XML Retrieval: A Survey

Sukomal Pal
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute, Kolkata
sukomal_r@isical.ac.in

June 30, 2006

1 Introduction

Rapid advances in electronic and computer technology have ushered in what we call an age of information. The exponential growth of the Internet and the Web has flooded us with huge quantities of data in different formats on a variety of subjects. To manage this colossal amount of data and extract useful information out of it, we need efficient and effective means of organizing and indexing the data. Though these data are available in different forms and the amount of available multimedia data (images, speech, audio, video etc) is rapidly increasing, textual data continues to be a fundamental and very widely prevalent form of storing information. Textual information can be broadly classified into three categories based on their structure : (i) unstructured data (ii) structured data and (iii) semi-structured data.

1.1 Unstructured data

Unstructured data means raw text, separated by punctuation and syntactic tags. Information available in the Web or the Internet which are crawled and indexed by popular search engines comprise mainly of HTML files. The tags in HTML are of only syntactic nature, they hardly indicate any semantic information about the content. Moreover, HTML does not have stringent structure; HTML allows files to contain start-tags without proper nesting and corresponding end-tags. While this feature gives a free-hand and lot of ease to the developer, we loose the structural information or meta-data from HTML. Our present-day Internet search engines which index and search mostly HTML pages perform retrieval by returning hundreds of matched documents for a keyword-based single-query given by user. While some of these documents are relevant, most of them constitute irrelevant and imprecise data. User often feels sort of getting lost in finding the right information out of piles of data.

1.2 Structured data

Structured data means data within a pre-defined stringent format, e.g. database records. Here the meta-data gives a clear idea about the content with its type, length and other attributes. The user can have specific and exact answer to his/her information need but, (s)he is constrained by the simple, fixed and parameterized mode of query. One can not extract information without an idea of the database meta-data.

1.3 Semi-structured data

Semi-structured data, as the name suggests, come in between the above two. It has a more stringent structure than unstructured data but not as rigid as structured one. From the information retrieval perspective, retrieving requested information is not only enough, but information of the right granularity with as little non-relevant content as possible is equally important to the user. Hence we need to incorporate structural

information into our search technique which is not possible with present-day search engines crawling and indexing the *surface web* that comprise mainly of unstructured data. Moreover 75% of data in web pages are derived from *deep web* which are stored in databases as structured data and hence not crawlable by search engines. We certainly need to access them. But at the same time, we cannot constraint users to have fair idea of meta-data as pre-requisite for information query. To circumvent the problem hence we need easy conversion and interoperability between these structured and unstructured data-formats. The eXtensible Markup Language or XML[1] as a semi-structured data is a right candidate to fit the bill and has emerged as a new standard for data representation and exchange on the Internet. Unlike HTML, XML can have user-defined tags which identify the meaning of the data contained between them. Each data-component is mandatorily encapsulated within a start-tag and an end-tag. Relationship between different data-elements is specified via nesting and references. XML data files can be rendered via specifications in XSL[2], the eXtensible Stylesheet Language. Guidelines for visually rendering XML data files can be specified using XSL, XSLT etc. Using XML, the issues of information content and information rendering can be separated, making it easy to provide multiple views of the same data. Laborious, error-prone, and unmaintainable "screen-scraping"[3] [Wid99] as a method for extracting useful data from HTML Web pages is greatly reduced, since XML is designed for data representation – it is simple, easily parsed, and self-describing. Readers can refer to [Wid99] for an example of HTML to XML conversion, and the W3 consortium [4] for XML representations of database tables. Moreover XML can serve as a platform for large-scale data integration among organizations based on some common syntax or DTD [Lev99]. From a user's perspective, XML enables him/her to pose more expressive and precise queries using XQuery or other XML query languages hitherto impossible with unstructured HTML pages like:

1. Find all authors with two or more SIGIR publications in the same year.
2. Which professors from IIT-Kharagpur are teaching IR and have research projects on XML ?
3. What work has been done by the Computer Science departments at IITs during 2000-2005 on semistructured data ?

and hence get more precise and less irrelevant results by exploiting the structural information represented via XML.

Since XML forms a bridge between unstructured data like HTML and structured data like database records, efforts are continuously being made in to address the issues of storing, indexing and querying XML data from both the Information Retrieval and Database communities. There have already been a couple of surveys focusing on some specific areas pertaining to XML. One of the earliest among them was by Luk et al.[LCDL00], where existing commercial systems or prototypes of XML retrieval were discussed from three different angles, namely: DB-oriented systems, IR-oriented systems and Hybrid (DB + IR) systems. However Luk et al. do not discuss retrieval models at all. Another survey [CMS01] sums up the ACM SIGIR 2000 workshop on XML and IR, examining query languages for searching XML documents as well as text annotations using XML, but does not contain any discussion on retrieval models. Luk et al.'s updated survey [LLD+02] addresses this point, and is a good starting point for studying indexing and searching techniques for XML documents. The authors provide an elaborate discussion of different indexing and searching techniques and retrieval models implemented and practised up to that point of time. However, the area has evolved dramatically and taken a more mature shape since then. INEX, a TREC-like evaluation forum dedicated to XML retrieval was started in 2002 and the growing participation at INEX indicates the importance of research on XML Retrieval. This survey will essentially discuss these recent developments. While we do provide a brief overview of XML retrieval from the Database perspective, our primary focus is on XML retrieval from the IR point of view.

The rest of this survey is organized as follows. First we outline some issues pertaining to querying data from the web applications, querying databases through an XML interface, and querying the XML data

[1] http://www.w3.org/TR/REC-xml/
[2] http://www.w3.org/TR/xsl/
[3] http://en.wikipedia.org/wiki/Screen_scraping
[4] http://www.w3.org/XML/RDB.html/

itself with various XML query languages(section 2). Section 3 first summarizes the work done to formalize evaluation metrics for XML retrieval and then analyzes various models for storing, indexing and retrieving XML data and the corresponding ranking strategies. We conclude in section 4.

2 Database Approach

The database community has attacked the problem of managing and querying the information available on the Internet mainly by applying database concepts to the web. Florescu et al., in their excellent survey [FLM98] divided the whole task into three classes: (a) modeling and querying the web, (b) information extraction and integration and (c) web site construction and restructuring.

(a) Two different models of the Web were considered. First, a graph data model views the Web as a directed graph whose nodes are web pages and edges are the hyperlinks between them. The second one was a semi-structured data model like XML. Though XML was in a very nascent stage at that time, the survey discusses different query languages for semi-structured data like Lorel[AQM+97], UnQL [PSGD96], StruQL [FFLS97], WebOQL[AM98], FLORID [HLLS97].

(b) Data integration among different web sources was done mainly with the help of wrappers. Unfortunately wrapper construction suffered a major bottleneck because of heterogeneous structure of different web sources. The authors stressed that the emergence of XML would help web-data integration by simplifying the construction of wrappers.

(c) [FLM98] discussed the issues pertaining to web site construction based on a query and creating multiple versions of a webpage tailored to different classes of applications or users. Their proposed architecture is based on a semi-structured data model for both the underlying data where query is made as well as the web site structure.

The problems discussed in [FLM98] were largely solved by Luk et al.[LCDL00]. In their survey of commercially available XML search engines, they opined that since XML can be applied to represent data in databases as well as in HTML files, it will enable transfer of data in a standard format across different systems on different platforms. Hence data integration, web-page construction and restructuring would be much simpler. Creating multiple views for different class of users or applications with the same underlying data can also be done easily by rendering XML data through XSL. Nowadays,the contents of increasing number of databases are published on the Internet in XML formats. Thus a traditional database engine can provide search facilities using SQL and present the results using dynamic webpages. This approach requires XML documents to conform to a pre-defined Document Type Definition, or DTD (meta-data that provides a legal structure to XML so that it can work as a platform-independent, interoperable language). But still DTD at its current stage suffers from poor data and referential integrity constraints. Hence while transferring data from database to XML is comparatively easy and straight-forward, putting XML documents into database is more difficult and requires more careful handling. The survey [LCDL00] also discussed the improvement of XML query languages like XML-QL[FAA+99].

Querying XML data has been one of the major thrust areas among the database community. Uniquely identifying an XML element is considered as the first step to indexing and querying the XML data. Xpath[5] is used to address parts of an XML document.[RG03] discuss some of the query languages like Quilt, XQL and XQuery[6]. Quilt is influenced by its earlier predecessors Lorel and YATL. XQuery is basically an improved and tuned version of Quilt, XQL and XML-QL.

[PHN01] and [BGK+02] explored the potential of XML as an interface between web application and back-end database. They describe ROLEX (Relational On-Line Exchange with XML), a system developed at Bell-labs for providing (i) data-integrity and consistent interoperability with relational database and (ii) the performance required by busy web or e-commerce applications. ROLEX provides better performance

[5] http://www.w3.org/TR/xpath/
[6] http://www.w3.org/TR/xquery/

than usual XML-relational middleware which generally caches application data for speed but sacrifices data-integrity and concurrency.

The system uses here main-memory DB platform(DataBlitz) to reduce latency in the interaction with DBMS. It provides web applications LIVE virtual XML views of relational data through standard APIs like parser-based SAX[7] or tree-based DOM[8]. They considered navigation by document traversal("sequential scan" and "DOM breadth-first traversal") and parameterized queries. The system keeps the statistics of navigation profile while it supports navigation of view queries through virtual DOM interface. When ROLEX query-processor executes, it uses this navigation-probability associated with the arcs of DOM-tree to optimize the view Queries [ROLEX uses modified VOLCANO-style rule-based optimizer] and takes suitable and cost-effective execution plan chosen from its 'decorrelation space' (Decorrelation Plan space corresponding to a correlated query at a particular node in the DOM-tree is a set of all possible SQL-like queries varying number of parameters to the query). They showed that a node with small navigation probability but having correlated query performs better with its deeply nested query whereas the node with high navigation probability performs better with greater decorrelation. Though the thrust was on better optimization and better performance bypassing application caching and parsing of XML data it envisages coming of DB technology to more fore-front than ever-before and plans to incorporate more complex navigation profile and multi-query optimization strategy in ROLEX and passing some of the job of XSLT [9] processor into query engine.

Apart from effectively querying XML data or exploring its application for interoperability, efforts are made in the direction of storage issue. Whether XML should be kept in a file structure or within database ? Storing XML data within traditional relational database is difficult [RG03]. Object-Relational database(ORDB) or Object-Oriented(OODB) are being considered as more viable options[cor00]. Retrieval of information from XML documents is also seen as a Object-view problem [Abi99] which can be based on ODMG model [Cat97].

3 Information Retrieval Approach

Traditional Information Retrieval techniques have successfully been applied to www as to index, store, query and retrieve mostly unstructured documents available in the Internet. As XML turns out to be close cousin of HTML, IR community relates the issues pertaining to XML retrieval alike. There has been a lot of work to explore different aspects of XML data retrieval. Here we shall highlight some of these recent advances.

