INQUERY and TREC-7

James Allan, Jamie Callan, Mark Sanderson, Jinxi Xu, and Steven Wegmann*

Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, Massachusetts USA

*Dragon Systems, Inc.
320 Nevada Street
Newton, Massachusetts USA

This year the Center for Intelligent Information Retrieval (CIIR) at the University of Massachusetts participated in only four of the tracks that were part of the TREC-7 workshop. We worked on ad-hoc retrieval, filtering, VLC, and the SDR track. This report covers the work done on each track successively. We start with a discussion of IR tools that were broadly applied in our work.

1 Tools applied

Although UMass used a wide range of tools, from Unix shell scripts, to PC spreadsheets, three major tools were applied across almost all tracks: the Inquery search engine, the InRoute filtering engine, and a a query expansion technique known as LCA. This section provides a brief overview of each of those so that the discussion does not have to repeated for each track.

1.1 Inquery

All tracks other than the filtering track used Inquery[6] as the search engine, sometimes for training, and always for generating the final ranked lists for the test. We used Inquery V3.2, an in-house development version of the Inquery system made available by the CIIR (V3.1). The differences between the two are not consequential for this study.

The current belief function used by Inquery to calculate the belief in term t within document d is:

$$w_{t,d} = 0.4 + 0.6 \times \frac{\mathrm{tf}_{t,d}}{\mathrm{tf}_{t,d} + 0.5 + 1.5\frac{\mathrm{length}(d)}{\mathrm{avg\ len}}} \times \frac{\log \frac{N+0.5}{n_t}}{\log N + 1}$$

where n_t is the number of documents containing term t, N is the number of documents in the collection, "avg len" is the average length (in words) of documents in the collection, length(d) is the length (in words) of document d, and $\mathrm{tf}_{t,d}$ is the number of times term t occurs in document d.

1.2 InRoute

The InRoute filtering system is based on the same Bayesian inference network model as InQuery. It is designed to operate efficiently in high-volume filtering environments, where incoming documents must be processed rapidly, one at a time. It uses the same document indexing techniques, query language, and scoring algorithms as InQuery. Corpus statistics for the document stream are estimated using an archival corpus or are learned as documents stream by. InRoute also incrementally learns improved profiles [1] and improved thresholds [4], as relevance judgments become available for documents that have been disseminated.

InRoute was used only in the filtering track.

1.3 Local Context Analysis (LCA)

In SIGIR '96, the CIIR presented a new query expansion technique that worked more reliably than previous "pseudo relevance feedback" methods.[17] That technique, Local Context Analysis (LCA), locates expansion terms in top-ranked passages, uses phrases as well as terms for expansion features, and weights the features in a way intended to boost the expected value of features that regularly occur near the query terms.

LCA has several parameters that affect its results. The first is the choice of LCA database: the collection from which the top ranked passages are extracted. This database could be the test collection itself, but is often another (perhaps larger) collection that it is hoped will broaden the set of likely expansion terms. In the discussion below, if the LCA database is not the test collection itself, we identify what collection was used.

LCA's other two parameters are the number of top passages used for expansion, and the number of expansion features added to the query. The LCA features were put into a query construct that allows a weighted average of the features. Assuming n features, f_1 through f_n, they are combined as:

$$\begin{array}{lll}\texttt{\#wsum(1.0} & 1.0 & f_1 \\ & \vdots & \vdots \\ & 1-(i-1)*0.9/s & f_i \\ & \vdots & \vdots \\ & 1-(n-1)0.9/s & f_n\ \texttt{)}\end{array}$$

Here, s is scaling factor that is usually equal to n. The weighted average of expansion features is combined with the original query as follows:

$$\texttt{\#wsum(1.0}\ 1.0\ \text{original-query} \qquad w_{\text{lca}}\ \text{lca-wsum}\ \texttt{)}$$

where w_{lca} is the weight that the LCA features are given compared to the original query. Note that the final query is a weighted combination of the original query and the expansion features. As will be discussed below, in the SDR track the combination was unintentionally done differently, slightly shifting the balance between the original query and the expansion concepts.

2 Ad-hoc track

Considering the excellent results of our TREC-6 runs (had we indexed all the documents), we basically copied what we did in TREC-6 with few changes. The main techniques used for TREC-7 are still phrase recognition and query expansion using local context analysis (LCA) [17]. Phrases are recognized from the topic text (title, description and narrative) using a phrase dictionary and are added to a query. Phrase acquisition and recognition are described in the UMass TREC-6 report[2].

As in TREC-6, query expansion is carried out using a database consisting of TREC volumes 1 to 5 except the Federal Register documents. LCA concepts are extracted from the top 30 passages retrieved from the database for a query. 50 concepts are added to the query with decreasing weights in proportion to their ranks. Referring to the discussion of Section 1.3, for the ad-hoc work we set $s = 70$ and $w_{\mathrm{lca}} = 1.25$.

The effectiveness of all query expansion techniques that are based on the top retrieved documents (passages) obviously depends on the quality of the top ranked set of documents (passages). As an attempt to improve the quality of the top ranked set, in TREC-7 we used a filter to modify the ranks of the passages (obtained by normal tf·idf ranking) based on whether a passage contains all the title words of the topic. The idea is that the title words are the essential requirements of relevancy and passages missing any of such words are unlikely to be relevant. The effect of the filter is to rank passages containing all the title words before those missing some title words. This is done using the INQUERY `#filreq` operator. We call the technique *filter-require* on the title words (to pass the filter a document is required to match the title words).

