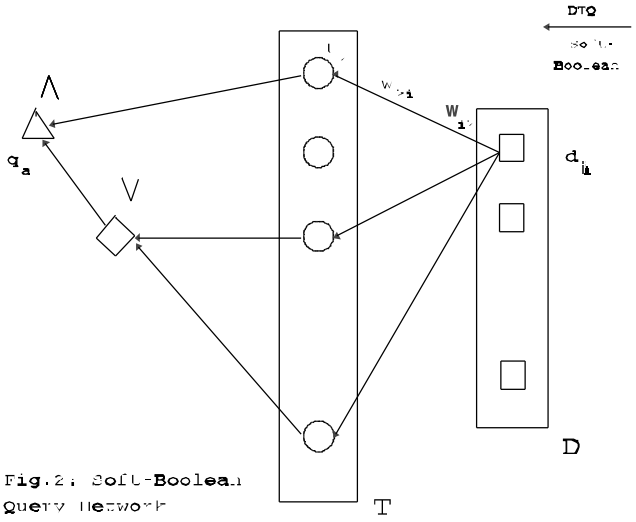
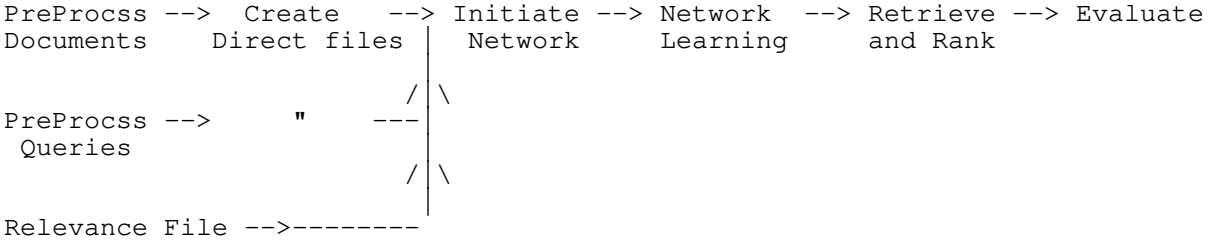


Fig.2: Soft-Boolean Query Network

3. System Design

Our previous software for TREC1 has extraneous processing that produces several intermediate files for other purposes. These consume a lot of disk space for large scale collections. We spent a substantial amount of time to revamp our system, resulting in a more streamlined flow-chart as follows:



In addition, we also took the opportunity to re-design our system resulting in several innovative approaches described in the following subsections:

3.1 Full Inverted File Eliminated

It is useful to view a textbase as a document by term matrix. If one stores the matrix rowwise, we call it a direct file, in contrast to an inverted file which stores the matrix columnwise. The inverted file is useful to support fast retrieval while the direct file is useful for feedback learning and query expansion when given certain documents being relevant to certain queries. In addition, the raw textfile is useful for display purposes after a retrieval ranked list is produced. If one assumes that each of these three files are approximately equal in size of N bytes, we need a minimum of 3N bytes, which is quite substantial. Removing the raw text may not win user support during display unless the direct file encodes all stopwords, punctuations and paragraph structures of the original. Most systems delete the direct file and re-produce a subset of it from raw text when needed. This results in a requirement of 2N bytes. We choose, however, to keep the direct file and produce our network with respect to the queries dynamically as needed without first producing an inverted file first. Our network actually contains both direct and 'inverted' data. It resides in memory to support learning and retrieval. By this

approach we achieve several objectives for our system: a) reduce the 'dead' time between a collection being acquired and its availability for searching, since the full inverted file is not produced; b) satisfy the 2N bytes of space requirement; c) support fast feedback learning and query expansion by having the direct file available. The price we pay is to have to create the network dynamically, unlike the full inverted file which is produced once. However, since the full inverted file is too large to reside in memory, reading parts of it in for retrieval is also time consuming.

3.2 Reduced Network Size

We produce our network in memory according to the queries under attention and the terms used in them. Since memory is limited, documents are divided into subcollections (Section 3.3) and queries are linked in five to ten at a time. We define the active term set (ATS) of a network as the set of terms used by the current queries and their feedback documents if any. The latter will provide terms for query expansion. Only edges that connect items to the active term set are initiated in the network, reducing network space requirements substantially. With the current implementation for 1GB subcollection and 7 queries, the node and edge files together take about 40 MB. Using clock time as a measure, producing the network requires about 40 minutes. Learning is fast, about 2 minutes, and retrieval and ranking another 8 minutes. We hope that further improvements in design and faster hardware in the future can improve these figures substantially.

