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. As in last year's INEX, our primary approach is the combination of a probabilistic methods using a Logistic regression 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 and multiple search elements for any given query. All of our runs were fully automatic with no manual editing or interactive submission of queries.

Keywords

Information Retrieval, IR Evaluation, XML Retrieval

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[10, 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[7], 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. At the time of last year's runs for INEX the only operators for combining the results of multiple subqueries within a single query were the Boolean operators AND, OR, and NOT. The Fusion search facility itself used only a simple weighted sum of scores from each input resultset. In Berkeley's 2002 INEX runs, our approach typically involved between 7 and 14 separate queries of the system that were then combined using the fusion search facility to determine the final ranking of the documents or components.

One of the main questions we were investigating in last year's 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 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 the script that converted the results to the INEX submission format, not in retrieval itself, where only the first occurrence of component retrieved for some of the queries 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 apparently 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 a 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. This bug was difficult to detect due to the complex interactions in combining multiple resultsets.

For the official INEX 2003 runs, the bugs noted above were corrected. But, unfortunately, another one was discovered rather late in the evaluation process, which may explain (in part) the worse-than-expected results obtained for the official runs (described below in the discussion of MERGE_NORM normalization).

Our principle approach this year was to expand on the basic fusion approach used last year, using both new implementations of algorithms, and new fusion operators. A major addition this year is that we have implemented, and employed, a version of the Okapi BM-25 algorithm (as well as other algorithms not used in the submitted runs). We have not, however, used blind relevance feedback with the Okapi algorithm which may explain (in part) the poor results obtained. The remainder of this paper describes the retrieval algorithms used, new methods for combining results for different elements, and discusses the comparative results for the different official runs. We hope to have versions of the official runs with all (known) bugs repaired to present at the meeting.

2. THE RETRIEVAL ALGORITHMS AND OPERATORS

The original design rationale and features of the Cheshire II search engine have been discussed elsewhere [9, 8] and will only be briefly repeated here with an emphasis on those features that were applied in the INEX evaluation. (This part is virtually identical to the discussion in last year's paper [7], and repeated only for reference). We will then describe our newly implemented algorithms and operators.

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.

The primary ranked retrieval operation for Cheshire II makes use of a *logistic regression* algorithm to estimate probability of relevance that was originally developed by Berkeley researchers and shown to provide excellent full-text retrieval performance in the TREC evaluation of full-text IR systems[5, 4, 3]. In this algorithm, the estimated probability of relevance given a particular query and a particular record in the database $P(R \mid Q, D)$ is calculated and the documents or components are presented to the user ranked in order of decreasing values of that probability. The actual implementation for calculating $P(R \mid Q, D)$ uses the "log odds" of relevance $\log O(R \mid Q, D)$, where for any events A and B the odds $O(A \mid B)$ is a simple transformation of the probabilities $\frac{P(A|B)}{P(\overline{A}|B)}$. The Logistic Regression model provides estimates for a set of coefficients, c_i, associated with a set of S statistics, X_i, derived from the query and database, such that

$$\log O(R \mid Q, D) \approx c_0 \sum_{i=1}^{S} c_i X_i$$

where c_0 is the intercept term of the regression.

For the set of M *terms* (i.e., words, stems or phrases) that occur in both a particular query and a given document or document component, the equation used in estimating the probability of relevance for the Cheshire II search engine is essentially the same as that used in [4] where the coefficients were estimated using relevance judgements from the TIPSTER test collection:

$X_1 = \frac{1}{M}\sum_{j=1}^{M} logQAF_{t_j}$. This is the log of the absolute frequency of occurrence for term t_j in the query averaged over the M terms in common between the query and the document or document component. The coefficient c_1 used in the current version of the Cheshire II system is 1.269.

$X_2 = \sqrt{QL}$. This is square root of the query length (i.e., the number of terms in the query disregarding stopwords). The c_2 coefficient used is -0.310.

