Cheshire II at INEX '03: Component and Algorithm Fusion for XML Retrieval

Ray R. Larson
School of Information Management and Systems
University of California, Berkeley
Berkeley, California, USA, 94720-4600
ray@sherlock.berkeley.edu

ABSTRACT

This paper describes the retrieval approach that UC Berkeley used in the 2003 INEX evaluation, and the subsequent analysis and correction of search failures in the "official runs". As in last year's INEX, our primary approach is a combination of probabilistic methods using a Logistic Regression (LR) algorithm for estimation of document (article) relevance and/or element relevance, along with Boolean constraints. This year we also used data fusion techniques to combine results from multiple probabilistic retrieval algorithms, specifically the Okapi BM-25 algorithm, and multiple search elements for any given query.

1. INTRODUCTION

Early in the TREC evaluations a number of participating groups found that fusion of multiple retrieval algorithms provided an improvement over a single search algorithm[13, 2]. With ongoing improvements of the algorithms used in the TREC main (i.e., ad hoc retrieval) task, later analyses[9, 1] found that the greatest effectiveness improvements appeared to occur between relatively ineffective individual methods, and the fusion of ineffective techniques, while often approaching the effectiveness of the best single IR algorithms, seldom exceeded them for individual queries and never exceeded their average performance.

Our approach to XML retrieval in last year's INEX, as reported in our 2002 INEX paper[6], was to use a "Fusion Search" facility in the Cheshire II system that merged the result sets from multiple searches. For the majority of the content-only and content and structure queries separate searches from different indexes and different elements of the collection were merged into a single integrated result set. This facility was developed originally to support combination of results from distributed searches, but has proved to be quite valuable when applied to the differing elements of a single collection as well.

One of the main questions we were investigating in the 2002 INEX was how to take advantage of more precise search matches (e.g. Boolean title searches) when they are possible for a given query, yet to permit the enhanced recall that probabilistic queries can provide. We found in subsequent analysis of the INEX 2002 results, that our implementation of this approach suffered significantly from a number of bugs. As noted in the final INEX 2002 paper, some of the bugs were found in the script that converted the results to the INEX submission format, not in retrieval itself, where only the first occurrence of component in a document was converted to an entry for the submission (this was most signicant in one query where all of the relevant components were in a single article).

We also discovered in analysis of the results from last year that Fusion Searches were not correctly accumulating scores for each component search in some cases. This turned out to be a particularly costly bug (in terms of the INEX performance measures) caused by a failure to sort some of the intermediate resultsets in searches before they were merged, leading to an incorrect ranking sequence in the final resultsets, and in some particularly pathological situations resulting in the effective reversal of the correct ranking sequence.

For the official INEX 2003 runs, the bugs noted above were corrected. But, unfortunately, others were discovered rather late in the evaluation process, which led to the worse-than-expected results obtained for the official runs (these bugs are described below in the discussion of retrieval score normalization in combining results from different indexes and algorithms). We now believe that most of the bugs have been corrected, which has led to significant improvements in the performance of both CO and SCAS searches in our "post-INEX" experiments.

Our principle approach this year was to expand on the basic fusion approach used last year, using a combination of new implementations of additional algorithms, and new operators for merging intermediate results from different algorithms and search elements. The major addition this year is that we have implemented, and employed, a version of the Okapi BM-25 algorithm. The remainder of this paper describes the retrieval algorithms, new methods for combining results for different elements, and discusses the comparative results for the different official runs and our subsequent runs with bugs corrected.

2. THE RETRIEVAL ALGORITHMS AND OPERATORS

The original design rationale and features of the Cheshire II search engine have been discussed elsewhere [8, 7] and will only be briefly repeated here with an emphasis on those features that were applied in the INEX evaluation. We will also describe our newly implemented algorithms and operators used in the official and subsequent runs.

2.1 Original Probabilistic and Boolean Operations

The Cheshire II search engine supports various methods for translating a searcher's query into the terms used in indexing the database. These methods include elimination of "noise" words using stopword lists (which can be different for each index and field of the data), particular field-specific query-to-key conversion or "normalization" functions, standard stemming algorithms (a modified version of the Porter stemmer) and support for mapping database and query text words to single forms based on the WordNet dictionary and thesaurus using a adaption of the WordNet "Morphing" algorithm and exception dictionary.

In his analysis of fusion approaches to improving retrieval performance, Lee[9] found that the best results were obtained by combining algorithms where similar sets of relevant documents were returned but that retrieved different sets of non-relevant documents. With this in mind, we chose for this research two probabilistic algorithms that at least partially fulfill this criteria. The first algorithm is based on logistic regression and the second is the well-known Okapi BM-25 algorithm. In this section we describe each algorithm as it was implemented for this evaluation.

