# Information Retrieval for XML Documents
# Seminar Work

**Marcel Braasch**
Ludwig-Maximilians-Universität München
marcelbraasch@gmail.com

## 1 Introduction

Information retrieval (IR) in the context of natural language processing (NLP) can be referred to as finding data in a structured or unstructured corpus to meet an information need (Schütze et al., 2008). The information need is usually framed in terms of a query the user expresses. Corpora may contain, e.g., newspaper articles, published papers or conference results. While newspaper articles are often subject of classical IR (Iwayama et al., 2003), especially journal articles (along other sources) induce structural information onto the document at hand (Fuhr et al., 2005).

Documents are typically organized within sections, subsections (and possibly deeper levels) as well as paragraphs containing the actual textual information. Unlike unstructured, running text (even though the structure might be implicitly induced), XML documents encode this tree structure explicitly. If the corpus at hand is comprised of XML documents, one may call the IR process *XML retrieval*. The prevalent research question at hand is whether this explicit structure is informative enough to boost retrieval performance (Fuhr et al., 2006).

Although XML documents induce more information and potentially increases accuracy, the retrieval task becomes more complex. Indexing needs to be reconsidered. Does one want to index all elements? At what granularity? How does one index structural information? Similar questions arise for what parts of the document to present to the user. XML documents bring along a variety of tools such as XPath or XQuery. Would one assume that the user is capable of formulating a query accordingly, *and* has knowledge of the structure? By only a few, simple questions we can see that XML retrieval surely is a complicated matter.

The work presented here is structured as following. In Section 2 (XML Overview) I briefly review

```
<?xml version="1.0" encoding="UTF-8"?>
<LMU>
  <faculty no="13" dpt="CIS">
    <prof name="Adalbert">
      <focus>Deep NLP</focus>
      <papers in="2021">21</papers>
      <papers in="2020">41</papers>
      <papers in="2019">32</papers>
    </prof>
    <prof name="Berthold">
      <focus>Knowledge Extraction</focus>
      <papers in="2021">7</papers>
      <papers in="2020">6</papers>
      <papers in="2019">10</papers>
    </prof>
    <prof name="Christiansen">
      <focus>Deep MT</focus>
      <papers in="2021">3</papers>
      <papers in="2020">17</papers>
      <papers in="2018">10</papers>
    </prof>
  </faculty>
</LMU>
```

Figure 1: A typical XML document showing structural properties of CIS called `cis.xml`.

XML documents. Syntax, semantics and the document object model are concisely discussed. In Section 3 (XML Retrieval) I present the base of XML retrieval and manipulation. XPath, XQuery and XSLT are all subject of this section. In Section 4 (Advances in XML Retrieval) current methods, and the environment in which these were developed, are showcased. I introduce the driving force in XML retrieval, INEX, move on the a high-level overview of research attempts and lastly discuss a few selected approaches. The work is then finally concluded in Section 5 (Conclusion).

## 2 XML Overview

This section shall serve as a quick knowledge refresher on XML documents. I briefly cover the motivational and structural aspects of XML. In the case the reader is not familiar with XML and is looking for a more fine-grained introduction, I recommend reading Harold et al. (2004) or Fawcett

et al. (2012).

The **E**xtensible **M**arkup **L**anguage (XML) is a metamarkup language defined by the W3C-endorsed standard [1]. XML is a document markup for structurally representing entities in textual or numerical form. Documents can have various appearances. For example, XML is a common tool for describing configuration files, mathematical equations, API definitions or vector graphics (Walsh, 1998). Being human and machine readable simultaneously, XML brings along a powerful toolkit to describe and analyze complex structures. Since this essay explicitly discusses XML in the context of information retrieval (IR), I consider XML as a means of encoding *running text in a (semi-)structured* format only.

### 2.1 Syntax and Entities

Describing XML documents is simple as there are only a handful of rules to pay attention to. The most important are: **(1)** XML documents must have a unique root element. This corresponds to the opening `<LMU>` tag in Figure 1. **(2)** Every element needs a closing tag. This corresponds to the closing tag `</LMU>`. Further, **(3)** need to be properly nested. For example, while `<a><b></b></a>` is a legal declaration, `<a><b></a></b>` is clearly not. **(4)** The `<?xml> tag` in the beginning of the document is optional and may be omitted as it just indicates some meta information. There are a few other rules, however, the aforementioned are the most important (Beckett and McBride, 2004). This shows that describing XML documents is a quite simple endeavor.

XML elements can possess attribute values, for example, the `<faculty>`-element is augmented by its *number* and the *name*. Attributes append additional knowledge and can later be referenced and queried. Similarly, the three `<prof>` elements are defined, enclosing their research `<focus>` and the amount of `<papers>` published in a certain year. Note that the granularity of an XML document is arbitrary. It can be tailored to ones needs. For example, if we wanted to include doctoral researchers or administrative staff, we would likely enclose the `<prof>`-element inside a `<profs>`-element. Similarly, PhD students would be enclosed inside `<PhDs>`.

