Cheshire II at INEX: Using A Hybrid Logistic Regression and Boolean Model 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 Berkeley used in the INEX evaluation. The primary approach is the combination of a probabilistic methods using a Logistic regression algorithm for estimation of collection relevance and element relevance, along with Boolean constraints. The paper also discusses our approach to XML component retrieval and how component and document retrieval are combined in the Cheshire II system.

Keywords

Information Retrieval, IR Evaluation, XML Retrieval

1. INTRODUCTION

The Cheshire II system originally was developed to provide a bridge from conventional online library catalogs to full-text online resources. Early research (circa 1990) with the system concentrated on the application of probabilistic ranked retrieval to short documents consisting primarily of bibliographic metadata and not the kinds of full-text document collections encountered today.

Over the past several years we have started to use the system to implement production-level services providing access to full-text SGML and XML document for a number of digital library systems in the United States and the United Kingdom, including the UC Berkeley Digital Library Initiative project sponsored by NSF, NASA and ARPA, The Archives Hub sponsored by JISC in the UK, The History Data Service of AHDS in the UK and the Resource Discovery Network in the UK. The Cheshire system is also being used to provide scalable distributed retrieval for consortia of institutions providing access to online catalogs and archival collections (the WARM system and the Distributed Archives Hub).

This paper will review the characteristics of the Cheshire II system. It will also examine the approach taken in applying this system to a collection of large XML documents as part of the Initiative for the Evaluation of XML retrieval (INEX), some observations on its performance and behavior in this area will be presented as well.

2. THE CHESHIRE II SYSTEM

When the Cheshire system was first conceived (in the late 1980's) the aim was to develop a "next-generation" online library catalog system that could provide ranked retrieval based on probabilistic IR methods, while still supporting Boolean retrieval methods expected in the online catalog systems of that era. The decision was made early on to employ SGML as the single format used in the database (with conversion utilities to generate, for example, SGML versions of MARC format records).

Since that time the system has been constantly redesigned and updated to accommodate the information retrieval needs of a much broader world. The early choice of SGML made use of X1cgptML a natural growth path, and the system remains one of the few to accomodate both XML and its more complex parent, SGML. The Cheshire II system now finds its primary usage in full text or structured metadata collections based on SGML and XML, often as the search engine behind a variety of WWW-based "search pages" or as a Z39.50 [13] server for particular applications.

The Cheshire II system includes the following features:

1. It supports SGML or XML as the primary database format of the underlying search engine, and also provides support for raw-text data or HTML linked to SGML/XML metadata records. MARC format records for traditional online catalog databases are supported using MARC to SGML conversion software developed for the project.
2. It is a client/server application where the interfaces (clients) communicate with the search engine (server) using the Z39.50 v.3 Information Retrieval Protocol. The system also provides a general Z39.50 Gateway which supports mapping of Z39.50 structured queries to local Cheshire databases and to relational databases.
3. The system include multiple clients, all of which are scriptable using either Tcl/Tk[10] or the Python language. These include a programmable graphical direct manipulation interface under X windows on Unix and Linux systems as well as a Windows implementation. There is also CGI interpreter version that combines client and server capabilities for low-overhead access to local collections. All of the interfaces permit searches of the Cheshire II search engine as well as any other z39.50 compatible search engine on the network.
4. It permits users to enter natural language queries and

these may be combined with Boolean logic for users who wish to use it.

5. It uses probabilistic ranking methods based on the Logistic Regression research carried out at Berkeley to match the user's initial query with documents and document components in the database. In some databases it can provide two-stage searching where a set of "classification clusters"[5] for the database is first retrieved in decreasing order of probable relevance to the user's search statement. The clusters can then be used to provide feedback about the primary topical areas of the query, and retrieve documents within the topical area of the selected clusters. This aids the user in subject focusing and in discriminating between variant treatments of a topic.

6. It supports distributed search across multiple collections using features of the Z39.50 protocol to harvest and create collection representatives that can then be searched probabilistically to select the collections most likely to contain relevant documents.