2.1 Ad-hoc runs

We submitted three ad-hoc runs in TREC-7. The runs are labeled INQ501, INQ502 and INQ503. Our query processing includes 3 steps:

1. Basic query processing—removing stop words and stop phrases (such as "relevant documents") from the topic texts. Sentences discussing criteria of non-relevance in the narratives (such as "Documents discussing ... are not relevant") are also removed. After that, the narratives still contain a lot of verbiage not directly related to relevance. Therefore we further reduce the size of the narratives by removing the non-content bearing words from them. Words in the narrative of a topic are ranked by the formula

$$v(t) = freq(t) \times idf(t) \times avtf(t)^{0.7}$$

   where $freq(t)$ is the frequency of t in the narrative, $idf(t)$ is the inverse document frequency of t in the TREC-7 ad-hoc collection and $avtf(t)$ is the average frequency of t in TREC-7 documents when it occurs. Terms with a value less than 0.4 are discarded.

   INQ501 and INQ502 used the title, description and narrative fields. Narrative words are given 30% of the weight of the title and description words. INQ503 used only the title and description fields.

2. Phrase identification (as in TREC6)

3. Query expansion using LCA: adding 50 LCA concepts per query. INQ502 used filter-require on the title words in addition to tf·idf ranking for retrieving the top ranked passages while INQ501 and INQ502 used only tf·idf for that purpose.

2.2 Ad-hoc results of submitted runs

The retrieval results are shown in Table 1. The average precision for INQ501, INQ502 and INQ503 is 0.2739, 0.2815, and 0.2521 respectively. Comparing with the TREC-7 ad-hoc average 0.1822, our results are very satisfactory. Of 50 topics, INQ502 (our best run) is below the average for only 2 topics.

2.3 Ad-hoc analysis

This section considers the various stages of query processing and how they impact effectiveness individually and collectively.

Run:	INQ501	INQ502		INQ503	
Total number of documents over all queries					
Retrieved:	50000	50000	(+ 0.0)	50000	(+ 0.0)
Relevant:	4674	4674	(+ 0.0)	4674	(+ 0.0)
Rel_ret:	3206	3215	(+ 0.3)	3017	(- 5.9)
Interpolated Recall - Precision Averages at:					
0.00	0.7823	0.8314	(+ 6.3)	0.7730	(- 1.2)
0.10	0.5883	0.6070	(+ 3.2)	0.5586	(- 5.0)
0.20	0.4526	0.4579	(+ 1.2)	0.4301	(- 5.0)
0.30	0.3656	0.3848	(+ 5.3)	0.3434	(- 6.1)
0.40	0.3101	0.3269	(+ 5.4)	0.2843	(- 8.3)
0.50	0.2490	0.2571	(+ 3.3)	0.2097	(-15.8)
0.60	0.1919	0.1906	(- 0.7)	0.1597	(-16.8)
0.70	0.1393	0.1414	(+ 1.5)	0.1095	(-21.4)
0.80	0.0893	0.0790	(-11.5)	0.0715	(-19.9)
0.90	0.0417	0.0449	(+ 7.7)	0.0279	(-33.1)
1.00	0.0175	0.0175	(+ 0.0)	0.0151	(-13.7)
Average precision (non-interpolated) over all rel doc					
	0.2739	0.2815	(+ 2.8)	0.2521	(- 8.0)
Precision at:					
5 docs:	0.5760	0.6160	(+ 6.9)	0.5680	(- 1.4)
10 docs:	0.5540	0.5800	(+ 4.7)	0.5240	(- 5.4)
15 docs:	0.5240	0.5333	(+ 1.8)	0.4747	(- 9.4)
20 docs:	0.4850	0.4950	(+ 2.1)	0.4500	(- 7.2)
30 docs:	0.4240	0.4347	(+ 2.5)	0.4007	(- 5.5)
100 docs:	0.2502	0.2572	(+ 2.8)	0.2394	(- 4.3)
200 docs:	0.1833	0.1840	(+ 0.4)	0.1720	(- 6.2)
500 docs:	0.1063	0.1070	(+ 0.7)	0.0994	(- 6.5)
1000 docs:	0.0641	0.0643	(+ 0.3)	0.0603	(- 5.9)
R-Precision (precision after R (= num_rel for a query) docs retrieved):					
Exact:	0.3117	0.3178	(+ 2.0)	0.2899	(- 7.0)

Table 1: Submitted ad-hoc retrieval runs

2.3.1 Ad-hoc basic processing

Table 2 shows the retrieval results when we only use the basic query processing (i.e., no phrase recognition and no LCA). The results show that retrieval becomes better as queries get longer. However, even very short queries with 2–3 words (the title queries) can still achieve reasonable retrieval performance.

2.3.2 Using phrases in ad-hoc

Table 3 shows that adding phrases to queries causes a modest 3.6% improvement in average precision. The improvement is not statistically significant. This is consistent with our TREC-6 results: phrases can improve retrieval performance but the benefit is limited.

2.3.3 Expanding ad-hoc queries

Table 4 shows that query expansion by LCA causes a substantial improvement in average precision. Precision at all document cut-offs are improved as well. The improvement in average precision is 18.5% and statistically significant (t-test, $p = 0.00004$). This is also consistent with our TREC-6 results: query expansion using LCA can significantly improve retrieval performance.