2.2 Logistic Regression Algorithm

The *logistic regression* (LR) algorithm used in this study was originally developed at Berkeley by Cooper, et al.[4] and shown to provide good full-text retrieval performance in the TREC ad hoc task. As originally formulated, the LR model of probabilistic IR attempts to estimate the probability of relevance for each document based on a set of statistics about a document collection and a set of queries in combination with a set of weighting coefficients for those statistics. The statistics to be used and the values of the coefficients are obtained from regression analysis of a sample of a collection (or similar test collection) for some set of queries where relevance and non-relevance has been determined. More formally, given a particular query and a particular document in a collection $P(R \mid Q, D)$ is calculated and the documents or components are presented to the user ranked in order of decreasing values of that probability. To avoid invalid probability values, the usual calculation of $P(R \mid Q, D)$ uses the "log odds" of relevance given a set of S statistics, s_i, derived from the query and database, such that:

$$\log O(R \mid Q, D) = b_0 + \sum_{i=1}^{S} b_i s_i \qquad (1)$$

where b_0 is the intercept term and the b_i are the coefficients obtained from the regression analysis of the sample collection and relevance judgements.

Based on the structure of XML documents as a tree of XML elements, we define a "document component" as an XML subtree that may include zero or more subordinate elements or subtrees with text as the leaf nodes of the tree. Naturally, a full XML document may also be considered a document component. As discussed below, the indexing and retrieval methods used in this research take into account a selected set of document components for generating the statistics used in the search process and for extraction of the parts of a document to be returned in response to a query. Because we are dealing with not only full documents, but also document components (such as sections and paragraphs or similar structures) derived from the documents, we will use C to represent document components in place of D. Therefore, the full equation describing the LR algorithm used in these experiments is:

$$\begin{aligned}
\log O(R \mid Q, C) \quad = \quad & \\
& -3.70 + \left(1.269 \cdot \left(\frac{1}{|Q_c|}\sum_{j=1}^{|Q_c|} \log qtf_j\right)\right) \\
+ \quad & \left(-0.310 \cdot \sqrt{|Q|}\right) \\
+ \quad & \left(0.679 \cdot \left(\frac{1}{|Q_c|}\sum_{j=1}^{|Q_c|} \log tf_j\right)\right) \\
+ \quad & \left(-0.0674 \cdot \sqrt{cl}\right) \\
+ \quad & \left(0.223 \cdot \left(\frac{1}{|Q_c|}\sum_{j=1}^{|Q_c|} \log \frac{N - n_{t_j}}{n_{t_j}}\right)\right) \\
+ \quad & (2.01 \cdot \log |Q_d|)
\end{aligned} \qquad (2)$$

Where:

Q is a query containing terms T,

$|Q|$ is the total number of terms in Q,

$|Q_c|$ is the number of terms in Q that also occur in the document component,

tf_j is the frequency of the jth term in a specific document component,

qtf_j is the frequency of the jth term in Q,

n_{t_j} is the number of components (of a given type) containing the jth term,

cl is the document component length measured in bytes. and

N is the number of components of a given type in the collection.

This equation, used in estimating the probability of relevance in this research, is essentially the same as that used in [3]. The coefficients were estimated using relevance judgements and statistics from the TREC/TIPSTER test collection. In this evaluation we used the same coeffients for each of the main document components used. This means that we are treating all components smaller than a full document as if they were, in effect, small documents.

2.3 Okapi BM-25 Algorithm

The version of the Okapi BM-25 algorithm used in these experiments is based on the description of the algorithm in Robertson[11], and in TREC notebook proceedings[12]. As with the LR algorithm, we have adapted the Okapi BM-25 algorithm to deal with document components :

$$\sum_{j=1}^{|Q_c|} w^{(1)} \frac{(k_1+1)tf_j}{K+tf_j} \frac{(k_3+1)qtf_j}{k_3+qtf_j} \quad (3)$$

Where (in addition to the variables already defined):

K is $k_1((1-b)+b \cdot dl/avcl)$

k_1, b and k_3 are parameters , 1.5, 0.45 and 500, respectively, were used,

avcl is the average component length measured in bytes

$w^{(1)}$ is the Robertson-Sparck Jones weight:

$$w^{(1)} = \log \frac{(\frac{r+0.5}{R-r+0.5})}{(\frac{n_{t_j}-r+0.5}{N-n_{t_j}-R-r+0.5})}$$

Where, for a given query and a given term:

r is the number of relevant components of a given type that contain a given term,

R is the total number of relevant components of a given type for the query. (Note that these statistics do not take into account nested components.)

Our current implementation uses only the *a priori* version (i.e., without relevance information) of the Robertson-Sparck Jones weights, and therefore the $w^{(1)}$ value is effectively just an IDF weighting. The results of searches using our implementation of Okapi BM-25 and the LR algorithm seemed sufficiently different to offer the kind of conditions where data fusion has been shown to be most effective [9].

