INQUERY and TREC-9

James Allan, Margaret E. Connell, W. Bruce Croft,
Fang-Fang Feng, David Fisher, and Xioayan Li

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

This year the Center for Intelligent Information Retrieval (CIIR) at the University of Massachusetts participated in three of the tracks: the cross-language, question answering, and query tracks. We used approaches that were similar to those used in past years.

In the next section, we describe some of the basic processing that was applied across most of the tracks. We then describe the details for each of the tracks and in some cases present some modest analysis of the effectiveness of our results.

1 Tools and Techniques

Although UMass used a wide range of tools, from Unix shell scripts, to PC spreadsheets, three major tools and techniques were applied across almost all tracks: the Inquery search engine, query processing, 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 three tracks used Inquery[Callan et al., 1992] 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. The "tf" component is usually referred to as an "Okapi tf" function [Robertson et al., 1995], and the "idf" component is the normalized idf used by the CIIR for years.

1.2 Query Processing

The processing of queries—i.e., the transformation from TREC topic to InQuery query—was handled very similarly to the way it was managed for TREC-8, though there were some small changes. It includes three steps:

1. Basic Query Processing removes stop words and phrases (e.g., "relevant documents will include"), and stop structures—i.e., sentences discussing criteria of non-relevance in the narratives. For removing the stop structures the processor simply segments each sentence, then removes the sentence fragments that contain the stop structure (e.g., "Documents discussing ... are not relevant"), but keep those clauses on the negative part of the removed sentence (such as in "... not relevant, unless ...", where the "unless" clause should not be removed).

2. Query Formalizing identifies noun phrases (as in earlier TRECs), and proper names. The proper names were transformed to use the ordered proximity-one operator (e.g., `#passage25(#1(Golden Triangle)`, which requires that the two words occur immediately adjacent in the text in that order; the passage operator affects the weighting of the feature. For noun phrases we not only used the phrase operator but also duplicated the single terms used by the phrase (e.g., `#passage25(#phrase(tropical storms)), tropical, storms`). Note that for proper names, the single term duplication was *not* done.

   Query formalizing also identifies compound words, like *wildlife* and *airport*, that are formalized with the synonym operator (e.g., `#syn(#1(air port) airport)`). Also as part of query formalization, if there is any word concerned with foreign countries, like *international*, *world*, or *Europe*, a token `#foreigncountry` is added to the query. If there is a term concern with the United States, then a token `#usa` will be added. If both `#foreigncountry` and `#usa` are found in a query, all such tokens are removed.

   Finally a query is formed with the weighted sum operator (`#wsum`) with a weight for each term. For those terms occurring in the title and description fields the weight 1.0 is used, while 0.3 is used for those terms occurring in the narrative field. That is, we trust the title and the description more than the narrative.

3. Query Expansion adds 50 LCA concepts to each query. These 50 concepts are collected from top 30 passages (the passage database is built with TREC volumns 1 through 5). This is the same process that we used in past TRECs, but we did not use "filter-required" on the title words that has sometimes been used. LCA is described next.

1.3 Local Context Analysis (LCA)

In SIGIR 1996, the CIIR presented a query expansion technique that worked more reliably than previous "pseudo relevance feedback" methods.[Xu and Croft, 1996] 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:

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

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.

2 Cross-language Track

The TREC-9 cross-lingual track was done on three collections, *Hong Kong Commercial Daily*, *Hong Kong Daily News*, and *Ta Kong Pao*, totaling 127938 documents. All documents were encoded in BIG5 font. Since most Chinese tools work only for the GB font, the collection was converted into GB with the converter, hc3[1]. The database was built using Inquery's Chinese version of build (called "hanbuild") with each Chinese character indexed as a term.

In order to do the cross-lingual run, a machine translator is required to translate the English query into the target language, Chinese. A dictionary-based translation was used. The title and description fields of the topics were selected for constructing the queries in the source language, English. For each English query term (i.e., word or phrase), the translator consulted an English-Chinese dictionary, and replaced the English term with Chinese entries found in the dictionary. Using such a simple translator has weaknesses. The dictionary has a limited number of entries, so there may not be translations available for some English words or phrases. Also, there are multiple translations in Chinese for most English words, introducing ambiguities. An effort was made to both enlarge the dictionary and add query terms based on LCA to counteract the weaknesses.