3.1 INEX : Initiative for the Evaluation of XML retrieval

Growing enthusiasm centering XML has prompted IR community to take 'Initiative for the Evaluation of XML retrieval' (or INEX in short).Organized each year since 2002, INEX is a TREC[10]-like forum where participating researchers can discuss and evaluate their retrieval techniques using uniform scoring procedures over a reasonably large relevance-assessed test-collection[11]. Entire work-area of INEX is divided into several tasks/tracks:

1. Adhoc Track
   - Relevance feedback task
   - Natural query language task
2. Heterogeneous collection track
3. Interactive track

[7] http://www.saxproject.org/
[8] http://www.w3.org/DOM/
[9] http://www.w3.org/TR/xslt11/
[10] http://trec.nist.gov/
[11] http://inex.is.informatik.uni-duisburg.de/2006/

4. Document mining track
5. Multimedia track. And with newly added in INEX 2006
6. Use case studies track
7. XML Entity Ranking track.

We shall concentrate in this survey only on the Adhoc track which is the main track of INEX, specifically focusing on the Content-Only(CO) sub-task while touching some issues of Content-And-Structure(CAS) sub-task. But before going into the analysis of retrieval techniques under this track we should address two pre-requisites : a brief discussion about query languages and that about evaluation metrics used in INEX.

3.1.1 Query Languages

INEX 2002 provided two types of queries specified in XML [KLM02] : CO queries and CAS queries. Though CO query language did not need much change, CAS 2002 query language was insufficient for INEX 2003 problems. It necessitated extended version of XPath [KLM03] which also eventually proved too complex to use correctly. INEX query working group outlined a list of changes [ST03] based on which came 'Narrowed Extended XPath I' or NEXI [TS04a]. NEXI is a derivative language of XPath with added function *about*. Unlike XPath, semantics in NEXI is not defined but is to be deduced by retrieval engine - this significant feature differs from a database-oriented query language. For CO queries, NEXI does not allow any XML-element restriction, neither needs any specification of target element. But CAS-queries here may contain either explicit or implicit structural requirements.NEXI CAS queries come in two variants: Strict CAS(SCAS) queries need target element to be specified and to be met for retrieval,while for Vague CAS (VCAS) queries target path,if specified, serves as a hint and hence need not be necessarily fulfilled. NEXI has been successful in reducing error-rate from 63% (2003)to 12% (2004) for CAS queries for Adhoc track, but it is still not expressive enough for heterogeneous track and question answering [TS04b].

[ZBOW05] showed some improvement over NEXI with their GUI-based Bricks query system by reducing syntactical complexity of query formulation process,minimizing required knowledge of document structure but retaining full expressive power of NEXI.

[STW05] proposes a new query language named XXL (stands for FleXible XML Search Language) which has a core of XQuery style language but is enhanced with a similarity operator($\sim$) on element names and contents. Research is going on to formulate NEXI type query from natural language[WG05a].

3.1.2 Evaluation Metrics

The basic goal of XML-IR is like traditional IR, a ranked list of XML elements ordered by their relevance to the query. But unlike traditional IR where the whole document is returned, here only document-part(element) is returned. These document-parts or elements can be of varying granularity and may contain overlap among themselves. Hence traditional precision and recall are not enough to effectively rank XML elements. [LM05],[Lal05] state that "the general task of an IR engine has been defined in INEX as the task of returning, instead of whole documents, those components(XML elements) that are most specific and exhaustive". Hence we need to define specificity and exhaustivity in this context. *Exhaustivity* of an element is defined as the extent how much of the topic or query is covered by the element while how much of the element focuses on the topic is measured by its *specificity*(*i.e.* discusses no other irrelevant topics).

Both dimensions of a XML-element are measured in 4-point scale with degrees highly(3), fairly(2), marginally(1) and not(0) exhaustive/specific. Hence each assessed component is assigned one relevance degree combined by its exhaustivity and specificity as $(e, s) \in ES$ where

$$ES = \{(0,0), (1,1), (1,2), (1,3), (2,1), (2,2), (2,3), (3,1), (3,2), (3,3)\}$$

(10 meaningful combinations are taken out of total 16 possible cases). To apply different metrics two relevance dimensions are mapped to a single relevance scale by a quantization function $f_{quant}(e, s) : ES \rightarrow [0, 1]$. Two variants of $f(e, s)$ are used:

$$\begin{aligned} f_{strict}(e,s) &= 1 \text{ if (e,s) =(3,3)} \\ &= 0 \text{ otherwise.} \end{aligned}$$

$$\begin{aligned} f_{SOG}(e,s) &= 1 \text{ if (e,s) = (3,3)} \\ &= 0.9 \text{ if (e,s) = (2,3)} \\ &= 0.75 \text{ if (e,s)} \in \{(1,3),(3,2)\} \\ &= 0.5 \text{ if (e,s) = (2,2)} \\ &= 0.25 \text{ if (e,s)} \in \{(1,2,(3,1)\} \\ &= 0.1 \text{ if (e,s)} \in \{(2,1),(1,1)\} \\ &= 0 \text{ if (e,s) = (0,0).} \end{aligned}$$

inex-2002 :

[KL05b] nicely summarizes so-far-used metrics of INEX. The first metric inex-2002 or inex_eval [GK03] used a measure of *precall* [RJB89] to document-components and precision is interpreted as the probability $P(rel \mid retr)$ that a document viewed by a user is relevant :

$$P(rel|retr)(x) = \frac{x.n}{x.n + esl_{x.n}} = \frac{x.n}{x.n + j + \frac{s.i}{r+1}}$$

Here n is the total number of relevant document-components with regard to the user request in the collection; $x \in [0,1]$ denotes an arbitrary recall value. $esl_{x.n}$ denotes the *expected search length* [Coo68], expressed in estimated number of non-relevant elements retrieved until the recall point x is reached or $x.n$ -th relevant element is retrieved. [RJB89] considered weak ordering of XML elements where more than one element occur in a particular rank. Hence if l denotes the rank from which the $x.n$ th relevant component is drawn, j is the number of non-relevant document components within the ranks before rank l, s is the number of relevant components to be taken from rank l; r and i are the numbers of relevant and non-relevant components in rank l, respectively.

inex-2003 :

The inex_eval didnot consider overlap issue between elements and took descendant and ancestor elements independently. The inex-2003 or inex_eval_ng metric addressed the issue by incorporating component size and overlap within the definition of recall and precision. The metric is based on the notion of ideal concept space [WY95] and is formulated (for derivation refer [GKFL03]) as :

$$recall = \frac{\sum_{i=1}^{k} e(c_i).\frac{|c'_i|}{|c_i|}}{\sum_{i=1}^{N} e(c_i)}$$

$$precision = \frac{\sum_{i=1}^{k} s(c_i).|c'_i|}{\sum_{i=1}^{k} |c'_i|}$$

Components $c_1, ..., c_k$ form a ranked result list, N is the total number of components in the collection, $e(c_i)$ and $s(c_i)$ denote the quantized assessment exhaustivity and specificity value of c_i respectively defined [KLV04] as:

STRICT case :

$$\begin{aligned} e(c_i) &= 1 \text{ if e} = 3 \\ &= 0 \text{ otherwise} \end{aligned}$$

$$\begin{aligned} s(c_i) &= 1 \text{ if s} = 3 \\ &= 0 \text{ otherwise} \end{aligned}$$

GENERALIZED case :

$$e(c_i) = e/3$$

$$s(c_i) = s/3$$

where $|c_i|$ denotes the size of component c_i, $|c_i'|$ is the size of the component that has not been seen by the user previously, which, given a representation such as a set of (term,position) pairs, is calculated as :

$$|c_i'| = |c_i - \bigcup_{c \in C[1,n-1]} (c)|$$

where n is the rank position of c_i in the output list, $C[1, n-1]$ is the set of elements retrieved up to rank n.

XCG :

XCG metrics are an extension of Cumulated Gain(CG)-based metrics[JK02] in XML domain by [KLV04]. Gain vector G is formed from result-list by replacing document-id with its relevance-value (say, integer score in [0,3] where 3 is highest and 0 lowest) where G(i) gives the score of ith rank document. CG vector is defined as:

$$\begin{aligned} CG(i) &= G(i) \text{ for i} = 1 \\ &= CG(i-1) + G(i) \text{ for i} \geq 2 \end{aligned}$$

Similarly we can have an ideal gain vector I' in decreasing order of the score and hence therefrom an ideal CG' vector for each query. Normalized CG vector, nCG between actual and ideal is formed by dividing $CG$by CG' co-ordinate wise:

$nCG(V, I) =< v_1/i_1, v_2/i_2, ..., v_k/i_k >$ where CG vector $V =< v_1, v_2, ..., v_k >$ and ideal CG vector $I =< i_1, i_2, ..., i_k >$

Obviously for any rank i, $nCG(i) = 1$ means ideal performance and, in general, the area between $nCG(V, I)$ and $nCG(I, I)$ represents the quality of an IR technique.

The assessed generalized f-value is used as the relevance-value in XCG. To mitigate the problem of overlap,[KLV04] built an ideal-recall base by recursively choosing the element with highest f-value and removing the rest of elements from each path. If more than one elements have the same f-value in a path then the one with greater height is chosen. While I is derived from the ideal recall-base, G is taken from full recall-base. For any partly seen component c_i, relevance-value $rv(c_i)$ is:

$$rv(c_i) = \alpha . \frac{\sum_{j=1}^{k} rv(c_j).|c_j'|}{|c_i|} + (1 - \alpha) rv(c_i)$$

where k is the number of c_i's relevant child nodes and $0 \leq \alpha \leq 1$.

However this score calculation is kept within the constraint: for any $c_j \in S$,

$$\sum_{c \in S} rv(c) \leq rv(c_{ideal})$$

where S is the set of retrieved descendant nodes of the ideal node and c_{ideal} is the ideal node that is on the same relevant path as c_j.

Though there exists an accepted conjecture that system rank would affect significantly if the overlap is handled differently and hence overlap needs to be removed, [WG05b] experimented on the role of overlap

with XCG metric over INEX 2004 data. They showed that high scoring and high overlap systems perform well regardless of how overlap is handled.

T2I :

Since XML does not have fixed pre-defined retrieval unit, [VKL04] came up with evaluation metrics derived from user-effort-oriented view of IR. The user-effort is expressed as wasted time to inspect irrelevant elements. Basic assumption here is that a user does not bother for an accurate element to be retrieved but needs an entry-point into the document. A retrieval system is considered to produce a ranked list of entry-points. The user starts reading a retrieved article from an entry-point and continues reading until his/her *tolerance to irrelevance* (T2I) is reached when the user moves to next entry-point. T2I expressed by parameter τ_{NR} represents maximum time a user spends reading irrelevant text. [VKL04] used three T2I metrics:

1. average precision after a fixed amount of user effort in accordance with Hull's proposal [Hul93] which is given by :

$$\text{Average Precision } = \frac{\tau_{NR}}{T}. \sum_{t=1}^{T/\tau_{NR}} \text{Precision after } t.\tau_{NR} \text{ seconds wasted user effort}$$

   with suitable choice of τ_{NR} and T(some multiple of τ_{NR}).

2. precision after expected search length(ESL) which came following [Coo68]. ESL is expressed here in terms of expected search duration(ESD)[Dun97] as user-effort wasted inspecting irrelevant elements from system result list, augmented with the effort to find remaining relevant items by random search through the collection.

3. probability of relevance $P(Rel|Retr)$ for R relevant fragments computed in terms of ESL_R:

$$P_R(Rel|Retr) = \frac{R}{(R + ESL_R)}$$

   .

