Middle-East Journal of Scientific Research 19 (6): 817-820, 2014
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2014.19.6.1475

Structured Information Extraction System from Web Pages

B. Anantha Barathi

Department of CSE,
Bharath University, India

Abstract: The World Wide Web is a vast and rapidly growing source of information. Most of this information is in the form of unstructured text, making information hard to query. Many websites that have large collections of pages containing structured data. For example, Amazon lay out the author, title, comments, etc. in the same way in all its book pages. The values used to generate the pages (e.g. the author, title,..) typically come from a database. In this paper, we study the problem of automatically extracting the database values from template generated web pages without learning examples or other similar human input. We formally define a template and propose a model that describes how values are encoded into pages using a template. We present an algorithm that takes, as input, a set of template-generated pages, deduces the unknown template used to generate the pages and extracts, as output, the values encoded in the pages. Experimental evaluation on a large number of real input page collections indicates that our algorithm correctly extracts data in most cases.

Key words: Information Extraction · Wrapper · EXALG

INTRODUCTION

Many web sites contain large sets of pages generated using a common template or layout. For example, Amazon [1] lays out the author, title, comments, etc. in the same way in all its book pages. The values used to generate the pages (e.g. the author, title,...) typically come from a database. In this paper, we study the problem of automatically extracting the database values from such template generated web pages without any learning examples or other similar human input.

This paper studies the problem of *automatically* extracting structured data [2] encoded in a given collection of pages, without any human input like manually generated rules or training sets. For instance, from a collection of pages like those in Figure 1 we would like to extract book tuples, where each tuple consists of the title, the set of authors, the (optional) list-price and other attributes (Figure 2). Extracting structured data from the web pages is clearly very useful, since it enables us to pose complex queries over the data. Extracting structured data has also been recognized as an important sub-problem in information integration systems [3-5], which integrate the data present in different web-sites

Fig. 1: Two book pages from Amazon

Corresponding Author: B. Anantha Barathi, Department of Cse Bharath University, India.

Page	A	B	C	…
1	MySystem . . .	Aron . . .	(NULL)	…
2	Godel, . . .	Douglas . . .	20.00	…
.	…..	….	…..	…
.	…..	….	…..	…

Fig. 2: Extracted Data

Therefore, there has been a lot of recent research in the database and AI communities on the problem of extracting data from web pages (sometimes called information extraction (IE) problem).

An important characteristic of pages belonging to the same site and encoding data of the same schema, is that the data encoding is done in a consistent manner across all the pages. For example, in both the pages of Figure 1, the title of the book appears in the beginning followed by the word "by," followed by the a uthor (s). In other words, the above pages are generated using a common "template" by "plugging-in" values for the title, list of authors and so on. Most of the information extraction techniques proposed so far and the technique that we propose in this paper, exploit the template based encoding for extracting data from the pages. Specifically, the techniques use either a partial or complete knowledge of the template used to generate the pages, to extract the data. For example, in Figure 1, the price of a book can be extracted by retrieving the text immediately following the template-text "OurPrice."

The primary difference between various information extraction techniques lies in how the knowledge of the template is acquired by the extraction system. The earliest information extraction techniques rely on a human to encode knowledge of the template into a program called *wrapper*, which then extracts data. In Hammer *et al*. [8] a human expresses some part of the template as declarative rules and a "wrapper generator" converts these rules into a wrapper. More recent systems like XWRAP [6], WIEN [7], STALKER [8] and SOFTMEALY [9] use human generated training examples that identify data in a small number of pages, to learn knowledge of the template.

We know of only two previous work on automatic extraction, namely, Kumaravel etal14] ROADRUNNER [10] and IEPAD [11]. There are fundamental differences, both in problem formulation and solution approach, between our work and the above two. Both ROADRUNNER and IEPAD make the simplifying assumption that an HTML tag is always part of the template of the page. Although, statistically, HTML tags do tend to occur in template, there are a significant number of cases where they occur within data.

We make two clarifications regarding our assumptions and goals. First, our goal is not to try to semantically name the extracted data. We assume that renaming, for example, attribute *A* in Figure 2 as "TITLE", is done as a post processing step, possibly with human help. Second, we assume that our input pages conform to a common schema and template. We do not consider the problem of automatically obtaining such pages from web sites. It is reasonably easy for a human to identify web collections of interest that have a common schema and then run a crawler to gather the pages.

EXALG: In this paper, Kumaravel etal [12] we present an algorithm, EXALG to solve the EXTRACT problem. Figure 3 shows the different sub-modules of EXALG. Broadly, EXALG works in two stages.

In the first stage (ECGM), it discovers sets of tokens associated with the same type constructor in the (unknown) template used to create the input pages.

In the second stage (Analysis), it uses the above sets to deduce the template. The deduced template is then used to extract the values encoded in the pages.