$X_3 = \frac{1}{M}\sum_{j=1}^{M} logDAF_{t_j}$. This is is the log of the absolute frequency of occurrence for term t_j in the document (or component) averaged over the M common terms. The c_3 coefficient used is 0.679.

$X_4 = \sqrt{DL}$. This is square root of the document or component length. In Cheshire II the raw size of the document or component in bytes is used for the document length. The c_4 coefficient used is -0.0674.

$X_5 = \frac{1}{M}\sum_{j=1}^{M} logIDF_{t_j}$. This is is the log of the *inverse document frequency*(IDF) for term t_j in the document averaged over the M common terms. IDF is calculated as the total number of documents or components in the database, divided by the number of documents or components that contain term t_j The c_5 coefficient used is 0.223.

$X_6 = logM$. This is the log of the number of terms that are in both the query and in the document or component. The c_6 coefficient used in Cheshire II is 2.01.

These coefficients and elements of the ranking algorithm have proven to be quite robust and useful across a broad range of document and component types.

The system also supports searches combining probabilistic and (strict) Boolean elements, as well as the FUZZY, RESTRICT and MERGE operators discussed above. 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 only one type of search strategy is used then the result is either a probabilistically ranked set or an unranked Boolean result set (these can also be sorted). When both are used in a single query, combined probabilistic and Boolean search results are evaluated using the assumption that the Boolean retrieved set has an estimated $P(R \mid Q_{bool}, D) = 1.0$ for each document 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:

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

where $P(R \mid Q_{prob}, D)$ is the probability estimate from the probabilistic portion of the search, and $P(R \mid Q_{bool}, D)$ the estimate from the Boolean. This has the effect of restricting the results to those items that match the Boolean portion, with ordering based on the probabilistic portion.

2.2 New Fusion Operators

The runs submitted this year make use of a set of new operators for the Cheshire II system. These are the FUZZY, RESTRICT and MERGE operators. 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 [7], "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 components. (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 + a probabilistic result and 1.5 for boolean results with matching document or component ids in both lists, and the original value 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 normalizes the weights using MIN_MAX normalization 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. This was where (one) bug occurred in the official runs, because items from a single list were neither normalized or divided in half. This implications of this bug, was that for our official runs using MERGE_NORM, items that occurred 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 ranked results than items that occurred in many (or all) of the partial results.

With the new fusion operators described above, the system has available an entire spectrum of combination methods ranging from strict Boolean operations to fuzzy Boolean and normalized mean scores for probabilistic and Boolean results. The motivation for the 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, 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[6]. 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.

2.3 Okapi Implementation

The version of the Okapi BM-25 algorithm implemented in the Cheshire II system is based on the description of the algorithm in Robertson[11], and in TREC notebook proceedings[12].

$$\sum_{T \in Q} w^{(1)} \frac{(k_1+1)tf}{K+tf} \frac{(k_3+1)qtf}{k_3+qtf}$$

Where:

Q is a query containing terms T

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

k_1, b and k_3 are parameters , usually 1.2, 0.75 and 7-1000

tf is the frequency of the term in a specific document

qtf is the frequency of the term in a topic from which Q was derived

dl and *avdl* are the document length and the average document length measured in some convenient unit

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

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

Where, for a given query and a given term:

r is the number of relevant items indexed with a given term

R is the total number of relevant items for the query

n is the number of items indexed with a given term

N is the number of items in the collection

The reason for adopting the Okapi BM-25 algorithm was twofold. First we wanted to provide to users of the Cheshire II system one of the most widely used and best performing algorithms. Secondly, we wanted to be able to do comparative testing and evaluation using this algorithm. The documents weights generated by the Okapi algorithm are on a quite different scale from those generated by the logistic regression algorithm, and they result in subtle differences in the final ranks of documents for the same database and query. Our current implementation uses only the *a priori* version (i.e., without relevance information) of the Robertson-Sparck Jones weights, and thus $w^{(1)}$ is effectively just an IDF weighting. However, because the results from this implementation of Okapi and the Logistic Regression were sufficiently different, they seemed to offer the kind of conditions where data fusion has been shown to be be most effective [10, 1].