[1] w3.org/TR/xml

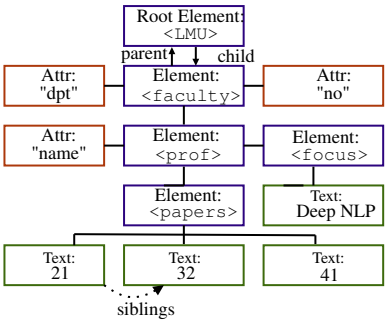


Figure 2: DOM tree realization of *cix.xml* shown in Figure 1. Note that two professors have been omitted. They follow the same structure as the one shown.

### 2.2 Document Object Model

The Document Object Model (DOM) is a general API for describing, accessing and manipulating tree structures such as XML or HTML (Wood et al., 1998). In principle, the DOM provides a clearly defined set of operations accessible from all common programming languages. This allows for more flexibility in terms of convertibility. DOM operations range from accessing nodes (e.g., getting a list of nodes, retrieving node information, etc.) to manipulation the tree (e.g., removing or adding nodes, traversing the tree, etc.). Figure 1 described as a DOM-tree can be seen in Figure 2.

### 2.3 Semantics

Unlike HTML, XML does not inherently encode semantics (Lie and Saarela, 1999). They have to be induced by the programmers themselves. Either, developers implicitly agree on a convention (which of course can be disregarded) or define explicit schemas to follow (Skulschus and Wiederstein, 2008). Though XML schemas (which I choose not discuss in this work) define a strict set of rules in terms of syntax, they do not encode clear semantics (Renear et al., 2002). There have been attempts in analyzing XML schemas inducing semantics (Li and Miller, 2005). However, it seems that current work solely assesses ways of analyzing how an XML documents conforms to a schema, not the inherent meaning of an XML document (as for HTML documents).

## 3 XML Retrieval

Now that I have covered the basics of XML, I am moving on to XML retrieval. Information retrieval is typically defined as finding data in a structured or unstructured corpus to meet an information need

(Schütze et al., 2008). In the context of XML retrieval, usually the corpora at hand are at least semi-structured. Either a clear hierarchy with distinct element names is defined, or elements are depicted as running text, and possibly need further processing (Lalmas, 2009).

Similar to relational database (RDB) systems, a corpus of XMLs can be queried, too. However, there are significant differences to note (Schütze et al., 2008). Using the structured query language (SQL), one queries inter-connected tables defined by a relational model. Table entries are set up according to a fixed datatype. Algebraic operators such as joins, projections or unions are another important property of RDBs (Codd, 1989).

In contrast to that, in XML retrieval, the model at hand is defined by a tree. Nodes can be textually augmented by tags and leaves contain text. The text at hand does not have to be human language but can contain any type of information (Hosoya et al., 2005; Murata et al., 2001).

There is a variety of tools for XML document manipulation. Each serves its own purpose. The most common standards are XSLT and XQuery, which at its core are based on XPath. There are other important components of XML processing, for example, XLinks, XPointer, XForms or extensions of XPath, like Narrowed Extended XPath I (NEXI) (Trotman and Sigurbjörnsson, 2004; Wilde and Lowe, 2003). However, these are not in the scope of this work and I will focus on the most important standards below.

### 3.1 XPath

XPath is used to query a document, navigate through a tree and extract information accordingly. Queries can be formulated in absolute and relative terms allowing for powerful selections (Harold et al., 2004). XPath is one of the (if not the) most important concepts in XML retrieval. It lies at the core of XQuery and XSLT, as well as many other manipulation concepts such schema constraints (Van der Vlist, 2002). This is owed to its simplistic functional nature. XPath is a simple, non-XML language which enables users to address and retrieve specific parts of an XML (Benedikt and Koch, 2009).

The XPath syntax looks a lot like classical web page URIs combined with the Unix shell (Harold et al., 2004). In XPath a multitude of operations can be executed, that is: **(1)** selecting nodes, **(2)** using predicates, **(3)** specifying multiple paths, **(4)** absolute and relative path selection and **(5)** a variety of functions including string, number, node and boolean operators.

For example, if we wanted to select *"all professors"* from Figure 1 we would formulate it as `"/LMU/faculty/prof"`, which would return three distinct elements, one for each professor. `"/"` selects the specified node, starting from the root node. We could equivalently formulate the above query as `"//prof"`, which uses `"//"` to look for matching nodes, independently of its location. Similarly to the Unix file system, `".."` and `"."` can be used to navigate back and forth respectively.

Further, we can extend the above query to specify our needs. For example, the question *"Which professors have published 10 papers at least once?"* can be formulated as `"//prof[papers > 10]"`, where `"papers > 10"` is the boolean predicate in use. Questions related to a node attribute can be formulated alike. For example `"//prof[@name]"` would select all professors which have a name, whereas `"//prof[@name="Christiansen"]"` would specifically select the professor with a name we might be looking for.

Clearly, the above is only a portion of what is possible to express with XPath. Since I want to use the space to not only discuss the basics I am concluding this section and refer to great tutorials like Javatpoint's [2] and RIP Tutorial's introduction [3].

