LOCATING AND EXTRACTING PRODUCT SPECIFICATIONS FROM PRODUCER WEBSITES

Maximilian Walther, Ludwig Hähne, Daniel Schuster, Alexander Schill
Technische Universität Dresden, Faculty of Computer Science, Institute of Systems Architecture
Helmholtzstr. 10, 01062 Dresden, Germany
{maximilian.walther, ludwig.haehne, daniel.schuster, alexander.schill}@tu-dresden.de

Keywords: Document Retrieval, Information Extraction, Federated Search.

Abstract: Gathering product specifications from the Web is labor-intensive and still requires much manual work to retrieve and integrate the information in enterprise information systems or online shops. This work aims at significantly easing this task by introducing algorithms for automatically retrieving and extracting product information from producers' websites while only being supplied with the product's and the producer's name. Compared to previous work in the field, it is the first approach to automate the whole process of locating the product page and extracting the specifications while supporting different page templates per producer. An evaluation within a federated consumer information system proves the suitability of the developed algorithms. They may easily be applied to comparable product information systems as well to minimize the effort of finding up-to-date product specifications.

1 INTRODUCTION

Today, customers as well as retailers and service companies use the Web for gathering detailed product information. As this information is distributed on different websites and presented in heterogeneous formats, this process is both time-consuming and error-prone.

There are already a number of secondary sources bundling product information like online shops (e.g., amazon.com), product review sites (e.g., dpreview.com) or shopping portals (e.g., ciao.de). The information in individual online shops is restricted to only the sold products and often error-prone and not comprehensive. The information on product review sites is collected and verified manually and thus of higher quality but restricted to a special product domain such as dpreview.com to the domain of digital cameras. Shopping portals often rely on information gathered from online shops, thus again only offering incomplete and error-prone information.

For gathering product information from online shops these systems are generally able to query available Web Services, extract the information from websites using web scraping technologies or receive the offers directly by feed-like mechanisms. Gathering product information first-hand from producers is more reliable, but this requires a lot more manual work as this data is not offered in a standardized way by the producers. The operators of the shopping portals or other product information systems have to locate the producer's website, find the website presenting the product of interest, pinpoint the product information and extract it. As this process evidently requires a lot of man hours, information providers tend to either specialize on concrete product domains or reduce the presented information to very general details all products have in common, such as a product name, a producer name, a picture, prices, etc.

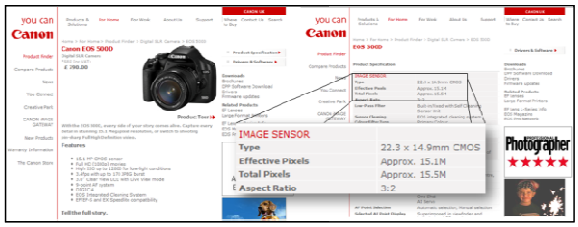

Figure 1: Example product page (left) and product detail page (right) - source: canon.co.uk.

From the consumer's point of view, product specification data provided by producer websites (see example in Figure 1) is the most important product information, as it creates a general view on the product of interest and makes it comparable with

related products. Easing the automatic retrieval of such information would yield a great advantage for product information systems.

To reach this goal the following conditions have to be met:

(Req1)
*The system has to **retrieve** the producer's **product detail page** while only being supplied with a product name and its producer's name. If multiple description pages with different templates exist for the same product, the page with the specification data is to be selected.*

(Req2)
*The system has to be able to **extract information** being supplied with few similar or even only one product detail page.*

(Req3)
***Different page templates** for one manufacturer have to be managed by the extraction process, e.g., in case of different product categories or families.*

Current methods for information extraction already cover part of (Req2) while they do not yet take the product page retrieval into account (Req1). Different page templates (Req3) are also not yet considered by existing work.

Thus, the contributions of this paper are techniques to fit all three requirements mentioned above. We present an algorithm to locate the product information page provided by a producer for an arbitrary product (Section 3) and three complementary algorithms for finding the product details on this page and extracting them (Section 4). The four mentioned algorithms were implemented and evaluated as described in Section 5. We conclude the paper in Section 6 discussing future research directions.

2 RELATED WORK

As shown in the introduction, the presented work is located in the area of product information retrieval putting a special focus on product document retrieval (DR) and information extraction (IE), more precisely, product specification extraction. Several systems dealing with similar problems were developed in related research works.

Considering the product information domain, these systems mostly handle vendor information provided by online malls or third-party information in the shape of user reviews.

