Obtaining Product Attributes by Web Crawling

Peter Gurský, Róbert Novotný, Michal Vaško, Milan Vereščák

Ústav informatiky, Prírodovedecká fakulta, Univerzita P.J. Šafárika v Košiciach
Jesenná 5, 040 01 Košice
{peter.gursky, robert.novotny, michal.vasko, milan.verescak}@upjs.sk

Abstract. Current e-shop products aggregators like heureka.sk, najnakup.sk or pricemania.sk, which integrate and compare product offers, obtain their data directly from e-shops through XML data sent by the e-shops having XML schema specified by the aggregator. Our goal is to obtain e-shop product offers by methods of web crawling. That way, the comparison of product offers may cover a wider spectrum of e-shops, including simpler e-shops from the same country without the ability of generating such XML files, or the e-shops all over the world without an agreement of data transmission with the local aggregator. This work-in-progress paper presents the main challenges of the crawling, extracting and unification process, our first implementation proposal and the outcomes of the first experiments.

Keywords: web data mining, web crawling, web annotation, web extraction, information integration, unification

1 Introduction

E-shop product aggregators are popular product offers comparators. One of the disadvantages of current aggregators is that they present offers of their partner e-shops only. Obviously, the same products are offered on many other e-shops not covered by the aggregators. There are 2 obstacles to expand the comparison to a wider scope. First, there are many local e-shops that cannot offer their data in format specified by an aggregator, because it requires complex and expensive programming. Second, there are many e-shops (e.g. in other countries) that do not bother to communicate with the aggregators.

Our vision of the future e-shop product aggregator allows users to add new e-shops to the aggregator by short-time and easy-to-use e-shop content annotation. The annotation would be done in user's web browser with an annotation add-on.

The extraction should not only provide the name and price of the product, but also product pictures, user comments and ratings as well as other attributes like size, color, etc. Product attributes are not collected for product comparisons only, but they are important for the unification with the products from other e-shops too.

2 Components

The integration of the annotated e-shops in the aggregator requires 4 main parts of the system:

1. web crawling, that periodically crawls the relevant pages of the e-shops;
2. identification of product attributes and product data extraction from the crawled web pages to a common data structure;
3. processing and conversions of values of product attributes;
4. unification of identical products.

2.1 Web Crawling

A web crawler is used to gather all detail pages needed for further extraction process. We have implemented our crawler using slightly modified crawler4j java library [14]. In our approach, we collect and store detail (product) pages only. Our crawler uses regular expressions to decide which pages will be saved to our storage to minimize amount of stored data.

The crawler also stores images of objects (products) and other data like user ratings and comments into MySQL database.

The process of crawling a specific web domain should be done repeatedly and frequently due to constantly changing data (e. g. fluctuating product prices). The crawler should also be polite, adhere to robots.txt file and execute commands with reasonable delay, so the server is not flooded with requests which may lead to banning. The crawling of media that do not change so often (e. g. images) has been optimized, so it is only crawled if there is a change. It is done by HTTP request header fields Last-modified and ETag. The optimization was enabled by a change of the core of crawler4j library.

Although crawler4j is designed as a multithreaded crawler, we can further think of parallelizing the whole process of crawling to distribute the workload across computing resources and network resources evenly. Further, it is possible to automatize the process of selection and identification of product detail pages with machine learning techniques used in information extraction systems.

2.2 Web Information Extraction

The web information extraction module uses the semiautomatical user-assisted methods to retrieve annotated product attribute values from HTML web pages. Via tree-difference method [8] we compare multiple HTML tree sources pair-by-pair, thus retrieving a set of candidate XPath locations of object-attribute values within the HTML tree. This approach accelerates the manual annotation of product page which is made via a web browser add-on (see Figure 1). User highlights the relevant object values and annotates them with attribute names and data types. These highlighted values correspond to XPath values and may be used in further executions of tree-difference method to improve the semiautomatical extraction process. To allow

nesting into non-atomic values, we plan to improve the extraction process by introducing support for regular expressions induced from user text selection as a part of the add-on. By combining the extracted XPath expression and regular expressions, we would extract the value of variously presented product attribute data. We further think of trying OXPath [15] instead of XPath. OXPath is an extension of XPath aimed at data extraction, web automation and crawling. It is a relatively new minimalistic language developed and researched at Oxford University.