### 3.2 XQuery

XQuery can be viewed as a functional programming language which defines a set of operations to query, combine and manipulate XML document collections (Boag et al., 2002). In principle, XQuery was designed as a functional equivalent of SQL for RDBs (DeHaan et al., 2003). Besides querying documents, XQuery can be used to build web pages (Gui et al., 2010) and transform XML to XHTML documents (Kaufmann and Kossmann, 2009). While XSLT is a powerful tool designed to transform entire collections, XQuery is typically the more light-weight option to perform simple operations expressible in few lines of code.

XQuery is based on the FLWOR principle (Beyer et al., 2005), which is an acronym for *for*,

[2] `javatpoint.com/xpath-tutorial`
[3] `riptutorial.com/xpath`

*let*, *where*, *order*, *return*. *For* in combination with *let* yield an ordered sequence of elements to be processed. The *where* term allows to specify a boolean expression to denote which parts to retrieve. The function of *order* arranges elements as wished and *return* lets the user specify which parts of the structure at hand to present. Additionally, XQuery provides a collection of sophisticated aggregation functions (Van Cappellen et al., 2008), string operations, if-then-else constructs as well as support for regular expressions (Harold et al., 2004). If you view the following query, the expressiveness of XQuery's toolkit becomes apparent quickly.

```
for $x in doc("cis.xml")//prof
where matches($x/focus, $Deep)
return $x[@name]
```

We loop over every professor in the xml-file defining the bound variable *"$x"*. Next, we look for regular expressions which match *"Deep"*. For every such match we want to return the name of the respective professor. Note that, compared to solely using XPath, we can be much more concise in formulating information needs. If we only used XPath we would have to use a combination of relative path selection, going back the tree and string functions. Clearly, XQuery is much more convenient for this purpose.

Again, the above is just an extract of what is possible to express with XQuery. For the sake of the scope of this work, I am concluding this section by referring to Javatpoint's tutorial [4] for further reading.

### 3.3 XSLT

Extensible Stylesheet Language Transformations (XSLT) process documents and transform its representation for further use. Transformations range from simply capitalizing headings to performing knowledge extraction from documents (McManigal, 2005). One of the most common uses of XSLT is transforming an XML document into an XHTML representation (Myllymaki, 2001). This allows most browsers to semantically interpret the contents of the document and display it accordingly, which would not be possible if the document was kept in XML format.

In Figure 3 one can see a simple example for transforming `cis.xml` into `cis.xhtml`. The

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
  <h2>Professors at CIS</h2>
  <table border="1">
    <tr bgcolor="#ffffff">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <xsl:for-each select="//prof">
    <tr>
      <td><xsl:value-of select="focus"/></td>
      <td><xsl:value-of select="papers"/></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Figure 3: The XSLT sheet extracts entities from `cis.xml` and further processes them into in XHTML file.

procedure is simple. First, one defines the XML, XHTML and XSLT boilerplate in rows 1 to 5 (which do not want to further discuss). This is followed by defining the HTML structure we want to define. In principle, the contents we want to retrieve from the XML document gets embedded into the HTML structure. Therefore, one needs to define *what* to extract and *where* to put it. In this example, this is expressed as `<xsl:for-each select="//prof">`, which specifies to loop over every professor. Since we want to extract each professor's research focus we formulate `<xsl:value-of select="focus"/>`. As can be seen, this gets embedded into an HTML table accordingly.

The example presented in this work only showcases a small portion of available operations. Besides the `<value-of>` and `<for-each>` elements other typical operations include `<sort>`, `<sort>`, `<if>`, `<choose>`. This incorporates a wide range of applications. As a matter of fact, both XSLT and XQuery are Turing complete programming languages, suggesting a mere infinite range of applications (Kepser, 2004).

## 4 Advances in XML Retrieval

From 2002 up until 2014 the driving force in XML retrieval was the **IN**itiative for the **E**valuation of

[4] www.javatpoint.com/xquery-tutorial

**X**ML retrieval program, short INEX [5]. In an annual meeting, advances in the field were presented, reference collections created and approaches systematically compared on common benchmarks. For example, the initial test collection was comprised of 12.000 IEEE articles, a set of queries and targets.

The meeting was eventually discontinued as a wide range of topics were already covered in breadth and depth. In order the provide a review on previous XML retrieval methods I systematically reviewed the INEX meetings from 2004 to 2012. Meeting notes prior to 2004 ('02, '03) and after 2012 ('13, '14) are not publicly available, thus are omitted. In Figure 6 one can examine the available tracks for every year.

In the following, I will show a variety of interesting approaches. The topics selected strongly orient themselves towards Figure 6. I will cover two important areas and point to interesting papers in the field, that is; **(1)** *Ad Hoc* retrieval, which was the most vital research direction and the main track until 2010, **(2)** *Relevance Feedback*, which has seen much play in the early emergence of the meetings. It was discontinued for a few years until in the last years of INEX it was resumed.