2.1 Cross-language dictionary

The dictionary was built primarily from the English to Chinese dictionary obtained on-line from LDC.[2]. When experiments were run trying to translate the English version of the topics given in TREC-5 and TREC-6, this dictionary of 110,818 entries was found not to have translations of some critical English terms. In order to enlarge the dictionary, two Chinese-English dictionaries, one from the LDC[3] and one from elsewhere on the Web[4] were converted into English-Chinese dictionaries and merged with the original. With the merged dictionary performance on training queries improved.

The process of converting and merging the Chinese-English dictionary into English-Chinese was simple. Each English word or phrase given as a translation of the Chinese entry became an English entry with the original Chinese word as its translation. Explanations in parentheses or brackets were omitted. The conversion process merged identical converted English entries. For example, there are three Chinese entries concerning *nuclear power*,

[1] Available at http://www.cnd.org/software/UNIX.html.
[2] http://morph.ldc.upenn.edu/Projects/Chinese/
[3] http://morph.ldc.upenn.edu/Projects/Chinese/
[4] http://www.chinapage.com/dictionary/dictionary.html

```
核大国   /a nuclear power (country)/
核电     /nuclear power/
核能源   /nuclear power/
```

The converted English entries are identical, so they are merged as

```
nuclear power /核大国 /核电 /核能源 /
```

The converted English-Chinese dictionaries were merged with the original. If an entry already existed in the dictionary the new Chinese translations were just merged, otherwise a new English entry was added. After the merge, the resulting dictionary had 120,003 entries. This was the dictionary used in all training experiments described below.

Four smaller English-Chinese word lists were added to the dictionary. They were ITglossary[5] (1361 entries), Economy-Indus-glossary[6] (2037 entries), computer-terms[7] (602 entries), and sehk-stock[8] (2419 entries). After reformatting these dictionaries and merging them, the dictionary had 124,013 entries. This dictionary was used in the cross-lingual, TREC-9, final runs.

2.2 Cross-language query translation

The query translation was based on the English-Chinese dictionary lookup. The lookup procedure was a binary search on the alphabeticly sorted dictionary. The query text was segmented automatically with a list of stopwords. Each stopword indicated a new segment. A dictionary lookup was done for every sequence of English words within a segment (a potential phrase) from long sequence to short recursively. The translator first tried to find English phrases from the dictionary. If none were found, it translated single words. For example, in the description of query 1 of TREC-5 Chinese track, there is a sequence:

most-favored nation status

The translator first looks in the dictionary for *most-favored nation status.* If it is not found, it then trys *most-favored nation.* If *most-favored nation* is found in the dictionary then the output is its Chinese translation otherwise it tries *nation status.* If a phrase lookup fails, the translator will do simple stemming of the plural head noun, in this case *most-favored nations* to *most-favored nation.* If no phrase is found in the dictionary, the translator translates single words. When a word-lookup fails the translator tries a stemmed or unhyphenated lookup. If all lookups fail the untranslatable term is discarded.

Name translations are important for queries about particular people, organizations, companies, or locations. There does not seem to be a dictionary covering proper names. An attempt was made to translate proper names using a parallel corpus. This approach was aborted because of time limitations.

When an English phrase or word was translated to a Chinese word consisting of more than one Chinese character, the multi-character word was represented by a proximity operator which forced the glyphs to be in order and adjacent. If more than one Chinese translation existed for an English phrase or word, all translations were wrapped in a synonym operator which treats all translations as the same word. While this avoids the exclusive use of wrong translations, it risks retrieving irrelevant material.

[5] http://www.iscs.nus.sg/ colips/archives/glossary/glossary.html
[6] http://news.cens.com/glossary/
[7] http://home.ust.hk/ lbsun/terms.html
[8] http://www.sehk.com.hk/index.asp?id=glossary.htm

2.3 Cross-language query expansion

Query expansion can be done either before translation, after translation, or both before and after translation.

Experiments were done expanding the English query prior to translation with LCA (described in Section 1.3). The expansion terms were chosen from TREC volumes 1-5.

Translated queries were also expanded using LCA. The translated query was used to extract expansion words from the top ranked passages of the original, TREC-9, Chinese database. The chosen passage size was 1500 words. Segmented words were used for expansion concepts. The segmenter was an improved version of the automatic segmenter used previously by the CIIR.[Ponte and Croft, 1996] It is based on hidden Markov models.

Stop phrases and phrases containing stopwords, digits and Chinese punctuation were automatically filtered from the expanded query.

Experiments were done using English topics from the TREC-5 and TREC-6 Chinese track. The results could be assessed using the available relevance judgements from this track.