Systems for gathering vendor information either access online malls using Web Services or web scraping wrappers and rank resulting product lists by federated ranking mechanisms. Detailed information on such systems including a feasible approach for federated ranking can be found in (Walther et al., 2009b). Wong and Lam (2009) present algorithms for feature mining especially applicable for extracting product information from vendor sites. Their evaluation proves the algorithms' feasibility in comparison to other systems.

Concerning third-party information like user reviews, TextRunner (Banko et al., 2007) offers a facts-based search engine using the principles of Open Information Extraction. Sources treated by TextRunner do not only comprise product reviews. Red Opal (Scaffidi et al., 2007) offers effective product search mechanisms based on the evaluation of a product review database. Reviews are examined concerning special product features, thus enabling the system to provide a set of products to the consumer that is expected to satisfy their needs concerning a chosen feature.

As mentioned above, the presented systems do not focus on product information provided by producers. In effect, such information is of particular interest for the consumer as producers offer complete, correct and up-to-date information.

In the field of information extraction, many research results have been published as well. Those may be divided in supervised, semi-supervised and unsupervised approaches.

The approach of learning extraction rules from labeled training documents is referred to as supervised IE. Rapier (Califf and Mooney, 1997) is a supervised extraction system that uses a relational learning algorithm to extract information from job postings. It initializes the system with specific rules to extract the labeled data and successively replaces those with more general rules. Syntactic and semantic information is incorporated using a part-of-speech tagger. Other supervised IE systems are SRV (Freitag, 1998), WIEN (Kushmerick et al., 1997), SoftMealy (Hsu and Dung, 1998), STALKER (Muslea et al., 1999) and DEByE (Laender et al., 2002).

Labeling training data in advance is a labor-intensive process limiting the scope of the IE system. Instead of requiring labelled data, semi-supervised IE systems extract potentially interesting data and let the user decide what shall be extracted. IEPAD (Chang and Lui, 2001) is a semi-supervised system. Apart from extraction target selection, such systems are very similar to unsupervised IE systems.

Automatic or unsupervised IE systems extract data from unlabeled training documents. The core concept behind all unsupervised IE systems is to

identify repetitive patterns in the input data and extract data items embodied in the recurrent pattern. Unsupervised IE systems can be subdivided into record-level extraction systems and page-level extraction systems. The former assume multiple data records of the same type are available being rendered by a common template into one page while the latter extract data from multiple pages having the same page-wide template.

Evidently, record-level extraction systems can only operate on documents containing multiple data records and require means to identify the data regions describing the individual data records. The latter problem can be tackled with string or tree alignment techniques. Examples for such systems are DEPTA (Zhai and Liu, 2005) and NET (Liu and Zhai, 2005). DEPTA stands for Data Extraction based on Partial Tree Alignment and is an unsupervised IE system. It extracts data records from list pages (e.g., Amazon search result lists) with an algorithm called MDR, taking advantage of the tree structure of the HTML page. MDR was first presented by Liu et al. (2003). The design of MDR is based on two observations about data records. The first observation states that similar objects are likely located in a contiguous region and formatted with almost identical or at least similar HTML tags. The second observation is that similar data records are built by sub-trees of a common parent node. Unfortunately, multi-record IE systems like DEPTA are not well-suited for our extraction problem, as product detail pages are rarely multi-record pages and typically describe only a single product.

Page-level extraction systems can treat the whole input page as a data region from which the data record shall be extracted. However, multiple pages for induction of extraction wrappers need to be fetched in advance. Thus, the problem of collecting training data is shifted into the DR domain and is rarely addressed by IE researchers. Examples for page-level extraction systems are RoadRunner (Crescenzi et al., 2001) and ExAlg (Arasu and Garcia-Molina, 2003). RoadRunner is an unsupervised web IE system that compares multiple pages and generates union-free regular expressions based on the identified similarities and differences. RoadRunner initializes the wrapper with a random page of the input set and matches the remaining pages using an algorithm called ACME matching. The wrapper is generalized for every encountered mismatch. Text string mismatches are interpreted as data fields, tag mismatches are treated as indicators of optional items and iterators. ExAlg is an IE system for automatically deducing the template from a set of template-generated pages. It has a hierarchically structured data model and supports optional elements and disjunctions. A web page is modeled as a list of tokens in which a token might either be an HTML tag or a word from a text node. ExAlg builds equivalence classes of the tokens found in the input documents. Based on these sets of tokens, the underlying template is deduced.