### 4.1 Ad Hoc Retrieval

The *Ad Hoc* retrieval track is the most important track introduced by the INEX (Fuhr et al., 2006). Ad Hoc methods usually refer to promptly meeting a user's information need, formulated through a query, by providing a subset of relevant documents from a designated corpus (Schütze et al., 2008). The main research question which emerged is whether structural document information (as the one XML documents provide) are informative enough to boost retrieval performance (Fuhr et al., 2008).

#### 4.1.1 Tasks

It is important to note that tasks at INEX, typically one distinguishes between two high-level objectives. These objectives are comprised of Content-only (CO) retrieval and Content-and-structure (CAS) retrieval. CO retrieval solely focuses on providing relevant documents, given a user query. It does not specify in which parts of the document information may occur (Fuhr et al., 2005). It assumes the user has no prior structural knowledge about the document collection , which fits most

[5] inex.mmci.uni-saarland.de

**INEX CAS topic 149**
//article[about(.//(abs|kwd), "genetic algorithm")]
//bdy//sec[about(., simulated annealing)]

Figure 4: A CAS query formulated in NEXI style. The query specifies that "genetic algorithm" shall be contained in the *article* part of the document, and "simulated annealing" in an arbitrary section.

**INEX CO topic 166**
+"tree edit distance" +XML -image

Figure 5: A CO query. The query specifies that retrieved results shall include "tree edit distance", "XML" and exclude "image" (similar to a simple Google search).

casual users using such a system (Fuhr et al., 2005). CAS retrieval, on the other hand, assumes that users can explicitly induce structural information when formulating a query. This style of retrieval usually refers to librarians. In Figure 4 and Figure 5 typical CAS and CO queries provided by INEX can be viewed.

Further, there are four rather fine-grained sub tasks introduced starting from INEX 2007 (Fuhr et al., 2008). That is, the **(1)** *Focused Task*, which returns the best possible match for the user in terms of elements or passages. Importantly, choosing the right granularity poses a major challenge, as ancestors of important element, and longer passages are likely relevant, too. The **(2)** *Relevant in Context Task*, which returns a ranked list of documents, where each document inhabits relevant elements (which do not have to be adjacent in any way). The **(3)** *Best in Context Task*, which shall provide the user with a single "best-entry-point" to start reading. And lastly, the **(4)** Thorough Task, which focuses on returning relevant elements only.

Though knowing of the respective sub tasks is not important to understand the following sections, it shall demonstrate that research efforts have been channeled into multiple more fine-grained directions. Most models proposed in the meetings indeed address all tasks.

#### 4.1.2 Methods

Approaches for Ad Hoc retrieval are manifold. Most ideas emerge from a view on query extension (Ibekwe-Sanjuan and SanJuan, 2008; Mass and Mandelbrod, 2004a)

| | INEX | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Topics** | **04** | **05** | **06** | **07** | **08** | **09** | **10** | **11** | **12** |
| Methods | X | X | X | | | | | | |
| Ad Hoc | X | X | X | X | X | X | X | | |
| Relevance Feedback | X | X | | | | | X | X | X |
| Heterogenous Documents | X | X | X | X | X | X | X | X | X |
| NLP Query | X | X | X | | | | | | |
| Interactive | X | X | X | X | X | X | X | | |
| Document Mining | | X | X | X | X | X | X | | |
| Multimedia Track | | X | X | X | | | | | |
| Entity Ranking | | | | X | X | X | | | |
| Link-the-Wiki | | | | X | X | X | X | | X |
| Efficiency | | | | | X | X | | | |
| Question Answering | | | | | | X | X | | |
| Data Centric | | | | | | | X | X | |
| Web Service Discovery | | | | | | | X | | |
| Snippet Retrieval | | | | | | | | X | X |

Figure 6: Tracks available from INEX 2004 until INEX 2012.

logic (Karimzadegan et al., 2004; Tanioka, 2006), rational algebra or databases (Fujimoto et al., 2005; Vittaut et al., 2004), language modelling (Sigurbjörnsson et al., 2004; Li and Van Der Weide, 2009; SanJuan and Ibekwe-SanJuan, 2009), indexing (Lehtonen and Doucet, 2008; Sigurbjörnsson and Kamps, 2005), vector space models (Pal et al., 2009; Weigel et al., 2004), machine learning (Larson, 2005; Gao et al., 2009), and others (Wang et al., 2009; Carpineto et al., 2006). There are also mixtures of these methods (Popovici et al., 2006; Hubert, 2004). The prior classify the various directions taken in a rather coarse grained manner. Since presenting work from all categories is not in the scope of this work, I will limit the selection to approaches in language modelling and machine learning as they coincide with my personal research interests.

**Language Modelling (LM) for XML retrieval** Language model (LM) approaches in information retrieval have seen early play in the 90s (Ponte and Croft, 1998; Song and Croft, 1999). Usually, the general idea is to estimate the probability of the query, given the document. Following Ponte and Croft (1998), this can be written as

$$\mathbb{P}(q|d) = \prod_{k=1}^{|q|} \frac{tf(t_i, d)}{l_d}$$

where $q$ is the query, $d$ the document, $t_i$ the query terms, $tf$ is the raw term frequency of $t_i$ in $d$ and $l_d$ the document length. Note that this formulation assumes query term independence. Clearly, this is a faulty assumption. However, assuming Markov property has proven to be an effective measure to simplify expressions in NLP (Blunsom, 2004; Roth, 1999).