2.4 Boolean Operators

The Cheshire II system used in the evaluation supports searches combining probabilistic and (strict) Boolean elements, as well as operators to support various merging operations for both types of intermediate result sets. Although strict Boolean operators and probabilistic searches are implemented within a single process, using the same inverted file structures, they really function as two parallel *logical* search engines. Each logical search engine produces a set of retrieved documents. When a single search strategy is used the result is either a probabilistically ranked set or an unranked Boolean result set. When both are used within in a single query, combined probabilistic and Boolean search results are evaluated using the assumption that the Boolean retrieved set has an estimated $P(R \mid Q_{bool}, C) = 1.0$ for each document component in the set, and 0 for the rest of the collection. The final estimate for the probability of relevance used for ranking the results of a search combining strict Boolean and probabilistic strategies is simply:

$$P(R \mid Q, C) = P(R \mid Q_{bool}, C)P(R \mid Q_{prob}, C)$$

where $P(R \mid Q_{prob}, C)$ is the probability of relevance estimate from the probabilistic portion of the search, and $P(R \mid Q_{bool}, C)$ is the Boolean. In practice the combination of strict Boolean "AND" and the probablistic approaches has the effect of restricting the results to those items that match the Boolean portion, with ranking based on the probabilistic portion. Boolean "NOT" provides a similar restriction of the probabilistic set by removing those document components that match the Boolean specification. When Boolean "OR" is used, the probabilistic and Boolean results are merged (however, items that only occur in the Boolean result, and not both, are reweighted as in the "fuzzy" and merger operations described below.

A special case of Boolean operator in the experimental system is that of proximity and phrase matching operations. In proximity and phrase matching the matching terms must also satisfy proximity constraints (both term order and adjacency in the case of phrases). Thus, proximity operations also result in Boolean intermediate result sets.

2.5 Result Combination Operators

Cheshire II provides a number of ways to using "FUZZY", "RESTRICT" and "MERGE" operators to combine intermediate results of a search from different components or indexes. With these operators we have available an entire spectrum of combination methods ranging from strict Boolean operations to fuzzy Boolean and normalized mean scores for probabilistic and Boolean results.

Fuzzy operators are versions of the Boolean operators that are less "strict" than the conventional Boolean operators, applied to weighted result lists. In place of Boolean AND, the "!FUZZY_AND" operator takes the mean of the two weights in the result sets for the same record (this differs from the conventional fuzzy AND that take the minimum of the two weight). The "!FUZZY_OR" takes the largest of the two weights for the same record. "!FUZZY_NOT" currently behaves the same way as strict Boolean "NOT". Otherwise these operators are used the same way as the strict Boolean operators.

The "!RESTRICT_TO" and "!RESTRICT_FROM" operators take either a component result and a document result, or two component results (where one component contains the other). As discussed in [6], "components" in the Cheshire II system can be the contents of any tag (or of a set of tags) that are treated as separate documents for the purposes of indexing and retrieval. In the case of component and document results the component list is restricted to components that are in the document result – the matching components only are returned retaining their weight from the original component result. When two nested component results are used with these operators the result is larger components that include one or more of the smaller compo-

nents. (Note that with component and document results !RESTRICT_TO and !RESTRICT_FROM may be used interchangibly and the type of operation to be performed is determined by the nature of the result sets, but with two component results the nesting of the elements must be taken into account in constructing the query (i.e, Parent_set !RESTRICT_FROM Child_set or Child_set !RESTRICT_TO Parent_set). Naturally Parent and Child can be any sub-query that results in the appropriate kind of component.

The !MERGE_SUM operator combines the two resultsets (like a Boolean OR) but adds the weights (actually the resulting raw ranking adds 1.0 to the probabilistic result and sets 1.5 for Boolean results with matching document or component ids in both lists, and the original values for items found only in a single result). Note that !MERGE_SUM weights may exceed 1 and are not probabilities.

The !MERGE_MEAN operator combines the two resultsets (like a Boolean OR) but takes the MEAN (or average) of the weights from items in both lists and half of the weight of items in only a single list. This is the (currently) recommended operator for merging probabilistic resultsets.

The !MERGE_NORM operator combines the two resultsets (like !MERGE_MEAN) but it performs the min-max normalization of the weights suggested by Lee[9] before it takes the mean of the weights from items in both lists and half of the weight of items in only a single list. There was a bug in this process in the official runs, because items in only one of the two input lists were neither normalized nor divided in half. This effect of this bug was that items occurring in only a *single* result set, among the many partial results merged for each of the queries, were likely to receive *higher* weights in the final results than items occurring in many (or all) of the partial results.

The motivation for these new operators follows from the basic observation that has driven all research into data fusion methods in IR, that no single retrieval algorithm has been consistently proven to be better than any other algorithm for all types of searches. By providing a set of operators for combining the retrieved sets from different search strategies, we are hoping to capitalize the strengths of particular algorithms while reducing their limitations. In general, the assumption behind any implementation of data fusion is that the more evidence the system has about the relationship between a query and a document (including the sort of structural information about the documents found in the INEX queries), the more accurate it will be in predicting the probability that the document will satisfy the user's need. Other researchers have shown that additional information about the location and proximity of Boolean search terms can be used to provide a ranking score for a set of documents[5]. The inference net IR model has shown that the exact match Boolean retrieval status can be used as additional evidence of the probability of relevance in the context of a larger network of probabilistic evidence[14]. In the same way, we treat the set of documents resulting from the exact match Boolean query as a special case of a probabilistically ranked set, with each retrieved document having an equal rank.