Run:	title	desc		title_desc		title_desc_narr	
Total number of documents over all queries							
Retrieved:	50000	50000	(+ 0.0)	50000	(+ 0.0)	50000	(+ 0.0)
Relevant:	4674	4674	(+ 0.0)	4674	(+ 0.0)	4674	(+ 0.0)
Rel_ret:	2070	2391	(+15.5)	2547	(+23.0)	2712	(+31.0)
Interpolated Recall - Precision Averages at:							
0.00	0.6984	0.7546	(+ 8.0)	0.7815	(+11.9)	0.8046	(+15.2)
0.10	0.4359	0.4864	(+11.6)	0.5058	(+16.0)	0.5630	(+29.2)
0.20	0.3088	0.3415	(+10.6)	0.3568	(+15.5)	0.4100	(+32.8)
0.30	0.2169	0.2674	(+23.3)	0.2731	(+25.9)	0.3036	(+40.0)
0.40	0.1484	0.2014	(+35.7)	0.2054	(+38.4)	0.2262	(+52.4)
0.50	0.1055	0.1412	(+33.8)	0.1554	(+47.3)	0.1803	(+70.9)
0.60	0.0783	0.1075	(+37.3)	0.1037	(+32.4)	0.1241	(+58.5)
0.70	0.0396	0.0602	(+52.0)	0.0641	(+61.9)	0.0762	(+92.4)
0.80	0.0218	0.0205	(- 6.0)	0.0286	(+31.2)	0.0333	(+52.8)
0.90	0.0140	0.0045	(-67.9)	0.0091	(-35.0)	0.0123	(-12.1)
1.00	0.0010	0.0000	(-100.0)	0.0000	(-100.0)	0.0004	(-60.0)
Average precision (non-interpolated) over all rel doc							
	0.1634	0.1924	(+17.7)	0.2008	(+22.9)	0.2231	(+36.5)
Precision at:							
5 docs:	0.4640	0.5120	(+10.3)	0.5200	(+12.1)	0.5400	(+16.4)
10 docs:	0.4020	0.4600	(+14.4)	0.4820	(+19.9)	0.5120	(+27.4)
15 docs:	0.3667	0.4067	(+10.9)	0.4320	(+17.8)	0.4640	(+26.5)
20 docs:	0.3360	0.3790	(+12.8)	0.3950	(+17.6)	0.4290	(+27.7)
30 docs:	0.2840	0.3220	(+13.4)	0.3353	(+18.1)	0.3727	(+31.2)
100 docs:	0.1658	0.1960	(+18.2)	0.1978	(+19.3)	0.2154	(+29.9)
200 docs:	0.1178	0.1358	(+15.3)	0.1434	(+21.7)	0.1568	(+33.1)
500 docs:	0.0694	0.0772	(+11.2)	0.0829	(+19.5)	0.0884	(+27.4)
1000 docs:	0.0414	0.0478	(+15.5)	0.0509	(+22.9)	0.0542	(+30.9)
R-Precision (precision after R (= num_rel for a query) docs retrieved):							
Exact:	0.2074	0.2435	(+17.4)	0.2504	(+20.7)	0.2712	(+30.8)

Table 2: Basic query processing of ad-hoc queries

2.3.4 Filter-require in ad-hoc queries

When filter-require on title words is used for query expansion (INQ502), the average precision is improved by another 2.8% (Table 5). The improvement at low recall is improved more substantially. The technique does not dramatically improve average performance, but it has a significant impact on individual queries. A few queries are significantly improved. One example is topic 385 about "hybrid fuel cars". Standard tf·idf ranking tends to pick up passages containing many occurrences of "fuel" and "cars" but missing "hybrid", an essential element of relevance for this query. Filter-require corrects this problem by requiring the top ranked passages to contain "hybrid". As a result, it results in more effective query expansion. But a few queries are hurt by this technique. The problem is caused by non-essential words in the titles. One example is the word "risks" in the title of topic 354 "journalist risks"—a document can be relevant without literally using the word "risks". Further investigation is needed to determine the value of this technique.

3 Filtering Track

Our goal for the Filtering track was to test the InRoute filtering system [5] and the threshold-learning algorithm that we used last year [2, 4]. These were relatively modest goals, because there seemed to be *significant* changes from last year's system.

Run:	title_desc_narr	title_desc_narr_phr	
Total number of documents over all queries			
Retrieved:	50000	50000	(+ 0.0)
Relevant:	4674	4674	(+ 0.0)
Rel_ret:	2712	2753	(+ 1.5)
Interpolated Recall - Precision Averages at:			
0.00	0.8046	0.8003	(- 0.5)
0.10	0.5630	0.5483	(- 2.6)
0.20	0.4100	0.4058	(- 1.0)
0.30	0.3036	0.3179	(+ 4.7)
0.40	0.2262	0.2476	(+ 9.5)
0.50	0.1803	0.1880	(+ 4.3)
0.60	0.1241	0.1367	(+10.2)
0.70	0.0762	0.0914	(+19.9)
0.80	0.0333	0.0530	(+59.2)
0.90	0.0123	0.0242	(+96.7)
1.00	0.0004	0.0086	(+2050.0)
Average precision (non-interpolated) over all rel doc			
	0.2231	0.2311	(+ 3.6)
Precision at:			
5 docs:	0.5400	0.5320	(- 1.5)
10 docs:	0.5120	0.5200	(+ 1.6)
15 docs:	0.4640	0.4827	(+ 4.0)
20 docs:	0.4290	0.4350	(+ 1.4)
30 docs:	0.3727	0.3887	(+ 4.3)
100 docs:	0.2154	0.2242	(+ 4.1)
200 docs:	0.1568	0.1603	(+ 2.2)
500 docs:	0.0884	0.0910	(+ 2.9)
1000 docs:	0.0542	0.0551	(+ 1.7)
R-Precision (precision after R (= num_rel for a query) docs retrieved):			
Exact:	0.2712	0.2745	(+ 1.2)

