Integrated DB and IR Approaches

Ralf Schenkel[1] and Martin Theobald[2]
[1]University of Trier, Trier, Germany
[2]Stanford University, Stanford, CA, USA

Synonyms

Using efficient database technology (DB) for effective information retrieval (IR) of semi-structured text

Definition

Integrated DB&IR semi-structured text retrieval combines IR-style scoring and ranking methods for effective search with indexing techniques and processing algorithms from the database world for efficient query evaluation.

Historical Background

Database research has traditionally focused on semi-structured documents that represent structured data with a well-defined schema and only little unstructured, textual content (aka. "data-centric" XML). Typical examples for such documents are invoices, purchase orders, or even complete bibliographies.

Early work in the field concentrated on "classical" data management problems for XML: storing XML data in relational or native XML systems, defining query languages that integrate conditions on the structure and the content of results (like SQL for relational data), efficiently processing these queries on huge collections of documents, and auxiliary structures (like structural summaries and path indexes) to support processing. The focus of this work was on space and runtime efficiency.

Foundations

When semi-structured data formats, especially XML, became popular for storing and exchanging information throughout the 1990s, an abundance of different schemas for such data was developed independently. This created a challenge for existing database query languages like the Structured Query Language (SQL), focusing on exactly matching conditions on the content and structure of results. Now, similar structured and textual information was present in different, heterogeneous formats and schemas, which was often the case when information from different sources was integrated in a single application. In reaction to this, the strict, SQL-style querying paradigm prevalent at that time evolved towards a more relaxed IR-style vague search with partial and imprecise answers. This created three main scientific problems at the intersection of the DB and IR fields: (i) the definition of query

L. Liu, M. T. Özsu (eds.), *Encyclopedia of Database Systems*,
DOI 10.1007/978-1-4899-7993-3_206-3

languages to specify vague constraints on the structure and/or the content of results, (ii) the definition of relevance scores to rank results by their degree of matching with the query, and (iii) efficient algorithms and auxiliary structures to quickly compute the best results to a vague query, according to the relevance score. Solutions to these problems were developed mainly with data-centric documents in mind, and with a strong focus on query languages and algorithms (thus addressing problems i and iii).

One of the first systems to retrieve semi-structured data using a relaxed query language was the *Lore* [1] system developed at Stanford University. Its object-oriented, OQL-style, query language, coined *Lorel*, provided regular path expressions and tag wildcards to express structural vagueness, as well as keyword conditions over the content of subtrees matching the path condition. However, it was still more of a database query language as it did not yet foresee any ranking for the results.

Querying Semi-structured Data with IR Support

A large body of proposals have been made for query languages over semi-structured data that support IR-style vague conditions on structure and content. The simplest of them merely aim to extend keyword search as known from text retrieval to semi-structured data by enhancing the keyword conditions with tag names of elements that should be matched (like in the query "*author:widom*" which restricts occurrences of the keyword "*widom*" to elements with tag name "*author*"). Matches to such a query are subtrees of the document that contain matches to all (for conjunctive evaluation) or at least one (for disjunctive evaluation) keyword. An important aspect here is the selection of subtrees of the right granularity, as large subtrees (such as the complete document) would often not be specific enough. The proposed solutions usually consider some variant of lowest common ancestor (LCA) search to identify suitable root nodes of the result trees, sometimes allowing for additional path conditions from elements containing the keywords towards the root element. Hardly any of the early proposed systems consider ranking of results; instead, they focus on efficient methods to retrieve all possible matches. Note that these techniques were primarily developed for data-centric XML (such as bibliographies) and cannot easily be applied for true full-text search over semi-structured documents.

Among the most prominent approaches for keyword-based ranked retrieval of XML data is XRank [7]. It generalizes traditional link analysis algorithms such as PageRank for authority ranking of linked XML collections and conceptually treats each XML element as an interlinked node in a large element graph. Then the *element rank* of an XML element corresponds to the PageRank value computed over a mixture of containment edges, obtained from the XML tree structure, and hyperlink edges, obtained from the inter-document link structure.