3. INEX APPROACH

Our approach in INEX was to use all of the original and new features of the Cheshire II system in generating the results submitted for our official runs. This section will describe the indexing process and indexes used, and also discuss the scripts used for search processing. The basic database was unchanged from last year's. We did, however, create and use a number of additional indexes and performed a complete reindexing of the INEX document collection. This section will first describe the indexes and component definitions created for INEX 2003.

3.1 Indexing the INEX Database

All indexing in the Cheshire II system is controlled by an SGML Configuration file which describes the database to be created. This configuration file is subsequently used in search processing to control the mapping of search command index names (or Z39.50 numeric attributes representing particular types of bibliographic data) to the physical index files used and also to associated component indexes with particular components and documents.

As noted above, any element or attribute may be indexed. In addition particular values for attributes of elements can be used to control selection of the elements to be added to the index. The configuration file entry for each index definition includes three attributes governing how the child text nodes of the (one or more) element paths specified for the index will be treated.

Each index can have its own specialized stopword list, so that, for example, corporate names have a different set of stopwords from document titles or personal names.

Most of the indexes used in INEX used keyword or keyword with proximity extraction and stemming of the keyword tokens. Exceptions to this general rule were date elements (which were extracted using date extraction of the year only) and the names of authors which were extracted without stemming or stoplists to retain the full name.

Other than the conversion of some indexes from keyword to keyword with proximity, the indexes and component elements for INEX 2003 were the same as those used in the 2002 evaluation[6].

Altogether, 27 separate indexes and 5 types of components (in addition to article-level) were used in search evaluation runs of the 2003 INEX topics. The official submitted runs in INEX are described in the next section.

3.2 INEX '03 Official Runs

Berkeley submitted six retrieval runs for INEX 2003, three CO runs and 3 SCAS runs. We did not submit any VCAS runs. This section describes the individual runs and general approach taken in creating the queries submitted against the INEX database and the scripts used to do the submission. All of the official runs were automatic, with queries generated by scripts that used title and keyword sections for the CO runs, and the title only for the SCAS runs. (The corrected runs described later also use automatic query generation with the same topic elements).

Berkeley_CO01: This run used LR ranking combined with Boolean phrase matching and MERGE_MEAN partial result combinations. Only article level results are returned in this run.

Berkeley_CO_Okapi: This run employed the Okapi BM-25 algorithm for ranked search components, combined with Boolean elements for proximity and term restrictions. Results from multiple components where combined using MERGE_MEAN merging of results. RSV scores were normalized and multiple result sets combined to include Article-level, section-level and paragraph-level results.

Berkeley_CO_MergePrOk: This run was a fusion of LR and Okapi algorithms using a score-normalized merging algorithm (MERGE_NORM). Results from multiple components where combined using MERGE_MEAN and MERGE_NORM merging of results. Separate retrieval of Articles, Sections and paragraphs were combined using score normalized merges of these results.

Berkeley_SCAS01: Used LR ranking combined with Boolean phrase matching and MERGE_MEAN partial result combinations. FUZZY_AND and FUZZY_OR operators were used in combining AND and OR elements within an "about" predicate.

Berkeley_SCAS_Okapi: Used the Okapi BM-25 ranking instead of LR and used normalized scores in merging results from different aspects of the queries. Results from multiple components used the MERGE_NORM operator for merging of results.

Berkeley_SCAS_Okapi2: Was similar to the above run, except for the use of some different indexes (including more of the document text).

4. EVALUATION

The summary average precision results for the official runs described above are shown in Table 1.

Run Name	Short name	Avg Prec (strict)	Avg Prec (gen.)
Berkeley_CO01	Prob	0.0467	0.0175
Berkeley_CO_Okapi	Okapi	0.0318	0.0314
Berkeley_CO_MergePrOk	MergePrOK	0.0546	0.0557
Berkeley_SCAS01	Prob_SCAS	0.1970	0.1545
Berkeley_SCAS_Okapi	Okapi_SCAS	0.0865	0.0682
Berkeley_SCAS_Okapi2	Okapi2_SCAS	0.0869	0.0687

Table 1: Cheshire Official Runs for INEX 2003

Figures 1 and 2 show, respectively, the Recall/Precision curves for generalized quantization of each the SCAS and CO results of the officially submitted Berkeley runs. None of Berkeley runs appeared in the top ten for all submitted runs. The results, as discussed above, particularly for the the Okapi-based runs have relatively poor results due to implementation errors. It is, however, worth noting that the fusion results (MergePrOk) did perform better than either the probabilistic or (flawed) Okapi runs for the CO task. Thus, the issue that we were seeking to investigate (whether XML retrieval would benefit from data fusion methods operating across both elements and algoriths, had some cautious confirmation from the official runs. The MergePrOK run which combined results for both LR and Okapi algorithms showed a marked improvement over the Okapi run alone. However The high-end precision in that run was less than in the Prob run, this may however be due to the bug described previously. In addition, it is likely that if the logistic regress algorithm run (Prob) had included section and paragraph elements, it would probably have had much better overall performance.

