Improving the Performance of Search Engine With Respect To Content Mining

Kr.Jansi, L.Radha
[1]Asst. Professor, Srm University, Chennai
[2]Mtech, Srm University, Chennai

Abstract
R- Google is a dedicated stream search engine, can help users retrieve dynamic real-time streams, focusing on quality and topic relevance rather than simple query matching. The popularity of Text mining[1] has created overwhelming quantities of Web data that has led to an evolution in the way people produce and consume information. Content is increasingly being delivered in streams—a series of text documents that arrive over time as content based document. The stream search engine that attempts to provide topic-focused searches for various live streams from the web. The proposed search engine architecture accounts for the structural and temporal characteristics unique to content based search, focusing on the problems of content relevance judgment and ranking.

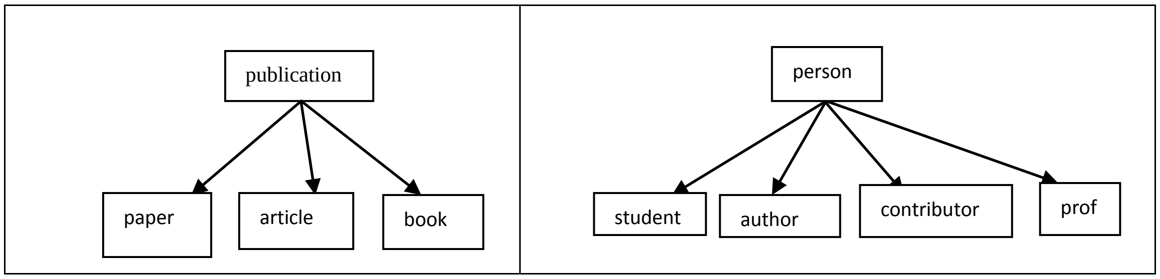
1 Introduction
There is an ever-growing availability of semi-structured information on the Web and in digital libraries. Increasingly, users, both expert and non-expert, have access to text documents equipped with some semantic hints through XML markup. The documents are queried using a database approach, which performs exact-match. But here recall [4]is often too low.

There are two alternatives: to issue either a Keyword-based query or a Loosely Structured query. The popularity of Keyword-based querying stems from the fact that it is user-friendly and does not require the user to learn a query language or to be aware of the structure of the underlying data. Loosely structured querying allows combining some structural constraints within a Keyword query by specifying the context where a search term should appear (combining keywords and element names). That is, it requires the user to know only the labels of elements containing the keywords, but does not require him or her to be aware of the structure of the underlying data. Thus, Loosely Structured querying combines the convenience of Keyword-based querying while enriching queries by adding structural conditions, which leads to performance enhancement.

Business executives and employees need flexible access to vast quantities of information. Since they are likely to be aware of some labels of elements/attributes containing data and are unlikely to be fully aware of the structure of the underlying data, they can access the data using Loosely Structured queries. On the other hand, business customers are most likely not aware of the elements' labels or the structure of the data. Thus, Keyword-based querying meets their needs. Some Internet-based businesses let their customers issue Loosely Structured queries by providing them with graphical user interfaces containing menus (e.g., drop-down menus and check boxes) and search fields (for submitting keywords). Each entry in a menu represents an element, and its name depicts the element's label.

2 Concepts Used In Xcdsearch
We model XML documents as rooted and labeled trees.Atree t is a tuple where n is the set of nodes. A node in a tree represents an element in an XML document. Weuse the term "data node" to denote a node of a tree data structure that has no child node and always has a value. Nodes are numbered for easy reference. The structure of an XML document can be partitioned into multiple units (e.g., subtrees), where each unit is associated with some document contents Thus, the framework ofXCDSearch partitions XML trees to subtrees, where each consists of a parent and its children data nodes and is treated as one unit. The idea of viewing each such set as one logical entity is useful as it enables filtering many unrelated groups of nodes when answering a query. Compared to filtering individual nodes, this methodology leads to more accurate results and less processing time. Each subtree is treated as a unified entity called a Canonical Tree (CT), which is a metaphor of real-world entities. Two real-world entities may have different names but belong to the same type, or they may have the same name but refer to two different types. To overcome that labeling ambiguity, we observe that if[2] we cluster Canonical Trees based on the ontological concept of the parent nodes' component of the Canonical Trees, we will identify a number of clusters.

Fig. 1.Example of ontology hierarchy



2.1 Ontology Label

The term Ontology Label (OL) used to refer to the ontological concept of a parent node. We now formalize the Ontology Label and Canonical Tree concepts. The framework of XCDSearch applies the above clustering concept to all parent nodes in an XML tree, and the label of each of these clusters[2] is an OL. The Ontology Label of a Canonical Tree is the Ontology Label of the parent node component of the Canonical Tree.

TABLE 1
OLs and OLAs of the Parent Nodes in Fig. 1

Parent nodes (with their IDs)	OL	OLA
paper(4,2), article(17,24), book(23)	publication	b
Student(1), contributor(9), reviewing Prof(20)	person	p
researchInterest(18,31), experience(29)	field	f
conference(6), journal(26)	Publication-proceedings	s

2.2 Canonical Tree

A Canonical Tree T is a pair, T(OL(n'),N) where (OL(n')) is the Ontology Label of an interior node n0 and N is a finite set of data nodes and/or attributes. Let (n',n) denote that there is an edge from node n' to node n in the XML tree.