PRUM :

[PGD05] proposes another metric PRUM(Precision Recall with User Modeling) where they extend the idea of probabilistic precision-recall by [RBJ89] to include user's browsing behavior. Instead of simple user model where user consults retrieved elements independently, PRUM allows the user to consult ancestor, descendants and siblings of a node.Each of these context element is seen by the user stochastically, *i.e.* with some pre-assigned probability.Hence PRUM is defined as

$$PRUM(i) = P(Lur|Retr, L = l, Q = q)$$

where

$$\begin{aligned} i &= \text{recall level lying between 0 and 1} \\ Lur &= \text{event that the element leads to an unseen Relevant unit} \\ Retr &= \text{event that the element in the list is consulted} \\ l &= \text{percentage of relevant elements user wants to see} \\ \text{and } q &= \text{the query topic.} \end{aligned}$$

[Piw05] further extended PRUM with an alternative definition of precision as the ratio of minimum number of ranks that a user has to consult in the list returned by an ideal system and the list returned by the system to be evaluated.EPRUM is computed, given three sets of parameters:

i. probability that a user consults an element in the corpus

ii. probability that a user browses from a considered element to any neighboring element and

iii. probability that a user finds an ideal element.

[Piw05] presented how it was used to precisely evaluate submissions in INEX 2005 considering more realistic user navigation across elements.

Metrics at INEX 2005 :

Among the five types of metrics discussed above, three XCG-based metrics were taken as official INEX 2005 metrics used to measure retrieval effectiveness of submitted runs [KL05a] :

Normalized xCG (or nXCG): This is exactly same as nCG discussed above. Systems are compared at different cut-off values, *e.g.* nxCG[1] and nxCG[100]. We may take average of nxCG[i] scores upto a given rank as

$$MAnxCG[i] = \frac{\sum_{j=1}^{i} nxCG[j]}{i}$$

ep/gr : Standing for effort-precision/gain-recall, the metric measures the amount of relative effort as the number of visited ranks a user spends compared to the effort needed in ideal system. ep(r) is measured for cumulated gain value r as[Ref. fig. 1] :

$$ep(r) = i_{ideal}/i_{run}$$

This ep is calculated at arbitrary gain-recall points where gain-recall is defined as the cumulated gain divided by total achievable cumulated gain :

$$gr[i] = xCG[i]/xCI[n],\ n \text{ being total number of relevant elements}$$

Q & R : nxCG suffers from the setback of not averaging well across the topics, [Sak04] suggests Q and R measure :

$$\begin{aligned} Q &= \frac{1}{Num_R}.\sum_{j=1}^{i} isrel(d_j).\frac{cbg(j)}{cig(j)+j} \\ \text{and } R &= \frac{cbg(Num_R)}{cbg(Num_R)+Num_R} \end{aligned}$$

where

$$\begin{aligned} Num_R &= \text{total number of relevant elements} \\ isrel(.) &= \text{boolean function, return 1 for relevant element and 0 otherwise} \\ cbg(i) &= \text{cumulated bonus gain function} \\ &= bg(i) + cbg(i-1) \\ bg(i) &= xG(i) + 1 \text{ if } g(i) > 0 \\ &= 0,\ \text{otherwise} \\ cig(.) &= \text{cumulated bonus gain vector for ideal vector} \end{aligned}$$

[PT05] propose an alternative XML retrieval evaluation metric solely based on the highlighted text which are generally ignored during evaluation for being 'too small' elements. Their metric 'HiXEval' credits

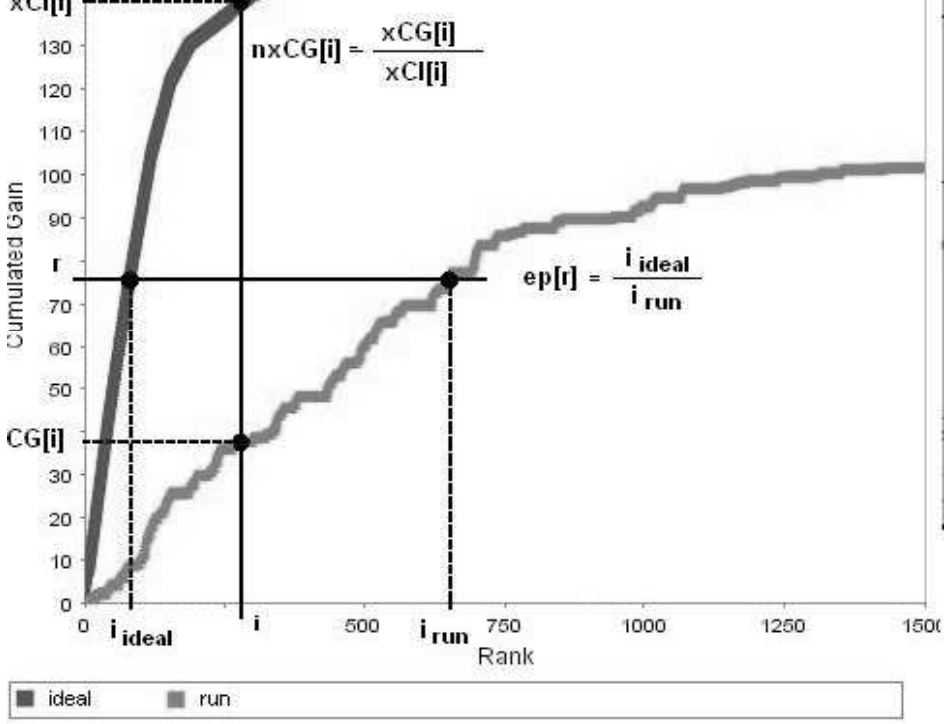

Figure 1: Cumulated Gain vs. Rank

systems that retrieve elements containing more highlighted textual information without containing non-relevant information. They define two functions for an element e at a given rank r as:

$$\begin{aligned} pre_r(e) &= \frac{rsize(e)}{size(e)} \text{ , if e is NOT-YET-SEEN} \\ &= \frac{(1-\alpha).rsize(e)}{size(e)} \text{ , if e is FULLY-SEEN} \\ &= \frac{\alpha(rsize(e) - rsize(e'))}{size(e)} + (1-\alpha).\frac{rsize(e)}{size(e)} \text{ , if e is PARTIALLY-SEEN} \\ rec_r(e) &= rsize(e) \text{ , if e is NOT-YET-SEEN} \\ &= (1-\alpha).rsize(e) \text{ , if e is FULLY-SEEN} \\ &= \alpha(rsize(e) - rsize(e')) + (1-\alpha).rsize(e) \text{ , if e is PARTIALLY-SEEN} \end{aligned}$$

where $rsize$ denote size of relevant part of element e, e' represents an already retrieved element & descendant of e but appears before r in the ranked list and α is weighting factor lying between 0 and 1.

Precision and Recall is defined in terms of above defined functions as :

$$Precision@r = \frac{\sum_{i=1}^{r} pre_i(e)}{r}$$

$$Recall@r = \frac{\sum_{i=1}^{r} rec_i(e)}{Trel}$$

where $Trel$ is the total amount of relevant information for an INEX topic.

To arrive at a single composite value they used F-measure as

$$F@r = \frac{2}{\frac{1}{Precision@r} + \frac{1}{Recall@r}}$$

[PT05] also proposed four different ways of measuring overlap to circumvent the problem associated to set-based overlap which does not differentiate among different types of overlap :

a. Overall overlap : identical to the set-based overlap

b. Ascendant overlap : measures percentage of elements that contain at least one other element in the set

c. descendants overlap : measures percentage of elements that are contained by at least one other element in the set

d. Probabilistic overlap : measures the probability that two randomly chosen elements from a set of retrieved elements overlap each other.

3.2 Retrieval Approaches

Information retrieval from XML documents belongs to any one of two broad categories : Model-Oriented Approaches and System-Oriented Approaches.

3.2.1 Model-Oriented Approaches

All the standard established IR models like e.g.Language model, Logistic Regression Model,Vector Space Model, Fusion Model, Bayesian Network Model *etc.* have been applied to XML domain. We shall look here at some of them.

Language Model: Statistical Language Model is introduced to IR by [Hie01] and then [OC03a] tried to apply it in structured document retrieval. They ranked structured documents by measuring negative of Kullback-Leibler divergence [LZ01] :

$$\begin{aligned} -KL(\theta_Q \| \theta_D) &= -\sum_w P(w|\theta_Q) log(\frac{P(w|\theta_Q)}{P(w|\theta_D)}) \\ &\propto \sum_w P(w|\theta_Q) log(P(w|\theta_D), \end{aligned}$$

or by estimating the probability of the query using the documents' language model :

$$\begin{aligned} P(Q|\theta_D) &= \prod_w P(w|\theta_D) \\ &\propto \sum_w log P(w|\theta_D) \end{aligned}$$

$$\begin{aligned} \text{where } \theta_D &= \text{ language model estimated from document} \\ \theta_Q &= \text{ language model estimated from query} \\ Q &= (q_1, q_2, ..., q_n) \text{ is the query string} \\ w &\in (q_1, q_2, ..., q_n) \end{aligned}$$

If query terms are assumed to be generated independently and query language model θ_Q used in KL-divergence is maximum-likelihood(ML) estimate, the generative model and KL-divergence produce the same rankings [OC02]. In ML-estimate, the probability is given by:

$$P(w|\theta_T) = \frac{count(w;T)}{|T|}, \text{ T being the observed text and } |T| \text{ its length.}$$

Since for a small text, count(w) may turn out to be zero, ML-estimate creates problem for document language model.Hence smoothing is applied to handle these very small probabilities [ZL01]. Linear interpolation with several language model is commonly used:

$$P(w|\theta) = \sum_{i=1}^{k} \lambda_i P(w|\theta_i)$$

where k is the total number of language models and $\lambda_i \geq 0$ is weight on the model θ_i with $\sum_{i=1}^{k} \lambda_i = 1$. In most cases, λ_i's are optimally set by Expectation Maximization(EM) algorithm iteratively.

[OC03b] applied hierarchical language model for CO queries on XML documents. Each document is assumed to have a tree-structure. For each node in the tree(*e.g. title, abstract, body, section, sub-section* etc.) a language model is considered. For leaf node,its text forms the language model whereas for intermediate nodes, language model is estimated by taking linear interpolation of language model formed from the text in the node itself and that formed from its children. The parameter for the model from a child node, λ is set as the accumulated length of the child divided by that of the node, where accumulated length of a node means number of words in the node plus accumulated length of all its children. They also considered prior probability of a node N as $P(N|Q) = P(Q|\theta_N)P(N)/P(Q)$. Storage and retrieval are done through Lemur toolkit. Lemur stores inverted indexes for document & node occurrences and document structure information. The first kind of inverted lists are indexed by term. Each such list contains document frequency of the term, a list of document entries each of which contain document id, term frequency, number of nodes the term occur in and a list of node entries where each node contains node id and term frequency within node. Document structure is stored in memory in compressed form for quick access. For each document-node, information regarding its parent node,type and length in number of words are maintained. When document structure is decompressed, its accumulated length, number of children of the particular node and a list of the node's children are calculated. Retrieval is made through constructing a query tree where leaf nodes contain query terms. Each term node reads the inverted list and create result object for XML node containing the term and calculate a term score individually. All term score at the leaf-nodes are propagated up to sum-node which merges the result lists from all term-nodes and combines the score estimates. The score adjuster node atop sum-node adjusts the combined score to get generation probabilities and applies priors. The heap-node over score adjuster maintains a list of top n ranked objects and returns a sorted result list. The exact details of the algorithm is explained in [OC03b].