Table 3: Addition of phrases to ad-hoc queries

One obvious departure from last year's system was that the user preference that influences threshold settings was raised to "high precision"; last year it was set halfway between "high precision" and "high recall", and results were reasonably good. Studies during the Winter and Spring indicated that raising the preferences was risky, because they pushed up the dissemination thresholds, which in turn reduced the amount of training data available to the system.

A second difference is that the initial queries for this year's system were intended to more closely match the query-creation process used for the Ad-hoc track. In the past, the initial filtering queries were very simple. Our hypothesis was that the initial query should be as good as possible, and then be further modified by incremental relevance feedback on documents disseminated during filtering.

An initial assessment suggests that the filtering results are quite poor. We do not yet know why, but it appears that there may have been several causes.

Raising the user dissemination preference was clearly a mistake. It caused thresholds to be too high, and hence no documents were disseminated for many profiles. It also reduced the amount of training data available for learning profiles, resulting in less accurate profiles. Lowering the preference to last year's value causes more documents to be disseminated for more profiles; the resulting increase in training data leads to dramatic precision improvements on many profiles. This accounts for some of the poor filtering results, but not all.

The use of more complex initial queries may also have been a factor, but we have not yet had an opportunity

Run:	title_desc_narr_phr	INQ501	
Total number of documents over all queries			
Retrieved:	50000	50000	(+ 0.0)
Relevant:	4674	4674	(+ 0.0)
Rel_ret:	2753	3206	(+16.5)
Interpolated Recall - Precision Averages at:			
0.00	0.8003	0.7823	(- 2.2)
0.10	0.5483	0.5883	(+ 7.3)
0.20	0.4058	0.4526	(+11.5)
0.30	0.3179	0.3656	(+15.0)
0.40	0.2476	0.3101	(+25.2)
0.50	0.1880	0.2490	(+32.4)
0.60	0.1367	0.1919	(+40.4)
0.70	0.0914	0.1393	(+52.4)
0.80	0.0530	0.0893	(+68.5)
0.90	0.0242	0.0417	(+72.3)
1.00	0.0086	0.0175	(+103.5)
Average precision (non-interpolated) over all rel doc			
	0.2311	0.2739	(+18.5)
Precision at:			
5 docs:	0.5320	0.5760	(+ 8.3)
10 docs:	0.5200	0.5540	(+ 6.5)
15 docs:	0.4827	0.5240	(+ 8.6)
20 docs:	0.4350	0.4850	(+11.5)
30 docs:	0.3887	0.4240	(+ 9.1)
100 docs:	0.2242	0.2502	(+11.6)
200 docs:	0.1603	0.1833	(+14.3)
500 docs:	0.0910	0.1063	(+16.8)
1000 docs:	0.0551	0.0641	(+16.3)
R-Precision (precision after R (= num_rel for a query) docs retrieved):			
Exact:	0.2745	0.3117	(+13.6)

Table 4: Query expansion (LCA) of ad-hoc queries

to investigate this change sufficiently to draw any conclusions.

4 Spoken Document Retrieval track

This section describes the work by CIIR on the SDR track. Repeating the partnership from last year's TREC submission with Dragon Systems, the CIIR submitted a number of runs primarily investigating alternate configurations of the retrieval system, Inquery. This centered on investigating different sources of evidence for use in automatic query expansion. The experimental set up of the system is first described followed by the motivations for the various configurations tested. The results of the experiments are presented and briefly discussed.

4.1 Recognition

The TREC-7 system used by Dragon is a faster version of their 1997 Hub4 evaluation system, and it was also used to transcribe 1000 hours of broadcast data automatically for the TDT task. This system is fully described in [11] and [12], but we give a brief description here.

Run:	INQ501	INQ502	
Total number of documents over all queries			
Retrieved:	50000	50000	(+ 0.0)
Relevant:	4674	4674	(+ 0.0)
Rel_ret:	3206	3215	(+ 0.3)
Interpolated Recall - Precision Averages at:			
0.00	0.7823	0.8314	(+ 6.3)
0.10	0.5883	0.6070	(+ 3.2)
0.20	0.4526	0.4579	(+ 1.2)
0.30	0.3656	0.3848	(+ 5.3)
0.40	0.3101	0.3269	(+ 5.4)
0.50	0.2490	0.2571	(+ 3.3)
0.60	0.1919	0.1906	(- 0.7)
0.70	0.1393	0.1414	(+ 1.5)
0.80	0.0893	0.0790	(-11.5)
0.90	0.0417	0.0449	(+ 7.7)
1.00	0.0175	0.0175	(+ 0.0)
Average precision (non-interpolated) over all rel doc			
	0.2739	0.2815	(+ 2.8)
Precision at:			
5 docs:	0.5760	0.6160	(+ 6.9)
10 docs:	0.5540	0.5800	(+ 4.7)
15 docs:	0.5240	0.5333	(+ 1.8)
20 docs:	0.4850	0.4950	(+ 2.1)
30 docs:	0.4240	0.4347	(+ 2.5)
100 docs:	0.2502	0.2572	(+ 2.8)
200 docs:	0.1833	0.1840	(+ 0.4)
500 docs:	0.1063	0.1070	(+ 0.7)
1000 docs:	0.0641	0.0643	(+ 0.3)
R-Precision (precision after R (= num_rel for a query) docs retrieved):			
Exact:	0.3117	0.3178	(+ 2.0)