3.3 Master and Subcollection File Structure

We view the TREC experiments as document retrieval from multiple collections, but reporting retrieval results in one single ranked list of documents for each topic (query). Although only four or five collections such as Wall Street Journal, Associated Press, etc. are given in TREC, in reality it could be many more. We consider three methods in file design to approach the problem:

a) A centralized approach where all documents from all collections are processed as if from a single document stream, producing a centralized dictionary containing full usage statistics and a giant direct file. From these, networks can be initialized. The idea is simple, but it has drawbacks in that eventually the direct file would exceed file/disk size limitations and software has to be designed to handle data crossing file/disk boundaries. Moreover, it is inherently fragile to create single files of this size. The advantage of this approach is that RSVs calculated are directly comparable for all documents and a single ranked list is produced without difficulty.

b) An independent collections approach where each source collection of about 0.5-1.0 GB say, forms a textbase with its own local dictionary and direct file, for network initiation and retrieval. One simply repeats the process for as many collections as necessary. This is the preferred approach, and if one has n processors and sufficient disk space, n separate textbases can be created for learning and retrieval in parallel saving substantial time. The problem is how to combine the retrieval lists from each into a single ranked list, since each textbase has its own term usage statistics and calculates RSVs for ranking within its own environment. Classical Boolean retrieval and coordinate matching pose no problem. Some retrieval strategy may produce RSVs that are comparable across collections in theory; but after approximations are taken, it is questionable that this is still true. Similar problem exists for retrieval from distributed databases such as the WAIS environment.

c) For TREC2 we settle on a hybrid subcollections approach, treating each source as a subcollection within a master. We create a master centralized dictionary as in a) capturing full usage statistics serving all the subcollections, and create separate direct files for each subcollection as in b). The central dictionary has about 620,000 unique terms after processing 2 GB from Disk1 and Disk2, and is relatively small. It captures global term usage statistics, while the individual direct files capture local usage statistics within items. Separate networks are then created for each subcollection with edge weights based on the correct global and local statistics as in a), assuring that retrieval lists contain RSVs that are directly comparable. This approach combines the advantages of both a) and b), and can also function in a parallel distributed environment.

4. Item Representation

As in TREC1, a number of preprocessing mainly for the purpose of improving the representations of documents and queries are done as follows:

4.1 Vocabulary Control

In addition to a manual stopword list of about 630 words and another 528 manually identified 2-word phrases, we also process samples from Disk1 and Disk2 using all five source types (WSJ, AP, FR, ZIFF and DOE, of about 100MB each) to produce two-word phrases based on adjacency within sentence context. Our objective is to remedy losses of recall and precision due to the removal of high frequency terms. Our criteria for phrases is that each word pair must have a frequency of 40, and either one or both components must be high frequency (>=10000). A casual scan of the resultant list generated led us to remove

some obvious non-content phrases (such as 'author describ', paper attempt', 'two way'). The result is 13,787 pairs. These plus the previously prepared manually list (which contains mostly of phrases with at least one stopword as well as some phrases identified in the 'topics') are then treated as if they were single index terms to be identified during document and query processing [BuSA93]. They have their own global and local usage statistics, and can improve individual collection retrieval effectiveness from a few to 10%.

After documents are processed, we invoke Zipf's law to remove low (=4) and high frequency (=16000) words from being used for representation. Low frequency terms lead to too many nodes, and high frequency terms lead to too many edges, both consuming valuable memory space. Unfortunately, our high frequency cut-off of 16000 was a bad choice. In fact, we used high = 12000 for routing because at that earlier period, we were short of resources. The effect is that our queries become too short and many useful terms (such as 'platform' in query #80, 'crimin' for criminal in query #87, etc.) are screened out. We discover that this is a major factor in our disappointing results. Later experiments use a high cut-off of 50000.

4.2 Subdocuments

As in TREC1 we segment documents into subdocument units to deal with the problems of WSJ documents having multiple unrelated stories, and long documents in general. A really long document is FR89119-0111 which has 400748 words. Our criteria is to break documents either on story boundaries or on the next paragraph boundary after a run of 360 words for all collections. We have not found convenient story boundary indicators in other collections as in WSJ (which uses three dashes '---' on a line). With this scheme, the total number of subdocuments from Disk 1&2 becomes 1,281,233 compared with an original 742,611.