[OC04] advanced their earlier work by exploring role of context with incorporation of parent's model.First smoothing is done from bottom to top of the tree and then ranking is done by passing model from the root to the tree down or what is known as shrinkage.They have shown that very small shrinkage parameter values boost precision moderately.Moreover use of extreme priors greatly improves retrieval performance.

[OC05] proposed generalized EM algorithm to estimate parameters for hierarchical language model. Instead of recursively smoothing the language models, they linearly interpolated parent's unsmoothed language model, each child's language model, document's unsmoothed language model and the collection language model. This way of learning parameters maximizing the likelihood of training examples using only relevant components, resulted in poor estimation as it also increased the likelihood of non-relevant components. Hence they introduced negative examples in the process as well by maximizing the likelihood that the language models estimated for the non-relevant components do not generate query-terms.

[SKR05] investigated different selective indexing strategies. To retrieve elements they built 4 indexes:

i. Element index with traditional overlapping elements

ii. Length based index where elements crossing a pre-set threshold length are indexed only

iii. Qrel based index where elements with certain tag-elements are indexed like *article, bdy, sec, ss1, ss2, p, ip1* and *fig* .

iv. Section index where non-overlapping passages are indexed based on structure of passage boundary. For full article retrieval also they built 2 indexes named *article index* and *Query fields* containing both content and a selection of fields respectively.

For each case, they used multinomial language model and elements were ranked by $P(e|q) \propto P(e).\prod_{i=1}^{k} P(t_i|e)$, where query q is made of terms $t_1, t_2, ..t_k$ and each of $P(t_i|e)$ is taken as linear interpolation of three language models from *element*, *document* and *collection*. Though retrieving from full hierarchy of elements outperformed that from linear segmentation, quite interestingly, its performance is comparative enough at low precision.

[KRS05] studied the length normalization issue of XML-IR based on [SSMB96] which was done for general document retrieval. [KRS05] observed that distribution of XML elements-length is more skewed towards short lengths unlike distribution of document-length for a general document collection, which is nearly normal. But like assessors of TREC collections, XML-assessors too have a strong bias towards retrieval of long elements. Moreover it is a proven fact that retrieval system performance increases if retrieval probability of elements of certain length is proportional to the relevance probability of the element. With Generative Language Modeling approach, they focus on three aspects which control the bias required towards retrieving relatively long elements, namely : smoothing parameter(λ), priors based on the element length[$p(e)$] and cut-off length of elements to be indexed(N).

Since $P(e,q) = P(e).P(q|e) = P(e).P(t_1, t_2, t_3...t_n|e)$ and $P(t_1, t_2...t_n|e)$ is estimated as a continuous product of linear combinations between maximum likelihood estimation of $P(t_i|e)$ and $P(t_i)$, smoothing parameter(λ) plays a crucial role in determining how much importance will be given to each term (t_i). Higher value of λ puts bias to long element.

Prior probability of an element e, $P(e)$ is estimated as a ratio between summation of occurrence of the terms present in the element raised to the power (β) and the same expression over all the elements. Here a parameter (β) is introduced to tune the importance of length prior. $\beta = 0$ means no length prior, $\beta = 1$ means normal length prior and $\beta = 2$ means squared length prior (prior is proportional to the square of the length of the element). Though higher λ individually favors long documents, in association with high β, it does smoothing better.Element cut-off length N forcefully prevents smaller elements from being retrieved.

The experiment is done with variation of $\lambda \in [0, 1]$, $\beta \in \{0, 1, 2, 3, 4, 5\}$ and $N \in \{0, 1, .., 60\}$ and their various combination. Length of retrieved elements increases with β up to 3 above which performance decreases at $N = 0$. Higher N improves the performance over baseline($\lambda = 0.2, \beta = 1, N = 0$) but it does not vary much for the range $N = 20$ and $N = 50$ above which performance decreases. With optimal smoothing, extreme length prior($0 < \beta < 4$) and suitable cut-off ($N \approx 40$) they got better results. They also showed that while baseline retrieves mainly para or section as most retrieved tag-type, different combination of smoothing setting, extreme prior and element cut-off retrieves mainly article or body as the most frequently retrieved tag-type.

Okapi-based Model

There have been some works with okapi model as well. [LRM05] gives a vivid description how BM25 [RZT04] and Robertson's field-weighted BM25F model for document retrieval is extendable to XML element level retrieval with BM25E. Suppose there are nE elements $e = 1, 2, ..., nE$in given collection C. *el* is element length and *avel* is average element length. Then term-weight for query term j within element e of document d in collection C is given by :

$$w_j(e, d, C) = \frac{(k_1 + 1)tf_{e,j}}{k_1((1 - b) + b\frac{el}{avel}) + tf_{e,j}} log \frac{N - df_j + 0.5}{df_j + 0.5}$$

where

$$\begin{aligned} tf_{e,j} &= \text{frequency of query-term } j \text{ in element } e \\ df_j &= \text{document-frequency of query-term } j \\ N &= \text{total no. of document} \end{aligned}$$

and element score is given by

$$W(e, q, C) = \sum_j w_j(e, d, C)q_j$$

Similarly, weighted BM25E can be given by :

$$wf_j(e,d,C) = \frac{(k_1'+1)tf'_{e,j}}{k_1'((1-b)+b\frac{el'}{avel'})+tf'_{e,j}} log \frac{N-df_j+0.5}{df_j+0.5}$$

where

$$\begin{aligned} tf'_{e,j} &= \text{ weighted term-frequency of } j\text{-th term in } e \\ &= w_j tf_{e,j} \\ el' &= w_e el \\ &= \sum_{t \in q} w_t tf'_{e,t} \\ avel' &= \frac{1}{M}\sum el', \ M \text{ being the total number of elements in } C \\ \text{and } k_1' &= k_1 \frac{avel'}{avel} \end{aligned}$$

They adjusted the weights of three elements(*article title*, *abstract* and *section title*) on INEX data and got good results with very high weight for *article title*, moderate weight for *section title* and very low weight of *abstract* for CO.Thorough and CO.FetchBrowse tasks.

[PG05] also used one of the okapi-variants as local baseline model for document element. Okapi[WR99] is adapted to compute retrieval status value(RSV) of an element X for a query q as:

$$Okapi(q,X) = \sum_{j=1}^{length(q)} w_{j,X} \frac{(k_1+1)tf_{X,j}}{k_1((1-b)+b\frac{el}{avel})+tf_{X,j}} . \frac{(k_3+1)qtf_j}{k_3+qtf_j}$$

where

$$\begin{aligned} w_{j,X} &= log\frac{M-ef_j+0.5}{ef_j+0.5}, ef_j \text{ being element-frequency of term j} \\ \text{and } length(q) &= \text{ number of terms in q} \end{aligned}$$

[PG05] used mainly term-frequency with parent-length normalization or tag-length normalization in the above formula. Since okapi score does not range in [0,1] they normalized okapi score via a logistic regression function to have a probability estimate as discussed in [Rob02].

Logistic Regression Model :

LR model of probabilistic IR estimates the probability of relevance for each document or document component based on a set of statistics about a document collection on a set of queries along with a set of weights attached to the statistics. The statistics and their weights are obtained from the regression analysis of a relevance-judged sample document collection for some set of queries. Probability $P(R|Q,C)$ is calculated with the help of "log-odds" of relevance $logO(R|Q,C)$ where for any two events A and B odds $O(A|B)$ is a simple transformation of the probabilities $P(A|B)/P(A'|B)$ in the following way :

$$logO(R|Q,C) = b_0 + \sum_{i=1}^{S} b_i s_i$$

$$P(R|Q,C) = \frac{e^{logO(R|Q,C)}}{1+e^{logO(R|Q,C)}}$$

b_0 being intercept term, b_i the coefficient for statistic s_i from the set S. [R03] used this algorithm for their Cheshire II system with:

$$s_1 = \frac{1}{|Qc|}\sum_{j=1}^{|Qc|} log(qtf_j)$$

$$
\begin{aligned}
s_2 &= \sqrt{|Q|} \\
s_3 &= \frac{1}{|Qc|}\sum_{j=1}^{|Qc|} log(tf_j) \\
s_4 &= \sqrt{\text{component length in bytes}} \\
s_5 &= \frac{1}{|Qc|}\sum_{j=1}^{|Qc|} log\frac{N-n_j}{n_j} \\
s_6 &= log(|Qd|)
\end{aligned}
$$

where Q is a query with $|Q|$ terms,

$$
\begin{aligned}
|Qc| &= \text{number of query terms in } Q \text{ that also occurs in component } C \\
tf_j &= \text{frequency of } j\text{-th term in a document component} \\
qtf_j &= \text{frequency of } j \text{ -th term in query } Q \\
n_j &= \text{number of component containing } j\text{-th term} \\
N &= \text{number of components in the collection} \\
Q_d &= \text{number of terms in both query and component}
\end{aligned}
$$

Machine Learning techniques :

Several machine learning techniques are contemplated to be used for XML retrieval. [PG05] propose a model based on Bayesian Networks. To reduce the complexity from 10 (Exhaustivity,Specificity) values, they consider only three states a particular document node X may have : Irrelevant, Big or Exact. Each document is considered as a bayesian network. Retrieval process starts from the root node of document and propagates downwards. Probability that a node X has a particular state given a query q,$P(X = v_x|q)$ is calculated as a linear combination of conditional probabilities of its parent nodes and various local baseline models. Conditional probabilities are learnt by cross-entropy training criterion where target distribution is known for relevance-judged element's (E,S) values and mapping them to three states:I,B,E and minimizing entropy function via gradient descent method.

[VG05] presented a Machine Learning based ranking model which automatically learns its parameters from a training set of annotated document elements for a set of queries. For a given set of elements X, they define a partial order among all the elements for a particular query. They associate a real function$f$ to each element xs.t. $x_i < x_j$ if and only if $f(x_i) < f(x_j)$ for any two elements x_i, x_j in X and a query q. The function f is defined as :

$$f(x) = 1 + w_1 * okapi(x) + w_2 * \text{ okapi } (parent(x)) + w_3 * \text{ okapi } (document(x)).$$

Learning function f or learning the weights w_i gives the total ranking. This is achieved by defining a ranking loss function and minimizing the loss function by gradient descent method.

Vector Space Model :

Though VSM is the most popular and effective IR model for unstructured documents, few attempts have been made to use VSM in the field of XML-IR. One of the earliest work was by [SM02] where they applied VSM and tree matching technique. Document collection is seen as a single tree where documents are sub-trees. Query is also a tree and every logical document rooted at a node with the same label as the query root is a potential candidate to be returned as result. But instead of returning a ranked list of document elements, a ranked list of whole documents was returned.