Table 5: Use of filter-require on ad-hoc runs

4.1.1 Front End

A total of 36 parameters are computed every 10 milliseconds: 12 cepstral parameters, 12 cepstral differences, and 12 cepstral second differences. This set of 36 parameters is linearly transformed using IMELDA techniques [8] to a set of 24 parameters which are used for training and recognition. We use PLP-based cepstra [14], computed in the style of Cambridge/HTK, as reported in [9].

Speaker normalization [10] is used to reduce variability among speakers due to vocal tract length. During the signal processing stage, the frequency scale is "warped" using a piecewise linear transformation.

4.1.2 Acoustic Modeling

The model for a sentence hypothesis is obtained by concatenating models Dragon calls PICs (for "phonemes-in-context"). For this evaluation, Dragon used triphone models as described below. A 51-element phoneme set was used that has syllabic consonants and two stress levels for certain vowels. PICs are modeled using from 3 to 4 nodes, with each node having an output distribution (PEL) and a duration distribution. Which PEL model to employ for any given position of any PIC is determined based on a decision tree whose nodes ask linguistic questions about neighboring phonemes as well as questions about the position of word boundaries. The PEL models themselves are general mixture models with basis components given by multivariate Gaussian distributions with diagonal covariance.

In addition to speaker normalization, Dragon also makes use of rapid adaptation, using linear regression techniques to construct transformations of acoustic parameter space mapping speaker-independent model means to speaker-specific ones. This approach was inspired by, and represents a simplification of, speaker adaptation strategies implemented by Cambridge [15]. Dragon also used speaker adaptation techniques (SAT) during training ([7, 16]): Training speech is force-aligned to transcripts and the usual adaptation transformations are computed mapping speaker-independent model to speaker-specific data, and then a sort of "inverse" transformation is performed on the speech frames. This permits the training of new models with the transformed data which should behave well under test-time adaptation. Dragon used four transformation classes at training, determined by grouping related phonemes. (For another approach to speaker-adaptive training, see [13].)

A thumbnail sketch of the system used follows.

1. The system overall:
   - A amplitude based silence detector is used to break the input into chunks that are 20 to 30 seconds long.
   - A phoneme recognizer is used to produce a more refined chopping of these chunks.
   - The segments are clustered for speaker normalization and unsupervised adaptation.
   - Channel normalization is performed on each segment.
   - Speaker normalization is performed within each cluster by doing a quick, errorful recognition with small acoustic models (5000 PELs) and a small bigram LM (300,000 bigrams), and then re-scoring this transcript with each warp scale in order to pick the best scoring scale.
   - Speaker normalized, SAT models with 12,000 PELs are used along with an interpolated trigram LM to obtain an initial transcription for each cluster.
   - Unsupervised rapid adaptation is performed within each cluster, followed by the final recognition pass using the adapted acoustic models and the same trigram LM.
2. Acoustic training:
   - Used the 100 hours of Broadcast News training data to train the seed models for the SAT process.
   - The SAT models were trained from the above data plus the following:
     - A 24 hour subset of the WSJ1 si284 corpus.
     - The WSJCAM0 training corpus.
     - The 1995 Marketplace development corpus.
3. Grammar training:
   - A trigram language models were trained from 500 million words of text from three sources:
   - The Broadcast News acoustic training transcriptions plus the 1995 Marketplace development transcriptions.
   - The Broadcast News LM training corpus.
   - The 1995 Hub4 and Hub3 newswire texts were combined with 190 million words of commercially available newspaper data collected from the period January 1995 through June 1996.
4. Recognition lexicon description:
   - The three LMs share a 57K vocabulary list constructed by mixing the unigram probabilities from the three LM's and selecting the top 57K words.
   - The mixture weights were determined from preliminary recognition runs on the 1996 devtest: 0.30 for A, 0.50 for B, and 0.20 on C.
5. Speed:
   - The entire system ran at 16 x RT on a 300 MHz P-II.

4.2 Experimental components

Retrieval was performed using the Inquery IR system (see Section 1.1). In addition to use of standard IR techniques such as stop word removal, stemming and a tf·idf-like weighting scheme, Inquery was set up to use two additional proven methods.

First, SDR topics were pre-processed where phrases within the topics were recognized and some proper nouns were expanded with synonyms. This type of processing is the same that was done for the ad-hoc queries, and is described in last year's report.[2]

Second, SDR topics were automatically expanded using Local Context Analysis (LCA), essentially as described in Section 1.3. The difference is that the expansion terms were added to the query as "siblings" of the query features rather than balancing the two in an overall weighted sum. The combined version of the query therefore had the form:

$$
\begin{array}{lcc}
\texttt{\#wsum(1.0} & w_{q_1} & q_1 \\
 & \vdots & \vdots \\
 & w_{q_i} & q_i \\
 & \vdots & \vdots \\
 & w_{q_m} & q_m \\
 & & \\
 & 1.0 & f_1 \\
 & \vdots & \vdots \\
 & 1-(i-1)*0.9/s & f_i \\
 & \vdots & \vdots \\
 & 1-(n-1)0.9/s & f_n \texttt{)}
\end{array}
$$

where q_i are the original query features (after processing) and w_{q_i} is the weight assigned to that feature. In the case of SDR, 60 expansion features were add$(n = 60)$, expansion weights were assigned with $s = 60$. We discuss the impact of this alternate form of LCA expansion below.