3. INEX APPROACH

Our approach in INEX was to use all of the original and new features of the cheshire 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. These attributes are:

1. ACCESS: The index data structure used (all of the indexes for INEX used B-TREE indexes).
2. EXTRACT: The type of extraction of the data to be performed, the most common are KEYWORD, or EXACTKEY. EXACTKEY takes the text nodes as a string with order maintained for left-to-right key matching. KEYWORD takes individual tokens from the text node. In addition there is support for extraction of proximity information as well. We made use of proximity processing to manage phrase searches for INEX 2003. Some more specialized extraction methods include DATE and DATETIME extraction, INTEGER, FLOAT and DECIMAL extraction, as well as extraction methods for geographic coordinates.
3. NORMAL: The type of normalization applied to the data extracted from the text nodes. The most commonly used are STEM and NONE. STEM uses an enhanced version of the Porter stemmer, and NONE (in spite of the name) performs case-folding. Specialized normalization routines for different date, datetime and geographic coordinate formats can also be specified.

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_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.

Table 1 lists the document-level (/article) indexes created for INEX and the document elements from which the contents of those indexes were extracted. These indexes (with the addition of the topicshort index) are the same as those used last year. The major difference from last year is that the alltitles, kwd, title, topic, topicshort indexes were set up to support proximity searching.

As noted above the Cheshire system permits parts of the document subtree to be treated as separate documents with their own separate indexes. Tables 4 & 3 describe the XML components created for INEX and the component-level indexes that were created for them.

Table 4 shows the components and the path used to define them. The COMP_SECTION component consists of each identified section (<sec> ... </sec>) in all of the documents, permitting each individual section of a article to be retrieved separately. Similarly, each of the COMP_BIB, COMP_PARAS, and COMP_FIG components, respectively, treat each bibliographic reference (<bb> ... </bb>), paragraph (with all of the alternative paragraph elements shown in Table 4), and figure (<fig> ... </fig>) as individual documents that can be retrieved separately from the entire document.

Table 3 describes the XML component indexes created for the components described in Table 4. These indexes make

Name	Description	Contents
docno	Digital Object ID	//doi
pauthor	Author Names	//fm/au/snm //fm/au/fnm
title	Article Title	//fm/tig/atl
topic	Content Words	//fm/tig/atl //abs //bdy //bibl/bb/atl //app
topicshort	Content Words 2	//fm/tig/atl //abs //kwd //st
date	Date of Publication	//hdr2/yr
journal	Journal Title	//hdr1/ti
kwd	Article Keywords	//kwd
abstract	Article Abstract	//abs
author_seq	Author Seq.	//fm/au @sequence
bib_author _fnm	Bib Author Forename	//bb/au/fnm
bib_author _snm	Bib Author Surname	//bb/au/snm
fig	Figure Contents	//fig
ack	Acknowledgements	//ack
alltitles	All Title Elements	//atl, //st
affil	Author Affiliations	//fm/aff
fno	IEEE Article ID	//fno

Table 1: Cheshire Article-Level Indexes for INEX

Name	Description	Contents
COMP_SECTION	Sections	//sec
COMP_BIB	Bib Entries	//bib/bibl/bb
COMP_PARAS	Paragraphs	//ilrj|//ip1|//ip2| //ip3|//ip4|//ip5| //item-none|//p| //p1|//p2|//p3| //tmath|//tf
COMP_FIG	Figures	//fig
COMP_VITAE	Vitae	//vt

Table 2: Cheshire Components for INEX

Component or Name	Description	Contents
COMP_SECTION		
sec_title	Section Title	//sec/st
sec_words	Section Words	//sec
COMP_BIB		
bib_author	Bib. Author	//au
bib_title	Bib. Title	//atl
bib_date	Bib. Date	//pdt/yr
COMP_PARAS		
para_words	Paragraph Words	*†
COMP_FIG		
fig_caption	Figure Caption	//fgc
COMP_VITAE		
vitae_words	Words from Vitae	//vt