The drawback of these page-level IE systems relating to our extraction problem is the large number of training data to induce extraction rules. ExAlg draws upon the target attributes' occurrence characteristic which can hardly be derived from only two training pages, thus not meeting (Req2). Furthermore, the presented approaches do not take the problem of document retrieval into account and hence do not fulfil (Req1). Additionally, the support for multiple page templates (Req3) is not tackled yet. The conditions stated above are essential for a successful employment of such algorithms in federated consumer product information systems. In the following, we present our approach building upon some of the ideas presented here, extending them to fully fit (Req2) as well as finding new methods to tackle document retrieval (Req1) and multiple page templates (Req3).

3 DOCUMENT RETRIEVAL

The retrieval component's task is to supply the information extraction algorithm with a genuine product specification page. We use web search services such as Google, Bing and Yahoo for this purpose.

The document set to consider is the total number of publicly available websites W. Let the product whose specification page is to be found be p_i. Thus, all websites presenting information about this product can be subsumed as $W(p_i)$. Since only specification pages are of interest, these websites are defined by $W_S(p_i)$. Specification pages may be distributed all over the Web being offered by arbitrary sources. However, product manufacturers are accounted to be the most trustable sources concerning their own products. All websites provided by a manufacturer producing p_i can be summarized by $W(m(p_i))$. Hence, the document to be found is one of the websites $W(m(p_i)) \cap W_S(p_i)$. In the majority of cases, only one producer's specification page exists per product, therefore following through with $|W(m(p_i)) \cap W_S(p_i)| = 1$. If so, this page is curtly defined as w_i.

The formula shows that the DR component's task consists in determining the set of producer websites $W(m(p_i))$ for the producer of p_i filtering out the set of pages presenting information about p_i and finally detecting w_i or choosing one of the found product specification pages. Thus, the retrieval is laid out as

a two-step process. In a first step, the producer page is located and, in a second step, the product specification page is searched restricting the requests to the producer domain.

3.1 Producer Page Retrieval

A producer site comprising $W(m(p_i))$ is searched for by querying the mentioned web search services with the producer's name, e.g., "Siemens Home Appliances". The results returned by all search engines are ordered using Borda ranking (Liu, 2007). In Borda ranking, every participant announces an ordered list of preferred candidates. If there are n candidates, the top-ranked candidate of each voter receives n points and each lower ranked candidate receives a decremented score. For being able to search on the producer's site, the producer domain is extracted. It includes the top-level domain of the host and one further level. For example, from the URL http://www.gigabyte.com.tw/ the domain name gigabyte.com.tw is extracted. If the product page cannot be retrieved on-site, the algorithm falls back to the next producer site candidate from the phrase search.

3.2 Product Detail Page Retrieval

For locating the actual product page, that is, building the intersection of $W(m(p_i))$ and $W_S(p_i)$, again different web search services are queried, this time using the product's name as query and restricting the search space to the retrieved producer domain. First, the result sets of the individual search engines are combined using Borda ranking to form an initial candidate list (see Figure 2).

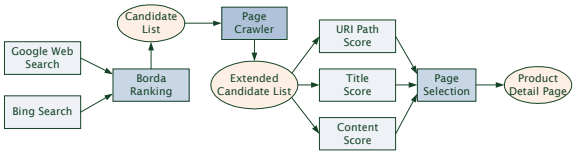

Figure 2: Scoring the product page candidates.

Under the supposition of a product page being discovered but the specification data being contained in a separate page, each page from the candidate list is scanned for product detail page links. Each link's text is compared with characteristic product detail page link patterns. The target of the best matching link is added to the extended candidate list. The prospective specification page inherits the Borda score of the linking page if it is not among the existing search results. Additionally, a specification score is assigned to this candidate.

Subsequently, each result from the extended candidate list is rated with a URI score, a title score and a content score. For the URI score, the URIs of the candidates are scanned for characteristic terms associated with positive or negative meanings in the context of searching for product specification data. For example, the terms "product" or "specification" in a URL might indicate that the candidate is indeed a product specification page. Contrariwise, terms like "forum", "news", "press" or "review" might signify an irrelevant page in this context and entail a negative score. Furthermore, the URL is scanned for substrings of the product name. Accordingly, a URI score is given to each candidate.

In a next step, the titles of the web pages are matched with the product identifier. The rationale behind this concept is to favor pages associated with the proper product in contrast to specification pages associated with similar products which might receive an otherwise high score. Depending on the percentage of matching terms a title score is calculated for every candidate.