The best results were found, using Chinese LCA only, selecting the top 20 passages and expanding the query with ten concepts. The expansion section of the query was weighted by 1.25 and each expansion concept was assigned a weight value $w(i) = 1.0 - 0.9(i - 1)/70$ where i was the rank of the concept.

The experiments on TREC-6 Chinese LCA only were: a) Top 30 passages adding 5,10,15,20,30 ,40,and 50 terms. b) Top 20 passages adding 5,10,15,20,30 ,40,and 50 terms.

The experiments run on TREC-6 English LCA only were: a) Top 20 passages adding 5, 10, 15, 20, 30, 40, 50 terms. b) Top 30 passages adding 5, 10, 15, 20, 30, 40, 50 terms.

The experiments on TREC-6 with both English and Chinese LCA were: a) Top 20 passages in English, adding 10 terms. Top 20 passages in Chinese adding 5, 10, 15, 20, 30, 40, 50 terms. b) Top 30 passages in English, adding 10 terms. Top 30 passages in Chinese adding 5, 10, 15, 20, 30, 40, 50 terms.

2.4 Cross-language, monolingual contrast

In the monolingual run queries were processed in much the same way as was done for TREC-6. Queries were composed from the title and description fields of the topics. The queries in BIG5 font were converted to GB using the hc3 converter[9].

Segmentation was done at query time. The automatic segmenter was the same as that used for the cross-lingual run. Every segmented Chinese word was wrapped in a proximity operator, forcing its characters to be ordered and adjacent. When more than one word, represented by a single character, occurred in a sequence, the characters were enclosed in a phrase operator and restricted to be within 25 terms of each other. This was to correct for possible errors in the segmenter. Single isolated terms were down weighted by 0.3.

Stop phrases and terms were automatically filtered from the query.

Queries were expanded using LCA. The expansion concepts were extracted from the top 10 passages of the TREC-9 converted Chinese database of passages and concepts. The expansion was similar to the cross-lingual run, segmented words being used as concepts. The expansion part of the query was weighted by 2.0 and 40 weighted concepts were added. The weights of the concepts were the same as those in the cross-lingual run. The expansion words were automatically filtered to remove stop phrases.

Experiments were run on TREC-6 to find the best configuration of number of top passages, number of

[9] http://www.cnd.org/software/UNIX.html

expansion concepts and weight of the expansion section of the query. The experiments were done with a weight of 2.0 and 2.5 for the expansion part, with the number of expansion terms at 10, 20,30,40, 50,60 and 70, and with the top 10,20 and 30 passages.

2.5 Cross-language conclusions

The submission consisted of four runs.

INQ7XL-1 was an official automatic run. It was cross-lingual using Chinese LCA only. The top twenty passages and ten expansion terms were used. The weight of the expansion part of the query was 1.25.

INQ7XL-2 was an official run. It was an official monolingual run using Chinese LCA. The top ten passages and forty expansion terms were used. The weight of the expansion part of the query was 2.0.

INQ7XL-3 was an optional run. It was cross-lingual using both English and Chinese LCA. The top 30 passages and 20 expansion terms were used for English and the top 20 passages and 10 expansion terms were used for Chinese. The weight of the expansion part of the query was 1.25

INQ7XL-4 was an optional run. It was cross-lingual using only English LCA. The top 30 passages and 20 expansion terms were used. The weight of the expansion part of the query was 1.25.

2.6 Cross-language results

The following table summarizes the cross-language results. We include the counts of comparisons with other systems because it gives a sense of where our approach worked well relative to others, and where it did not.

		AvgPrec comparison			
Run	AvgPrec	best	>avg	=avg	<avg
INQ7XL1	0.2416	3	12	0	10
INQ7XL2	0.2681	1	10	5	9
INQ7XL3	0.2425	2	17	1	5
INQ7XL4	0.2201	1	15	0	8

The table shows that all approaches were somewhat similar in their performance relative to other systems.

Not surprisingly, INQ7XL2, the monolingual run, did the best overall, though it is not clear why that is true from the table (i.e., the distribution compared to other sites is different, but almost seems worse). Looking at other results shows that INQ7XL2 had a substantially higher precision at 20 and at 100 documents retrieved, meaning that staying within a single language did a better job at the high-precision end of the curve.

3 Question Answering Track

Our work in the question answering track is similar to the previous year. Numerous post hoc experiments indicated that performance on the TREC-8 long (250-byte) answer runs could have been improved by using a different query generator and a different passage selection method. This prompted us to submit two 250-byte runs to test if this result was an artifact of the nature of the TREC-8 question set, or generally applicable to question processing.