After the TREC2 deadline, we have the resources to investigate the effects of subdocuments on retrieval. Experiments were performed on individual subcollections WSJ1, FR2, FR1 and AP2, using segmentation sizes of 360, 720 and 1080 words. For WSJ1, we further break on story boundaries only. Results are tabulated in Table 1. It appears that for the abnormally long documents of FR, breaking into subdocuments is definitely worthwhile, achieving improvements of over 20% compared with no segmentation. However, for the newswire documents of

	AsIs	Break on "Stories"	1080	720	360
WSJ1					
Av11	0.421	0.432	0.428	0.424	0.418
# of docs	98733	127151	134819	149611	193881
FR1					
Av11	0.289		0.351	0.354	0.354
# of docs	26207		50055	64650	108374
FR2					
Av11	0.333		0.372	0.420	0.421
# of docs	20108			51607	86787
AP2					
Av11	0.423		0.423	0.404	0.414
# of docs	84678		85616	95867	146354

Table 1: Document Segmentation Av11 Precision using 50 Queries Q2

WSJ1 and AP2, subdocuments have marginal performance effects: a little better for large chunk sizes, and a little worse for small chunks. It seems that a chunk size of about 720-1000 words would get the benefits of both types. Using different chunk sizes for different collections would probably not be worth the effort.

We like to point out that other than effectiveness, considerations such as isolating relevant sections for output display and for more precise feedback judgment would also make document segmentation worthwhile. In particular, a number of long documents in feedback for query expansion would easily overload memory space in our network.

4.3 Queries

Topics are preprocessed to remove introductory phrases and

non-content terms based on word sequence patterns. We continue to use words from the title, description, narrative and concept sections of the topics to form queries. Experiments without the narrative section show slight decrease in performance for our system in contrast to [Crof93]. We also try to produce automatically Boolean expressions as queries from the description and concepts sections. This is done by using punctuations to delineate phrases and ANDing the words within them. Phrases are then ORed. This is a very crude way of getting Boolean expressions for later soft-Boolean retrieval processing.

5. Learning Procedures

In our network of Fig.1 the edge weights determine the retrieval outcome. w_{ki} and w_{ka} captures the proportion of term t_k in d_i or q_a. They are fixed and obtained from the manifestation of terms in the respective items. w_{ak} (and w_{ik}) has a log odds factor, and an Inverse Collection Term Frequency (ICTF) factor which is regarded as a constant for a term t_k, as follows:

$$w_{ak} = \ln [r_{ak}/(1-r_{ak})] + ICTF_k \qquad (1)$$

Here r_{ak} is the conditional probability that given relevance to q_a that term t_k occurs, and needs to be learnt. It is unknown unless one has a sample of relevant documents to q_a. This is not applicable to initial ad hoc retrievals where a document collection is being processed against a new query with no known results. Relevance feedback information can remedy this later, but it is not available at the beginning. One way is to ignore the log odds factor in Eqn.1, as done in our TREC1 experiments, resulting in ICTF weighting. A better way, which we use for TREC2, is to include item self-learning to determine r_{ak} and initiate the term weights. This is shown in Fig.3 and is based on the following argument. Consider a document d_i containing certain concepts and topics. Imagine this author wishing to inquire the textbase for the same topics as this document s/he has written; what query would be most suitable? Naturally the author's own words in the document can serve as the 'query', and there is also known to be one relevant item to this 'query' in the collection, viz. the document itself. In other words, every item is assumed to be self-relevant. One relevant item is however not sufficient for estimating r_{ak}. Our method is to consider each document as constituted of many independent conceptual components each being described by a list of terms. We therefore work in a component universe rather than in the document collection. Components can be units such as sentences or phrases, but we have used single terms for simplicity. Right from the start then, even without any relevance feedback, we can divide the component universe into two parts: one set relevant, and the other non-relevant to each the 'query'. Standard probabilistic retrieval theory now enables us to define this 'query' optimally -- meaning that the defined 'query' will rank its set of relevant components optimally with respect to the other components when used for retrieval. The definition of this 'query' (i.e. its terms and weights) becomes the initial indexing representation of the document, and it is used in our ad hoc QTD retrieval. This is the principle of document self-recovery introduced in [Kwok90] and implemented as a self-learning process in a network [Kwok89,9x] shown in Fig.4. One can argue that this relevant set of components from one document is too small. But that is all the information one has at this stage. Previous experiments [Kwok90] show that this kind of weighting can outperform ICTF weight by a few percent. Moreover, our network self-learning parameters can be adjusted to provide a smooth transition from ICTF to full self-learn weights, or any value in between. We invoke the