In a last step, the document contents are scanned for customary attribute key phrases. For this purpose, possibly available attribute keys from former extractions and their occurrence counts are retrieved. The set of text nodes contained in the page is matched with these phrases to calculate the candidates' content scores.

All computed scores are combined. The candidate with the highest score is returned as the alleged product page w_i. An example can be seen in Table 1.

Table 1: Final rankings for "Sony Ericsson W595a".

Document	Borda	Spec.	URI	Title	Cont.	∑
/cws/products/mobilephones/overview/w595a?lc=en&cc=us	20	0	4	9	0	33
/cws/support/phones/w595a?lc=en&cc=us	16	0	-2	9	0	23
/cws/corporate/products/phoneportfolio/specification/w595a	15	0	6	9	10	40
/cws/products/mobilephones/specifications/w595a?lc=en&cc=us	9	10	6	9	10	44
/cws/support/softwaredownloads/w595a?lc=en&cc=us	7	0	-2	9	0	14

4 INFORMATION EXTRACTION

The information extraction component is designed to extract key-value pairs of product information from product detail pages on the producers' sites. Thus, the following algorithm takes a product detail page as input and retrieves the product's specifications from this page. As keys and values in one product

detail page share similar XPaths, we try to find those XPaths and create an extraction wrapper out of them. An overview of the procedure is given in Figure 3.

In a first step the given product specification page is fetched and the DOM tree is created. Then, extraction wrappers already residing in the system can be applied. A wrapper consists of an attribute XPath and a relative key XPath. To extract the attributes, the wrapper retrieves the node set via the attribute XPath from the DOM representation of the input document. The key node is located by the relative key XPath. Subsequently, the key node is removed from the attribute node and the remaining text in the node is presumed to be the value component. If the extraction fails, the wrapper is not valid and a new one has to be induced.

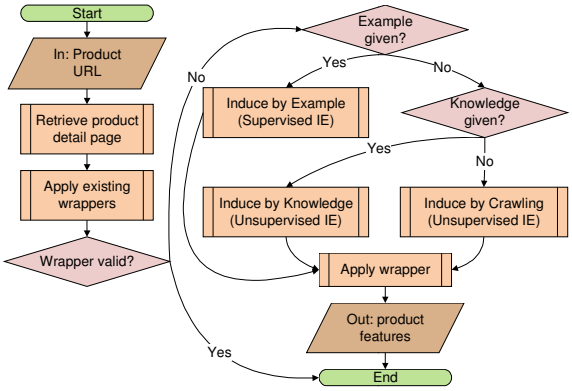

Figure 3: General information extraction procedure.

4.1 Wrapper Induction

As depicted in Figure 3, different wrapper induction algorithms, namely a supervised and two unsupervised algorithms, are executed. The supervised algorithm (Induce by Example) is applicable when provided with key examples (e.g., "Optical Zoom" for the digital camera domain) directly giving a hint on the product attributes to be extracted from the product detail page. The different algorithms are shown in Figure 4.

Independent of the chosen algorithm, the first step comprises the creation of phrase clusters that might contain the product details to be extracted. The phrases are all text nodes of the website's DOM tree. Only unique phrases are considered, all recurrent phrases are discarded during clustering. The clustering is based on the phrases' generalized occurrence paths and enclosing tag signatures. The former is defined as an XPath query unambiguously selecting a node set only containing the respective nodes and being stripped of all indices. For example, the generalized occurrence path of "//table/tr[4]/td[1]" is "//table/tr[]/td[]". The latter consists of the enclosing tag element including its attributes.

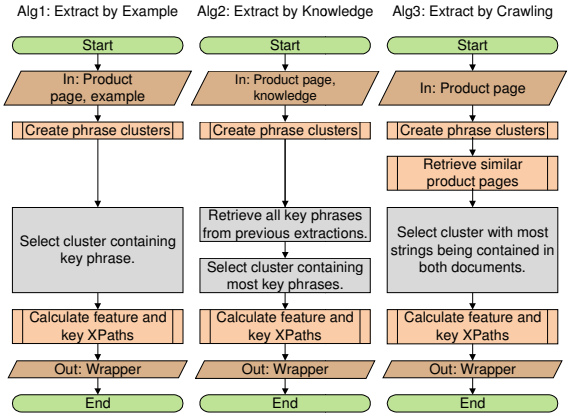

Figure 4: Wrapper induction algorithms.