For the first run, INQ9AND, we generated initial queries from the questions by stripping the question words (who, what, where). We then added in the following expansions, which were derived from the TREC-8 questions:

1. a few synonyms, such as `#syn`(length long), `#syn`(far distance)
2. a few alternations, such as `#syn(#uw5`(between somewhere and here) `#uw5`(from here to somewhere))
3. a proximity operator around superlative constructions, `#5`(largest mountain)

The resulting query was then wrapped in a probabilistic and (`#and`) operator. This approach resulted in roughly 10% higher performance on the TREC-8 question set.

The second run, INQ9WSUM, used the query generator from TREC-8, augmented with the the expansions listed above. These queries use the weighted sum (`#wsum`) operator, with all of the weights equal to 1.0 for the additional expansions. These augmentations produced a small (roughly 3%) improvement over the official INQ635 run from TREC-8.

For both runs, we used the LCA expansion of these queries to retrieve the top 20 documents. We then used the unexpanded query to retrieve the top passage from the top 5 documents from that set of 20. Based on our experiments with the TREC-8 questions, the unexpanded queries produce better answer passages than the LCA expanded queries. The passages are then reordered so that the rank order of the passages matches the rank order of the 20 documents retrieved with the LCA expanded query. The reordering step is performed because the LCA expanded queries tended to rank the answer containing documents higher than the unexpanded queries in our experiments with the TREC-8 question set.

Comparing the two runs, INQ9AND edges out INQWSUM on the aggregate score, 34.6% versus 33.6%. This is nowhere near the difference seen on the TREC-8 question set. Additionally, if we break down the scores by comparison against the median ranks for all of the systems, we find

	INQ9AND	INQ9WSUM
above	159 (23.3%)	171 (25.1%)
equal	370 (54.3%)	344 (50.4%)
below	153 (22.4%)	167 (24.5%)

that there is little difference between the two. INQ9WSUM produced 3 more answers than INQ9AND, and it produced 16 fewer rank 1 answers than INQ9AND (154 vs. 170). From this we conclude that there is some benefit from the changes based on the TREC-8 question set, but that the nature of the questions make them less than representative of real questions.

4 Query Track

The TREC-9 query track included 43 query files, variations of topics 51-100. There were three categories of the variations:

1. Very short: (2-3 words) based on topic.
2. Sentence: natural language, based on topic and judgements.
3. Manual Feedback: Manual NL sentence based on reading 5 or so relevant documents without reference to the topic (done by someone who doesn't have the topics memorized and who might use different vocabulary than the topic). This is an attempt to get a sentence which might use different vocabulary than the topic.

In this running of the track, we should get some feeling for whether query processing and expanding can improve the performance. So we did many experiments for different query processing techniques, and parameter settings for LCA query expansion.

4.1 Gathering queries

The queries were gathered from undergraduates in an information systems course at the University of Massachusetts. The information needed for query formulation was put on Web pages and the students were asked to formulate queries and submit them as part of a homework exercise. They received points for handing in validly formatted queries; there was no check to ensure that the quality of their queries were reasonable.

The students were randomly assigned two sets of five topics, based upon the last two digits of their student ID number (processing in advance determined that generated an even distribution over all the topics). The students were given these instructions:

> Each topic is listed with lots of descriptive information. In addition, there is a list of 5 or more documents that are known to be relevant to the topic. For each topics in set A of your batch do the following:
>
> - [short] Read the topic and come up with a short 2-3-word query that describes the topic as succinctly but completely as you can manage. You can think of this as queries that someone is likely to type on the Web, though that is not how they'll be used. Note that the `<title>` of the topic is not an example of a short query. An example of short query might be "popping corn methods."
> - [long] Now look at about 5 of the relevant documents for that same topic and generate a new and improved version of the query, this time coming up with a natural language, full sentence question or sentence that is on the topic specified, but also captures what it is about the relevant documents that makes them similar. Note that the relevant documents are known to be relevant, so there has to be something. An example might be, "What are the methods for popping corn that do not use a microwave?".
>
> Then for the five topics in set B of your batch, do the following:
>
> - [notopic] Look at about 5 of the relevant documents for the topic–without looking at the topic–and decide what it is about those documents that makes them all related to each other. Form a natural language, full-sentence statement or question that you think would be appropriate for retrieving those documents. For example, "How did people used to pop popcorn?" might be a query describing a bunch of documents that talk about ways to pop corn that seem to avoid talking about the microwave. Obviously there is a large amount of judgement here, so do not expect that everyone will have the same queries.