[MM03],[MM05a] achieved remarkable success in applying VSM to XML data. Instead of keeping tf-idf statistics at the document level they stored them at XML component level. To get rid of overlapping problem, they made separate indexes for full articles, sections, paragraphs etc. Each component was thus separately ranked by the similarity measure with its own kind of tf-idf. Weights for both query Q and component C are defined as the ratio of $log(tf)/log(average\_tf)$ and traditional definition of idf. But this separate indexing

crippled from the problem of missing data *i.e.* the fine-grained indices lacked data that is outside their scope but which is not indexed. Hence [MM05a] incorporated document pivot factor(DocPivot)[[SKR03],[SBM96]] to scale final element score by its containing article score.Final score of component C with original score S_c and its article score S_a is then $DocPivot * S_a + (1 - DocPivot) * S_c$.

[HB05] elegantly incorporate vector space model in their user-driven XML retrieval system. From the query they segregate structural constraints and content part. Then they measure similarity separately for meta-data and content considering different term-space depending on structure/level. Final RSV is calculated as a linear combination of these 2 similarity-values and this RSV is used for ranking.

[Hub05] comes up with an adapted VSM. Here unlike similarity of vector space model, score is determined by the combination of three functions: $f(t, E)$ measuring contribution of term t in element E; $g(t, T)$ measuring contribution of t in the topic T; function $p(T, E)$ measuring global presence of topic T in element E. $f(t, E)$ is taken as the ratio of occurrence of term t in E to the occurrence of all query-terms in E. Factor $g(t, T)$ is a function of occurrence of term t in T, total no. of occurrences of all the terms in T, number of elements containing term t and rank of the term according to the number of elements where the term t is present. Factor $p(T, E)$ is an exponential function of number of terms in query topic T and number of query-terms appearing in E as well. Though this score is calculated at each element, the score is propagated upwards in the hierarchical structure of XML tree. The score propagated is gradually decreased by a tunable reducing factor. Structural constraints are incorporated in $f(t, E)$ and in final score function through intervening coefficients. Their approach shows better performance in CO.Focused than CO.Thorough task and considerably fair results for CAS tasks.

[Leh05] make a naive effort to apply VSM in their EXTIRP system. Here information coded in tags is outrightly ignored while considering CO queries. Two indices are built, one for words and the other for phrases. The entire document collection is divided into disjoint fragments which are independently treated as full-documents under traditional tf-idf based VSM. The scheme suffers from the problem of fragment combination as indexed fragments are static.

Fusion Model:

There have been some efforts to fuse results of different models to produce the final ranking of XML components as no single retrieval algorithm could be shown to be consistently better than any other algorithm for all types of searches [C00,SF94,BKFS95]. [Lar05] studies different fusion approaches. In Cheshire II system[Lar03] within a single process, both strict Boolean operators and probabilistic searches are implemented as two parallel logical search engines. Each one individually produces a set of retrieved documents. When both are used in a single query final ranking is done by combining their result as $P(R|Q, C) = P(R|Q_{boolean}, C) * P(R|Q_{prob}, C)$ where the components retrieved by boolean operator as relevant are ranked. They also include new result-fusion operators like FUZZY, RESTRICT and MERGE operators. In [Lar05] they report the result with MERGE_MEAN(arithmetic mean of two result lists is given in the final rank),MERGE_NORM(combines two results with min-max normalization of the weights as suggested by [Lee97]), MERGE_NSUM (performs min-max normalization and then sums the normalized scores) and MERGE_CMBZ operator(based on COMBMNZ by [SF94], doubling scores of components common to both the lists, unmodifying high-ranking components in single list but penalizing low-ranked components in single list). Their experiments with LR and Okapi BM25 methods over INEX data strengthen [Lee97]'s observation that most effective fusion algorithms are those that combine base algorithms that retrieve similar set of relevant components but different sets of non-relevant components. They also infer that while okapi is better at identifying wide range of degrees of perceived relevance, LR is better at identifying highly relevant items. Though they could not strongly prove that fusion of multiple algorithms is guaranteed to show better retrieval performance, their results show fusion as highly effective in INEX evaluation. Other than fusion of different algorithms or models, fusion of the results of different components of a XML document is also tried. [Lar05] also shows that fusion of results from indexes for topic, title, section words, section-title and paragraph words improve retrieval performance but also cautions that arbitrary combination may turn the table down.

[MM05b] also show fusion of different indexes with highly promising results. They used 6 indices for 6 different levels :*article*, *bdy*, *abs*, *sec*, *ss1*, *ss2* & *p*. For each index they ran the search and made a ranked

list. To obviate redundancy then a merged result list is made based on normalization and they got good results for all 10 types of tasks(3 CO,3 CO+S and 4 CAS). Their *CO.Thorough* task was with base AQR algorithm using Lexical affinity Refinement algorithm [MM05a]. But for *CO.Focussed*, they first applied *CO.Thorough* and then filter out the overlaps. This filtering out is done in 2 steps. First they identify highly ranked clusters and pick the most relevant one node and deleting the other nodes from the clusters. Here they use some novel technique. They build result tree from the merged ranked list and then pruned the tree by eliminating redundant components (parent or child or sibling) depending on some set thresholds. In the second step they take the pruned tree and compared score of different nodes and eliminated nodes unless we get one node from each tree.For $CO + S$ runs, the structural run improve *Thorough* result but not Focused run. Their different CAS showed that using '+' tag for mandatory inclusion gives good result. Their consecutive top performances at INEX prove the effectiveness of fusion approach.

3.2.2 System-Oriented Approaches

These are approaches that focus more on system aspects like *e.g.* using relational database for storing and query processing, applying XML-specific post-processing to traditional text retrieval engine or performing retrieval in a distributed environment. These approaches can be loosely classified into two broad groups: Database-Oriented system approach and IR-system approach. We shall look at each one of these two. Though we must mention that the classification is not mutually exclusive and the boundary is often blurred with overlap and there are approaches that can be termed as hybrid of the two.

Database Systems :

[LMR$^+$05], in their TIJAH XML-IR system design a standard layered database architecture separating conceptual, logical and physical levels. At Conceptual level they classify XPath-based NEXI queries into three different query patterns. For each pattern they map distinct query execution strategy. Here they also apply language model for the special *about* function of NEXI query to rank XML components. Next at the logical level they use score region algebra(SRA) as the basis of query processing. Here the whole XML document is considered as a linearized string and each component as a text region or a contiguous subset of the linearized string.Each region is considered as the retrieval unit and identified by a 5-tuple with *startpoint*, *endpoint*, *name*, *type* and *score*. They defined different region algebra operators to represent different query patterns and manipulated their scores. At the physical level, they implemented physical region operators and execution strategy for each pattern. Here they physically computed length-prior and count term-frequency, document frequency, collection frequency of different term-region to estimate different probabilities. These three level of abstraction not only ensure *data independence*, but also separated structural query processing from underlying probabilistic model used for ranked retrieval, what is known as *content-independence*.

[STW05] present XXL(FleXible XML search Language) search engine based on Oracle 9i database with a powerful XML query language. It supports SQL-like schematic style of logical search coupled with pattern matching and IR-style relevance ranking. Their XML data model represented a collection of XML documents by a directed graph with two types of nodes : *n-node*(containing element or attribute) *c-node* (element content or attribute value) and weighted edges denoting ontological similarity value between connecting nodes. The search engine, based on a client-server architecture, took the XXL query through a GUI-based applet and passed to a query processor which evaluates the query and transfers it to three different indexes: *Element Path Index* , *Element Content Index* and *Ontology Index* . The indexes are built from XML files via a stand-alone crawler and physically stored in relational tables. During query processing, a XXL query which looks like a SQL, is decomposed into subqueries. A global evaluation order is chosen to evaluate the subqueries which are evaluated in top-down fashion. Each subquery is associated with a regular expression and local score based on semantic similarity of the ontology-based model is assigned to the query as it undergoes state-transition along its Non-deterministic Finite Automata(NFA). Final score is obtained by combining local scores of subqueries from $n-nodes$ and $c-nodes$. The components thus obtained are listed to the user in decreasing order of relevance score.

[YHSS05] combine usual strategy of considering content and structure of XML data with database-scoring. The idea is implemented in STRUX system which is used as a platform for experimenting with XML scoring on top of GALATEX, an open-source implementation of XQuery Full-Text [YBS04], which

is an upcoming standard for extending XPath/XQuery with full-text search primitives. Here score of an element s is

$$score(s) = \sum_{t \in T} \frac{tf_d(s,t) * ipf_d(s,t) * tf_q(t) * ipf_q(t)}{length(s)}$$

where $tf_d(s,t)$ and $tf_q(t)$ imply frequency of term t in s and query q respectively, T the set of key-word terms in q, $length(s)$ total number of words in s and ipf-scores are inspired from *path-scoring* method [YKA$^+$05] as

$$ipf_d(s,t) = 1 + log_2 \left(\frac{\text{number of answers meeting datapath}}{\text{number of such answers containing t}} \right)$$

$$ipf_q(t) = 1 + log_2 \left(\frac{\text{number of answers meeting query path}}{\text{number of such answers containing t}} \right)$$

The scheme is a basis for considering query relaxation on structure of XML and allows near matches not-exact according to query structure.

[Gev05] describes their retrieval system of XML file inversion and GPX database structure which is based on building of a collection sub-tree consisting of all-nodes that contain one or more search-terms. Each node there is assigned a score using tf-idf variant and then results are ranked.

Here each term posting in the collection consists of three path elements: *file name*, *absolute XPath context* and the *ordinal position within XPath context*. Entire collection is inverted and stored in MS Access database. Queries in NEXI formats are first parsed and then incrementally result-tree is built so that result-tree contains all the elements which contain at least one query term. Once the tree is constructed, it is traversed from leaves to root and scores are calculated. A leaf's score L is calculated as $L = K^{n-1} \sum_{i=1}^{n} \frac{t_i}{f_i}$ where n = number of unique terms contained within leaf, K integral constant, t_i & f_i are frequency of i-th query term within leaf node and in the collection respectively. Score of a branch node $R = D(n) * \sum_{i=1}^{n} L_i$, where $D(n)$ is a decay-factor, function of number of its children and L_i score of its i-th child. When score of all nodes are calculated, the score of support-filter(score of structures meeting queries) are added to the nodes so that overall score of the nodes meeting both content and structures increases.

Another database approach is depicted in [HB05] where the system is well-divided in three modular sub-systems: Indexing, Retrieval and RMI(Remote Method Invocation). During Indexing documents are mapped to XML schema(DataMapper),then stored in DB(DataStorer) and then indexed for retrieval(DataIndexer). Here a document is taken to consist of three main elements DOC(full document), SEC(section) FRA(fragment, generally paragraph). DOC is root, SEC is sometimes recursively defined in the path and FRA is smallest retrievable unit(leaf node). Any node is viewed as a tuple <*metadata, content*>. Metadata viz. structural info like docID, DOC/SEC/FRA, path, parentID, pre-node, post-node, tagID is stored in some DB tables and used for fast retrieval while content viz. mainly text, tables, figures, formulas, images, sound, video with external link is stored in separate tables and used for partial matching with VSM. Indexing is done mainly on content involving NLP tasks like tokenization, tagging, term extraction, stemming, filtering and term-frequency calculation. Term-frequency(tf) is stored in a VSM table and idf is stored in two tables for two levels (localIDF and CombinedIDF). Weight-vector is calculated dynamically with the help of three tables (VSM, localIDF and CombinedIDF). Query interface RMI accepts two types of query: Keyword-based query and NLP-based free-text query and then transforms it into NEXI with both metadata-query [looks like: meta(path,condition)] and content-type [looks like: about (path,terms)]. When searched element is found and satisfies the specified path then the particular element is returned, otherwise its parent element is returned. Ranking is done using Vector Space Model and then final filtering is done based on the requirement like thorough retrieval (overlapping allowed) or focused retrieval (no overlap).