4.1 Post-INEX CO Results

A large number of subsequent tests were run evaluate the causes of the relatively poor performance shown in the the official results, and to track down and correct the bugs discussed above. After correction of these problems, a number of tests were run to evaluate the corrected baseline performance for the LR and Okapi algorithms for the CO task. The result for these runs are shown in Tables 2 and 3. Run names that include "_full" in the name include expansions of the topic terms in the queries to include proximity-based search for quoted phrases, query term weight enhancements for "+" terms and Boolean NOT. Thus, "prob_full" and "okapi_full" use the LR and Okapi algorithms, respectively, and include the full expansion. Run names with "_base" use just the particular algorithm with no term expansions or reweighting.

For fusion operations between different indexes for a particular document component, the MERGE_NORM operator was used to combine the sub-query results. In Tables 2 and 3 "fusion_full" combines full queries of only the topic, sec_words, and para_words indexes for both LR and Okapi, "fusion_t_full" combines both the topic, alltitles, sec_words, sec_title, and para_words, "fusion_ta_full" adds the abstract index to this. As in the preceding, "fusion_t_p_abs_full" and "fusion_t_p_abs_full" use the same indexes, but perform an additional LR search of the abstract and extract and merge the abstract in the final results used in evaluation.

The fusion approaches that we have been been exploring attempt to consider both the optimal combinations of search elements and algorithms that should used in the retrieval process. For this evaluation we have not re-estimated the logistic reression parameters or examined the possibility of differential weightings that could be applied to the search elements to best estimate the probability of relevance for a given query and document element, or combination of elements.

The summary average precision results for the runs described above are shown in Tables 2 and 3 for the strict and generalized quantization of the INEX evaluation metrics. In these tables ΔP shows the percentage difference for the test from the "prob_base" baseline and 'ΔO shows the difference from "okapi_base".

Figures 3 and 4 show the Recall/Precision curves for generalized quantization of the base algorithms (prob_base and okapi_base) in combination with the full expanded queries (Figure 3) or the best performing fusion query (fusion_t_full).

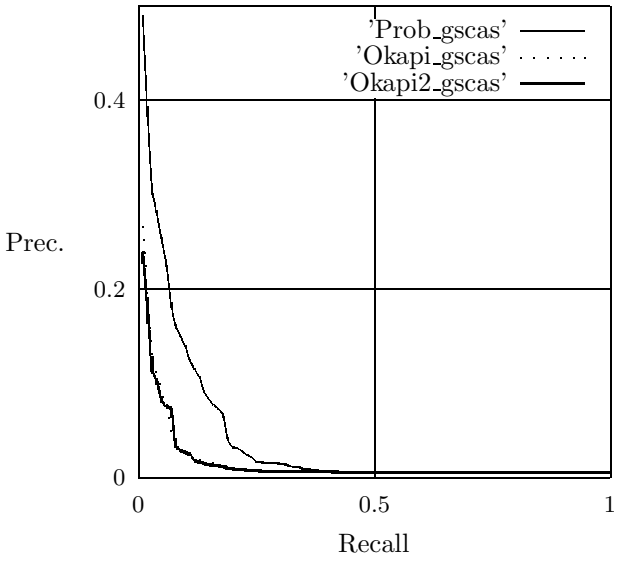

Figure 1: Official SCAS Runs (generalized)

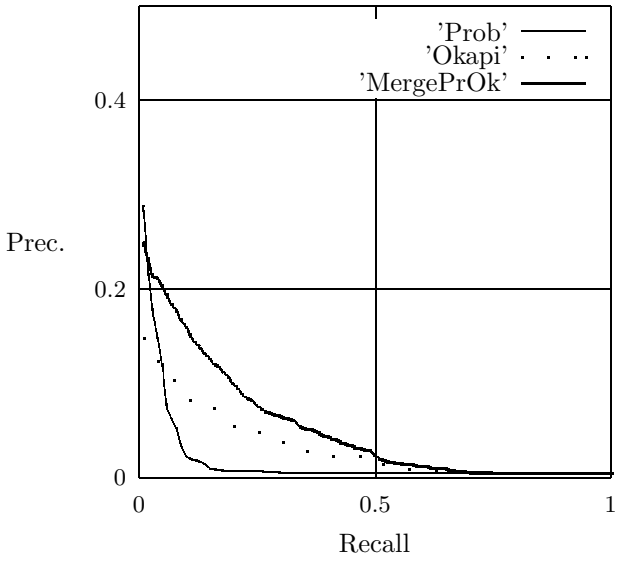

Figure 2: Official CO Runs (generalized)