As can be seen in Figure 5, phrases not occurring in all input documents are discarded in case multiple documents are considered. Thus, a phrase cluster contains all text nodes residing on the same level of the DOM tree having an identical enclosing tag. Apparently, the attributes must not reside in the very same tag to be syntactically distinguishable by the algorithm. Otherwise, approaches like ExAlg operating on the token-level have to be adopted.

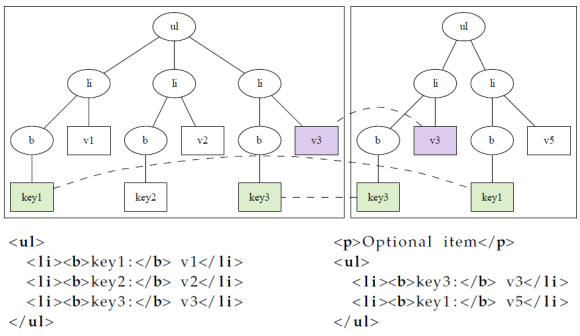

Figure 5: Clustering text nodes from multiple documents.

After all text nodes have been assigned to a cluster, a score for each cluster is computed using a rating function. Different rating functions are used depending on the wrapper induction algorithm. The different induction variants and employed rating functions are discussed below. Subsequently, an XPath query is derived from the path of all nodes in the best rated cluster and a wrapper object is created. The XPath is converted to a generalized occurrence path and split on the last eliminated index into attribute XPath and relative key XPath. If the wrapper is able to extract any attributes using these

XPaths, the wrapper is returned. Otherwise, the next cluster with a lower score is considered.

4.1.1 Induction with Example Phrase

The wrapper induction process can be facilitated by specifying a key phrase of one product attribute as an example. When such an example is provided, no additional training document is crawled and the cluster rating just looks for the example phrase in the available clusters (left side of Figure 4).

4.1.2 Induction with Domain Knowledge

If no example key phrase is given, but extracted and confirmed product data is already stored in the system, all key phrases of this known product data are matched with each element in the cluster and a hit is recorded in case of success. The cluster score is simply modelled as the number of hits (central part of Figure 4).

4.1.3 Induction with 2nd Product Page

If neither example key phrase nor domain knowledge is available, the wrapper induction relies on a training approach. Related product pages are retrieved to provide a training set. Phrases not occurring in all training documents are discarded and the clusters are rated based on their size scaled by the fraction of non-discarded nodes. The scaling shall prevent mixing up key and value components. The latter may happen when individual attributes have value tuples which build a larger cluster than the key components of the respective attributes (right side of Figure 4). How to retrieve related product pages is described in the following.

4.2 Related Product Page Retrieval

If no domain knowledge is available for the information extraction component to identify relevant data on a given product page, the generator of the IE wrapper requires at least one other page sharing the same template to detect recurrent patterns. Thus, similar pages are crawled starting from the product detail page and selecting a page with a similar URL, content and structure. This approach is feasible, as it often takes no more than two clicks to navigate from a product detail page to another one of a similar product. Additionally, similar URLs are more likely to reference template-sharing pages, e.g., “/product.html?name=D60” and “/product.html?name=S550”. This is due to routing mechanisms in template-based web application frameworks.

The crawler starts at the product detail page and recursively extracts all referenced URLs until a given recursion depth is reached. In practice a depth of two showed suitable results. The URLs then are sorted by similarity to the original product URL and provided to the IE component. The URL similarity is modelled as the weighted Levenshtein distance of the individual URL components. E.g., differences in the host name have a larger impact on the final score than differences in the URL’s query part.

4.3 Text Node Splitting

In practice, product attributes are often stored in a single text node with a colon separating key and value items, e.g., "Key: Value". Therefore, text nodes are split along the alleged separator and only the first part is stored in the cluster. If such joint phrases are predominant in a cluster, the wrapper stores the occurrence path of the cluster as the attribute path without specifying a key path. In these cases, the wrapper performs the extraction of the key and value terms from the joint attribute phrases.

Whether a cluster features joint attribute phrases is determined based on the fraction of phrases with an alleged separator. Either most phrases in the cluster contain a separator or known key phrases are found featuring a separator. For this reason, individual phrases are voted for in the respective rating functions.

4.4 Wrapper Selection

As stated above, product pages residing on the same producer site often share the same template. Still the algorithm should be able to handle more than one template per manufacturer, e.g., considering different product categories. When extracting information from an arbitrary product page, it needs to be decided which wrapper will extract information from the page or, in case none is applicable, whether a new one is to be generated.