Information Retrieval Systems

Some of the system approaches can be categorized as Information Retrieval systems. TopX search engine system by [TSW05] is certainly one of the top runners. Here query processor decomposes a query into *content conditions* each of which is a tag-term pair. For each such tag-term pair, an index list sorted in descending order of scores is maintained and stored in relational table. For content-only queries sequential

search is performed on tag-term pairs, but for structural query condition, a judicious mix with more expensive random look-up is made. The system returns top-K (K is user specified) elements and they are selected dynamically based on global score of the elements. All intermediate potential data-items are collected in main-memory. Here each data-item d is associated with three fields: $E(d)$ which is the set of evaluated index lists where d is seen; *worstscore(d)* which stores summed-up score of d from seen list; *bestscore(d)* which stores *worstscore(d)* added with maximum possible score of d predicted from unvisited index lists. Two priority queues of pointers are also maintained in memory : one, Top-k queue-based on *worstscore*, the other, candidate queue, based on *bestscore*. Both the queues contain two disjoint subsets of items currently in the cache and are continuously updated. If an item's *bestscore* falls below current *min-k* threshold (min-k is the score of the last item of current top-k list) the item is dropped from candidate queue as well as from cache. The final top-k result comes when *min-k* threshold is at least as high as the highest *bestscore*(d) among remaining candidates. Here content score is measured according to Okapi BM25 model and structural score is added as a constant score mass c for each structural match. For CAS, overall score of a sub-tree is sum of content score and structure score whereas for CO, the score is only the sum of all scores over all terms. For text predicates (“”, +, -) separate scoring mechanism is there. Negation is not strictly followed as there is a chance of losing a highly relevant element which somehow contain a few negated terms.But for each negated term, if an element does not contain it, constant static score like structure is added. So is done for mandatory terms(+). But phrases(“”) are treated as binary filters. Though their performance for CO task is average, the system perform well with CO+S and SSCAS tasks.

[AKJ05] discuss their TRIX (Tampere Information Retrieval and Indexing of XML) approach for indexing, weighting and re-weighting with *CO.Focussed*, *CO.FetchBrowse* and *CO.Thorough* and CAS tasks. TRIX is based on structural indexing of XML tree where root has index <1>, its children have <1,1>, <1,2>, <1,3> and node <1,1> has children like <1,1,1>, <1,1,2> and so on. For each node and each keyword, weight is calculated in the okapi-like style based on frequency of the term, total number of content-elements, number of descendant elements of the corresponding node. The relevance of a node to a query is determined by the average of the weights of the node for all the keywords in the query. After this basic weighting scheme a node is re-weighted based on different CONTEXTUALIZATION: root,parent, tower (the element along with all its ancestors) and root+tower. In root contextualization, final weight of a node is calculated as average of its own weight and 1.5 times weight of the root. Parent contextualization gives final weight of a node as the average of the weight of the node itself and that of its parent. In tower, final weight of a node is the average of all weights upto root from the node. Similarly, for Root+Tower final weight is the average of all weights upto root from the node plus the weight of root (i.e. tower contextualization + 2 times the weight of root). For CO runs they use Root+Tower contextualization or Root contextualization. Results show that Root contextualization is not better than Root+Tower, except for top categorization. For CAS they consider SSCAS, SVCAS, VSCAS, VVCAS queries.

[Dop05] describes two novel post-processing enhancement approach to improve the baseline results obtained from Lucene IR engine with application of *element relationship graph(ERG)* and *context patterns* . During query processing they discarded ‘-’ sign and use of ‘and’ & ‘or’. They take part in only vague CAS tasks and hence for CAS queries they consider only the last element(i.e. *atl* only) of the NEXI query-path (//article//fm//atl) and not the preceding structural constraints(//article//fm). They index all the elements of a document, even inline elements containing only a few words(*e.g.* bold or italics text) with all the words they used ‘length-based score correction’ such that each element's score is appropriately multiplied by a suitable factor according to its length(very short text may be *title*, so not an element to be retrieved. Again very long element is not desired as an XML result-element).

After this baseline approach, in ERG semantic relations between tag-names are explored by placing them in a semi-hierarchical graph. Tag-names occurring as leaves of similar kind are grouped in a category and each category is assigned a coherence value in [0,1] to give a measure of similarity of their name (*e.g.* first level of heading is given 0.2, second-level 0.5 etc). As the category for last tag-name is used to search in and if a match is found there its score is taken as is, otherwise the layer above in the ERG is taken and searched reducing the score by multiplying it with the corresponding coherence. CONTEXT patterns is a schema-independent method of enhancement of baseline search. Here for each non-leaf node its score(Y)

is plotted against its position and length(X). Within the whole length of the non-leaf node occur the leaf-nodes.Depending on the context patterns of the leaf nodes within this non-leaf node, we recognize certain structure (*e.g.* very short-lengthed leaf-node with very high score may be title of a section...hence reduce its score while increase its parent's score since its parent is potentially more relevant than the title node itself) and modify the score of corresponding nodes based on some fuzzy-logic-driven decisions. They experiment with different combination of ERG, length-based score correction and context pattern and obtain good result for length-based and context-pattern with generalized quantization.

[PMM05] report a lightweight indexing and search-engine SIRIUS based on approximate matching scheme of structural and textual content. Instead of DOM matching they split document object model in a set of paths which are indexed using optimized data structures and searched based on Levenshtein editing distance [Lev66], [WF74] and information fusion heuristics. Here every node n is assumed to inherit a path $p(n)$ that links n to the root of the tree. Hence $p(n)$ is represented as: $p(n) =< n_0, A(n_0) >< n_1, A(n_1) > ... < n_n, A(n_n) >$ where $A(n_i)$ is XML context in which n_i is occurring. Inverted index list contains two kinds of entries: for each textual sub-element or string s of a tree-node they attach

(i) a link to the URI of the original document

(ii) an index specifying the location of the sub-element within the document

(iii) a link towards its XML context $p(n)$

and for each node n of the DOM tree, they attach

(a) link to the URI of the document

(b) a link to its XML context $p(n)$.

A query is taken as a path goal p^R, with some conditions/constraints to be met on attributes or attribute values, and is matched with a tree T^D. The similarity is measured based on modified Levenshtein edit-distance scheme which measures minimal sequence of elementary transformation like substitution, deletion and insertion needed to get from p_i^D(a path cut from T^D) to p^R. Path-similarity function is so formulated that it gives 1 for a perfect match whereas it tends to zero as the number of mismatching nodes and attribute condition increases. Complex requests are built using low-level request p^R and merging operators (boolean or specialized operators like *seq*, *in*, *same*, +, *same+*, *in+* etc). They translate INEX CAS queries to SIRIUS query language and perform retrieval based on them with average and good quality results for top 25 answers.

Hybrid Systems

If both the database systems approach and information retrieval systems approach can be present why not mixing them up and try to get best of the two ? [PTV05] is certainly good work exploring the hybrid system approach. They first investigate a full-text IR approach using Zettair system with pivoted cosine document length normalization scheme [SBM96] tuning slope parameter with a value of 0.55. They also index XML documents using eXist, a native XML database in parallel. The eXist query is run through a topic translation module which translates INEX CO topic into both AND and OR queries. The resulting eXist answer list comprises elements of AND resulting answer list followed by that of OR in increasing document id order. This list doesnot support any ranking of the answer elements. Hence in the hybrid approach first Zettair is run for each INEX topic which extracts 1500 highly-ranked full-articles. For the same topic, eXist is run on these 1500 articles with both AND OR query. As a result two answer lists (one for each of AND and OR) of non-ranked elements are obtained. To merge small elements they define Coherent Retrieval Element(or CRE) which is a collection of at least two matching elements, two other CREs or a combination of a matching element and a CRE. These CREs are ranked according to 3 criteria:

(i) number of times a CRE appears in each matching element's absolute path in answer list - more matches(M) or less matches(m)

(ii) length of the CRE's absolute path from root- long path(P) or short path(p)

(iii) the ordering of XPath sequence in CRE's absolute path - near to Beginning(B) or near to end(E).

They obtain best results by using MpE or pME combination. In MpE, first CREs are sorted in descending order of number of times of elements. If two CREs have same number of matching elements then the one with short length will be ranked higher.If two CREs have same number of matching elements and similar path length then the one with nearer to End will be ranked higher. Their hybrid-CRE approach perform best much ahead of any individual approach with Mean Average Precision. Their hybrid approach appears to give best results with all kinds of queries e.g. *broad* or *narrow* as well.

Relevance Feedback Sub-Track at INEX :

As adding feedback is a proven method of enhancing retrieval results for traditional IR, the approach is being applied in XML domain as well[Cro05]. [SW05] propose two independent concepts of feedback in XML IR. One re-ranks the result of a keyword-based search engine incorporating structural features derived from results with known relevance using feedback approach by [Roc71]. The other one, extending feedback approach of [RJ76], expands a key-word query into a CAS query specifying new constraints on structure alongwith standard content-based query expansion.

[SW06] details the first approach where three classes of features are used as candidates for query expansion from an element of known relevance:

(i) C features (all terms from the content of the element together with their score)

(ii) P features (features derived from the path of the element)

(iii) D features (tag-term pairs within element's document)

For each of the feature classes, weights of each feature is re-calculated according to Rocchio[Roc71] with suitable choice of positive(β) and negative(γ) parameter for relevant and non-relevant elements respectively. Among the many possible features, the one with highest absolute weights are selected and number of features are heuristically chosen. If there are many features with same weights then mutual information of the feature's score distribution among the elements with known relevance and the relevance distribution decides as a tie-breaker. If mutual information is zero then least document frequency of the feature act as tie-breaker. Expanded query is evaluated by parts, the added part is evaluated separately and combined for each element afterwards. For each element, additional score is computed for each feature class as a separate vector space where each dimension corresponds to a particular feature. The score of the element for a class is computed as cosine of the new vector with old feature vector in the baseline run and normalized to [-1,1]. The overall score of an element is sum of its baseline run and additional score for different feature classes. Their result using both MAP and precision@10 show retrieval improves with introduction of C, C+D and C+D+P features gradually.

The second feedback approach [SW05] produces CAS queries by expansion using four candidate classes:

(i) C candidates(all terms of the element alongwith score)

(ii) D candidates(all tag-term pairs of the descendants of the element)

(iii) A candidates(all tag-term pairs of the ascendants of the element)

(iv) AD (all tag-term pairs of descendants of ancestors of the element).

To assign weight to feature they used Robert-Sparck-Jones weight. The candidates are ordered based on the Robertson Selection Values [a modification of RSJ weight of a candidate] except the elements with known relevance value. Top-b (b is a configuration parameter of the system) candidates from the ordered list are chosen to generate expanded CAS query. Each of the expansion is weighted according to RSJ value adjusted with a factor depending on class of the candidate. C and D class are adjusted with high weight as they would help find more relevant results.