Normally, the two retrieval steps of LCA are performed on the same collection; however, this is not required and it is possible that other collections of text can be used in the initial retrieval to provide expansion terms. All the variation in configurations of Inquery centered around what form of collection the initial retrieval was performed on. Three possibilities were tried.

1. The default configuration of just using the collection being searched on: e.g,. this year's SDR test collection as recognized by the Dragon speech recognizer. This was the strategy used in the umass-b1.dragon, umass-b2.dragon, umass-r1.dragon, umass-s1.dragon runs.
2. A combination of the SDR recognized collection and a corpus of AP newswire documents produced at the same time as the SDR audio broadcasts were made. This was the approach used in the umass-s2.dragon run.
3. A combination of different versions of the SDR collection each version produced by a different speech recognizer. The Sheffield, Cambridge, Dragon, NIST-B1, NIST-B2, and DERASRU recognizer transcripts were used. It was hoped that the different recognizers would make different recognition mistakes and that by combining the transcripts the mistakes would be less prevalent and, therefore, the expanded query would produce better retrieval. This approach was used in a number of additional runs that were not assessed by TREC.

In experiments on the SDR training set, this final strategy proved to be the most successful of the three. Results on the test set, however, proved to be different.

4.3 Main SDR results

The best retrieval effectiveness on the test collection was found on the umass-s1-dragon run. The average (non-interpolated) precision for this run was 0.5075, which was 89% the effectiveness of the r1 (hand transcribed) reference run: 0.5668. Such a result indicates that retrieval on audio data of the quality used in the SDR collection can be expected to be almost as good as retrieval on a hand transcribed version.

Description	AvgPrec
s1-dragon	0.5075
r1 (LTT)	0.5668

Comparison of the overall "Best, Median, Worst" statistics (compiled by TREC from all of the groups' data) against the s1 result revealed that s1 had the best average precision for 6 of the 23 topics, above median for 13, median for 1 and below the median for 3. That turned out to represent "top" performance among the participating systems.

4.4 Cross-recognizer runs

The CIIR had interpreted the cross-recognizer run description to allow a run such as the final approach listed earlier, where we combined the results of *several* recognizer outputs to help generate higher-quality expansion features. (It does not appear to have worked.) The TREC evaluation process chose to eliminate our run because—although it was a valid interpretation of the request—there were no similar runs to compare it to!

We have expended some effort to run various parts of our query processing across different recognizer outputs to see the impact of differing word error rates on the results. The following table includes a row for each of several recognizer outputs, and the columns represent different amounts of query processing. All numbers are non-interpolated average precision.

Recognizer	WER	SWER	No QP	Basic QP	With LCA
Human (ltt)	–	–	0.4401	0.5240	0.5677
Cambridge (htk)	24.8	24.6	0.4294	0.5001	0.5305
Dragon	29.8	29.9	0.3935	0.4711	0.5089
AT&T-s1	33.1	36.0	0.4197	0.4843	0.5243
Sheffield	36.8	34.1	0.3771	0.4551	0.5061
NIST-B1	34.1	34.7	0.3837	0.4581	0.5066
NIST-B2	46.9	49.2	0.3053	0.3725	0.4193
DERA/SRU-2	61.5	61.7	0.3427	0.4043	0.3858
DERA/SRU	66.2	64.1	0.3053	0.3598	0.4064

In all cases, the same techniques were applied. When LCA was used, it was used on the same database that was being run against. The type of LCA used was as described above (the style submitted for our SDR runs) in all cases, although later experiments showed that a modest improvement was possible.

What is interesting to note is that the effectiveness of retrieval degrades uniformly with increases in WER. We had hoped that either the query processing ("Basic QP") or query expansion ("With LCA") would help compensate for the recognition errors. Unfortunately, the results show that although those techniques are useful in all cases, they degrade at the same rate as the others so appear to be equally impacted by WER.

The one exception to that uniform drop in effectiveness is the AT&T run that had higher WER than Dragon, but also had better effectiveness across the board. We have not investigated this aberration, though hypothesize is may be the result of AT&T's recognizer that worked with larger portions of the word lattice.

4.5 Variants on LCA

We were interested in the effects of varying how we did query expansion. We tried several options, varying the number of expansion features, and varying the way of combining the original query and the expansion concepts. For the latter trials, we used either the method submitted (as described above) or our more standard method (in Section 1.3)—in the former, the expansion features are each treated as equal to original query features; in the latter, the expansion features are treated as a group and balanced against the original query as a group.

First we consider the result of different numbers of expansion concepts. In all cases, the expansion concepts were added to the query as above—that is, the first expansion concept was weighted equal to query features, and the rest had monotonically dropping weights. The runs are done with three different levels of WER to explore the impact of ASR errors on the expansion.