Run Name	MAP	Δ P	Δ O
fusion_t_full	0.0690	22.97	18.99
fusion_t_p_abs_full	0.0635	16.26	11.94
fusion_ta_full	0.0632	15.89	11.55
fusion_full	0.0600	11.37	6.80
prob_full	0.0589	9.72	5.07
fusion_ta_p_abs_full	0.0584	9.00	4.30
okapi_full	0.0563	5.51	0.63
okapi_base	0.0559	4.90	0.00
prob_base	0.0532	0.00	-5.16

Table 2: Post Evaluation of CO Queries: Mean Average Precision of different algorithms and search element combinations (strict)

Run Name	MAP	Δ P	Δ O
fusion_t_full	0.0741	15.89	4.85
fusion_full	0.0739	15.61	4.53
okapi_full	0.0730	14.58	3.37
fusion_ta_full	0.0725	14.07	2.79
fusion_t_p_abs_full	0.0712	12.49	1.01
okapi_base	0.0705	11.60	0.00
fusion_ta_p_abs_full	0.0698	10.75	-0.96
prob_full	0.0690	9.70	-2.15
prob_base	0.0623	0.00	-13.12

Table 3: Post Evaluation of CO Queries: Mean Average Precision of different algorithms and search element combinations (generalized)

As Tables 2 and 3 indicate, the use of query expansion, as discussed in section 3.2, appears to offer some benefit of the unexpanded query for both quantizations, prob_full shows improvement over prob_base and okapi_full shows improvement over okapi_base. What is somewhat more interesting is that under strict quantization the LR approach in prob_full performs better than either okapi test, but for generalized quantization both Okapi tests perform better than either LR test (and indeed better than some of the fusion approaches. This implies that the Okapi algorithm is better at identifying a wider range of degrees of perceived relevance, while the LR algorithm is better at identifying the highly relevant items.

When the two algorithms are combined (with only topic and word searches in fusion_full) the results for both the strict and generalized measures are better than any of the single algorithms. This is different from the kind of results reported in [1], and seems to confirm the improvements from data fusion reported by Lee[9]. When the searches include a separate ranking of title searches merged with the topic searches the performance is further improved and performs the best for both quantizations of all of the query forms examined here. However, it appears that element indexes cannot be arbitrarily combined in attempting to improve performance, adding the abstract index results in reduced performance relative to topic and titles alone.

4.2 Post-INEX SCAS Results

Some of the subsequent SCAS runs are shown in Table 4. The table shows that the LR-based queries (indicated by "scas.p" in the names) seem to be generally less effective than the Okapi-based queries (including "scas.o" in the run names). Of course, the SCAS queries are in general more complex than the CO queries, and make use of many additional merging operations (such as the "RESTRICT" operators) driven by the individual Xpath queries. The runs with the same number, used the same combinations of merge operators and differ only in the ranking algorithm employed. The Fusion runs (indicated by name with "scas.fus" each combine results from different runs, those with numbers only in the last part of the name are Okapi only runs, and

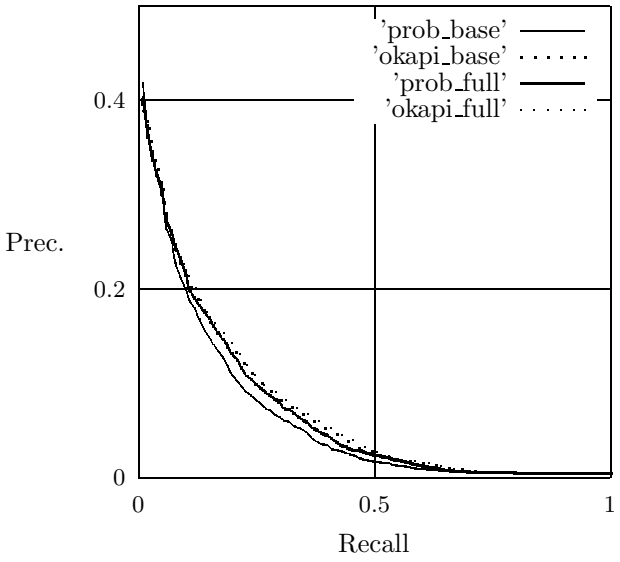

Figure 3: Recall-Precision of LR and Okapi retrieval algorithms for CO (generalized quantization)