Evaluation of feedback run is done differently compared to the baseline run as the feedback run has information about relevant elements. Hence in 2004 for FB run all known elements alongwith their descendants were removed. But in 2005, six different collections were made:

(i) resColl-result(only the elements for which FB is given are removed)

(ii) resColl-desc(elements for which FB is given are removed alongwith their descendants)

(iii) resColl-anc(respective elements alongwith ancestors are removed)

(iv) resColl-doc(the whole documents with respective elements are removed)

(v) freezeTop (all elements with known relevance are removed) and

(vi) plain or baseline run

For all types of document-collection they got substantial improvements over baseline using feedback and best results when all A+C+D+AD class features are included.

[MM05c] also tried adding feedback on their XML component ranking algorithm [MM05a]. After having top N results from the merged list of component ranking scheme, they ran Relevance Feedback algorithm for each index (article,section, paragraphs *etc*) with the parameters R, NR and Res_i where R is the set of relevant elements,NR the set of non-relevant elements out of N chosen elements and Res_i is the result set for index i. Thus query Q is refined to query Q' and the result set Res'_i is computed for index i. Res'_i is normalized and then scaled each score by its containing article score from Res'_0.Ultimately all Res'_i are merged to a single result set Res' composed of all components sorted by their score. They tried two RF algorithms: Rocchio [Roc71] and Lexical Affinity and reported the result obtained from Rocchio run with $\alpha = 1$ and different values of β and γ and got best results with $\beta = 0.8$.

NLP Sub-Track at INEX :

Prevalent XML-IR systems expect user to express their requirement in the form of structured queries like XPath or NEXI.But even the experts in the domain of XML feel difficulty in correctly using the complex queries, let alone then inexperienced casual users. This serves as a strong motivation for formulating a methodology to automatically translate user's content and structural need expressed in natural language to one of the usual structured query.

[WG05] propose a system NLPX which does exactly the same by accepting Natural Language Queries (NLQ) and then translating them into NEXI queries. Here the system uses topics' Description rather than Title part. First they tag words in NLQ as special connotation or their part of speech into 3 major divisions: Structures(e.g.*section*, *abstract*), Boundaries(e.g.*contains*, *about*) and Instructions(e.g. $find$ or *retrieve*) and the remaining words are tagged by their part of speech by Brill Tagger. In the second step this tagged queries are matched with query templates and then finally outputted in NEXI format based on request's structural and content attributes. Support requests are inserted by comparing the structural attributes of return and support requests and by finding their longest shared descendant. For 2005 they add some special connotations as NEGATORS (like 'except' or element user 'doesnot want' categorically - represented by '-'), STRENGTHENERS (like 'major' or 'particularly' within NLQ and represented as '+'), REVERSE BOUNDARIES (here NLQ is in passive voice and boundary occurs after the content like 'talked about' or 'described'), SELF-REFERENCE TOPICS(like 'that topic') and some stopwords. They also introduce SHALLOW PARSING as an intermediate step between lexical tagging and template matching to divide sentences into atomic segments (or chunks) alongwith definition of some additional templates. In most of the cases(CO, +S, both focused and thorough) their score are much better than baseline NEXI; for some CAS queries their nXCG and MAP scores are almost similar to baseline NEXI, if not better.

[Tan05] is another approach of NLQ2NEXI system. On the NLQ they first perform part-of-speech(POS) tagging with TreeTagger[Sch94] and then assign a POS-dependent semantic representation. Context-free syntactic rules describing the most current grammatical constructions in queries and questions are defined to give a final logical representation. Semantic representation is then reduced with the help of above-defined rules to recognize some typical constructors of a query and to distinguish between semantic elements mapping to structure and to the content. A treatment of relations existing between different elements are explored and finally NEXI query is formed from them. They take special care of noun phrases with two types:

1. simple noun phrases with rule $NP \rightarrow (ADJ|NOUN) + NOUN$, *e.g.* "graph theory" or "semantic networks"

2. complex noun phrase with rule $NP \rightarrow NP(PREPNP)+$, *e.g.* "annotation in image retrieval"

However their system suffers from usual lexical ambiguity (synonymy, hyponymy, homographs), syntactic ambiguity and corpus-dependent dissimilarity. Moreover these NLQ2NEXI systems also face a general problem of evaluation as they produce only the inputs Hence evaluation is done indirectly through the use of a common search engine. Their system top at INEX for VVCAS and VSCAS queries and come second to in CO+S queries with general quantization.

4 Conclusion

With the advent of the information age and the widespread use of the Internet, there is an unprecedented demand for effective and efficient techniques for handling these huge amount of data. Querying this data and extracting relevant information out of it is not enough today, user wants more focussed and specific information with as little non-relevant information as possible. XML as semi-structured data seems a potential candidate to meet our needs. The survey attempts to summarize the recent efforts that have been made in this direction. Database community sees the solution to the problem by applying traditional database techniques to XML data. They formulate SQL-like query languages, tries to address the problem of data-integrity and referential integrity. On the other hand, IR community applies with some modification standard IR techniques for focussed element-level retrieval. But despite some similarities with unstructured text, XML needs special treatment in terms of relevance of its elements to a user query and in its evaluation. Hence we need a new paradigm in its retrieval techniques and evaluation metrics. On the whole, XML as an research area holds immense prospect which is not still extensively explored and therefore remains an interesting field of further research.

Acknowledgments

My sincere thanks to my supervisor Dr. Mandar Mitra for his constant guidance and inspiration, Prof. S.K.Parui for his academic assistance, my senior colleague Prasenjit Majumdar for his valuable suggestion. I owe to Yosi Mass, Shlomo Geva,Borkur Sigurbjornsson and Paul Ogilvie who helped me a great deal by sending copies of some of the articles listed in the bibliography.

References

[Abi99] Serge Abiteboul. On views and xml. In *Proceedings of ACM SIGMOD Records*, pages 30–38, New York, NY, USA, 1999. ACM Press.

[AKJ05] P. Arvola, J. Kekalainen, and M. Junkkari. Trix experiments at inex 2005. In *Pre-proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 110–124, 2005.

[AM98] Gustavo Arocena and Alberto Mendelzon. Weboql: Restructuring documents,databases and webs. In *Proc. of Int. Conf. on Data Engineering(ICDE)*, Orlando, florida, USA, 1998.

[AQM+97] Serge Abiteoul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet Wiener. The lorel query language for semi-structured data. 1(1):68–88, 1997.

[BGK+02] P. Bohannon, S. Ganguly, H. F. Korth, P. P. S. Narayan, and P. Shenoy. Optimizing view queries in rolex to support navigable result trees. http://www.bell-labs.com/project/rolex/, 2002.

[Cat97] R. G. Cattell. *The Object Database Standard:ODMG 2.0*. Morgan Kaufmann, 1997.

[CMS01] D. Carmel, Y. Maarek, and A. Soffer. Xml and information retrieval:a sigir 2000 workshop. In *ACM SIGMOD Record*, volume 30(1), pages 62–65, 2001.

[Coo68] W.S. Cooper. Expected search length : A single measure of retrieval effectiveness based on the weak ordering action of retrieval systems. 19(1):30–41, 1968.

[cor00] Excelon corp. Objectstore: object data management system. http://www.exceloncorp.com/products/objectstore/os_overview.html, 2000.

[Cro05] C. Crouch. Relevance feedback at the inex 2004 workshop. In *SIGIR Forum*, volume 39(1), pages 41–42, 2005.

[Dop05] P. Dopichaj. The university of kaiserlautern at inex 2005. In *Pre-proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 158–168, 2005.

[Dun97] M. Dunlop. Time, relevance and interaction modelling for information retrieval. In *Proceedings of the 20th ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–213, Philadelphia,PA, USA, 1997. ACM Press.

[FAA+99] D. Florescu, A.Deutch, A.Levy, D.Suciu, and M. Fernandez. A query language for xml. In *Proc. 8th International World Wide Web Conference*, 1999.

[FFLS97] Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. A query language for a web-site management system. In *SIGMOD Record*, volume 26(3), pages 4–11, 1997.

[FLM98] Daniela Florescu, Alon Levy, and Alberto Mendelzon. Database techniques for the world-wide web : A survey. In *SIGMOD Record*, volume 27(3), pages 59–74, 1998.

[Gev05] S. Geva. Gpx-gardens point xml ir at inex 2005. In *Pre-proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 195–198, 2005.

[GK03] N. Gövert and G. Kazai. Overview of the initiative for the evaluation of xml retrieval (inex) 2002. In *N. Fuhr, N. Gövert, G. Kazai, and M. Lalmas, editors, Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)*, pages 1–17, Dagstuhl,Germany, 2003.

[GKFL03] N. Gövert, G. Kazai, N. Fuhr, and M. Lalmas. Evaluating the effectiveness of content-oriented xml retrieval. Technical Report Technischer bericht, Computer Science 6, University of Dortmund, 2003.

[HB05] M. Hassler and A. Bouchachia. Searching xml documents - preliminary work. In *Pre-proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 95–109, 2005.

[Hie01] D. Hiemstra. *Using language models for information retrieval.* PhD thesis, University of Twente, 2001.

[HLLS97] Rainer Himmeroder, George Lausen, Bertram Ludascher, and Christian Schlepphorst. On a declarative semantics for web queries. In *Proc. of the Int.Conf. on Deductive and Object-Oriented Databases(DOOD)*, pages 386–398, Singapore, December 1997. Springer.

[Hub05] G. Hubert. Xml retrieval based on direct contribution of query components. In *Pre-proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 138–149, 2005.

[Hul93] D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–338, Pittsburgh,PA, USA, 1993. ACM Press.

[KL05a] G. Kazai and M. Lalmas. Inex 2005 evaluation metrics. http://inex.is.informatik.uni-duisburg.de/2005/index.html, 2005.

[KL05b] G. Kazai and M. Lalmas. Notes on what to measure in inex. In *Proceedings of the INEX 2005 Workshop on Element Retrieval Methodology*, pages 22–38, Glasgow,UK., 2005.

[KLM02] G. Kazai, M. Lalmas, and S. Malik. Inex guidelines for topic development. In *Proceedings of the 1st workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 178–181, 2002.

[KLM03] G. Kazai, M. Lalmas, and S. Malik. Inex '03 guidelines for topic development. In *Proceedings of the 2nd workshop of the initiative for the evaluation of XML retrieval (INEX)*, 2003.

[KLV04] G. Kazai, M. Lalmas, and A.P. Vries. The overlap problem in content-oriented xml retrieval evaluation. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 7279, Sheffield,UK, July 2004. ACM.

[KRS05] J. Kamps, M. Rijke, and B. Sigurbjörnsson. The importance of length normalization for xml retrieval. 8:631–654, 2005.

[Lal05] M. Lalmas. Inex 2005 retrieval task and result submission specification. Technical Report Technical report, Queen Mary, University of London, 2005.

[Lar03] R.R. Larson. Cheshire ii at inex 03: Component and algorithm fusion for xml retrieval. In *Proceedings of the Second Annual Workshop of the Initiative for the Evaluation of XML retrieval(INEX)*, pages 38–45, Dagstuhl, Germany, 2003.

[Lar05] R.R. Larson. A fusion approach to xml structured document retrieval. 8:601–629, 2005.