Sigurbjörnsson et al. (2004) provide an interesting LM approach. They explicitly exploit the XML structure by indexing on different levels. Three indices are created. The first builds an inverted index at element level. This can go as far as literal words, e.g., marked in italic. The second builds an inverted index at article level. That is, the entire document is indexed as if it was one running text. Thirdly, they structurally index the document according to Grust (2002), that is, the pre- and post-order tree information is encoded.

Once indexed, the following multinonomial language model is employed (Hiemstra, 2001).

$$\mathbb{P}(x|q) \propto \mathbb{P}(x) \cdot \mathbb{P}(q|x) \quad (1)$$

$$= \mathbb{P}(x) \cdot \prod_{i=1}^{|q|} \mathbb{P}(t_i|x) \quad (2)$$

where $x$ denotes the XML document, $\mathbb{P}(x)$ is the prior belief about $x$, $q$ the query composed of terms $t_i$. Again, this model assumes term independence between query terms, hence the factorization into a product. Further, $\mathbb{P}(q|x)$ describes the likelihood of the query given a document. Since we want to take into account element, document and structural information we further decompose the product term in (2) into

$$\lambda_e \cdot \mathbb{P}_{mle}(t_i|e) \quad (3)$$

$$+ \lambda_d \cdot \mathbb{P}_{mle}(t_i|d) \quad (4)$$

$$+ (1 - \lambda_e - \lambda_d) \cdot \mathbb{P}_{mle}(t_i) \quad (5)$$

where (3) refers to the probability of query term $t_i$ given XML element $e$, (4) denotes the same on document level and (5) accounts for the structural index. The $\lambda$ parameters form a convex combination of the prior and present a hyperparameter. All probabilities are estimated using corpus statistics through maximum likelihood estimation. The final results imply that exploiting structural information indeed enhances retrieval accuracy. Further, the authors note that CAS retrieval is much more accurate than CO retrieval, that is, adding structural constraints increases accuracy significantly (which is not too surprising as we induce additional information).

**Machine Learning (ML) for XML retrieval** Machine Learning (ML) approaches, refer to algorithms which do not receive explicit instructions to

perform a certain task, but infer appropriate actions by extracting knowledge through latent patterns induced by data (my definition, hence no citation). ML is deployed in a broad range of disciplines ranging from physics (Sarma et al., 2019), chemistry (Mater and Coote, 2019), operations research (Sra et al., 2012), or the now ubiquitous autonomous driving (Shalev-Shwartz et al., 2016).

ML in information retrieval has a long history, too. As a matter of fact, even the earliest work of Gerard Salton's group at Cornell University in the 70s, the vector state model, can be interpreted as an ML model (Dubin, 2004). Learning-to-rank has seen much play ever since the mid 2000s (Cao et al., 2007; Xia et al., 2008).

Gao et al. (2009) apply a learning-to-rank method to XML retrieval. Their approach is fairly straightforward as they simply learn the parameters of the well known Okapi BM25 algorithm adapted to XML retrieval (Robertson and Zaragoza, 2009). Formally, $score(E, Q)$ is defined as

$$\sum_{i=1}^{n} IDF(q_i) \cdot \frac{tf(q_i, E) \cdot (k_1 + 1)}{tf(q_i, E) + k_1 \cdot (1 - b + b \cdot \frac{|E|}{avgdl})}$$

where $E$ is an indexed XML element, $Q$ is the query and $q_i$ its terms, $tf$ the term-frequency of term $t_i$ in element $E$, $IDF$ denotes the inverse-document-frequency, $avgdl$ is the avarage document length across all XML elements, and $k_1$ and $b$ are learnable scaling parameters to denote an importance trade-off between average XML element length and term-frequency.

The model is trained through the INEX 2008 data collection (this idea was implemented at INEX 2009). The data is comprised of a set of queries $Q = \{q^1, q^2, \ldots, q^m\}$, where each is associated with a ranked list of correct search results $D^i = \{d_1^i, d_2^i, \ldots, d_n^i\}$. Next, for each query BM25 retrieves a set of results $R^i = \{r_1^i, r_2^i, \ldots, r_n^i\}$. The "distance" between $R^i$ (the prediction) and $D^i$ (the ground truth) is then compared in terms of the loss function

$$\sum_{i=1}^{m} L(D^i, R^i) = \sum_{i=1}^{m} \frac{mn^2}{\sum_{j=1}^{n} rank_j^i}$$

where $rank_j^i$ is defined as

$$rank_j^i = \begin{cases} 0 & \text{if the j-th result in } R^i \text{ is not in } D^i \\ k & \text{otherwise} \end{cases}$$

In principle, the loss can be interpreted as a counting mechanism (also, it looks much like a 0-1, or 0-k, loss (Friedman, 1997)). $rank_j^i$ increases if documents retrieved are a hit. Thus, due to its inverse nature, $L$ decreases. The ListBM algorithm proposed by Gao et al. (2009) then iteratively computes the loss for each query. As long as the loss decreases, $k_1$ is updated as $k_1 = k_1 + \frac{1}{L(D^i, R^i)}$. The parameter $b$ is independently trained just alike. Unfortunately, the authors do not cover jointly training the parameters, which, theoretically should be computationally tractable as the search space is not very large.