7. It supports relevance feedback searching where a user's selection of relevant documents is used to expand upon the initial query and automatically construct a new query derived from the contents of the selected documents.

8. It allows parts or "components" of complete SGML or XML documents (e.g., paragraphs) to be defined, indexed and retrieved as if they were individual documents, with separate indexes and ranking statistics used during retrieval.

9. It provides flexible document retrieval, including the ability to request any individual XPATH specification from any document selected during searching.

The Cheshire II search engine supports both probabilistic and Boolean searching. The 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.

The Cheshire II search engine supports both Boolean and probabilistic searching on any indexed element of the database. In probabilistic searching, a natural language query can be used to retrieve the documents that are estimated to have the highest probability of being relevant given the user's query. The search engine supports a simple form of relevance feedback, where any items found in an initial search (Boolean or probabilistic) can be selected and used as queries in a relevance feedback search.

The search engine also 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[11]) 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.

However, the primary functionality that distinguishes the Cheshire II search engine is support for probabilistic searching on any indexed element of the database. This means that a natural language query can be used to retrieve the documents or document components that have of highest probability of being relevant given the user's query. In both cluster searching and direct probabilistic searching of the database, the Cheshire II search engine supports a very simple form of *relevance feedback*, where any items found in an initial search (Boolean or probabilistic) can be selected and used as queries in a relevance feedback search.

The probabilistic retrieval algorithm used in the Cheshire II search engine is based on the *logistic regression* algorithms developed by Berkeley researchers and shown to provide excellent full-text retrieval performance in the TREC evaluation of full-text IR systems[3, 2, 1]. Formally, the 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. In the Cheshire II system $P(R \mid Q, D)$ is calculated as 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

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

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 [2] 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.

In recent work we have developed alterative forms of the ranking algorithms discussed above for application in distributed search collection selection. Because the "collection documents" used for our method of distributed search [6, 7] represent collections of documents and not individual documents, a number of differences from the usual logistic regression measures were used. In addition, analysis showed that different forms of the TREC queries (short titles only, longer queries including the concepts fields and the very long title, concepts, description and narrative) behaved quite differently in searching the distributed test collections, so three different regression equations were derived and applied automatically based on the length of the query.

Probabilistic searching, as noted above, requires only a natural language statement of the searcher's topic, and thus no formal query language or Boolean logic is needed for such searches. However, the Cheshire II search engine also supports complete Boolean operations on indexed elements in the database, and supports searches that combine probabilistic and Boolean elements. Although these are implemented within a single process, they comprise 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 the parallel search strategies merge the two result sets as a single set.

At present, 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 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.

Besides allowing users greater flexibility, the motivation for having two search methods follows from the observation that no single retrieval algorithm has been consistently proven to be better than any other algorithm for all types of searches. By combining the retrieved sets from these two search strategies, can leverage the strengths and reduce the limitations of each type of retrieval system. In general, 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[4]. 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[12]. 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. The Boolean result set is combined with the ranked result set from the probabilistic query to form a single ranked result set using evidence from both logical retrieval engines to determine a more accurate probability of relevance.

In addition we have implemented a "Fusion Search" facility in the Cheshire II system that can be used to merge the result sets from multiple searches. These will typically be from different indexes and different elements of the collection which are then 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. We have exploited this facility in our retrieval processing for INEX (as discussed below). When the same documents, or document components, have been retrieved in differing searches, their final ranking value is based on combining the weights from each of the source sets. It should be noted, however that this final ranking value is not a probability but a combination of probabilistic weights and weighted Boolean values.

Relevance feedback has been implemented quite simply in the Cheshire II system, as probabilistic retrieval based on extraction of content-bearing elements (such as titles, subject headings, etc.) from items that have been seen and selected by a user. Because the INEX runs were done as a batch process, no relevance feedback was performed for them.