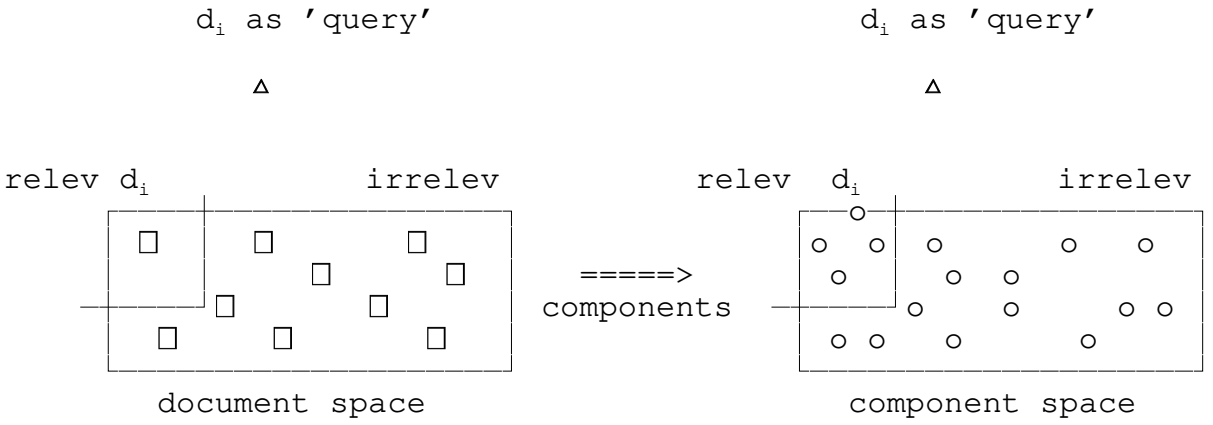


Fig.3 Item Self-Learn Using Its Own Self-Relevant Components

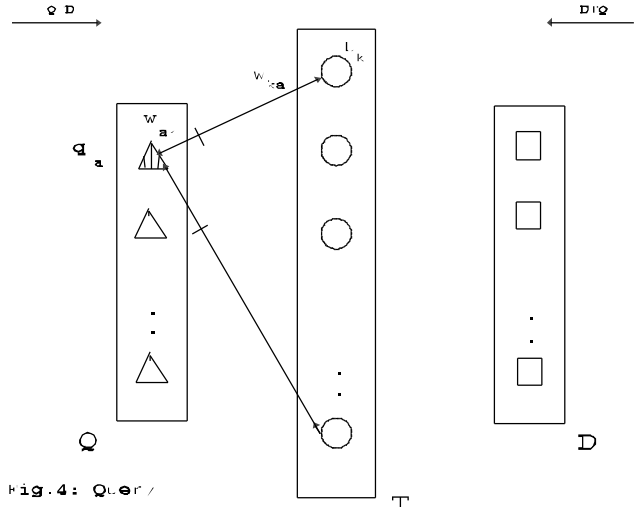


Fig.4: Query Self-learning

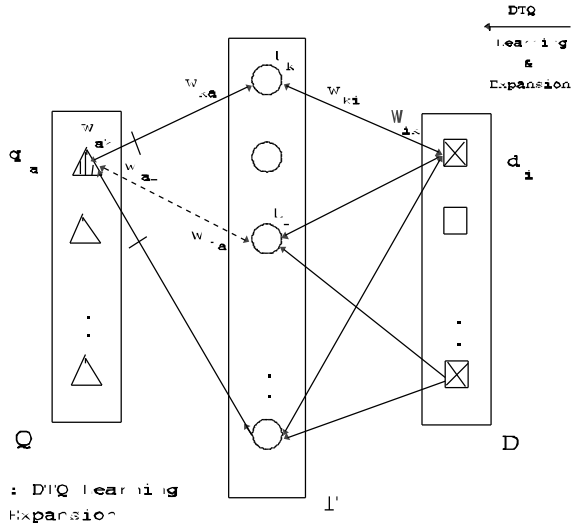


Fig.5: D'Q learning with Expansion

same self-learning procedure for a query by adding the query to the collection as a 'document' temporarily, resulting in self-learn initial weights for the indexing representation of the query. This weight is used in our ad hoc DTQ retrieval. The bottom line is that this component consideration enables the probabilistic model to self-bootstrap, allows term frequencies in items to be employed instead of 'binary', and takes into account of query-focused and document-focused retrieval in a cooperative fashion.

In the case of routing retrieval, relevant documents are available for training each topic q_a. These documents are again considered as constituted of components, and are used in the network to estimate r_{ak}. The estimate should be better than self-learning because the sample size of relevant components is much larger. Our learning algorithm updates the conditional probability r_{ak} (and r_{ik}) as follows (Fig.5):

$$\Delta r_{ak} = \eta_Q * (x_k - r_{ak}^{old}) \qquad (2)$$

Here, η_Q is a learning rate for training on the query side and x_k is the average activation deposited on term t_k by the given relevant set. If a term t_l not in the original query is highly activated by the relevant set, it can also be linked to q_a with edge weights given by:

$$w_{la} = \alpha * x_l$$
$$r_{al} = \beta * \eta_Q * x_l \qquad (3)$$

This implements query expansion as was also done in TREC1.