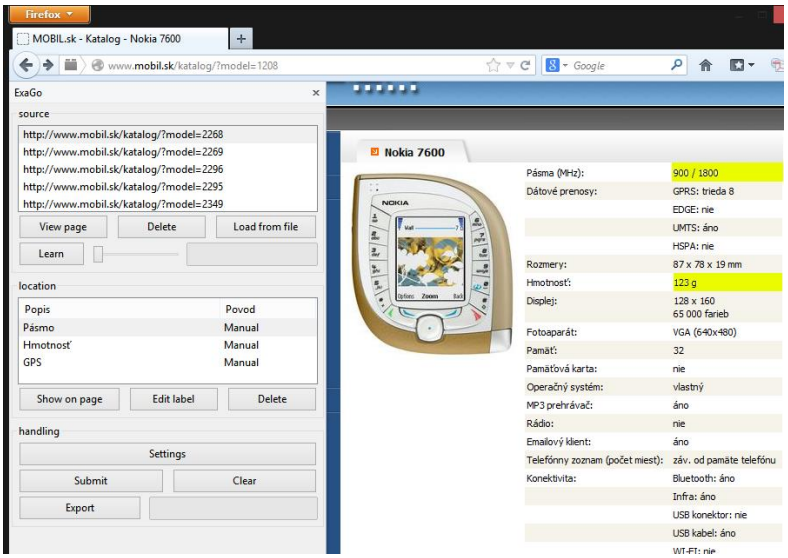

Figure 1: Web-browser add-on. Manually annotated values ("Pásmo", "Hmotnost"' and "GPS") are combined with URLs that would be submitted to the tree difference method.

2.3 Processing and Conversions of Values of Product Attributes

Moving on from the extracted data, we perform the value data type specification for each product attribute. First, we estimate the possible range of the attribute value, especially in the case of attributes which are represented by multi-typed values (e. g. Energy Class A+/150 kWh). Therefore, we need to take into account as many objects with the particular attribute as possible. Further issues are connected with various formats of a single particular value (localized date formats, real numbers etc.), units of measurement conversions, ambiguous names (usually synonyms) or artificially non-atomic or unintentional atomic attribute values.

These issues are eliminated via user-advised configuration scripts so far. A list of attributes, their types, locales, units and attribute names (including synonyms) is presented to the user. By using these data, the user assigns a data type to each presented attribute. Furthermore, all other attribute metadata can be customized by the

user, along with the attribute relevance and value comparison function. Both these values are used in the unification process.

2.4 Unification Process

In the unification process, we compare the attribute values of newly-extracted products with the attribute values of products that have already been processed and are available in the storage. The extracted products usually have many missing attribute values, which complicates the evaluation of product equality. We define 3 types of attribute relevance. The *identifying* attributes (e.g. EAN code) denote equal entities or indicate different objects. The *relevant* attributes (e.g. size, name, weight) contribute to the probability of equal objects. The *general* attributes (e.g. price) neither improve nor degrade the equality ratio. Some attributes are e-shop specific (e.g. local product id, product URL) and can be used in product unification after repeated crawling process.

The unification does pairwise comparison of new objects with the old ones. For each pair of products the attribute similarity is evaluated starting with the identifying attributes. The presence of identifying attributes in both products returns the product equivalency result. If there was no identifying attribute values to compare (e.g. because of the missing values in any of the products), the relevant attribute values are compared using the similarity functions. We compute a product of the attribute similarities. The similarity with the missing value is considered to be 1, thus it has no negative influence on the overall product similarity. We use threshold values to classify products pair to one of the following categories: equal, possibly equal and different. The "possibly equal" group is intended to be analyzed by an administrator.

3 Related Work

Research related to our work generally concerns with web information extraction, web crawling, data matching, specifically record linkage and deduplication of data, information integration, schema matching, etc. Many of the related topics can be found together in a comprehensive overview of web data mining methods [1].

Unlike a universal crawler that gathers all pages irrespective of their content, we use a type of preferential crawler [1] that gathers specific pages (product detail pages) of certain web domains (web e-shops). We had tried several crawlers before we decided to use crawler4j [14]. We had been trying and discussing Heritrix [16], Apache Nutch [17], HTTrack [18] and other crawlers. Heritrix is a crawler written in Java with main emphasis on archiving internet data and it is a robust crawler, but configuration of crawler within xml expects knowing a lot of implementation details of Heritrix crawler. Apache Nutch is one of the most popular crawlers written in Java, but it is more often used in context of full-text search and is designed to work best with Apache Lucene or other search frameworks based on Lucene. Nutch is customizable mainly through creating its own plugins and its main advantage is in its scalability. HTTrack is more suitable for copying complete web domains, but it is not

very customizable as an application. We finally decided to use crawler4j due to the Java platform, its simplicity, and ability to be easily extended: for example with methods responsible for the link selection and page handling.