A total of 69 students provided queries in each of the three categories, meaning that 345 queries were provided in each category. Because some students failed to hand in the assignment, not all topics were evenly distributed. The end result was that the queries could be combined into five different sets of 50 queries for each of the different categories (long, short, notopic). The queries were combined somewhat randomly, though all five of any students queries were included in the same file. That means that any set of 50 queries includes queries generated by 10 different students.

Query sets INQ_a through INQ_e were created in Autumn 1998 and submitted for TREC-8 (in the same manner). Query sets INQ_f through INQ_j were created in Autumn 1999 and submitted for TREC-9.

Note that the generated queries were never checked for quality. It is theoretically possible that a student could submit nonsense queries. Some small problems of that nature happened and forced all participating sites to re-run their query track runs for some query sets.

4.2 Processing queries

For the query processing we ran experiments with four different query processors, the TREC-8 processor, the two QA processors described in the previous section, and another. The experiment results showed that the modified query processor performed better than all others.

The major difference between the modified one and the one used for TREC-8 concerns the handling of proper names. The TREC-8 query processor used to treat proper names differently from other phrases. When the TREC-8 processor identified a proper name, it used the unit proximity operater (i.e., #1) for a stronger constraint for the proper name occurrence. For the other phrases, it used the phrase operator (i.e., #phrase) and duplicated every single phrase term (i.e., terms occurring in the phrase). For example, in the topic 52, the country name *South Africa* was treated as a proper name, in the formatted query, it was

#passage25(#1(South Africa))

If *South Africa* was treated as other phrases, it would be

#passage25(#phrase(South Africa)) South Africa

The modification to that process loosens the constraint and treats all proper names as phrases. Overall experimental results from 5 query sets from TREC-8 (i.e., INQ3a, INQ3b, INQ3c, INQ3d, INQ3e) suggested this modification can improve the performace from 2% to 14%, with the average improved precision being 4.4%. However, this modification cannot guarantee improving the performance for every query, in which proper names occur. For example, the current processor can improve the topic 52 (containing *South Africa*) and 91 (containing *US Army*) with increased the average precision +83.5% and +860.2% respectively; but for the topics 69 (*SALT II*) and 70 (*Baby M*), the current processor makes the performance worse with decreased average precision -13.8% and -15.1% respectively.

There were two sets of experiments run for figuring out the best parameter setting for running LCA query expansion. One set was run with INQ3a and the other one was with INQ3d. In TREC-8 we did not run such experiments, we just borrowed the setting from the previous results of ad-hoc runs. The setting used in TREC-8 was selecting 50 expanded concepts from top 30 passages. For determining TREC-9 LCA setting we ran experiments trying from top 10 to 50 passages, and selected from 10 to 50 concepts for both query file INQ3a and INQ3d. The results showed that using top 50 passages and expanding 50 concepts can always improve the performance, the average precision was about 4% higher than using the TREC-8 settings (i.e., top 30 passages and 50 expanded terms).

We ran all 43 query sets that were part of the track using each of three different approaches: the query as is, basic query processing, and the addition of query expansion. More specifically:

- INQ7a — The queries were run by InQuery as is. InQuery stopped and stemmed the queries and treated them as if they were entered with InQuery's default `#sum` operator. That operator calculates the belief of every query term for a document, and then averages them.
- INQ7p — The queries were first passed through the query processing described above.
- INQ7e — After query processing, the queries were then expanded using LCA as described above.

For those who are curious: the INQ stands for InQuery, the name of the system used; the number 7 is an historical artifact of submission numbering from the start of the CIIR's participation in TREC; "a" and "p" and "e" should be self-explanatory.

4.3 Query track results

Overall, the three query processing approaches worked as expected. Query processing improved the quality of the queries, and expanding them improved the accuracy more. A brief summary over all 43 query sets shows:

Run	Rank	Below Average	Above Average	Top Rank	AvgPrec
INQ7a*	8	6	37	0	0.1799
INQ7p*	6	2	41	0	0.1848
INQ7e*	2	0	41	2	0.2288

Where "rank" respresents where (on average) the indicated run was in the set of 18 runs submitted. Note that the ranking is over query sets: so there were two query sets (of 43) that INQ7e did better than any of the other 18 runs.

4.4 Query track analysis

We did some more careful analysis of two of the queries to get a sense of how variant forms of the query failed or did not. In several parts of this analysis, we ran queries using the InQuery engine.