6. Retrieval Methodology

To satisfy TREC2 requirements, we submitted results as named in the following:

pircs1: routing, no training;
pircs2: routing, with training from Disk1 relevants, Disk2 not used and no query expansion;
pircs3: ad hoc, no soft-Boolean added;
pircs4: ad hoc, with soft-Boolean added.

Routing allows training the old Q2 topic set (topics 51-100) before doing retrieval on new Disk3 documents. Disk3 term usage statistics are not used. Ad hoc retrieval involves using new Q3 topic set (topics 101-150) to do retrieval on old documents in Disk1 and Disk2. All our queries are automatically constructed. We did not perform feedback experiments using Q3.

For baseline routing (pircs1) and ad hoc (pircs3) retrievals, we use the item self-learn (SL) edge weights. Routing pircs2 denotes retrieval based on further learning from known relevant samples and represent improvements from pircs1. There can be hundreds of known relevants for each of the Q2 topic set from documents in Disk1 and Disk2 as given from the results of TREC1. One way of employing them is to do a retrieval (ranking) of Disk1 and Disk2 documents, and then make use of the first n (say n=100) relevant documents, as for feedback learning. However, we did not have enough resources to create a network and do retrieval (ranking) on 2GB of documents at that time and have to settle on a simplified strategy. First, we decided to use Disk1 only (1 GB) for training. Secondly, we believe

a retrieval (ranking) of Disk1 is still expensive and perhaps not necessary; rather we just select 'nonbreak' relevants from Disk1 for training. 'Nonbreak' means documents that do not get split into multiple subdocuments based on our criteria given in Section 3. The idea is that the quality of documents for training is important, and short relevants are the choice. They may not be those ranked early during a retrieval. With these simplifications, a network is produced with Disk1, the query-term edges are trained, and then stored for later routing retrieval using Disk3.

Ad hoc pircs4 denotes retrieval based on combining the baseline pircs3 with a soft-boolean retrieval. The pircs4 ranking formula becomes $r*W_i+s*S_i$ (see definitions in Section 2). Our Boolean expressions for queries are produced automatically as discussed in Section 4.3, and edge weights are used to initiate the leaf nodes of the Boolean expression tree.

7. Discussion of Submitted Results

From our routing retrieval table in the master appendix of this volume, it can be seen that pircs2 improves over pircs1 by about 7% based on average non-interpolated precision (.266 vs .249) and about 3.8% based on relevants retrieved (6135 vs 5913), showing that our simplified method of using only the Disk1 'nonbreak' training documents still works. We did not do a retrieval and rank. Compared with the other sites, our result is below median both using the average non-interpolated precision for individual queries (18 better, 2 equal and 30 below median), and using the relevants retrieved at 100 documents (18 better, 8 equal and 24 below median). If we assume the existence of an overall 'maxi-system' that produces the best non-interpolated precision values among all sites for all 50 queries, then its average precision over all queries is 0.5054 and 8348 relevants retrieved. Our pircs2 achieves only .266/.505 = 52.7% of the average precision but 6135/8348 = 73.5% of the relevants retrieved.

From the ad hoc retrieval table in the appendix of this volume it can be seen that pircs4, which is pircs3 combined with automatic soft-Boolean retrieval, improves over pircs3 only by about 1%. Processing time however increases substantially. Our automatic Boolean expressions are crudely formed; manual Boolean queries may do better. Compared with other sites, our result is above median both using the average non-interpolated precision for individual queries (34 better, 2 equal and 14 below median), and using the relevants retrieved at 100 documents (36 better, 4 equal and 10 below median). The 'maxi-system' has an average precision over all queries of 0.4354 and 9027 relevants retrieved. pircs4 achieves about 0.298/0.435 = 68.5% of this best precision value and 7464/9027 = 82.7% of the relevants retrieved. They are much better than for routing. It would be most useful and interesting if one can choose the best reported result for each query before the answers are known. For these experiments our high frequency term cut-off is 16000, which is still too low. The next Section discusses our later results.

8. Further Experimental Results

After the TREC2 Conference, we decided to repeat both experiments. We realize that our disappointing results are due to several factors: 1) bad high frequency term cut-off leading to insufficient representation; 2) no query expansion; 3) insufficient training samples; and 4) parameters need tuning. Except for 4) these are remedied as follows: high-frequency cut-off is set at 50000, learning for routing is done from both disk1 and disk2 and only documents that 'break' into six or less subdocuments are used, and query expansion is also done. The runs are named in Table 2 as:

pircs5: routing, with learning but no query expansion;
pircs6: routing, query expansion level of 20;
pircs7: routing, 'upperbound', no expansion;
pircs8: ad hoc without Boolean queries.