2.3 Canonical Tree Graph

A CTG is a hierarchical representation depicting the relationships between CTs. A CTG is a pair of sets, CTG=(Ts,E) where TS is a finite set of CTs and E, the set of edges is a binary relation on TS.

2.4 Semantically Related CTs

CTs T and T0 are semantically related if the paths from T and T0 to their LCA, not including T and T', do not contain more than one CT with the same Ontology Label. The LCA of T and T0 is the only CT that contains the same OL in the two paths to T and T'.

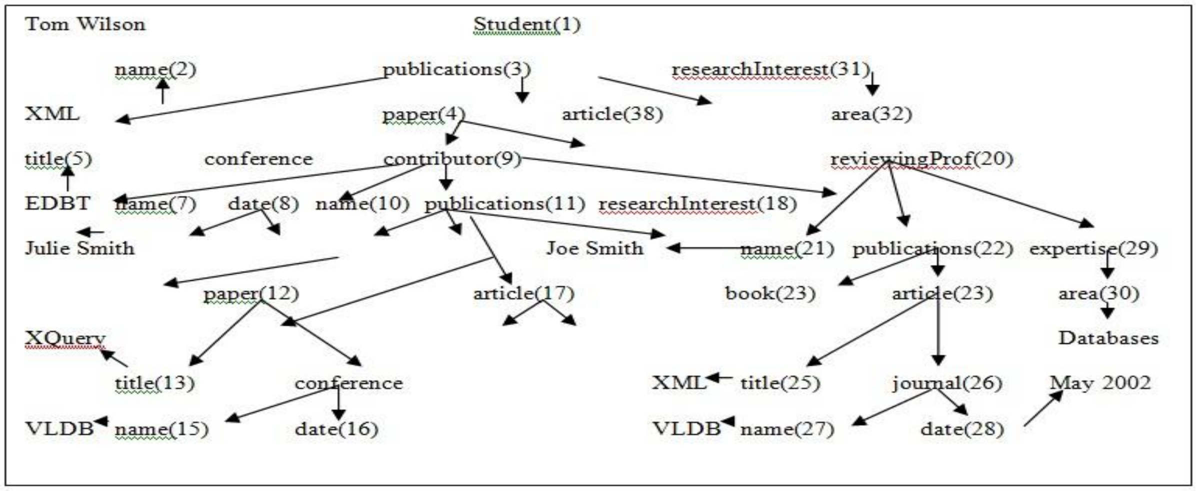


Fig3. Canonical Trees Graph

3, context-Driven Techniques

3.1 (Keyword Context (KC)). KC is a CT containing a keyword of a query. That is, one of the data nodes of the KC holds a value matching one of the query's keywords

3.2 (Intended Answer Node (IAN)). IAN is a data node in the XML tree containing the data that the user is looking for.

3.3(Immediate Relatives of a KC (IRKC)). We can determine IRKC by pruning from the CTG all CTs and the remaining ones would be IRKC.

4, System Implementation And Architecture

Fig. 4 shows the XCDSearch system architecture. We describe below the processing steps and modules in the system architecture that create Ontology Labels, CTGs,
and IRT.

4.1 Creating Ontology Labels

Many ontologies[5] are already available in electronic form and can be imported into an ontology-development environment.The imported ontologies[5] are stored in database Ontology_DB for future references. After an XML schema describing the structure of an XML document is input to module OntologyBuilder, the module will access database Ontology_DB to determine the Ontology Labels corresponding to the interior nodes of the XML schema. If the ontology for creating the Ontology Label of a node is not found in the database, the module will first tokenize the node's label by parsing it into a set of tokens using delimiters.The
resultant OLs will be added to database Ontology_DB for future reference.

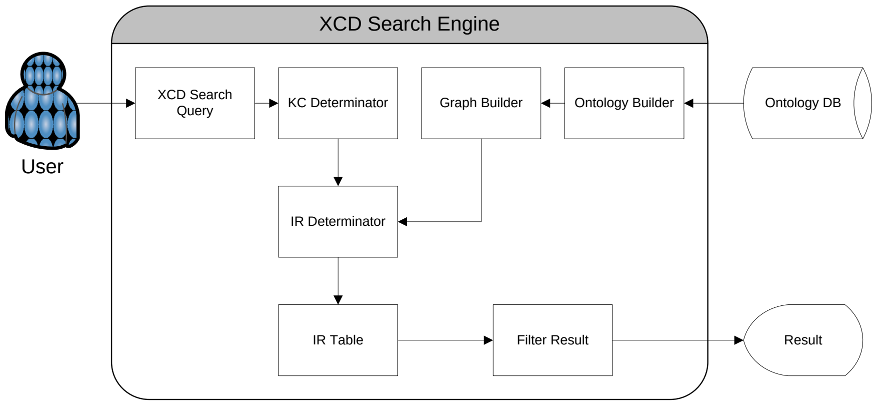


Fig. 4. XCDSearch system architecture

4.2 Constructing CTGs

Module OntologyBuilder outputs to module GraphBuilder a table called OL_TBL, which stores the OLs of the interior nodes in the input XML schema. Module GraphBuilder,
creates a CTG corresponding to the input schema using algorithm BuildCTreesGraph. The inputs to the algorithm are table OL_TBL and the list of nodes adjacent to each node.