The proposed algorithm nicely showcases how well-known procedures can be slightly modified to obtain reasonably good results. The authors state that ListBM outperforms classical BM25 by a large extend in all subtasks.

### 4.2 Relevance Feedback

Relevance feedback (RF) in the context of information retrieval usually refers to an iterative process which refines the user query $Q$ to obtain better results. Documents originally returned to the user are marked as relevant ($R$) or non-relevant ($NR$). Based on this feedback, $Q$ gets modified such that relevant documents $R$ are highly ranked and non-relevant documents $NR$ get a low score. In the best case, an additional, not seen before, set of relevant documents emerges through this process (Mass and Mandelbrod, 2004b).

RF has seen early play in information retrieval. The first of algorithm of this type was presented in Rocchio (1971), called Rocchio's Algorithm. The algorithm operates on documents and queries embedded in a semantic space, i.e., assumes an underlying vector space model. Mass and Mandelbrod (2004b) use Rocchio's Algorithm for XML retrieval. Their algorithm can be examined in Figure 7.

The main challenge for RF in XML retrieval lies in the rather complicated indexing. While in classical IR only documents themselves are indexed, in XML retrieval we can choose an arbitrary indexing strategy. Usually, paragraphs, sections and entire documents are indexed, however, other strategies may apply, too (Li et al., 2001).

The above algorithm therefore iterates over each index, computes a result set $Res_i$ and performs a series of normalization methods, including DocPivot. DocPivot in principle makes up for varying doc-

**Algorithm 1**
1. For each index $i$:
   a) Compute result set $Res_i$ given $Q$
   b) Normalize results in $Res_i$ to $[0, 1]$
   c) Apply DocPivot to $Res_i$
2. Merge all $Res^i$ into $Res$ sorted by score
3. Take the top $N$ results from $Res$, extract relevant documents $R$ and non-relevant documents $NR$
4. For each index i:
   a) Apply a RF algorithm, refine $Q$ to $Q'$
   b) Compute the result set $Res'_i$ given $Q'$
   c) Normalize results in $Res'_i$ to $[0, 1]$
   c) Apply DocPivot to $Res'_i$
5. Merge all $Res'_i$ to a single result set $Res'$ sorted by score

Figure 7: Iterative RF algorithm proposed by Mass and Mandelbrod (2004b)

ument length, as they can substantially distort results (Singhal et al., 1996). Once results are retrieved, we merge them into one common, ranked set $Res$. Until here, the algorithm has not yet applied RF. Next, we annotate results as *relevant* or *non-relevant*. Proceeding, for each index we apply a RF algorithm to refine $Q$, and again, perform a series of normalization steps. The RF algorithm at hand may be chosen arbitrarily. However, Rocchio's Algorithm has proven to be an effective way of employing RF. The equation for query refinement can be examined in Equation (6).

$$Q' = \alpha \cdot Q + \frac{\beta}{n_1} \sum_{i=1}^{n_1} R_i - \frac{\gamma}{n_2} \sum_{i=1}^{n_2} NR_i \quad (6)$$

It is important to note that $Q$, $R_i$, $NR_i$ are all vectors, thus the refined query $Q'$ will be vector too. Note that Rocchio's Algorithm tries to maximize the query towards relevant documents $R$ (hence the "+"-sign) and minimize it toward non-relevant documents $NR$ (hence the "−"-sign). $\alpha$, $\beta$, $\gamma$ are tuning parameters which control the importance of each term. In order for this algorithm to work on the document collection at hand, documents are embedded using $tf\text{-}idf$ (Ramos et al., 2003). Unfortunately, the authors only report marginal gains through their approach.

## 5 Conclusion

In this work a broad overview of methods in XML retrieval was showcased. I started from from a general introduction to XML documents, the most common tools at hand to conclude with advances in the field and a few selected methods. XML retrieval was most active during the annual INEX meeting from 2002 to 2014. It seems that ever since these foundations have been laid, the fundamentals of XML retrieval have become less attractive as a research field. However, research results still find application in a wide range of approaches (Dabrowska and Larsen, 2015; Ramya et al., 2016).

## References


Dave Beckett and Brian McBride. 2004. Rdf/xml syntax specification (revised). *W3C recommendation*, 10(2.3).

Michael Benedikt and Christoph Koch. 2009. Xpath leashed. *ACM Computing Surveys (CSUR)*, 41(1):1–54.

Kevin Beyer, Don Chamberlin, Latha S Colby, Fatma Özcan, Hamid Pirahesh, and Yu Xu. 2005. Extending xquery for analytics. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 503–514.