As in TREC1, our query expansion level of 20 actually adds less than 20 terms because some of the top-ranked terms may already appear in the query. It can be seen that results are substantially better than those in Section 7. In particular, pircs6 routing with query expansion have average precision of 0.355 and the number of relevants retrieved are 7476 out of 10489. These are 12% and 5% respectively better than pircs5 (0.318, 7098): routing with learning but no query expansion, and achieving 70.3% and 89.6% of the maxi-system values. The same average precision value and relevants retrieved for ad hoc retrieval pircs8 are 0.344 and 8279 out of 10785, representing 79% and 91.7% of the ad hoc maxi-system respectively. At 20 documents retrieved, the precision values for routing and ad hoc are respectively 0.583 and 0.564. This means that averaging over 50 queries, out of the first 20 retrieved over 11 are relevant. Considering the size of these textbases, these are quite good results. These numbers are user-oriented, and users naturally hope to see 100% precision. As discussed in TREC1, from a system point of view the precision at n documents retrieved should not be compared to the theoretical value of 1.0, but to an operational precision value x/n if the total number of relevants x for a query is less than n. For example, at n=100 documents retrieved 20 routing and 16 ad hoc queries have total relevants x less than 100. The operational maximum precision averaged over 50 queries for routing is only 0.8, and that for ad hoc is 0.871. At 100 documents, routing pircs6 value of 0.439 and ad hoc pircs8 value of 0.468 therefore achieves 54.9%

	---------- Revised Routing ----------					New Ad hoc
	pircs5	**pircs6**	**%**	**pircs7**	**%**	**pircs8**
	---	---	---	---	---	---
Total number of documents over all 50 queries						
Retrieved:	50000	50000		50000		50000
Relevant:	10489	10489		10489		10785
Rel_ret:	**7098**	**7476**	**+ 5.3**	**7385**	**+ 4.0**	**8279**
Interpolated recall - precision averages:						
0.0	.765	.814	+ 6.4	.793	+ 3.7	.823
0.1	.554	.628	+13.3	.594	+ 7.2	.561
0.2	.491	.546	+11.2	.534	+ 8.6	.505
0.3	.421	.469	+11.4	.469	+11.4	.456
0.4	.380	.413	+ 8.7	.416	+ 9.5	.417
0.5	.336	.363	+ 8.0	.368	+ 9.5	.368
0.6	.270	.310	+14.8	.314	+16.3	.320
0.7	.212	.240	+13.2	.241	+13.7	.246
0.8	.150	.165	+10.0	.171	+14.0	.160
0.9	.079	.099	+25.3	.098	+24.0	.087
1.0	.014	.006	-57.1	.015	+ 7.1	.012
Average precision (non-interpolated) over all rel docs						
	.318	**.355**	**+11.6**	**.350**	**+10.1**	**.344**
Precision at						
5 docs:	.600	.660	+10.0	.624	+ 4.0	.612
10 "	.582	.632	+ 8.6	.598	+ 2.7	.572
15 "	.545	.617	+13.2	.572	+ 5.0	.573
20 "	.527	.583	+10.6	.563	+ 6.8	.564
30 "	.507	.553	+ 9.1	.534	+ 5.3	.540
100 "	**.402**	**.439**	**+ 9.2**	**.427**	**+ 6.2**	**.468**
200 "	.334	.360	+ 7.8	.353	+ 5.7	.396
500 "	.222	.238	+ 7.2	.232	+ 4.5	.264
1000 "	.142	.150	+ 5.6	.148	+ 4.2	.166
R-precision						
Exact	**.358**	**.385**	**+ 7.5**	**.385**	**+ 7.5**	**.378**

Table 2: Revised Routing and New Ad Hoc Retrieval Results

and 53.7% of the operational maximum. The R-precision Exact calculates the precision value at x retrieved, where x is the known number of relevants for each query, and can be compared with the theoretical value of 1.0.

The 'Upperbound' retrieval pircs7 (suggested by Sparck-Jones) means performing learning from the known Disk3 (not Disk 1&2) relevant documents before retrieval. In other words, we assume the answer documents are known for training and represent the best that probabilistic theory can provide using our system. This is however not the true upperbound [Spar79] for retrieval from Disk3, because the vocabulary and usage statistics are still those of Disk 1&2. The vocabulary is retained for comparison with routing results. pircs7 retrieval achieves average precision of 0.350, which improves over pircs5 (training from Disk 1&2) by about 10% in average precision and about 4% in relevants retrieved. Of course in real life, the answer documents are not known; but it is interesting to note that query expansion using Disk 1&2 documents can provide similar performance, showing the importance of query expansion.