This section outlines the execution of EXALG for our running example. In the first stage, EXALG (within Sub-module FINDEQ) computes "equivalence classes" — sets of tokens having the same frequency of occurrence in every page in *Pe*. An example of an equivalence class (call *Ee*1) is the set of 8 tokens *{*<html>, <body>, Book,..., </html>*}*, where each token occurs exactly once in every input page. There are 8 other equivalence classes. EXALG retains only the equivalence classes that are large and whose tokens occur in a large number of input pages. We call such equivalence classes LFEQs (for Large and Frequently occurring EQuivalence classes). For the running example there are two LFEQs. The first is *Ee*1

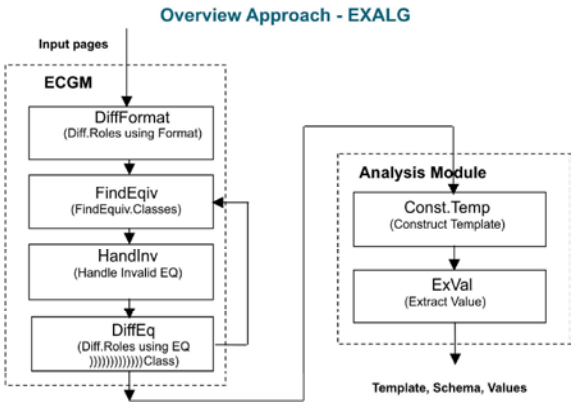

Fig. 3: Modules of EXALG

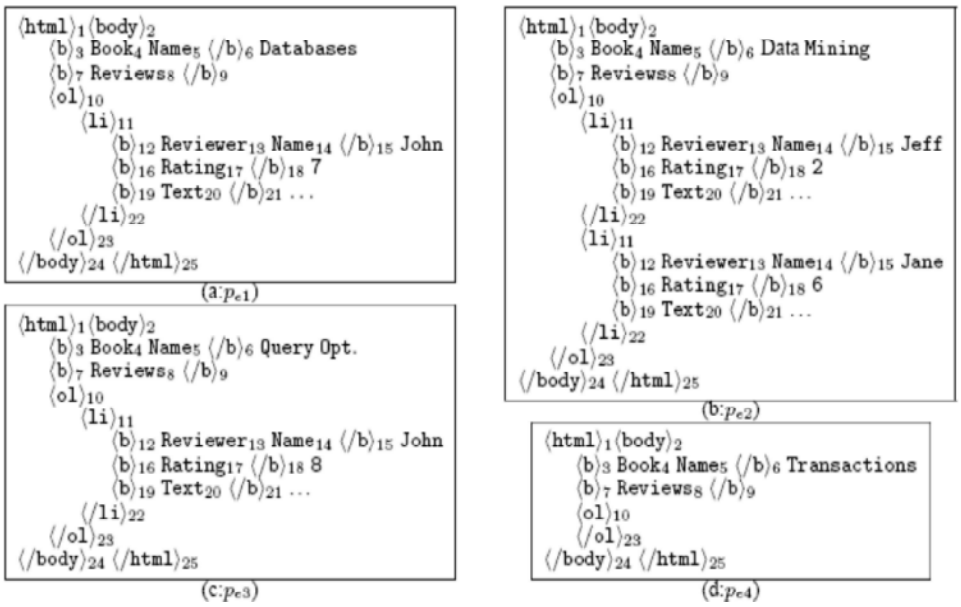

Fig. 4: Template

shown above. The second, which we call *Ee*3, consists of the 5 tokens: *{*
, Reviewer, Rating, Text, *}*. Each token of *Ee*3 occurs once in *pe*1, twice in *pe*2 and so on. *The basic intuition behind* LFEQ*s is that it is very unlikely for* LFEQ*s to be formed by "chance". Almost always,* LFEQ*s are formed by tokens associated with the same type constructor in the (unknown) template used to create the input pages*. This intuition is easily verified for the running example where all tokens of *Ee*1 (resp. *Ee*3) are associated with *ôe*1 (resp. *ôe*3) of *Se* in *Te*2. For this simple example, Sub-module HANDINV does not play any role, but for real pages HANDINV detects and removes "invalid" LFEQs — those that are not formed by tokens associated with a type constructor.

EXALG enters the second stage when it cannot grow LFEQs, or find new ones. In this stage, it builds an output template *TS'*using the LFEQs constructed in the previous stage. In order to construct *S'*, EXALG first considers the root LFEQ — the LFEQ whose tokens occur exactly once in every input page.

Assumptions: Kumaravel *et al* [13] EXALG makes several assumptions regarding the unknown template and values used to generate its input pages. We summarize the important assumptions:

A1: A large number of tokens occurring in template have unique roles, to bootstrap the formation of equivalence classes and subsequent differentiation.

A2: A large number of tokens is associated with each type constructor. Further, each type constructor is instantiated a large number of times in the input pages. This assumption is ensures that the equivalence class derived from a type constructor is recognized as an LFEQ.

A3: There is no "regularity" in encoded data that leads to the formation of invalid equivalence classes.