It is feasible to let all existing wrapper objects of the current domain extract information out of a given product page. Improper extraction rules either yield no data at all or might mine bogus data. Therefore, the returned data is matched with existing data previously extracted by the same wrapper. The assumption is that a certain template encodes similar content. For example, the pages of a producer’s product line share one template, while pages from another product line are encoded with a different template. This way, every wrapper of a producer is assigned a score whose role is twofold. On the one hand, the score is used to select the proper wrapper object. On the other hand, a minimal threshold

ensures that a new wrapper is generated in case no eligible extraction rules exist yet. The threshold is based on the amount of prior knowledge available for a certain wrapper.

5 EVALUATION

To evaluate the effectiveness of the approach, the described algorithms were implemented in Fedseeko (Walther et al. 2009a), a system for federated consumer product information search. The presence of domain knowledge is denoted by the D superscript in the charts. The gold standard used for testing consisted of 100 products from 40 different manufacturers and 10 diverse application domains. These products were picked to get a broad coverage of different product categories. For the tests concerning domain knowledge, 262 key phrases were inserted into the database gathered from representative products for each of the 10 domains.

5.1 Product Page Retrieval

For evaluating the retrieval component the gold standard consisted of the proper domain and page URL of each product and its producer. The automatic retrieval results were matched with the prestored locations. If the locator was able to identify the proper product page URL, the retrieval was filed as a success. Due to URL aliases and localized versions of product pages, non-matching URLs were checked manually again to identify false negatives. The results are illustrated in Figure 6.

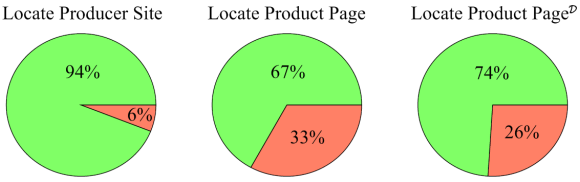

Figure 6: Evaluation of page retrieval.

With a success rate of over 90%, the producer site identification is quite robust. However, failures to find the producer site occur when a producer has distinct sites for different product groups. For instance, the home appliance product line of Siemens is not featured at the primary Siemens site "siemens.com" but offloaded to another domain, namely "siemens-home.com". Another frequently encountered source of error are localized producer sites. These might list diverse sets of products or use different product identifiers. However, often traces of product names are found on other sites of the producer, e.g., in the news section or in a support forum. Better retrieval results could be accomplished by searching the product page on multiple producer candidates in parallel and combining the results from all pages.

It is worth stressing the point that only the retrieval of the product detail page was filed as a success. In the majority of the failure cases, an overview page associated with the proper product was returned. In other cases, an index or comparison page listing multiple products was identified. Other failures can be attributed to the retrieval of wrong products' pages, ineligible content like product reviews or news entries.

On the right side of Figure 6 it can be seen that incorporating domain knowledge (reference attribute keys) increases the retrieval performance. At the same time, there is a slightly greater chance of retrieving a wrong product's page because the domain knowledge embodied in the accumulated key phrases is generic.

Overall, the retrieval component proved a very good performance concerning producer sites, while having some deficiencies in the field of product detail page retrieval. This is especially adverse, as the information extraction relies on the retrieval of a correct product detail page. However, the usage of domain knowledge ameliorates the situation and thus makes the algorithm quite suitable. Domain knowledge is already gathered automatically in the system. Still, optimizations in this field might yield even better results for the product page retrieval and thus will be part of future work.

5.2 Related Product Page Retrieval

The goal of the web crawler is to identify a page generated from the same template as the reference page. During the tests, this succeeded in 69% of the cases. Half of the failure cases can be attributed to the fact that different views being associated with the same product often have a high page and URL similarity. However, if there are few common attributes and only slight deviations in the product-specific parts of the page, it is more likely that another view of the original product will be taken for a related product page. Another problem occurs in case template-sharing pages are not reachable through the designed link distance. Though the recursion limit could be increased, the execution time easily rises tenfold with every followed link level. A recursion depth of two was found to be a suitable trade-off.

In order to make the employed related page retrieval algorithm comparable, a component for autonomously locating random product pages of the same producer has been implemented. However, one

has to take product page detection failures into account. The considerable chance that the found page was built from a different template has to be regarded as well. Moreover, the other product page might feature a completely distinct set of key phrases. It was observed that such a system performed rather poorly in comparison to the crawler-based approach.