We later concentrate on routing and discover that additional gains can be achieved by fine tuning of the parameters in our model. For learning: 1) we find that our original method of using only 'nonbreak' documents in the given set of relevants actually outperforms other document selection strategies including using all relevants, 'break six' or

	New Routing Results										
	No Train.	**Trained Exp 0**	**%**	**Trained Exp 20**	**%**	**Trained Exp 40**	**%**	**Trained Exp 80**	**%**	**Trained Exp 100**	**%**
Total number of documents over all 50 queries											
Retrieved:	50000	50000		50000		50000		50000		50000	
Relevant:	10489	10489		10489		10489		10489		10489	
Rel_ret:	**6551**	**6961**	**+ 6.0**	**7496**	**+14.0**	**7646**	**+17.0**	**7695**	**+17.0**	**7712**	**+18.0**
Interpolated Recall - Precision Averages:											
0.00	0.7200	0.7480	+ 4.0	0.8471	+18.0	0.8475	+18.0	0.8646	+20.0	0.8660	+20.0
0.10	0.5124	0.5815	+13.0	0.6645	+30.0	0.6751	+32.0	0.6810	+33.0	0.6801	+33.0
0.20	0.4431	0.5254	+19.0	0.5981	+35.0	0.6116	+38.0	0.6135	+38.0	0.6115	+38.0
0.30	0.4016	0.4728	+18.0	0.5371	+34.0	0.5413	+35.0	0.5465	+36.0	0.5452	+36.0
0.40	0.3486	0.4402	+26.0	0.4751	+36.0	0.4774	+37.0	0.4829	+39.0	0.4878	+40.0
0.50	0.2970	0.3862	+30.0	0.4167	+40.0	0.4288	+44.0	0.4229	+42.0	0.4214	+42.0
0.60	0.2382	0.3048	+28.0	0.3496	+47.0	0.3681	+55.0	0.3699	+55.0	0.3690	+55.0
0.70	0.1945	0.2430	+25.0	0.2772	+43.0	0.2880	+48.0	0.2815	+45.0	0.2843	+46.0
0.80	0.1284	0.1865	+45.0	0.1911	+49.0	0.1937	+51.0	0.1999	+56.0	0.2005	+56.0
0.90	0.0740	0.0860	+16.0	0.1130	+53.0	0.1144	+55.0	0.1219	+65.0	0.1238	+67.0
1.00	0.0119	0.0187	+57.0	0.0140	+18.0	0.0107	-10.0	0.0114	- 4.0	0.0171	+44.0
Average precision (non-interpolated) over all rel docs											
	0.2905	**0.3517**	**+21.0**	**0.3962**	**+36.0**	**0.4050**	**+39.0**	**0.4084**	**+41.0**	**0.4095**	**+41.0**
Precision at:											
5 docs:	0.5600	0.5760	+ 3.0	0.6960	+24.0	0.7160	+28.0	0.7320	+31.0	0.7280	+30.0
10 docs:	0.5440	0.5820	+ 7.0	0.6880	+26.0	0.6860	+26.0	0.6980	+28.0	0.7000	+29.0
15 docs:	0.5173	0.5627	+ 9.0	0.6573	+27.0	0.6707	+30.0	0.6800	+31.0	0.6813	+32.0
20 docs:	0.4910	0.5510	+12.0	0.6470	+32.0	0.6540	+33.0	0.6630	+35.0	0.6610	+35.0
30 docs:	0.4653	0.5313	+14.0	0.6147	+32.0	0.6173	+33.0	0.6240	+34.0	0.6267	+35.0
100 docs:	**0.3698**	**0.4396**	**+19.0**	**0.4824**	**+30.0**	**0.4930**	**+33.0**	**0.4974**	**+35.0**	**0.5002**	**+35.0**
200 docs:	0.3049	0.3562	+17.0	0.3887	+27.0	0.3945	+29.0	0.4002	+31.0	0.4004	+31.0
500 docs:	0.2038	0.2241	+10.0	0.2452	+20.0	0.2490	+22.0	0.2500	+23.0	0.2498	+23.0
1000 docs:	0.1310	0.1392	+ 6.0	0.1499	+14.0	0.1529	+17.0	0.1539	+17.0	0.1542	+18.0
R-Precision (precision after R (= num_rel for a query) docs retrieved):											
Exact:	**0.3346**	**0.3942**	**+18.0**	**0.4251**	**+27.0**	**0.4283**	**+28.0**	**0.4281**	**+28.0**	**0.4291**	**+28.0**