Full-fledged XML query languages with rich IR models for ranked retrieval were proposed by [6, 13]. *XIRQL* [6], a pioneer in the area of ranked XML retrieval, presents a path algebra based on XQL, an early ancestor of W3C's XQuery, for processing and optimizing structured queries. It combines Boolean query operators with probabilistically derived weights for ranked result output, thus transferring the probabilistic IR paradigm to the XML case. It defines data-type-specific vague predicates for similarity search over differently typed XML elements such as person names or numbers, and it introduces a notion of *index objects* that serve as anchors from which the probabilistic weights are derived (in the classic IR notion of a document). Using index objects follows the idea that only nodes of specific type and granularity in the document hierarchy should be presented as results to the end-user. Defining these index objects, however, may be strongly schema-dependent and assumes substantial knowledge about the general document structure and user intent, which typically requires their manual pre-selection from a – preferably compact – document type definition (DTD).

The *XXL* search engine [13] specifies a full-fledged, SQL-oriented query language for ranked XML IR with a high semantic expressiveness that made it stand way apart from the predominant

XQL and XPath language standards at its time. For ranked result output, XXL leverages both a standard IR vector space model and an ontology-oriented similarity search for the dynamic relaxation of structure and term conditions. The principal structure of the query, however, is evaluated in a strictly Boolean manner.

More recently, IR-style keyword conditions and full-text search were added to existing XML query languages. The NEXI (for Narrowed Extended XPath I) query language used in the INEX (INitiative for the Evaluation of XML Retrieval, see http://inex.mmci.uni-saarland.de) benchmark series aims at a simplified and easy-to-comprehend subset of XPath, the W3C standard language for path matches within a document, with extended IR functionality. Here the *about* operator already anticipates the role of the *ftcontains* operator in the later W3C Full-Text extensions of XPath 2.0 and XQuery 1.0. TeXQuery [2], on the other hand, has been the foundation for the W3C's official Full-Text extension to XPath and XQuery, which extends these languages with the option to express actual full-text queries over XML documents. It provides many retrieval options known from text retrieval, such as phrases, weighting terms, and expanding terms using ontologies, but leaves details of the scoring model used to rank results up to the implementation.

Pioneering work in the area of vague structural matches was done by [3, 11] (and refined later by [4]), who proposed relaxing queries with structural constraints to find matches in structurally similar, but not exactly matching documents. The *FlexPath* [4] algorithm integrates structure and keyword queries and regards the query structure as templates for the context of a full-text keyword search. The query structure (as well as the content conditions) can be dynamically relaxed for ranked result output according to predefined tree editing operations when matched against the structure of the XML input documents.

Query Processing for Semi-structured Text Retrieval

Efficient evaluation and ranking of conditions on content and structure of semi-structured data has been a very fruitful and popular research area in recent years. The majority of the proposed algorithms for efficient query evaluation combine some form of precomputed auxiliary indexes (like inverted files) with top-k aware processing algorithms, most notably Fagin's family of threshold algorithms (TA), which provide threshold-based candidate pruning and early termination.

XRank uses inverted lists containing – for each term – the elements that contain the term, sorted in descending order of element rank, along with a threshold algorithm for pruning the search space. The *FlexPath* query processor uses separate index structures for storing and retrieving the structure- and content-related conditions of a path query. The *Whirlpool* system introduced by Marian et al. [10] provides a flexible architecture for processing top-k queries on XML documents *adaptively*. Whirlpool allows partial matches to the same query to follow different execution plans, and takes advantage of the top-k query model to make dynamic choices during query processing. The key features of Whirlpool are: (i) a partial match that is highly likely to end up in the top-k set is processed in a prioritized manner, and (ii) a partial match unlikely to be in the top-k set follows the cheapest plan that enables its early pruning. Whirlpool provides several adaptivity policies and also supports parallel evaluations.

TopX [12], the actual successor of XXL, focuses on a small, XPath-like subset of the XXL query language which allows for a radically different query processing architecture that outperforms XXL in terms of efficiency by a large margin. As a native top-k engine for XML, TopX also uses sorted index lists, but keeps a candidate queue in-memory and therefore is able to focus on sequential disk access and on minimizing random disk access through sophisticated index structures and judiciously scheduled index access decisions.