Table 3: Cheshire Component Indexes for INEX †Includes all subelements of paragraph elements.

individual sections (COMP_SECTION) of the INEX documents retrievable by their titles, or by any terms occurring in the section. These are also proximity indexes, so phrase searching is supported within the indexes. Bibliographic references in the articles (COMP_BIB) are made accessible by the author names, titles, and publication date of the individual bibliographic entry, with proximity searching supported for bibliography titles. Individual paragraphs (COMP_PARAS) are searchable by any of the terms in the paragraph, also with proximity searching. Individual figures (COMP_FIG) are indexed by their captions, and vitae (COMP_VITAE) are indexed by keywords within the text, with proximity support.

Almost all of these indexes and components were used during Berkeley's search evaluation runs of the 2003 INEX topics. The official submitted runs and scripts used 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. The paragraphs below briefly describe Berkeley's INEX 2003 runs.

Berkeley_CO01 was an automatic CO run using only the topic's title and keyword sections. This run uses automatic query generation with Logistic regression matching combined with boolean phrase matching and MERGE_MEAN partial result combinations. Only article level results are returned in this run (definitely a mistake).

Berkeley_CO_Okapi was an automatic CO run using only the topic's title and keyword sections. This run uses also uses the same automatic query generation script as Berkeley_CO01 but employs the new implementation of 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 was an automatic CO run. This run uses automatic query generation with both Okapi BM-25 and Logistic regression retrieval algorithms combined 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. Only Titles and keywords were used in generating the queries, which also included Boolean operations for proximity searching and "negated" terms. This run was intended to demonstrate the effectiveness of fusion of results from two different retrieval algorithms.

Berkeley_SCAS01 was an automatic SCAS run using only the title (XPath specification). This run uses automatic query generation with Logistic regression matching 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 was a SCAS automatic run using only the topic title. This run uses automatic query generation from XPATH (like run 01) but uses Okapi BM-25 ranking instead of Logistic regression and uses normalized scores in merging results from different aspects of the queries. Results from multiple components where combined using MERGE_NORM merging of results.

Berkeley_SCAS_Okapi2 was another automatic SCAS run. This run also uses automatic query generation and is very similar to Berkeley_SCAS_Okapi. Results from multiple components where also combined using MERGE_NORM merging of results. The major difference is that different indexes were used (that included more of the text – i.e. the topic index vs. the topicshort index used in Berkeley_SCAS_Okapi) for some of the queries. Many of the queries will probably have identical results to the other okapi-based run.

No VCAS runs were submitted (although the SCAS runs should probably have been submitted for both, since path interpretation in matching was not completely strict).

3.2.1 General Script structure and contents

As noted in the overview of Cheshire II features, all of the Cheshire client programs are scriptable using Tcl or Python. For the INEX test runs we created scripts in the Tcl language that, in general, implemented the following sequence of operations:

1. Read and parse topics
2. Extract search elements and generate queries
   (a) Extract query type , title (and it's XPath specification when provided), and keywords.
   (b) Duplicate British spellings in queries to include both British and U.S. spelling (e.g. "colour" becomes "colour color").
   (c) Identify quoted phrases and required (+) or unwanted (-) terms.
   (d) Based on the query type (CO or CAS):
      i. For CO-type queries, construct multiple partial searches including:
         A. Boolean search words and phrases using the topic or topicshort index for all terms from query title and keywords.
         B. Ranked search of words and phrases using the topic or topicshort index for all terms from query title and keywords.
         C. Ranked search of words and phrases using kwd index for all terms from query title.
         D. Probabilistic search of words and phrases using the alltitles index for all terms from query title.
         E. Boolean search of words and phrases using the alltitles index for all terms from query title.
         F. All of the above were combined in a single query using various merge operators.
      ii. For CAS-type queries, the XPath specification was used to choose the indexes (and components) to be searched, and the RESTRICT operators described above were used to validate proper nesting of components.
         A. For each specified "about" clause in the XPath, a merger of phase, keyword, Boolean and ranked retrieval was performed.
3. Submit queries and capture resultsets
   (a) Each query constructed in the previous steps is submitted to the system, and the resultsets with one or more matching documents are stored.
   (b) The requested document element(s) in the XPath specification are extracted from the top-ranked documents and added to the final resultset for the query.
   (c) In the Berkeley_CO_Okapi and Berkeley_CO_MergePrOk runs each of these queries (or rather variant forms of the query) were submitted separately for article-level, paragraph-level and section level results (using the appropriate components). In post-processing these separate sets were combined using an MIN_MAX normalization and re-ordering of scores, resulting list of the top-ranked 100 documents.
4. Convert resultsets to INEX result format. (E.g., extract matching element XPath's, ranks, and document file ids from top-ranked results and output the INEX XML result format for each)

```
((topic @+ Smalltalk !MERGE_NORM (topic @+
{Smalltalk Smalltalk Smalltalk Smalltalk}) )
!FUZZY_OR (topic @+ Lisp !MERGE_NORM (topic @+
{Lisp Lisp Lisp Lisp}) ) !FUZZY_OR (topic @+
Erlang !MERGE_NORM (topic @+ {Erlang Erlang Erlang
Erlang}) ) !FUZZY_OR (topic @+ Java !MERGE_NORM
(topic @+ {Java Java Java Java}) )) !RESTRICT_FROM
((sec_words @+ {"garbage collection" algorithm}
!MERGE_NORM (sec_words $garbage collection$)
!MERGE_NORM (sec_words @+ {$garbage collection$
$garbage collection$ $garbage collection$ $garbage
collection$} ) !MERGE_NORM (sec_words @+ {algorithm
algorithm algorithm algorithm}) ))
TARGETPATH = XML_ELEMENT_sec
```

Figure 1: Berkeley SCAS Query for Topic #68

Figures 1 and 2 show examples of queries generated by the scripts for SCAS and CO topics respectively. The use of "$" is to limit phrases specified in the topic statement. The duplication of terms is to enhance weighting for those terms.

4. EVALUATION

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

The "strict" quantization is intended to be similar to the relevance assessments used in other IR evaluations. (One could argue, however, that a closer approximation to most relevance judgements might be to consider any full document containing a 3 as "relevant", and possibly some of the 2's).

```
(topicshort @+ {"query tightening" "narrow the
search" "incremental query answering" query
tightening,search narrowing,incremental query
answering,knowledge base,concept similarities}
!MERGE_MEAN (topicshort {$query tightening$})
!MERGE_MEAN (topicshort {$narrow the search$})
!MERGE_MEAN (top icshort {$incremental query
answering$})) !MERGE_NORM (alltitles @+ {"query
tightening" "narrow the search" "incremental
query answering"} !MERGE_MEAN (alltitles {$query
tightening$}) !MERGE_MEAN (alltitles {$narrow the
search$}) !MERGE_MEAN (alltitles {$incremental
query answering$})) !MERGE_NORM (kwd @+ {"query
tightening" "narrow the search" "incremental query
answering"} !MERGE_MEAN (kwd {$query tightening$})
!MERGE_MEAN (kwd {$narrow the search$}) !MERGE_MEAN
(kwd {$incremental query answering$})) !MERGE_NORM
(topicshort @ {"query tightening" "narrow the
search" "incremental query answering" query
tightening,search narrowing, incremental query
answering,knowledge base,concept similarities}
!MERGE_MEAN (topicshort {$query tightening$})
!MERGE_MEAN (topicshort {$narrow the search$})
!  MERGE_MEAN (topicshort {$incremental query
answering$})) !MERGE_NORM (alltitles @ {"query
tightening" "narrow the search" "incremental
query answering"} !MERGE_MEAN (alltitles {$query
tightening$}) !MERGE_MEAN (alltitles {$narrow the
search$}) !MERGE_MEAN (alltitles {$incremental
query answering$})) !MERGE_NORM (kwd @ {"query
tightening" "narrow the search" "incremental query
answering"} !MERGE_MEAN (kwd {$query tightening$})
!MERGE_MEAN (kwd {$narrow the search$}) !MERGE_MEAN
(kwd {$incremental query answering$}))
TARGETPATH = XML_ELEMENT_article
```