5.3 Information Extraction

Assessing the extraction performance is slightly more complicated than evaluating the page retrieval performance, as many attributes are associated with every product. Each of the three presented extraction algorithms was confronted with the task of extracting product attributes from the product pages. A reference attribute was manually retrieved for each product and matched with the extracted data. Whenever the reference attribute was contained within the set of extracted attributes, it indicated that (1) the proper data region had been selected, (2) the proper granularity level was chosen in a nested template and (3) the value could be mapped to its associated key phrase. Apparently, it does not indicate that the extracted data was complete or correct. Therefore, the first and last attribute were also recorded for reference and the number of extracted records was checked.

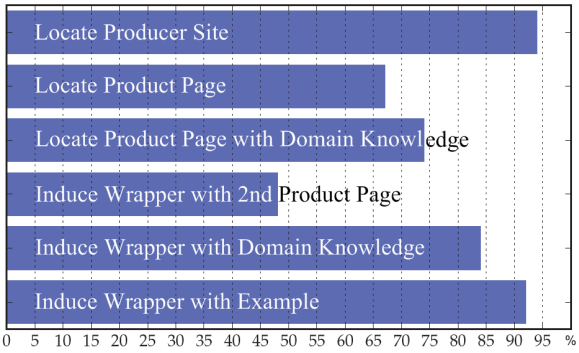

Figure 7: Evaluation of retrieval and extraction components.

As Figure 7 reveals, wrapper induction using crawled documents works for approximately half of the test products. However, significant extraction performance improvements were gained with the availability of domain knowledge. Unfortunately, some cases cannot be handled even when an example is provided. This applies to about one in ten products in the test data. A successful extraction implies that at least some product attributes were correctly extracted. More detailed results are given in Figure 8.

The per-product results can be classified in different success and failure categories, based on correctness and completeness of the extracted data. A perfect result indicates that the extraction results are correct and complete. In other words, all available attributes were extracted and no false positives were in the result set. The second category includes attribute sets which are complete but contain additional incorrect attributes. Finally, if some of the attributes were not extracted, the data set is filed as incomplete.

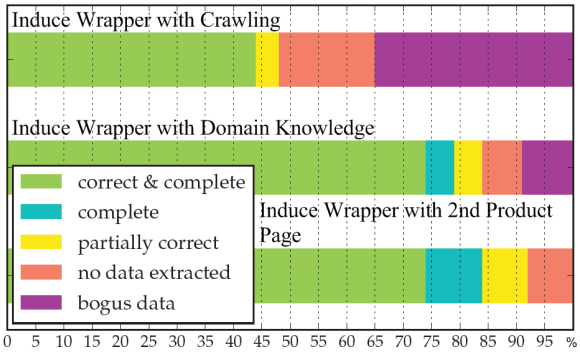

Figure 8: Correctness and completeness of extraction results.

The failures can be categorized into cases where no attributes were extracted at all and those where bogus attributes were mined. The former is less hazardous because it requires no guidance to mark these cases as failures. In the latter cases, however, the extracted data must be rejected manually.

Using automatic extraction with existing domain knowledge, 85% of the extracted product attributes were correct and 10% bogus data. On average, 23 of 27 available product attributes were correctly extracted and one false positive was mined.

Overall, the information extraction component showed feasible results. Assuming that the algorithms are included in an information platform used by consumers, it is expected that users provide extraction hints to the system in a wiki-like form. After some running time and the intensive collection of domain knowledge, the extraction success should even increase, thus only making the employment of information extraction by crawling inevitable in very few cases.

6 CONCLUSIONS

In this paper we presented algorithms for locating and extracting product information from websites while only being supplied with a product name and its producer's name. While the retrieval algorithm

was developed from scratch, the extraction algorithm extends previous works presented in Section 2 especially leveraging the special characteristics of product detail pages. The evaluation showed the feasibility of the approaches. Both the retrieval and extraction component generated better results when being supplied with domain knowledge used for bootstrapping. Thus, future research will focus on improving the system's learning component to automatically create extensive domain knowledge at runtime.

Currently, additional algorithms are being developed for mapping the extracted specification keys to a central terminology and converting the corresponding values to standard formats. Thus, product comparisons would be enabled at runtime. Evaluations will examine the success of these algorithms. Another direction of future research includes the automatic extension of the used product specification terminology being represented by an ontology. Thus, the mapping algorithm's evaluation results would be improved significantly.

The consolidated integration of this paper's algorithms as well as described future extensions in a federated consumer product information system would enable users to create an all-embracing view on products of interest and compare those products effectively while only requiring a fraction of today's effort for gathering product information from the information provider. In the same manner it may be integrated in enterprise product information systems as well as online shopping systems easing and accelerating the process of implementing product specifications.