The following section describes the approach taken using the Cheshire II system to construct the INEX database and conduct to searches based on the INEX structured and content queries.

3. INEX APPROACH

Our approach in the INEX evaluation was to use all of the features of the cheshire system required to support the searches produced by the participant in the evaluation. This section will describe the indexing process and the search processing along with specific comments on particular searches and the special approaches taken in some cases. In this discussion we will described some additional features of the Cheshire II system that were applied in processing the INEX queries.

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. There is also support for extraction of proximity information as well (true proximity indexes where not used for the INEX evaluation). 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 would have a different set of stopwords from document titles or personal names.

Most of the indexes used in the INEX evaluation used KEYWORD 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.

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

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

Table 1 lists the document-level (//article) indexes created for the INEX Evaluation and the document elements from which the contents of those indexes were extracted. Naturally the indexes created for the INEX collection were tailored to the needs of the retrieval task. Because it is simple to add a new index in the Cheshire system without reindexing the entire collection, indexes were added incrementally to support all of the specified content elements from the 60 INEX topics (i.e., the <ce> tags from the topic documents). Many of the indexes were document-level indexes, but, given the combination of target elements and content elements specified in some of the topics, a set of defined components and indexes to those components were created also.

<table>
<tr><th>Name</th><th>Description</th><th>Contents</th></tr>
<tr><td>COMP_SECTION</td><td>Sections</td><td>//sec</td></tr>
<tr><td>COMP_BIB</td><td>Bib Entries</td><td>//bm/bib/bibl/bb</td></tr>
<tr><td>COMP_PARAS</td><td>Paragraphs</td><td>//ilrj|//ip1|//ip2| //ip3|//ip4|//ip5| //item-none|//p|//p1| //p2|//p3|//tmath| //tf</td></tr>
<tr><td>COMP_FIG</td><td>Figures</td><td>//fig</td></tr>
</table>

Table 2: Cheshire Components for INEX

As noted above the Cheshire system permits parts of the document subtree to be treated as separated documents with their own separate indexes. Tables 2 & 3 describe the XML components created for the INEX Evaluation and the component-level indexes that were created for them.

Table 2 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 2), and figure (<fig> ... </fig>) as individual documents that can be retrieved separately from the entire document.

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

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

Table 3 describes the XML component indexes created for the components described in Table 2. These indexes make individual sections (COMP_SECTION) of the INEX documents retrievable by their titles, or by any terms occurring in the section. Bibliographic references in the articles (COMP_BIB) are made accessible by the author names, titles, and publication date of the individual bibliographic entry. Individual paragraphs (COMP_PARAS) are searchable by any of the terms in the paragraph, and individual figures (COMP_FIG) are indexed by their captions.

All of these indexes and components were used during Berkeley's search evaluation runs of the 60 INEX topics. The runs and scripts used in the INEX evaluation are described in the next section.

3.2 The INEX Search Approach

Berkeley submitted three retrieval runs for the INEX Evaluation. This section will describe the general approach taken in creating the queries submitted against the INEX database and the scripts used to do the submission. Then the differences between the three runs will be examined, including the handling of some special cases where the default query processing provided by the scripts did not appear to provide effective results.