Figure 2: Berkeley CO Query for Topic #92

Figures 3 and 4 show, respectively, the Recall/Precision curves for generalized quantization of each the SCAS and CO results of the officially submitted Berkeley runs. No Berkeley runs appeared in the top ten for all submitted runs. The issue that we were seeking to investigate (whether XML retrieval would benefit from data fusion methods operating across both elements and algoritms, has some cautious confirmation. The MergePrOK run which combined results for both logistic regression 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.

5. CONCLUSION

The INEX evaluation has once again proven very interesting, particularly as the first evaluation of the new fusion and resultset merging operators in the Cheshire system. As the above discussion shows, there remains considerable room for improvement of our results. We hope to present some "debugged" runs at the meeting.

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 4: Cheshire Components for INEX

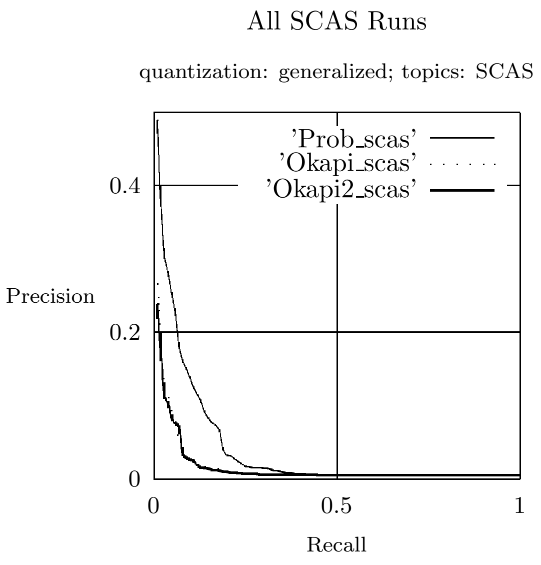

Figure 3: Berkeley SCAS Runs

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, A. Chen, and F. C. Gey. Experiments in the probabilistic retrieval of full text documents. In D. K. Harman, editor, *Overview of the Third Text Retrieval Conference (TREC-3): (NIST Special Publication 500-225)*, Gaithersburg, MD, 1994. National Institute of Standards and Technology.

[4] 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.

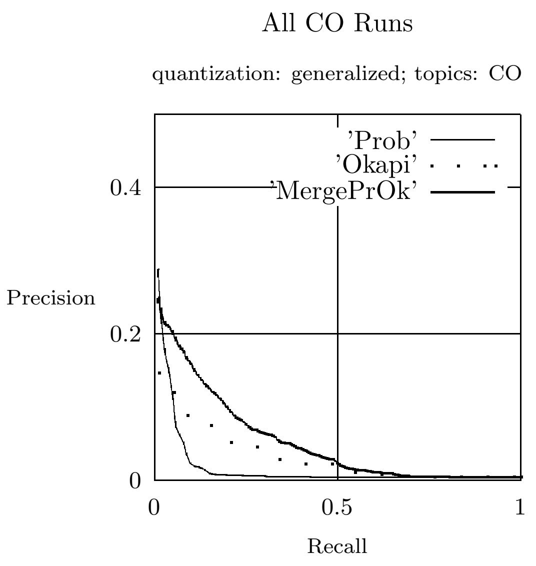

Figure 4: Berkeley CO Runs

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[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.