A4: There are "separators" around data values. In our model, this translates to the assumption that the strings associated with type constructors are non-empty. As we indicated in Section 1 this assumption is made in some related in most information extraction tasks.

CONCLUSION

This paper presented an algorithm, Kumaraveletal [14] EXALG, for extracting structured data from a collection of web pages generated from a common template. EXALG first discovers the unknown template that generated the pages and uses the discovered template to extract the data from the input pages [15-17].

EXALG uses two novel concepts, equivalence classes and differentiating roles, to discover the template. Our experiments on several collections of web pages, drawn from many well-known data rich sites, indicate that EXALG is extremely good in extracting the data from the web pages. Another desirable feature of EXALG is that it

does not completely fail to extract any data even when some of the assumptions made by EXALG are not met by the input collection. In other words the impact of the failed assumptions is limited to a few attributes [18-20].

REFERENCES

1. Amazon.com. http://www.amazon.com.
2. Abiteboul, S., R. Hull and V. Vianu, 1995. Foundations of Databases. Addison Wesley, Reading, Massachussetts.
3. Garcia-Molina, H., Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J.D. Ullman and J. Widom, 1996. The TSIMMIS project: Integration of heterogenous information sources. Journal of Intelligent Information Systems, 8(2): 116-132.
4. Levy, A., A. Rajaraman and J.J. Ordille, 1996. Querying heterogeneous information sources using source descriptions. In Proc. of the 1996 Intl. Conf. on Very Large Data Bases, pp: 251-262.
5. Haas, L., M.D. Kossmann, E.L. Wimmers and J. Yang, 1996. Optimizing queries across diverse data sources. In Proc. Of the 1996 Intl. Conf. on Very Large Data Bases, pp: 266-285.
6. Liu, L., C. Pu and W. Han, 2000. XWRAP: An XML-enabled wrapper construction system for web information sources. In Proc. of the 2000 Intl. Conf. on Data Engineering, pp: 611-621.
7. Kushmerick, N., D. Weld and R. Doorenbos, 1996. Wrapper induction for information extraction. In Proc. of the 1996 Intl. Joint Conf. on Artificial Intelligence, pp: 629-636.
8. Muslea, I., S. Minton and C.A. Knoblock, 1999. A hierarchical approach to wrapper induction. In Proc. of Third Intl. Conf. on Autonomous Agents, pp: 190-196.
9. Hsu, C.N. and M.T. Dung, 1998. Generating finite-state transducers for semi-structured data extraction from the web. Information Systems Special Issue on Semistructured Data, 23(8): 521-538.
10. Crescenzi, V., G. Mecca and P. Merialdo, 2001. ROADRUNNER: Towards automatic data extraction from large web sites. In Proc. of the 2001 Intl. Conf. on Very Large Data Bases, pp: 109-118.
11. Chang, C. and S. Lui, 2001. IEPAD: Information extraction based on pattern discovery. In Proc. of 2001 Intl. World Wide Web Conf., pp: 681-688.
12. Kumaravel, A. and K. Rangarajan, 2013. Algorithm for Automation Specification for Exploring Dynamic Labyrinths, Indian Journal of Science and Technology, ISSN: 0974-6846, 6(5S): 4554-4559.
13. Kumaravel, A. and R. Udayakumar, 2013. Web Portal Visits Patterns Predicted by Intuitionistic Fuzzy Approach, Indian Journal of Science and Technology, ISSN: 0974-6846, 6(5S): 4549-4553.
14. Kumaravel, A. and Oinam Nickson Meetei, 2013. An Application of Non-uniform Cellular Automata for Efficient Crytography, Indian Journal of Science and Technology, ISSN: 0974-6846, 6(5S): 4560-4566.
15. Brin, S., 1998. Extracting patterns and relations from the world wide web. In WebDB Workshop at 6th Intl. Conf. on Extending Database Technology.
16. Hammer, J., H. Garcia-Molina, J. Cho, A. Crespo and R. Aranha, 1996. Extracting semi structure information from the web. In Proceedings of the Workshop on Management of Semistructured Data.
17. Kumaravel, B. Anatha Barathi, 2013. Personalized image search using query expansion, Middle-East Journal of Scientific Research, ISSN: 1990-9233, 15(12): 1736-1739.
18. Pattanayak, Monalisa and P.L. Nayak, 2013. Green Synthesis of Gold Nanoparticles Using Elettaria cardamomum (ELAICHI) Aqueous Extract World Journal of Nano Science and Technology, 2(1): 01-05.
19. Chahataray, Rajashree. and P.L. Nayak, 2013. Synthesis and Characterization of Conducting Polymers Multi Walled Carbon Nanotube-Chitosan Composites Coupled with Poly (P-Aminophenol) World Journal of Nano Science and Technology, 2(1): 18-25.
20. Parida, Umesh Kumar, S.K. Biswal, P.L. Nayak and B.K. Bindhani, 2013. Gold Nano Particles for Biomedical Applications World Journal of Nano Science and Technology, 2(1): 47-57.