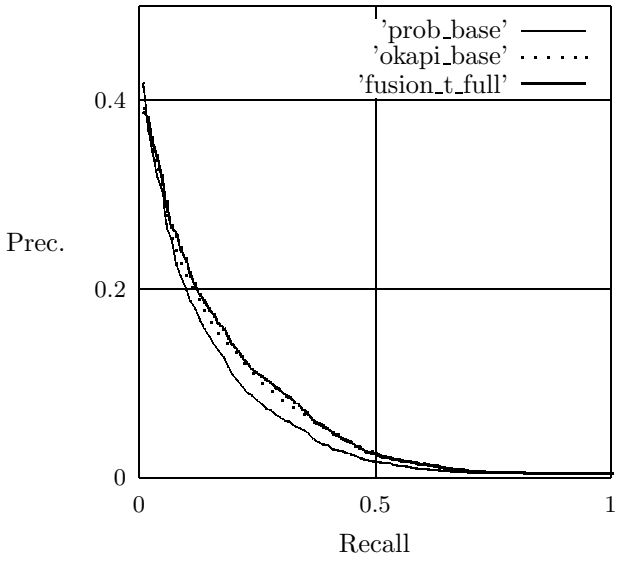

Figure 4: Recall-Precision of the best fusion method compared to algorithm baselines for CO (generalized quantization)

Run Name	Avg Prec (gen.)	Avg Prec (strict)
scas.fus.258	0.2107	0.2403
scas.fus.78	0.2075	0.2395
scas.fus.p28o8	0.1985	0.2304
scas.fus.p8o87	0.2020	**0.2444**
scas.o.2	0.2010	0.2205
scas.o.7	0.1996	0.2247
scas.o.8	**0.2120**	0.2308
scas.p.2	0.1877	0.2092
scas.p.8	0.1948	0.2174

Table 4: Post Evaluation of SCAS Queries: Mean Average Precision of different algorithms and search element combinations

the others mix LR and Okapi runs. The best performing SCAS run for the generalized evaluation metrics was an Okapi run that used the "MERGE_NORM" operator when a "AND" was used in an "about" clause in a query, and "MERGE_SUM" was used for "OR". For Xpath expression with separate "about" clauses in nodes on different levels in the document tree, the "RESTRICT_FROM" operators were used. Terms with "+", "-", and quotes were handled the same way as in the CO runs, with added search elements for exact phrase matching, additional query term weighting for "+" and use of Boolean "NOT" for "-".

Figures 5 and 6 show the generalized recall-precision metrics for the SCAS runs above. Figure 5 shows the LR and Okapi results and Figure 6 shows the different fusion results.

5. CONCLUSIONS

The results reported here are the first evaluation of the new fusion and resultset merging operators in the Cheshire II system. In exploring the fusion of different algorithms and document components in content-oriented and structured XML retrieval we have obtained some encouraging results. The results indicate that several of the fusion approaches that we tested do perform better than the individual algorithms, and that some Boolean constraints seem to be beneficial for XML retrieval. This is different from most studies of fusion methods, where the fusion is of different algorithms for the same collection of full documents [1, 10]. Because we are combining not only full document results, but also component elements of documents, we believe that the results benefit from the differing selectivity of different document components, when those can be merged into a single ranked list.

However, there is much room for further study, in particular this study did not include language models of XML, which have proved to be highly effective in the INEX evaluations. Future work will extend the Cheshire II system to include language model-based XML retrieval algorithms and test it in combination with the logistic regression and Okapi algorithms tested here.

When using the LR algorithms, as described above, the *same* weighting coefficients were applied to the statistics from *all* components ranging from full documents to paragraphs and titles. We plan to investigate a new implementation of the logistic regression algorithm where these coefficients will be estimated for each component type using a training sample of those components and their matching relevance judgements. Thus, the weighting coefficients applied to component length, for example, might be quite different depending on the component type. This can be expected to provide better tuned weighting coefficients and hence ranking values for the individual components and should, in turn, improve

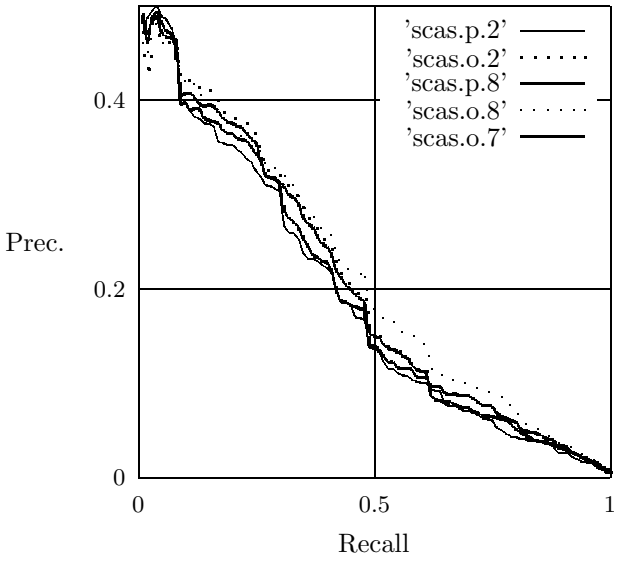

Figure 5: Recall-Precision of LR and Okapi retrieval algorithms for SCAS (generalized quantization)