A large effort has been made on mapping XML to relational schemas with highly specialized index structures for efficient support of approximate query processing, including support for IR-style retrieval functionality. *PF/Tijah* [8],

which is now a part of *MonetDB/XQuery*, is an example for such an XQuery engine.

Support for XML-IR in Commercial Database Systems

Meanwhile, all commercially available databases with XML support, relational or native, provide some support for IR-style content search in combination to structural queries. Some systems support the full-text extensions of XPath and XQuery, and all systems come with their own extensions of their query language that are incompatible with – and sometimes less powerful than – the W3C proposals. Frequently, existing text search components are extended for XML support and provide the standard text search features (like phrase search, proximity conditions, stemming, etc.) for searching XML elements, usually with some scoring function to rank results.

Key Applications

The techniques presented before can be applied for efficiently retrieving information from large, possibly heterogeneous collections of semi-structured data. This includes more data-centric collections like bibliographies, collections of textual documents (like abstracts or full-text of publications or books), heterogeneous data exported from different sources, and eventually documents on the Web.

Cross-References

- Top-k XML Query Processing
- XML Data Management: XML Prototypes/Systems
- XML Indexing
- XQuery Full-Text

Recommended Reading

1. Abiteboul S, Quass D, McHugh J, Widom J, Wiener JL. The lorel query language for semistructured data. Int J Digit Libr. 1997;1(1):68–88.
2. Amer-Yahia S, Botev C, Shanmugasundaram J. TeXQuery: a full-text search extension to XQuery. In: Proceedings of 12th international world wide web conference. 2004. p. 583–94.
3. Amer-Yahia S, Cho S, Srivastava D. Tree pattern relaxation. In: Advances in database technology, proceedings of 8th international conference on extending database technology. 2002. p. 496–513.
4. Amer-Yahia S, Lakshmanan LVS, Pandit S. FleXPath: flexible structure and full-text querying for XML. In: Proceedings of ACM SIGMOD international conference on management of data. 2004. p. 83–94.
5. Cohen S, Mamou J, Kanza Y, Sagiv Y. XSEarch: a semantic search engine for XML. In: Proceedings of 29th international conference on very large data bases. 2003. p. 45–56.
6. Fuhr N, Großjohann K. XIRQL: a query language for information retrieval in XML documents. In: Proceedings of 24th annual international ACM SIGIR conference on research and development in information retrieval. 2001. p. 172–80.
7. Guo L, Shao F, Botev C, Shanmugasundaram J. XRANK: ranked keyword search over XML documents. In: Proceedings of ACM SIGMOD international conference on management of data. 2003. p. 16–27.
8. Hiemstra D, Rode H, Van Os R, Flokstra J PF/Tijah: text search in an XML database system. In: Proceedings of 2nd international workshop on open source information retrieval. 2006.
9. Hristidis V, Papakonstantinou Y, and Balmin A. Keyword proximity search on XML graphs. In: Proceedings of 19th internatonal conference on data engineering. 2003. p. 367–78.
10. Marian A, Amer-Yahia S, Koudas N, Srivastava D. Adaptive processing of Top-k queries in XML. In: Proceedings of 21st international conference on data engineering. 2005. p. 162–73.
11. Schlieder T, Meuss H. Querying and ranking XML documents. J Am Soc Inf Sci Tech. 2002;53(6):489–503.
12. Theobald M, Bast H, Majumdar D, Schenkel R, Weikum G. TopX: efficient and versatile top-k query processing for semistructured data. VLDB J. 2008;17(1):81–115.
13. Theobald A, Weikum G. Adding relevance to XML. In: Proceedings of 3rd international workshop on the world wide web and databases. 2000. p. 105–124.
14. Xu Y, Papakonstantinou Y. Efficient keyword search for smallest LCAs in XML databases. In: Proceedings of ACM SIGMOD international confernce on management of data. 2005. p. 537–8.