Phil Blunsom. 2004. Hidden markov models. *Lecture notes, August*, 15(18-19):48.

Scott Boag, Don Chamberlin, Mary F Fernández, Daniela Florescu, Jonathan Robie, Jérôme Siméon, and Mugur Stefanescu. 2002. Xquery 1.0: An xml query language.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136.

Claudio Carpineto, Giovanni Romano, and Caterina Caracciolo. 2006. Information theoretic retrieval with structured queries and documents. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 178–184. Springer.

Edgar F Codd. 1989. Relational database: A practical foundation for productivity. In *Readings in Artificial Intelligence and Databases*, pages 60–68. Elsevier.

Anna Dabrowska and Birger Larsen. 2015. Exploiting citation contexts for physics retrieval. In *Second Workshop on Bibliometric-enhanced Information Retrieval: co-located with the 37th European Conference on Information Retrieval (ECIR 2015)*, pages 14–21. CEUR Workshop Proceedings.

David DeHaan, David Toman, Mariano P Consens, and M Tamer Özsu. 2003. A comprehensive xquery to sql translation using dynamic interval encoding. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, pages 623–634.

David Dubin. 2004. The most influential paper gerard salton never wrote.

Joe Fawcett, Danny Ayers, and Liam RE Quin. 2012. *Beginning XML*. John Wiley & Sons.

Jerome H Friedman. 1997. On bias, variance, 0/1—loss, and the curse-of-dimensionality. *Data mining and knowledge discovery*, 1(1):55–77.

Norbert Fuhr, Jaap Kamps, Mounia Lalmas, and Andrew Trotman. 2008. *Focused Access to XML Documents: 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle, Germany, December 17-19, 2007, Revised and Selected Papers*, volume 4862. Springer.

Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Gabriella Kazai. 2006. *Advances in XML Information Retrieval and Evaluation: 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2005, Dagstuhl Castle, Germany, November 28-30, 2005. Revised and Selected Papers*, volume 3977. Springer.

Norbert Fuhr, Mounia Lalmas, Saadia Malik, and Zoltán Szlávik. 2005. *Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2004, Dagstuhl Castle, Germany, December 6-8, 2004*, volume 3493. Springer.

Kei Fujimoto, Toshiyuki Shimizu, Norimasa Terada, Kenji Hatano, Yu Suzuki, Toshiyuki Amagasa, Hiroko Kinutani, and Masatoshi Yoshikawa. 2005. Implementation of a high-speed and high-precision xml information retrieval system on relational databases. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 254–267. Springer.

Ning Gao, Zhi-Hong Deng, Yong-Qing Xiang, and Yu Hang. 2009. Listbm: a learning-to-rank method for xml keyword search. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 81–87. Springer.

Torsten Grust. 2002. Accelerating xpath location steps. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 109–120.

Zhiming Gui, Husheng Liao, et al. 2010. Building web application with xquery. In *International Conference on Web-Age Information Management*, pages 106–109. Springer.

Elliotte Rusty Harold, W Scott Means, and Katharina Udemadu. 2004. *XML in a Nutshell*, volume 8. O'reilly Sebastopol, CA.

Djoerd Hiemstra. 2001. *Using language models for information retrieval*. Citeseer.

Haruo Hosoya, Jérôme Vouillon, and Benjamin C Pierce. 2005. Regular expression types for xml. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 27(1):46–90.

Gilles Hubert. 2004. A voting method for xml retrieval. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 183–195. Springer.

Fidelia Ibekwe-Sanjuan and Eric SanJuan. 2008. Use of multiword terms and query expansion for interactive information retrieval. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 54–64. Springer.

Makoto Iwayama, Atsushi Fujii, Noriko Kando, and Yuzo Marukawa. 2003. An empirical study on retrieval models for different document genres: patents and newspaper articles. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 251–258.

Maryam Karimzadegan, Jafar Habibi, and Farhad Oroumchian. 2004. Logic-based xml information retrieval for determining the best element to retrieve. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 88–99. Springer.

Martin Kaufmann and Donald Kossmann. 2009. Developing an enterprise web application in xquery. In *International Conference on Web Engineering*, pages 465–468. Springer.

Stephan Kepser. 2004. A simple proof for the turing-completeness of xslt and xquery. In *Extreme Markup Languages®*. Citeseer.

Mounia Lalmas. 2009. Xml retrieval. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 1(1):1–111.

Ray R Larson. 2005. Probabilistic retrieval, component fusion and blind feedback for xml retrieval. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 225–239. Springer.

Miro Lehtonen and Antoine Doucet. 2008. Enhancing keyword search with a keyphrase index. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 65–70. Springer.

Jian Bing Li and James Miller. 2005. Testing the semantics of w3c xml schema. In *29th Annual International Computer Software and Applications Conference (COMPSAC'05)*, volume 1, pages 443–448. IEEE.

Quanzhong Li, Bongki Moon, et al. 2001. Indexing and querying xml data for regular path expressions. In *VLDB*, volume 1, pages 361–370.