	AvgPrec	R-Prec	Prec@20
Human (LTT)			
10 LCA features	0.5648	0.5081	0.4022
60 LCA features	0.5677	0.5096	0.4087
100 LCA features	0.5645	0.4902	0.3935
Dragon (29.8% WER)			
10 LCA features	0.4923	0.4694	0.3761
60 LCA features	0.5089	0.4894	0.3671
100 LCA features	0.5097	0.4980	0.3804
DERA/SRU (66.2% WER)			
10 LCA features	0.3820	0.4007	0.3239
60 LCA features	0.4064	0.3992	0.3457
100 LCA features	0.3869	0.3948	0.3435

The results suggest that the number of expansion features is not critical, but that too few or too many can slightly damage performance. This supports our typical decision to expand with 50-70 features.

We also investigated the impact of different ways of adding the expansion features to the query. This set of runs was done because the expansion approach used by LCA was not intentional. We tried each of the following approaches. Recall that the query consists of a set of words or phrases, and the expansion features are a similar set of words or phrases.

equal In this run, the query features and the LCA features were all put together and treated equally, except that the LCA features were assigned descending weights starting with 1.0 and working down to 0.01 for the last one. (This was the submitted approach.)

add-group In this run, the LCA expansion features were treated as a single group and that entire group had the same potential contribution as a *single* query feature.

two-groups-Q In this run, the new query consisted of two parts: the original query and the expansion features, each treated as a single group. The query feature group was weighted 1.25 to the 1.0 weight of the LCA query group.

two-groups-LCA This is the same as the *two-groups-Q* run except that the weights were reversed. This is the way that queries were expanded in the ad-hoc track.

We ran these various forms of expansion on several recognizer outputs. In all cases, the queries were expanded with 60 LCA features.

	AvgPrec	R-Prec	Prec@20
Human (LTT)			
equal	0.5677	0.5096	0.4087
add-group	0.5374	0.4963	0.3761
two-groups-Q	0.5523	0.4906	0.4000
two-groups-LCA	0.5662	0.4976	0.4109
Dragon (29.8% WER)			
equal	0.5089	0.4894	0.3761
add-group	0.4823	0.4670	0.3587
two-groups-Q	0.5033	0.4927	0.3783
two-groups-LCA	0.5143	0.5011	0.3935
DERA/SRU (66.2% WER)			
equal	0.4064	0.3992	0.3457
add-group	0.3782	0.3760	0.3261
two-groups-Q	0.3945	0.3776	0.3283
two-groups-LCA	0.3987	0.3851	0.3348

Interestingly, our failing to group the concepts was a mixed blessing. For the human transcripts, it gave us a slight benefit overall, with a negligible drop in high precision. But for our official transcript (Dragon's), we got a slight drop: we would have done about 2% better (and the difference is significant by the sign test, $p = 0.03$). Given that the difference is small, however, we conclude that it did not have much of an impact. We expect to use the *two-groups-LCA* variant in the future, though, since that has been our preferred approach.

4.6 Other results

The s2 run, which used AP news wire as an additional source of LCA expansion terms proved to be worse than s1. The use of other groups transcripts in the LCA process proved to be even more detrimental to retrieval effectiveness than the s2 run. We have tentatively concluded that this approach is not valuable.

5 Very Large Corpus (VLC) Track

Our goals for the Very Large Corpus (VLC) track were modest: To gain experience with a larger corpus, and to contribute to the large-scale corpus-building effort by adding documents to the assessment pool.

Retrieval speed was *not* a research goal. The Inquery system, version 3.1, was used, with no special optimizations.

It was also not a goal to be the most accurate system. The queries created for the Ad-hoc track were used in the VLC track. Ordinarily query expansion with Local Context Analysis (LCA) is database-specific, but in the interests of doing the track with minimal effort, we simply used the query expansion terms created for the Ad-hoc track. Our hypothesis, based on other work with LCA, is that using the VLC database(s) for query expansion would produce more effective results.

5.1 VLC Query Sets

Three sets of queries were developed. All three are of the "Fixed Query" type, because query expansion was done on a separate database (the Ad-hoc database), as described above. The queries were:

inq5vlc1: The INQ501 query set, described above, which used the Title, Description, and Narrative fields;

inq5vlc2: The INQ502 query set, described above, which used the Title, Description, and Narrative fields; and

inq5vlc3: The INQ503 query set, described above, which used the Title and Description fields.

The judged runs were all done on inq5vlc1, because we believed it would be the most accurate query set.

5.2 Indices

The BASE1 collection was indexed as a single database.

Various data buffer size and index word size constraints prevented building the larger collections into a single database (in particular, the VLC2 collection contains more tokens than can be counted in a 32 bit word). Although these constraints could have been overcome, a multi-database approach fit more naturally with our other research interests, and was hence the approach adopted.

The BASE10 collection was indexed as two databases. Each indexed used its own *idf* values, i.e., global *idf* values were *not* created. Although this makes document scores slightly incompatible, it was assumed that queries would be long enough to minimize problems.

The VLC2 collection was indexed as 16 databases. As with the BASE10 collection, each index used its own *idf* values, i.e., global *idf* values were *not* created.

Indexing was done on a multi-user system during ordinary daily use, so little can be said about Inquery's indexing speed or resource requirements. It was not practical to devote one or more computers to the VLC effort; nor was it necessary, as Inquery is quite capable of handling this volume of data on a machine being used by a number of other processes.