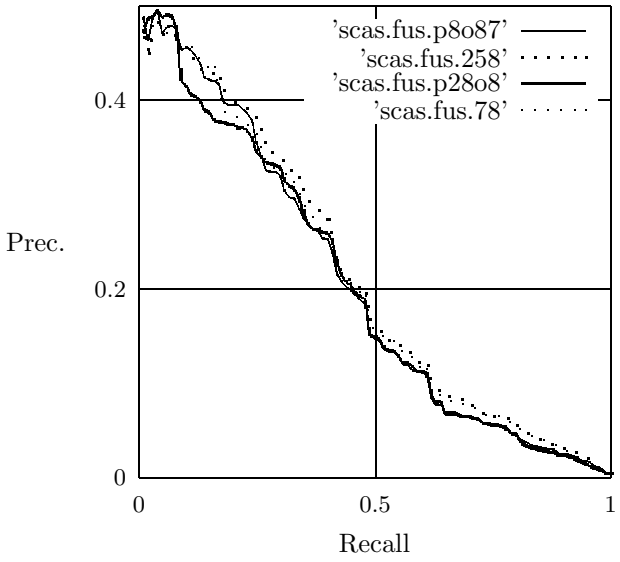

Figure 6: Recall-Precision of fusion approaches for SCAS (generalized quantization)

the fusion of components.

6. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation and Joint Information Systems Committee(U.K) under *NSF International Digital Libraries Program* award #IIS-9975164.

7. REFERENCES

[1] S. M. Beitzel, E. C. Jensen, A. Chowdhury, O. Frieder, D. Grossman, and N. Goharian. Disproving the fusion hypothesis: An analysis of data fusion via effective information retrieval strategies. In *Proceedings of the 2003 SAC Conference*, pages 1–5, 2003.

[2] N. Belkin, P. B. Kantor, E. A. Fox, and J. A. Shaw. Combining the evidence of multiple query representations for information retrieval. *Information Processing and Management*, 31(3):431–448, 1995.

[3] W. S. Cooper, F. C. Gey, and A. Chen. Full text retrieval based on a probabilistic equation with coefficients fitted by logistic regression. In D. K. Harman, editor, *The Second Text Retrieval Conference (TREC-2) (NIST Special Publication 500-215)*, pages 57–66, Gaithersburg, MD, 1994. National Institute of Standards and Technology.

[4] W. S. Cooper, F. C. Gey, and D. P. Dabney. Probabilistic retrieval based on staged logistic regression. In *15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Copenhagen, Denmark, June 21-24*, pages 198–210, New York, 1992. ACM.

[5] M. A. Hearst. Improving full-text precision on short queries using simple constraints. In *Proceedings of SDAIR '96, Las Vegas, NV, April 1996*, pages 59–68, Las Vegas, 1996. University of Nevada, Las Vegas.

[6] R. R. Larson. Cheshire II at INEX: Using a hybrid logistic regression and boolean model for XML retrieval. In *Proceedings of the First Annual Workshop of the Initiative for the Evaluation of XML retrieval (INEX)*, pages 18–25. DELOS workshop series, 2003.

[7] R. R. Larson and J. McDonough. Cheshire II at TREC 6: Interactive probabilistic retrieval. In D. Harman and E. Voorhees, editors, *TREC 6 Proceedings (Notebook)*, pages 405–415, Gaithersburg, MD, 1997. National Institute of Standards and Technology.

[8] R. R. Larson, J. McDonough, P. O'Leary, L. Kuntz, and R. Moon. Cheshire II: Designing a next-generation online catalog. *Journal of the American Society for Information Science*, 47(7):555–567, July 1996.

[9] J. H. Lee. Analyses of multiple evidence combination. In *SIGIR '97: Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 27-31, 1997, Philadelphia*, pages 267–276. ACM, 1997.

[10] M. E. Renda and U. Straccia. Web metasearch: Rank vs. score based rank aggregation methods. In *Proc. of the 18th Annual ACM Symposium on Applied Computing*, Melbourne, Florida, 2003. ACM Press.

[11] S. E. Robertson and S. Walker. On relevance weights with little relevance information. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 16–24. ACM Press, 1997.

[12] S. E. Robertson, S. Walker, and M. M. Hancock-Beauliee. OKAPI at TREC-7: ad hoc, filtering, vlc and interactive track. In *Text Retrieval Conference (TREC-7), Nov. 9-1 1998 (Notebook)*, pages 152–164, 1998.

[13] J. A. Shaw and E. A. Fox. Combination of multiple searches. In *Proceedings of the 2nd Text REtrieval Conference (TREC-2), National Institute of Standards and Technology Special Publication 500-215*, pages 243–252, 1994.

[14] H. Turtle and W. B. Croft. Inference networks for document retrieval. In J.-L. Vidick, editor, *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, pages 1–24, New York, 1990. Association for Computing Machinery, ACM.