[LCDL00] R. Luk, A. Chan, T. Dillon, and H.V. Leong. A survey of search engins for xml documents. http://www.haifa.il.ibm.com/sigir00-xml/final-papers/Luk/XMLSUR.htm, 2000.

[Lee97] 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*, pages 267–276, Philadelphia,USA, July 27-31 1997. ACM.

[Leh05] M. Lehtonen. When a few highly relevant answers are enough. In *Pre-Proceedings of the Fourth Annual Workshop of the Initiative for the Evaluation of XML retrieval(INEX)*, pages 215–216, 2005.

[Lev66] V. Levenshtein. Binary codes capable of correcting deletions,insertions and reversals. 10(8):707–710, 1966.

[Lev99] Alon Levy. More on data management for xml. http://www.cs.washington.edu/homes/alon/widom-response.html, 1999.

[LLD+02] R.W.P. Luk, H.V. Leong, T.S. Dillon, A.T.S. Chan, W.B. Croft, and J. Allan. A survey in inedexing and searching xml documents. 53(6):415–437, 2002.

[LM05] M. Lalmas and S. Malik. Inex 2004 retrieval task and result submission specification. In *Fuhr,N., Lalmas,M., Malik,S., Szlavik,Z.(Eds) Advances in XML Information Retreival. Third Workshop of the INitiative for the Evaluation of XML Retrieval INEX 2004*, volume 3493, pages 237–240, Schloss Dagstuhl, Germany, 2005. LNCS.

[LMR+05] J. List, V. Mihajlovic, G. Ramirez, A.P. Vries, D. Hiemstra, and H.E. Blok. Tijah : Embracing ir methods in xml databases. 8:547–570, 2005.

[LRM05] W. Lu, S. Robertson, and A. Macfarlane. Field-weighted xml retrieval based on bm25. In *Pre-Proceedings of the Fourth Annual Workshop of the Initiative for the Evaluation of XML retrieval(INEX)*, pages 126–137, 2005.

[LZ01] J. Laferty and C. Zhai. Document language models,query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval(2001)*, pages 111–119. ACM Press, 2001.

[MM03] Y. Mass and M. Mandelbrod. Retrieving the most relevant xml component. In *Proceedings of the Second Annual Workshop of the Initiative for the Evaluation of XML retrieval(INEX)*, pages 53–58, Dagstuhl, Germany, 2003.

[MM05a] Y. Mass and M. Mandelbrod. Component ranking and automatic query refinement. In *INEX 2004, Lecture Notes in Computer Science*, volume 3493, pages 73–84. Springer-Verlag GmbH, 2005.

[MM05b] Y. Mass and M. Mandelbrod. Experimenting various user models for xml retrieval. In *Pre-Proceedings of the Fourth Annual Workshop of the Initiative for the Evaluation of XML retrieval(INEX)*, pages 150–157, 2005.

[MM05c] Y. Mass and M. Mandelbrod. Relevance feedback for xml retrieval. In *INEX 2004, Lecture Notes in Computer Science*, volume 3493, pages 154–157. Springer-Verlag GmbH, 2005.

[OC02] P. Ogilvie and J. Callan. Experiments using the lemur toolkit. In *Proceedings of the Tenth Text Retrieval Conference,TREC 2001*, volume 500-250(2002), pages 103–108. NIST Special publication, 2002.

[OC03a] P. Ogilvie and J. Callan. Language models and structured document retrieval. In *Proceedings of the First Workshop of the Initiative for the Evaluation of XML Retrieval(INEX)*, pages 33–40, 2003.

[OC03b] P. Ogilvie and J. Callan. Using language models for flat text queries in xml retrieval. In *Proceedings of the Second Annual Workshop of the Initiative for the Evaluation of XML retrieval(INEX)*, pages 12–18, 2003.

[OC04] P. Ogilvie and J. Callan. Hierarchical language models for xml component retrieval. In *Proceedings of the Third Annual Workshop of the Initiative for the Evaluation of XML retrieval(INEX)*, 2004.

[OC05] P. Ogilvie and J. Callan. Parameter estimation for a simple hierarchical generative model for xml retrieval. In *Pre-Proceedings of the Fourth Annual Workshop of the Initiative for the Evaluation of XML retrieval(INEX)*, pages 169–181, 2005.

[PG05] B. Piwowarski and P. Gallinari. A bayesian framework for xml information retrieval : Searching and learning with the inex collection. 8:655–681, 2005.

[PGD05] B. Piwowarski, P. Gallinari, and G. Dupret. Precision recall with user modelling : Application to xml retrieval. submitted for publication, 2005.

[PHN01] P.Bohannon, H.F.Korth, and P.P.S. Narayan. The table and the tree:on-line access to relational data through virtual xml documents. In *Proceedings of the WebDB 2001 Workshop on Databases and the Web*, 2001.

[Piw05] B. Piwowarski. Eprum metrics and inex 2005: Draft. In *Pre-Proceedings of the Fourth Annual Workshop of the Initiative for the Evaluation of XML retrieval(INEX)*, pages 1–10, 2005.

[PMM05] E. Popovici, G. Menier, and P.F. Marteau. Sirius:a lightweight xml indexing and approximate search system at inex 2005. In *Pre-Proceedings of the Fourth Annual Workshop of the Initiative for the Evaluation of XML retrieval(INEX)*, pages 234–248, 2005.

[PSGD96] P.Buneman, S.Davidson, G.Hillebrand, and D.Suciu. A query language and optimization technique for unstructured data. In *Proc of ACM SIGMOD Conf. on Management of Data*, pages 505–516, Montreal,Canada, 1996.

[PT05] J. Pehcevski and J.A. Thom. Hixeval: Highlighting xml retrieval evaluation. In *Pre-Proceedings of the Fourth Annual Workshop of the Initiative for the Evaluation of XML retrieval(INEX)*, pages 11–24, 2005.

[PTV05] J. Pehcevski, J.A. Thom, and A.M. Vercoustre. Hybrid xml retrieval:combining information retrieval and a native xml database. 8:571–600, 2005.

[RG03] Raghu Ramkrishnan and Johannes Gehrke. *Database Management Systems, 3rd Edition.* McGraw-Hill, 2003.

[RJ76] Stephen E. Robertson and Karen Sparck Jones. Relevance weighting of search terms. 27:129–146, May-June 1976.

[RJB89] V. V. Raghavan, G.S. Jung, and P. Bollmann. A critical investigation of recall and precision a measures of retrieval system performance. 7(3):205–229, July 1989.

[Rob02] Stephen Robertson. Threshold setting and performance optimization in adaptive filtering. *Inf. Retr.*, 5(2-3):239–256, 2002.

[Roc71] J. Rocchio(Jr.). Relevance feedback in information retrieval. pages 313–323, 1971.

[RZT04] S. Robertson, H. Zaragoza, and M. Taylor. Simple bm25 extension to multiple weighted fields. In *Conference on Information and Knowledge Management (CIKM)*, pages 42–49, 2004.

[Sak04] T. Sakai. New performance metrics based on multigrade relevance : Their application to question answering. In *NTCIR Workshop 4 Meeting Working Notes*, June 2004.

[SBM96] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Research and Development in Information Retrieval,ACM SIGIR '96*, pages 21–29, 1996.

[Sch94] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, 1994.

[SF94] J.A. Shaw and E.A. Fox. Combination of multiple searches. In *Proceedings of the 2nd Text Retrieval Conference (TREC-2)*, pages 243–252. National Institute of Standards and Technology Special Publication 500-125, 1994.

[SKR03] B. Sigurbjörnsson, J. Kamps, and M. Rijke. An element based aapproach to xml retrieval. In *Proceedings of the 2nd workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 19–26, 2003.

[SKR05] B. Sigurbjörnsson, J. Kamps, and M. Rijke. University of amsterdam at inex 2005. In *Pre-Proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 84–94, 2005.

[SM02] T. Schlieder and H. Meuss. Querying and ranking xml documents. 53(6):489–503, 2002.

[SSMB96] A. Singhal, G. Salton, M. Mitra, and C. Buckley. Document length normalization. 32:619–633, 1996.

[ST03] B. Sigurbjörnsson and A. Trotman. Queries: Inex 2003 working group report. Technical Report Proceedings of the 2nd workshop of the initiative for the evaluation of XML retrieval (INEX), 2003.

[STW05] R. Schenkel, A. Theobald, and G. Weikum. Semantic similarity search on semistructured data with the xxl search engine. In *Information Retrieval*, volume 8, pages 521–545, 2005.

[SW05] R. Schenkel and G. Weikum. Relevance feedback for structural query expansion. In *Pre-Proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 260–273, 2005.

[SW06] R. Schenkel and G. Weikum. Feedback-driven structural query expansion for ranked retrieval of xml data. In *Y. Ioannidis et al.(Eds) EDBT 2006,LNCS 3896*, pages 331–348, 2006.

[Tan05] X. Tannier. From natural language to nexi, an interface for inex 2005 queries. In *Pre-Proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 289–303, 2005.

[TS04a] A. Trotman and B. Sigurbjörnsson. Narrowed extended xpath i (nexi). In *Proceedings of the 3rd workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 16–40, 2004.

[TS04b] A. Trotman and B. Sigurbjörnsson. Nexi, now and next. In *Proceedings of the 3rd workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 41–53, 2004.

[TSW05] A. Theobald, R. Schenkel, and G. Weikum. Topx xxl at inex 2005. In *Pre-proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 201–214, 2005.

[VG05] J.N. Vittaut and P. Gallinari. Machine learning ranking and inex '05. In *Pre-proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 251 – 259, 2005.

[WF74] R. Wagner and M. Fischer. String-to-string correction problem. 12(1):168–173, 1974.

[WG05a] A. Woodley and S. Geva. Nlpx at inex 2005. In *Pre-proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 274–288, 2005.

[WG05b] A. Woodley and S. Geva. Xcg overlap at inex 2004. In *Pre-proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 25–39, 2005.

[Wid99] Jennifer Widom. Data management for xml. http://www-db.stanford.edu/∼widom/xml-whitepaper.html, 1999.

[YBS04] S.A. Yahia, C. Botev, and J. Shanmugasundaram. Texquery: A full-text search extension to xquery. In *WWW '04*, 2004.

[YHSS05] S.A. Yahia, K. Hatano, J. Shanmugasundaram, and D. Srivastava. An evaluation of relevance ranking methods for xml using both document and query structures. In *Pre-proceedings of the 4th workshop of the initiative for the evaluation of XML retrieval (INEX)*, pages 249–250, 2005.

[YKA+05] S.A. Yahia, N. Koudas, A.Marian, D. Srivastava, and D. Toman. Structure and content scoring for xml. www.vldb2005.org/program/paper/wed/p361-amer-yahia.pdf, 2005.

[ZBOW05] R.V. Zwol, B. Baas, H.V. Oostendorp, and F. Wiering. Query formulation for xml with bricks. In *Trotman.A.,Lalmas,M., Fuhr,N.(Eds) Proc. of INEX 2005 Workshop on Element Retrieval Methodology*, pages 80–88, Glasgow, Scotland, UK, 2005.

[ZL01] C. Zhai and J. Laferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval(2001)*, pages 334–342. ACM Press, 2001.