Filtering and Ranking Schemes for Finding Inclusion Dependencies on the Web

Erika Yumiya
University of Tsukuba
yumiya@slis.tsukuba.ac.jp

Atsuyuki Morishima
University of Tsukuba
mori@slis.tsukuba.ac.jp

Masami Takahashi*
University of Tsukuba
mtaka@slis.tsukuba.ac.jp

Shigeo Sugimoto
University of Tsukuba
sugimoto@slis.tsukuba.ac.jp

Hiroyuki Kitagawa
University of Tsukuba
kitagawa@cs.tsukuba.ac.jp

ABSTRACT

This paper addresses the problem of finding *inclusion dependencies* on the Web. In our approach, we enumerate pairs of HTML/XML elements that possibly represent inclusion dependencies and then rank the results for verification. This paper focuses on the challenges in the finding and ranking processes.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

inclusion dependencies, data quality

1. INTRODUCTION

Data integrity constraints are fundamental in various applications, and *inclusion dependencies* are widely used as important data integrity constraints. This paper addresses the problem of ranking HTML/XML pairs to support the discovery of inclusion dependencies among HTML/XML elements on the Web. Finding inclusion dependencies among HTML/XML elements is important both from practical and theoretical viewpoints. Such dependencies occur in Web sites with lists of publications or members, and those with sets of sentences taken from an original document. Importantly, an inclusion dependency is a generalization of the equivalence constraint, which is a universal constraint that pervasively appears in many applications involving Web sites having the same data but maintained by different administrators.

Among important applications of data integrity constraints is the management of data quality and integrity. Although data quality and integrity have been known as crucial issues on the Web, it is well known that there are many inconsistencies in the data available on the Web. This is because there are many Web sites that publish the same or related content but are maintained by different administrators. For example, [7] reports that in a set of real estate Web sites, they found about 60% of data items with some inconsistency. Applying data integrity constraints to fix such inconsistencies have been widely discussed in various data integration and management problems [4].

*Current Affiliation: NTT corporation

Copyright is held by the author/owner(s).
WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

A widely known problem related to data integrity constraints is that they are not always given in an explicit manner [5]. Therefore, many studies have addressed the problem of helping users find integrity constraints from an existing data instance. A key technique to find inclusion dependencies is to enumerate inclusions, i.e., asymmetric set containments. There are already numerous studies about computing inclusions in the context of relational databases. Bauckmann and others [2] proposed an algorithm that takes as input a set of relations and efficiently enumerates all pairs of relational attributes one of which includes the other. The algorithm is designed to minimize the amount of I/O over the sets of attribute values. Sergey Melnik and others [6] proposed two hash-based partitioning algorithms called the Adaptive Pick-and-Sweep Join (APSJ) and the Adaptive Divide-and-Conquer Join (ADCJ), to efficiently compute set containment joins. Each of them produces a set of inclusions (a set of pairs that have inclusion relationship).

In the context of the discovery of inclusion dependencies, the set of enumerated inclusions is a set of *candidates* of inclusion dependencies. Therefore, a common approach is first enumerating all possible candidates (including false positives) and then verifying the enumerated candidates.

In this paper, we address the challenges in two problems of the approach. We first discusses the challenges in a filtering scheme to filter out irrelevant pairs in the enumeration process and then those in the ranking process to support a user in checking whether each enumerated pair actually suggests inclusion dependencies.

2. INCLUSION DEPENDENCIES ON THE WEB

We model the target Web data as a triple $(P, elem, words)$. Here, P $(= \{p_1, p_2, \ldots\})$ denotes a set of Web pages, and $elem$ and $words$ are functions to represent components of each Web page; $elem(p_k)$ $(= \{e_1, e_2, \ldots\})$ defines the set of *page elements* contained in Web page p_k, and $words(e_j)$ $(= \{|w_1, w_2, \ldots|\})$ defines the multiset $words(e_j)$ of words contained in the element e_j. We need one constraint to represent the hierarchical structure of page elements; If e_i is a sub-element of e_j, $words(e_i)$ has to be a subset of $words(e_j)$. For example, assume that $elem(p_k)$ represents a set of HTML elements in Web page p_k and let $words(e_j)$ be a multiset of words in each element $e_j \in elem(p_k)$. Then, the mapping satisfies the constraint. Each p_k does not necessarily have to be an actual Web page; it can be an XML document created from an HTML page by a wrapping process.

Now, we define an inclusion dependency among page elements as follows: Let e_i and e_j be page elements. Then, $e_i \subseteq_{ind} e_j$ is an inclusion dependency between e_i and e_j meaning that

$words(e_i) \subseteq words(e_j)$ should always be satisfied on the Web. In the rest of the paper, we often use e_i to denote $words(e_i)$ in set operations, when the meaning is clear from the context.

3. ENUMERATING INCLUSIONS WITH FILTERS

Let a set E of all page elements be $\bigcup_{p_k \in P} elem(p_k)$, and $pairs = \{(e_i, e_j) | e_i, e_j \in E\}$. Then, the first step is to compute and output a set of inclusions, denoted by $inclusions(pairs)$, where $inclusions(pairs) = \{(e_i, e_j) | (e_i, e_j) \in pairs \wedge e_i \subseteq e_j\}$

A *filter* for the first step is defined as follows. Let $filter(e_i, e_j)$ be a predicate that returns false only when $e_i \subseteq e_j$ is guaranteed not to hold. Assume that we compute the following set of pairs:

$$pairs' = \{(e_i, e_j) | (e_i, e_j) \in pairs \wedge filter(e_i, e_j)\}. \quad (1)$$

Then, $filter(e_i, e_j)$ should be designed to lead to the following results: (1) $|pairs'| \leq |pairs|$ and (2) $inclusions(pairs) = inclusions(pairs')$. The challenges here are twofold: First, web content is *error-prone* and has *variations in expression*. Therefore, if we search for *exact* inclusions, we would miss many inclusion dependencies. One possible approach is to adopt Jaccard containment [1] to deal with the situation. Second, efficient filters to support such non-exact inclusions are not trivial. We expect that a bit-based signature scheme can be used for efficient filtering. The detailed discussions on the possible approaches are given in [8].