There are many information extraction systems, mainly focused on wrappers that extract relevant information from HTML documents into a structured form [2, 3, 4, 5, 6]. Unsupervised approaches generate wrappers from unlabeled training examples by identifying patterns and structure of data records from samples of pages [8]. Semi-supervised systems [4] and supervised systems [10] usually use machine learning techniques to generate wrapper and require input from the user to label data records with the help of some GUI. In our approach, we use a supervised system in which general users optionally label a training set of web pages within the well-known web browser environment, thus improving the efficiency of the extraction algorithm. In our system we also focus on the web information integration or web data integration.

We face the problem of many heterogeneous data sets received from extraction process performed on different web pages. Schema integration has been studied in the database community since the early 1980s [9, 10, 11]. These approaches assume complete data without missing values, which is not the case of web-extracted data. A good survey about similarity functions, heterogeneous data unification and data cleaning can be found in [19]. Related research contains also Web Query Interface [12] and ontology, taxonomy or catalog integration [13].

4 Conclusions

Currently, the proposal and creation of the whole system are in the phase, where each part works with a "baseline algorithm" and the chain of modules operates in the test environment. Next, we will focus on the identification of weak parts of the procedure, testing with more data and improvements of each part of the system.

This work is supported by the Agency of the Slovak Ministry of Education for the Structural Funds of the EU, under project ITMS: 26220220158.

References

1. B. Liu: Web Data Mining: Exploring Hyperlinks, contents and Using Data, Second edition, Springer 2011. ISBN 978-3-642-19459-7
2. G. Gottlob, C. Koch, R. Baumgartner, M. Herzog, S. Flesca: The Lixto Data Extraction Project – Back and Forth between Theory and Practice. In: Proc. ACM PODS, 2004.
3. W. Thamviset, S. Wongthanavasu: Structured web information extraction using repetitive subject pattern. Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2012 9th International Conference on. pp. 1 –4 , Thailand, 2012.
4. C.-H. Chang, S.-C. Kuo: OLERA: A Semisupervised Approach for Web-Data Extraction with Visual Support. IEEE Intelligent Systems, 2004, 19 (6): 56-64.

5. C.-H. Chang, M. Kayed, M.R. Girgis, K.F. Shaalan:A Survey of Web Information Extraction Systems. IEEE Trans. Knowledge and Data Eng., vol.18, no. 10, pp. 1411-1428, Oct. 2006.
6. P. Sýkora, A. Janžo, P. Kasan, M. Jemala, I. Berta, V. Szöcs: Automatized Information Retrieval from Heterogenous Web Sources. In M. Bieliková (Ed.), Proc. of IIT.SRC 2006: Student Research Conference, Bratislava, Slovakia, 2006, pp. 137 -144.
7. S. Zheng, R. Song, J. Wen, C.Giles: Efficient record-level wrapper induction. In Proceedings of ACM International Conference on Information and knowledge management (CIKM-2009), 2009.
8. D. Maruščák, R. Novotný, P. Vojtáš: Unsupervised Structured Web Data and Attribute Value Extraction. Proceedings of Znalosti 2009.
9. C. Batini, M. Lenzerini, S. Navathe: A comparative analysis of methodologies for database schema integration. ACM Computing Surveys (CSUR), 1986, 18(4): p. 323-364.
10. A. Doan, A. Halevy: Semantic integration research in the database community: A brief survey. AI magazine, 2005, 26(1): p. 83.
11. P. Shvaiko, J. Euzenat: A survey of schema-based matching approaches. Journal on Data Semantics IV, 2005: p. 146-171.
12. E. C. Dragut, T. Kabisch, C. Yu, U. Leser: A Hierarchical Approach to Model Web Query Interfaces for Web Source Integration. In VLDB, 2009.
13. R. Agrawal, R. Srikant: On integrating catalogs. In Proceedings of International Conference on World Wide Web (WWW-2001), 2001.
14. Y. Ganjisaffar: Crawler4j – Open source web crawler for Java. https://code.google.com/p/crawler4j/
15. T. Furche, G. Gottlob, G. Grasso, C. Schallhart, A. Sellers: OXPath: A language for scalable data extraction, automation, and crawling on the deep web. The VLDB Journal 22(1): 47-72, 2013.
16. G. Mohr, M. Stack, I. Ranitovic, D. Avery, M. Kimpton: An introduction to Heritrix, an open source archival quality web crawler. In Proceedings of the 4th International Web Archiving Workshop, 2004.
17. Apache Nutch. http://nutch.apache.org/
18. HTTrack Website Copier. http://www.httrack.com/
19. V. Ganti, A. D. Sarma: Data Cleaning: A Practical Perspective, Morgan & Claypool Publishers, ISBN: 978-1-608-45678-9, 2013