REFERENCES

Arasu, A. and Garcia-Molina, H. (2003). Extracting Structured Data from Web Pages. In SIGMOD *International Conference on Management of Data.* San Diego, CA, USA 10-12 June 2003. ACM Press: New York.

Banko, M., Cafarella, M. J. Soderland, S., Broadhead, M. and Etzioni, O. (2007). Open Information Extraction from the Web. In IJCAI *20th International Joint Conference on Artificial Intelligence.* Hyderabad, India 9-12 January 2007. Morgan Kaufmann Publishers Inc.: San Francisco.

Califf, M. E. and Mooney, R. J. (1997). Relational Learning of Pattern-Match Rules for Information Extraction. In ACL SIGNLL *Meeting of the ACL Special Interest Group in Natural Language Learning.* Madrid, Spain July 1997. T. M. Ellison: Madrid.

Chang, C.-H. and Lui, S.-C. (2001). IEPAD: Information Extraction based on Pattern Discovery. In IW3C2 *10th International Conference on the World Wide Web.* Hong Kong, China 1-5 May 2001. ACM Press: New York.

Crescenzi, V., Mecca, G. and Merialdo, P. (2001). Roadrunner: Towards Automatic Data Extraction from Large Web Sites. In VLDB Endowment *27th International Conference on Very Large Data Bases.* Rome, Italy 11-14 September 2001. Morgan Kaufmann Publishers Inc.: San Francisco.

Freitag, D. (1998). Information Extraction from HTML: Application of a General Machine Learning Approach. In AAAI *15th National Conference on Artificial Intelligence.* Madison, WI, USA 26-30 July 1998. AAAI Press: Menlo Park.

Hsu, C.-N. and Dung, M.-T. (1998). Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web. *Journal of Information Systems,* 23(8), pp.521-538.

Kushmerick, N., Weld, D. S. and Doorenbos, R. (1997). Wrapper Induction for Information Extraction. In IJCAI *15th International Joint Conference on Artificial Intelligence.* Nagoya, Japan 23-29 August 1997. Morgan Kaufmann Publishers Inc.: San Francisco.

Laender, A. H. F., Ribeiro-Neto, B. and da Silva, A. S. (2002). DEByE - Data Extraction by Example. *Data and Knowledge Engineering*, 40(2), pp.121–154.

Liu, B. (2007). *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data.* Springer: Heidelberg.

Liu, B., Grossman, R. and Zhai, Y. (2003). Mining Data Records in Web Pages. In SIGKDD *International Conference on Knowledge Discovery and Data Mining.* Washington, DC, USA 24-27 August 2003. ACM Press: New York.

Liu, B. and Zhai, Y. (2005). NET - A System for Extracting Web Data from Flat and Nested Data Records. In WISE Society *6th International Conference on Web Information Systems Engineering.* New York, NY, USA 20-22 November 2005. Springer: Heidelberg.

Muslea, I., Minton, S. and Knoblock, C. (1999). A Hierarchical Approach to Wrapper Induction. In IFAAMAS *3rd International Conference on Autonomous Agents.* Seattle, WA, USA 1-5 May 1999. ACM Press: New York.

Scaffidi, C., Bierhoff, K., Chang, E., Felker, M., Ng, H. and Jin, C. (2007). Red Opal: Product-Feature Scoring from Reviews. In SIGECOM *8th ACM Conference on Electronic Commerce.* San Diego, CA, USA 11-15 June 2007. ACM Press: New York.

Walther, M., Schuster, D. and Schill, A. (2009a). Federated Product Search with Information Enrichment Using Heterogeneous Sources. In Poznan University of Economics *12th International Conference on Business Information Systems*. Poznan, Poland 27-29 April 2009. Springer: Heidelberg.

Walther, M., Schuster, D., Juchheim, T. and Schill, A. (2009b). Category-Based Ranking of Federated Product Offers. In IADIS *8th International Conference on WWW and Internet*. Rome, Italy 19-22 November 2009. IADIS Press: Lisbon.

Wong, T.-L. and Lam, W. (2009). An Unsupervised Method for Joint Information Extraction and Feature Mining Across Different Web Sites. *Data and Knowledge Engineering*, 68(1), pp.107-125.

Zhai, Y. and Liu, B. (2005). Web Data Extraction Based on Partial Tree Alignment. In IW3C2 *14th International Conference on the World Wide Web.* Chiba, Japan 10-14 May 2005. ACM Press: New York.