4.4.1 Analyzing Q51, Airbus subsidies

The original "query" was: *Document will discuss government assistance to Airbus Industrie, or mention a trade dispute between Airbus and a U.S. aircraft producer over the issue of subsidies.*

In looking at the various queries, there seemed to be four classes of terms that might be included in queries:

1. Key terms were terms without which the query did very poorly. For the query, any query without the term *airbus* did very poorly. The term as a query on its own yielded an average precision of 0.3364. Queries without that query tended to score about 0.01 average precision.
2. Supporting terms are those that are not useful on their own, but that improve results when they are included wiht a key term. For this query, the words *subsidy*, *Europe*, and *government* fell into this category. Some example queries and the effect on average precision:

Airbus	0.3364
Airbus *subsidy*	0.6029
Airbus *Europe*	0.4553
Airbus *government*	0.5543
Airbus *subsidy Europe government*	0.6483

3. Co-supporting terms help the query if supporting terms are present, but hurt the query if the supporting queries are missing. The word *industry* falls into that category here:

Airbus	0.3364
Airbus subsidy Europe government	0.6483
Airbus subsidy Europe government *industry*	0.6572
Airbus *industry*	0.3200

4. Noise terms are those that always hurt performance. For this query, *McDonnell* is an example of a noise term, even though on its surface it seems like it might be a useful term:

Airbus	0.3364
Airbus *McDonnell*	0.2253
Airbus subsidy Europe government	0.6483
Airbus subsidy Europe government *McDonnell Douglas*	0.4104

We also found several terms that affected the quality of the result, ranging from misspellings (e.g., "Concortium" or "Mcdonnel") to unusual stemming operations ("Boeing" to "boee"). The spelling errors clearly caused problems; the stemming areas are more amusing than anything else, since both queries are documents were stemmed with the same techniques. (Ironically, misspelling "McDonnell" turns out to be a win since "McDonnell" is a noise term.)

4.4.2 Analyzing Q93, NRA backing

The original "query" was: *What backing does the National Rifle Association have?*

We analyzed this query looking at the content of the query: what topics were mentioned and which were necessary. We found:

- Almost all queries had either *NRA* or *National Rifle Association* in them. The former was substantially less effective than the latter.
- Queries that included *gun* did well, but *handgun* (without *gun* also) performed terribly. When *gun* was combined with *National Rifle Association* the queries did very well.
- Words related to "support" or "backing" were important but there was no obvious pattern to what worked or did not.
- Additional "filler" words (e.g., "congress" or "people") had a range of effects, and those effects varied across systems. That is, "people" might hurt the query on one system, but have no effect on another.

For this query, we also looked at the effect of query expansion on the queries. We found that expanding queries with *NRA* made them as good as the original *National Rifle Association* queries, but that that latter set of queries doubled or tripled in effectiveness when expanded! The query expansion also removed the system differences do to handling of filler words.

5 Conclusions

We participated in three tracks this year. In the cross language track, we experimented some techniques for crossing the character encoding boundaries. Our efforts were moderately successful, but we do not believe that our approach worked well in comparison to other techniques.

In the question answering track, we focused on bringing answer-containing documents to the top of the ranked list. This is an important sub-task for most methods of tackling Q&A, and we are pleased with our results. We are now looking at alternate ways of thinking about that task that leverage the differences between retrieval for Q&A and for IR.

Finally, we continued to participate in the query track, providing large numbers of query variants, and running our system on the huge number of resulting queries. Our analysis showed how query expansion compensates for some of the problems that can occurs in query formulation.

6 Acknowledgements

This work was supported in part by the National Science Foundation, Library of Congress, and Department of Commerce under cooperative agreement number EEC-9209623, in part by the Air Force Office of Scientific Research under grant number F49620-99-1-0138, and in part by SPAWARSYSCEN-SD grant number N66001-99-1-8912. Any opinions, views, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

References

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

[Ponte and Croft, 1996] Ponte, J. and Croft, W. B. (1996). Useg: A retargetable word segmentation procedure for information retrieval. In *Proceedings of the Symposium on Document Analysis and Information Retrieval (SDAIR).*

[Robertson et al., 1995] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., and Gatford, M. (1995). Okapi at TREC-3. In Harman, D. K., editor, *The Third Text REtrieval Conference (TREC-3).* NIST.

[Xu and Croft, 1996] Xu, J. and Croft, W. B. (1996). 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. Association for Computing Machinery.