VLC2 indexing, from initiation to completion, took 43 hours and 36 minutes, with 4 processors busy for almost all that time, and some competition from other users and tasks. About 8 hours and 9 minutes of that time was spent uncompressing data. Different parts and stages of the build were running in parallel on the 3 processors of the computer, with an attempt to keep all processors busy without thrashing. Because of limited disk space, uncompressed versions of bundles were created as needed and then deleted immediately after use. All of the processes used were "niced". The computer used was a Sun Sparc server with 4 processors each running at 167 MHz and 1000 MB of memory.

The BASE10 build required 26:09 on the same processors but was overlapped with the VLC2 build for 14:30 of those hours.

The BASE1 build took 2:40 including re-compressing the source collection (the other builds made uncompressed copies and removed them; this build actually re-compressed the bundles after using them, taking much longer).

The indices required 0.68 gigabytes for BASE1, 6 gigabytes for BASE10, and 53 gigabytes for VLC2.

5.3 Retrieval

The BASE1 collection was organized as a single index. Queries were run against that index.

The BASE10 and VLC2 collections were organized as multiple indices. Queries were run against *all* indices associated with a collection. Document rankings from each index were merged to produce a final ranking for the collection.

The approach to merging document rankings was quite simple. The top-ranked 20 documents from each index were candidates for the final result set. These 20-document rankings were merged based upon the document scores. No attempt was made to normalize the scores returned by each index, or to favor documents returned by the "better matching" index.

Retrieval was done on a multi-user system during ordinary daily use, so little can be said about Inquery's retrieval speed or resource requirements. Inquery is clearly not one of the faster systems doing the VLC track. Based upon last year's results, it is also likely that Inquery uses longer queries than most of the other groups. The timing figures shown below are 'wall-clock' times.

	inq5vlc1	inq5vlc2	inq5vlc3
BASE1	7 min	8 min	6 min
BASE10	71 min	74 min	45 min
VLC2	593 min	598 min	438 min

The retrieval results appear to have been quite good. Precision at 20 documents is shown below.

	inq5vlc1	inq5vlc2	inq5vlc3
BASE1	0.202	0.204	0.208
BASE10	0.429	0.441	0.419
VLC2	0.625	0.624	0.598

We haven't done any interesting post-hoc analysis at this point.

Acknowledgments

We thank Daniella Malin, Michael Scudder, Fang-fang Feng, Chiinga Musonda, and Karen Priore of the CIIR for their assistance in the work described here.

This study is based on research support by several grants: the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623; the National Science Foundation under grant number IRI-9619117; and the NSF Center for Intelligent Information Retrieval at the University of Massachusetts, Amherst.

Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

References

[1] J. Allan. Incremental relevance feedback. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval*, pages 270–278, Zurich, 1996. Association for Computing Machinery.

[2] J. Allan, J. P. Callan, W. B. Croft, L. Ballesteros, D. Byrd, R. Swan, and J. Xu. INQUERY does battle with TREC-6. In D. K. Harman and E. M. Voorhees, editors, *The Sixth Text REtrieval Conference (TREC-6)*. National Institute of Standards and Technology, Special Publication, 1998.

[3] James Allan, Jamie Callan, W. Bruce Croft, Lisa Ballesteros, Don Byrd, Russell Swan, and Jinxi Xu. INQUERY does battle with TREC-6. In D. Harman, editor, *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*. National Institute of Standards and Technology Special Publication, (in press).

[4] J. Callan. Learning while filtering documents. In *Proceedings of the 21st annual international ACM SIGIR conference on research and development in information retrieval*, Melbourne, 1998. Association for Computing Machinery.

[5] J. P. Callan. Document filtering with inference networks. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval*, pages 262–269, Zurich, 1996. Association for Computing Machinery.

[6] J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83, Valencia, Spain, 1992. Springer-Verlag.

[7] B. Peskin et al. Progress in recognizing conversational telephone speech. In *Proc. ICASSP-97*, Munich, April 1997.

[8] M.J. Hunt et al. An investigation of plp and imelda acoustica representations and of their potential for combination. In *Proc. ICASSP-91*, Toronto, May 1991.

[9] P. Woodland et al. Broadcast news transcription using htk. In *Proc. DARPA Speech Recognition Workshop*, Chantilly, February 1997.

[10] R. Roth et al. Dragon systems' 1994 large vocabulary continuous speech recognizer. In *Proc. ARPA Spoken Language Systems Technology Workshop*, Austin, 1995.

[11] S. Wegmann et al. Dragon systems' 1997 broadcast news transcription system. In *Proc. of the Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA, February 1998.

[12] S. Wegmann et al. Dragon systems' automatic transcription system for the new tdt corpus. In *Proc. of the First International Conference on Language Resources and Evaluation*, Granada, Spain, May 1998.

[13] T. Anastasakos et al. A compact model for speaker-adaptive training. In *Proc. ICSLP'96*, Philadelphia, October 1996.

[14] H. Hermansky. Perceptual linear prediction (plp) analysis for speech. *J. Acoust. Soc. Amer.*, 87:1738–1752, 1990.

[15] C.J. Leggetter and P.C. Woodland. Speaker adaptation of continuous density hmms using multivariate linear regression. In *Proc. ICSLP'94*, Yokohama, September 1994.

[16] V. Nagesha and L. Gillick. Studies in transformation based adaptation. In *Proc. ICASSP-97*, Munich, April 1997.

[17] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval*, pages 4–11, Zurich, 1996. Association for Computing Machinery.