4. RANKING INCLUSIONS WITH THE COVER RELATIONSHIP

Because all inclusions enumerated for a data instance do not necessarily result in actual inclusion dependencies, the next step is to verify whether each inclusion implies an inclusion dependency. The challenge here is how to rank the candidate pairs because typically, the verification is performed manually.

We propose to rank the candidate pairs based on the *cover* relationship among the pairs. Let α and β be inclusions s.t. $\alpha, \beta \in inclusions(pairs)$. We say that α *covers* β, if we can verify β in parallel with the verification of α. We write $\alpha \geq \beta$ to denote that α covers β.

The notion of covers can be considered as a generalization of the removal of overlapping answers in XML search [3]. In XML search, it is often the case that the element hierarchy of XML data allows us to see an answer e_1 in parallel with seeing e_2, because some of the elements (e.g. e_1) satisfying a query condition are often included in the others (e_2). We extend the idea to deal with the relationships among element pairs in the context of the verification of inclusion dependencies. We define two types of cover relationships, namely, *deductive covers* that represent logical overlaps and *regional covers* that represent physical overlaps.

This section explains the deductive cover relationship ($\geq_d$). The regional cover relationship and efficient evaluation schemes are discussed in [8].

DEFINITION 1. *When we have two inclusions $\alpha = (e_1, e_4)$ and $\beta = (e_2, e_3)$, we say α deductively covers β (denoted by $\alpha \geq_d \beta$) if and only if the following conditions hold.*

- *e_2 is a descendent element of e_1 in the element hierarchy of a page (i.e., $words(e_2) \subseteq words(e_1)$), and*
- *e_4 is a descendent element of e_3 in the element hierarchy of a page (i.e., $words(e_4) \subseteq words(e_3)$).* □

This is illustrated by Figure 1, in which two element hierarchies are shown: (1) e_1 and e_2 and (2) e_3 and e_4. Then, $e_1 \subseteq e_4$ deductively covers $e_2 \subseteq e_3$ because the latter is deduced from the former

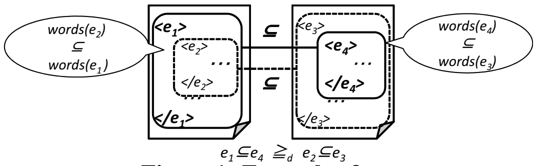

Figure 1: Example of $\geq_d$

and the inclusions $e_2 \subseteq e_1$ and $e_4 \subseteq e_3$, which are derived from the hierarchical structure.

When α deductively covers β, we can easily check whether β suggests an inclusion dependency in parallel with checking whether α does. This is because (1) the elements in β overlap those in α, and (2) the existence of inclusion $\alpha = (e_1, e_4)$ suggests the place of inclusion $\beta = (e_2, e_3)$. Namely, (a) e_2 exists inside e_1, and (b) e_3 can be every ancestor of e_4. The existence of β, however, does not imply that of α. Therefore, the user who was first told that β exists, could not know where and even whether related inclusions exist.

5. SUMMARY

This paper addressed the challenges in finding pairs of HTML/XML elements that suggest inclusion dependencies among them, focusing on the filtering and ranking mechanisms for candidate pairs of HTML/XML elements. In particular, we introduced the notion of covers to rank the candidate pairs in order to efficiently look through the enumerated inclusions.

6. ACKNOWLEDGEMENTS

The authors are grateful to Prof. Sakaguchi and Prof. Nagamori for the discussion in seminars. This research was partially supported by PRESTO from the Japan Science and Technology Agency, and by the Grant-in-Aid for Scientific Research (#21240005) from the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan.

7. REFERENCES

[1] Parag Agrawal, Arvind Arasu, Raghav Kaushik. On Indexing Error-Tolerant Set Containment. SIGMOD 2010, 927-938.
[2] Jana Bauckmann, Ulf Leser, Felix Naumann. Efficiently Computing Inclusion Dependencies for Schema Discovery. ICDE Workshops 2006, 2.
[3] Charles L. A. Clarke. Controlling overlap in content-oriented XML retrieval. SIGIR 2005, 314-321.
[4] Wenfei Fan. Dependencies revisited for improving data quality. PODS 2008, 159-170.
[5] Fabien De Marchi, Stephane Lopes, Jean-Marc Petit. Unary and n-ary inclusion dependency discovery in relational databases. J. Intell. Inf. Syst. Vol.32, No.1, 53-73, 2009.
[6] Sergey Melnik, Hector Garcia Molina. Adaptive Algorithms for Set Containment Joins. ACM Trans. Database Systems, 2003, Vol.28, No.1, 56-99.
[7] Ningning Wu, Irit Askira Gelman, Isaac O. Osesina. How consistent is web information - A case study on online real estate databases, AMCIS 2009, 437.
[8] Erika Yumiya, Atsuyuki Morishima, Masami Takahashi, Shigeo Sugimoto, Hiroyuki Kitagawa. A Comprehensive Study on Filtering and Ranking Schemes for Finding Inclusion Dependencies on the Web. available at `http://www.kc.tsukuba.ac.jp/~7Emori/papers/ymt+12.pdf`.