Rongmei Li and Theo Van Der Weide. 2009. Language models for xml element retrieval. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 95–102. Springer.

Håkon Wium Lie and Janne Saarela. 1999. Multipurpose web publishing using html, xml, and css. *Communications of the ACM*, 42(10):95–101.

Yosi Mass and Matan Mandelbrod. 2004a. Component ranking and automatic query refinement for xml retrieval. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 73–84. Springer.

Yosi Mass and Matan Mandelbrod. 2004b. Relevance feedback for xml retrieval. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 303–310. Springer.

Adam C Mater and Michelle L Coote. 2019. Deep learning in chemistry. *Journal of chemical information and modeling*, 59(6):2545–2559.

Chris A McManigal. 2005. Towards more comprehensive information retrieval systems: Entity extraction using xslt.

Makoto Murata, Simon St Laurent, and Dan Kohn. 2001. Xml media types. *RFC3023, January*.

Jussi Myllymaki. 2001. Effective web data extraction with standard xml technologies. In *Proceedings of the 10th international conference on World Wide Web*, pages 689–696.

Sukomal Pal, Mandar Mitra, and Debasis Ganguly. 2009. Parameter tuning in pivoted normalization for xml retrieval: Isi@ inex09 adhoc focused task. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 112–121. Springer.

Jay M Ponte and W Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281.

Eugen Popovici, Gildas Ménier, and Pierre-François Marteau. 2006. Sirius xml ir system at inex 2006: Approximate matching of structure and textual content. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 185–199. Springer.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.

S Thanga Ramya, P Rangarajan, and V Gomathi. 2016. Xml based approach for object oriented medical video retrieval using neural networks. *Journal of Medical Imaging and Health Informatics*, 6(3):794–801.

Allen Renear, David Dubin, and C Michael Sperberg-McQueen. 2002. Towards a semantics for xml markup. In *Proceedings of the 2002 ACM symposium on Document engineering*, pages 119–126.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

Joseph Rocchio. 1971. Relevance feedback in information retrieval. *The Smart retrieval system-experiments in automatic document processing*, pages 313–323.

Dan Roth. 1999. Learning in natural language. In *IJCAI*, pages 898–904.

Eric SanJuan and Fidelia Ibekwe-SanJuan. 2009. Combining language models with nlp and interactive query expansion. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 122–132. Springer.

Sankar Das Sarma, Dong-Ling Deng, and Lu-Ming Duan. 2019. Machine learning meets quantum physics. *arXiv preprint arXiv:1903.03516*.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*.

Börkur Sigurbjörnsson and Jaap Kamps. 2005. The effect of structured queries and selective indexing on xml retrieval. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 104–118. Springer.

Börkur Sigurbjörnsson, Jaap Kamps, and Maarten de Rijke. 2004. Mixture models, overlap, and structural hints in xml element retrieval. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 196–210. Springer.

Amit Singhal, Gerard Salton, Mandar Mitra, and Chris Buckley. 1996. Document length normalization. *Information Processing & Management*, 32(5):619–633.

Marco Skulschus and Marcus Wiederstein. 2008. *XML schema*. Comelio Medien.

Fei Song and W Bruce Croft. 1999. A general language model for information retrieval. In *Proceedings of the eighth international conference on Information and knowledge management*, pages 316–321.

Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. 2012. *Optimization for machine learning*. Mit Press.

Hiroki Tanioka. 2006. A method of preferential unification of plural retrieved elements for xml retrieval task. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 45–56. Springer.

Andrew Trotman and Börkur Sigurbjörnsson. 2004. Narrowed extended xpath i (nexi). In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 16–40. Springer.

Marc Van Cappellen, Wouter Cordewiner, and Carlo Innocenti. 2008. Data aggregation, heterogeneous data sources and streaming processing: How can xquery help? *IEEE Data Eng. Bull.*, 31(4):57–64.

Eric Van der Vlist. 2002. *XML schema*. O'Reilly Media, Inc.

Jean-Noël Vittaut, Benjamin Piwowarski, and Patrick Gallinari. 2004. An algebra for structured queries in bayesian networks. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 100–112. Springer.

Norman Walsh. 1998. A technical introduction to xml. *World Wide Web Journal*, pages 1–2.

Qiuyue Wang, Qiushi Li, Shan Wang, and Xiaoyong Du. 2009. Exploiting semantic tags in xml retrieval. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 133–144. Springer.

Felix Weigel, Klaus U Schulz, and Holger Meuss. 2004. Ranked retrieval of structured documents with the s-term vector space model. In *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pages 238–252. Springer.

Erik Wilde and David Lowe. 2003. *XPath, XLink, XPointer, and XML: A practical guide to Web hyperlinking and transclusion*. Addison-Wesley Professional.

Lauren Wood, Arnaud Le Hors, Vidur Apparao, Steve Byrne, Mike Champion, Scott Isaacs, Ian Jacobs, Gavin Nicol, Jonathan Robie, Robert Sutor, et al. 1998. Document object model (dom) level 1 specification. *W3C recommendation*, 1.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199.