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 topic-id, query type , title (identifying content words (<cw>), content elements (<ce>), and target elements (<te>)), description, narrative, and keywords, concatenating multi-line elements and store for each topic.
   - (b) Duplicate British spellings in queries to include both British and U.S. spelling (e.g. "colour" becomes "colour color").
   - (c) Based on the query type (CO or CAS):
     - i. For CO-type queries, construct 7 queries (run1 and run3) or 5 queries (run2) that include:
       - A. Boolean search of topic index for all terms from query title and keywords (run1 and run3).
       - B. Probabilistic search of topic index for all terms from query title and keywords (run1 and run3).
       - C. Probabilistic search of kwd index for all terms from query title and keywords (all runs).
       - D. Probabilistic search of abstract index for all terms from query title and keywords (all runs).
       - E. Probabilistic search of title index for all terms from query title and keywords (all runs).
       - F. Probabilistic search of alltitles index for all terms from query title and keywords (all runs).
       - G. Boolean search of alltitles index for all terms from query title (all runs).
     - ii. For CAS-type queries, construct all of the CO queries as in A-G above, but only for the keywords, then...
       - A. For each content element (<ce>) specified in the title of query construct both a probabilistic query and a boolean query of the index matching that content element, using the content words (<cw>) specified in the topic title for that content element.
     - iii. Construct extra or alternate queries for special cases (see below).
3. Submit queries and capture resultsets
   - (a) Each query constructed in the previous step is submitted to the system, and the resultsets with one or more matching documents are stored.
   - (b) All stored resultsets are combined using the resultset SORT/MERGE facility (discussed above), resulting in a single ranked list of the top-ranked 100 documents.

(c) The requested document elements (<te>) are extracted from the top-ranked 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)

3.2.2 “Fusion Search” and INEX Retrieval

As noted above, our INEX runs used the Cheshire “Fusion Search” facility in merging the result sets from multiple individual searches of different indexes. In the case of Berkeley's INEX runs, this 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.

The primary reason for this approach was largely 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. As described in the earlier section on Cheshire search, when the same documents, or document components, have been retrieved in differing searches, their final ranking value is based on combining the weights from each of the source resultsets. Therefore, a document that matches multiple searches will typically end up with a higher final rank than a document that matches fewer of the individual searches.

Thus, the goal in the search approach used in all of Berkeley's entries for INEX has been to try to achieve a good level of precision, without sacrificing too much recall.

3.2.3 Special Case handling

In reviewing the INEX topics, it was obvious that some of them would require special handling, because of unusual result requirements (e.g. topic #14 specifies that figures are to be retrieval along with paragraphs describing the figure). Others required special handling because of Boolean constraints on the requested results, unfortunately with inconsistent syntax for specifying those constraints (e.g. Topic #9 specifies that calendars are NOT to be retrieved by using “<cw> -calendar </cw> <ce> tig/atl </ce>” while Topic #17 uses “<cw>not(W. Bruce Croft) </cw> <ce> fm/au </ce>” for the same type of constraint.

In these situations special handling of the queries to apply the appropriate constraints was carried out by the run scripts for the Berkeley runs. The topics that were handled in this way were numbers 02, 04, 07, 09, 12, 16, 17, 20, 26, 27 and 30. All other queries were handled without special processing.

4. CONCLUSION

The results of the full INEX evaluation have not yet been revealed, but from “eyeballing” the results of the Cheshire runs for individual topics, and from the experience of evaluating the pooled results from all of the participating systems for two of the topics, it appears that the approach taken in the Berkeley runs was fairly effective. Although precision of results is usually a poor guide to overall performance, it appears that in many cases (from an admittedly biased view of the results) that the results using our “Fusion Search” approach have been quite good for many of the queries in the INEX test collection. We await the official analysis of the results to see if this belief is justified.

5. 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.

6. REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] R. R. Larson. Classification clustering, probabilistic information retrieval, and the online catalog. *Library Quarterly*, 61(2):133–173, 1991.

[6] R. R. Larson. Distributed resource discovery: Using Z39.50 to build cross-domain information servers. In *JCDL '01*, pages 52–53. ACM, 2001.

[7] R. R. Larson. A logistic regression approach to distributed ir. In *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11-15, 2002, Tampere, Finland*, pages 399–400. ACM, 2002.

[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. K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, Reading, Mass., 1994.

[11] M. Porter and V. Galpin. Relevance feedback in a public access catalogue for a research library: Muscat at the scott polar research institute. *Program*, 22:1–20, 1988.

[12] 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.

[13] A. Z39.50-1995. *Information Retrieval (Z39.50): Application Service Definition and Protocol Specification (ANSI/NISO Z39.50-1995).* NISO, Bethesda, MD, 1995.