4.3 Constructing IRT

Using the input XML document, CTG, and the keywords, the KCdeterminer identifies the KCs. Module IRdeterminer uses algorithm ComputeIR (recall Fig. 10) to compute for
each CT T in the CTG its IRT and saves it in a table called IR TBL for future reference.

4.4 Ontology Builder

Ontology structure[5] formed from Ontology database. Ontology structure contain main category inside this sub category.For example we consider student, author is a subcategory of person category. We cluster set that contains entities sharing the same domain.The clustering concept to all parent nodes in an XML tree, and the label of each of these clusters is an Ontology Structure.

4.5 Graph Builder

Each sub tree is treated as a unified entity called a **Canonical Tree (CT).** Canonical Tree Graph is a hierarchical representation depicting the relationships between Canonical Trees. Canonical Tree Graph builds using set of Canonical Tree and set of related edges. CTG = (Ts,E).

4.6 Keyword context determining

Keyword context determines Keyword Context is a Canonical Tree containing a keyword of a query. That is, one of the data nodes of the Keyword Context holds a value matching one of the query's keywords. Intended Answer Node is a data node in the XML tree containing the data that the user is looking for.

4.7 Immediate relative determining

The process of answering a query goes through three phases. In the first phase, the system locates the Keyword Contexts. In the second phase, the system selects from these Keyword Contexts subsets, where each subset contains the smallest number of Keyword Contexts that are closely related to each other, Contain at least one occurrence of each keyword. The Keyword Contexts in each subset are called **Related Keyword**.

4.8 Context-Driven Techniques

Keyword Context is a Canonical Tree containing a keyword of a query. That is, one of the data nodes of the Keyword Context holds a value matching one of the query's keywords. Intended Answer Node is a data node in the XML tree containing the data that the user is looking for. We call each Canonical Tree that can contain an Intended Answer Node for a Keyword Context an Immediate Relative of the Keyword Context. We denote the set of Canonical Trees that are Immediate Relatives of a Keyword Context. A Canonical Tree can contain an Intended Answer Node, if it has strong association with the Keyword Context.

5 Context-Driven Search Algorithm

The Sort-Merge Join (also known as Merge-Join) is an example of a join algorithm and is used in the implementation of a relational database management system.The key idea of the Sort-merge algorithm is to first sort the relations by the join attribute, so that interleaved linear scans will encounter these sets at the same time.

6, Ranking Function

Main Process of this algorithm finding the optimal ranked results of search engine.

6.1 Linear programming

Linear programming (LP) deals with the problem of minimizing or maximizing a linear function in the presence of linear equality and/or inequality constraints or a set of restrictions.When a query is passed to multiple search engines, each search engine returns a ranked list of documents.Researchers have demonstrated that combining results, in the form of a ‘‘metasearch engine’’, produces a significant improvement in coverage and search effectiveness. This paper proposes a linear programming mathematical model for optimizing the ranked list result of a given group of Web search engines for an issued query. An application with a numerical illustration shows the advantages of the proposed method.

7, Conclusion

XCDsearch, an xml context-driven search engine, which answers keyword based and loosely structured queries.xcdsearch accounts for nodes contexts by considering each set consisting of a parent and its children data nodes in the xml tree as one entity (CT).we proposed mechanisms for determining the semantic relationships among different cts. we also proposed an efficient stack-based sort-merge algorithm that selects from the set of CTs containing keywords (KCs) subsets, wherein each subset contains the smallest number of kcs that are closely related to one another and contain at least one occurrence of each keyword.we took as samples of non-context-driven xml search engines and compared them heuristically and experimentally with xcdsearch. The experimental results show that xcdsearch outperforms significantly the three other systems.

References

[1] M Eirinaki, M Vazirgiannis, (February 2003), Web Mining for Web Personalization, in ACM Transactions on Internet Technology (TOIT).

[2] N. Oikonomakou, M. Vazirgiannis, April (2003), A Review of Web Document Clustering approaches, in Proceedings of the NEMIS Launch Conference, International Workshop on Text Mining & its Applications, Patras, Greece.

[3] Omid Panah, AmirPanah and AminPanah, (2010), “Evaluating the data mining techniques and their roles in increasing the search speed data in web,” Islamic Azad University- Amol, Iran.

[4] Olfa Nasraoui and Leyla Zhuhadar, (2010),“Improving Recall and Personalized Sematic Search Engine for E-learning,” Proc. University of Louisville,KY 40292,USA.

[5] K.Saruladha, Dr.G.Aghila, Sajina Raj, (2010),” A survey of semantic similarity methods for ontology based information retrieval” Pondciherry University,Pondicherry,India.