Table 3: New Routing Results at Several Query Expansion Levels

ranking and selecting the n best. Moreover, these 'nonbreak' documents total only 5225, less than 1/3 of 16114 relevants used and is therefore very efficient. (There are actually 16400 relevants from Disk 1&2, but during processing a small percentage was lost). 2) All edges and their weights on the query side of the network are defined by the activations deposited by the relevant documents; this means the original query plays no part in their definition. 3) Negative edge weights are set to small positive weights of 0.1. For retrieval: 4) after ranking, several subdocuments of the same document ID may rank high, and we combine their largest three RSVs in the ratio of 1:0.2:0.05 as the single reported RSV for the whole document. Previously we ignored the third, and the ratio for combining the largest two was different. We choose to stop at two or three subdocuments because noise from long documents may creep back. Such tuning of parameters led to the results in Table 3 for our latest routing results. We use the convention 'Trained Exp K' to denote query expansion level K, with K=0 meaning weight adaptation without adding new

terms. The 'No Train.' column shows results without using any known relevants for training and serve as the basis for comparison. It can be seen using the measure of average precison over all recall points that training without query expansion improves over no training by 21%, and training with expansion at the 40 level improves over the basis by about 39%. This measure, as well as the R-precision, seems to level off from expansion level 40 onwards. However the number of relevants retrieved improves from 6551 (no training) to 7712 (expansion level 100) in a monotone fashion. Higher query expansion level appears to improve the high-recall region of the precision-recall curve without materially affecting the low-recall region as observed in [Kwok9x] using the WSJ collection only. Precision values at the different cutoffs of documents retrieved seem to level off at the expansion level of 80. At 20 retrieved documents cutoff, we now achieve a precision over 0.65, meaning that more than 13 of the 20 documents retrieved are relevant on the average. The tuning of parameters give us over 10% additional improvements above those obtained in the revised routing results of Table 2. It appears that a query expansion level of 40 achieves a compromise between good effectiveness and good efficiency for our system. We did not do massive query expansion at high levels of 200 or more. However, the results are comparable to the best of those reported in the TREC2 conference.

9. Conclusion

We have upgraded our PIRCS system to use dynamic network creation for learning and retrieval, and to handle files in a master-subcollection design. The former approach allows us to eliminate full inverted file creation resulting in 2 x collection size space requirement, reduced 'dead' time for a collection to be searchable, and provide fast learning. The latter approach renders our system to be sufficiently flexible to handle a large number of files in a robust fashion, yet produce a retrieval ranked list as if all documents were in one file. Although our submitted results for TREC2 were not up to expectation because of insufficient resources at the time of the experiments, the reasons for the behavior of our system were isolated. New experiments show that PIRCS can provide highly competitive retrieval effectiveness in both ad hoc and routing environments.

Acknowledgment

Ms. Lu Chi-Ni provided the program for our two-word phrase detection. We would like to acknowledge the continual support of the Department Chairman and the Dean of Mathematics and Natural Science at Queens College throughout the project. This work is partially supported by a grant from ARPA and a PSC-CUNY grant #663288.

References

[BuSA93] Buckley, C, Salton, G & Allan, J (1993). Automatic retrieval with locality information using SMART. In: The First Text REtrieval Conference (TREC-1). Harman, D.K. (Ed.). NIST Special Publication 500-207. pp.59-72.

[Crof93] Croft, W.B (1993). The University of Massachusetts Tipster project. In: The First Text REtrieval Conference (TREC-1). Harman, D.K. (Ed.). NIST Special Publication 500-207. pp.101-105.

[KwPK93] Kwok, K.L, Papadopolous, L & Kwan, Y.Y. Retrieval experiments with a large collection using PIRCS. In: The First Text REtrieval Conference (TREC-1). Harman, D.K. (Ed.). NIST Special Publication 500-207. pp.153-172.

[Kwok90] Kwok, K.L (1990). Experiments with a component theory of probabilistic information retrieval based on single terms as document components. ACM TOIS 8:363-386.

[Kwok9x] Kwok, K.L (199x). A network approach to probabilistic information retrieval. Accepted for publication in ACM TOIS.

[SaFW83] Salton, G; Fox, E.A & Wu, H (1983). Extended boolean information retrieval. Comm. ACM 26:1022-1036.

[RoSp76] Robertson, S.E & Sparck Jones, K (1976). Relevance weighting of search terms. J. ASIS. 27:129-146.

[Spar79] Sparck Jones, K (1979). Experiments in relevance weighting of search terms. Info. Proc. Mgmnt. 15:133-144.