Federated Product Search with Information Enrichment Using Heterogeneous Sources

Maximilian Walther, Daniel Schuster, and Alexander Schill

TU Dresden, Department of Computer Science, Helmholtzstraße 10,
01062 Dresden, Germany
{walther,schuster,schill}@rn.inf.tu-dresden.de

Abstract. Since the Internet found its way into daily life, placing product information at the user's disposal has become one of its most important tasks. As information sources are very heterogeneous concerning the provider as well as the structure, finding and consolidating the information for generating an all-embracing view on a product has become an important challenge. An auspicious approach for resolving the emerged problems are federated search techniques enriched with Information Extraction and Semantic Web concepts. We investigate possibilities for federated product search combining heterogeneous source types, i.e., structured and semi-structured sources from vendors, producers and third-party information providers. The developed approach is evaluated with a Ruby on Rails implementation.

Keywords: Federated Search, Information Extraction, Ontology, Product Information Management.

1 Introduction

Since the permeation of daily life by the WWW, product information search has become one of the central tasks carried out on the Internet. Producers increasingly realized the importance of presenting their products and associated information in an appealing and informative way online. Additionally, several types of online malls have emerged which today are widely used by consumers as starting points for collecting product information. Due to the fact that providing information on the Internet has become cheap and feasible, also average Internet users have started to enlarge the basis of information by adding so-called user-generated content. As a consequence the Internet now holds many different information sources created by different authors and structured in many different ways. Regarding the amount and the heterogeneity of product information on the Web, the task of information retrieval for one special product involves extensive research including the usage of different vendor, producer and 3rd-party websites.

Unfortunately most sources hold assets and drawbacks considering information quality. For instance, producer websites provide complete and correct information, but use advertising text for promotion purposes. Lexica like Wikipedia contain goal-oriented and fresh information, but are not immune to biased product characterizations.

As per description there are a lot of criteria for information sources to be called ideal. Table 1 presents all conditions that an ideal information source should fulfill.

Table 1. Requirements for an ideal product information source

- Completeness	All available information is included.
- Correctness	All included information is correct.
- Freshness	All included information is up-to-date.
- Neutrality	The information is not biased.
- Goal Orientation	All included information is relevant.
- Comparison	Information from similar products is available.
- Verification	Information is backed by corresponding references.

Obviously no current product information source on the Internet is able to fulfill all of these criteria. Instead a combination of data from many different product information sources could provide an opportunity for satisfying them.

Federated search is the keyword in this context, as it allows the user to gain an integral overview for the product of interest without requesting each information source separately. Additionally, the federation enables access to information sources the user would not have been searching himself. Current providers of applications for simultaneous search already allow the access of structured information sources typically offered by Web Services. However, important additional information for a special product is to be found in semi-structured and unstructured sources like producer websites and websites consisting of user-generated content.

We contribute a reference architecture for the heterogeneous federated consumer product search as well as methods for the extraction of semi-structured information from different information source types. The introduced architecture consolidates information from these sources to offer an all-embracing product view to the user.

The Fedseeko system is a research platform, which implements the described architecture. It uses multiple vendors for building a searchable information basis and enriches the given information automatically by details from other sources, such as producer websites or 3rd-party information providers using information extraction (IE) methods. Experimental results show the feasibility of the developed approach.

2 Related Work

In the field of federated product search, Shopbots [1] emerged already in the mid 90's and were the first step towards integration of multiple vendors in a federated product search using screen scraping. Scraping vendor websites caused a number of problems because it is error-prone and delivers incomplete information. The IPIS system [2] overcomes these problems as it uses Web Service interfaces and ontology mapping as key technologies. The user creates a semantic product query with the help of product categories. Those categories are organized in ontologies to enable an easy mapping between different information providers. As a main drawback, this approach relies on the assumption that each shopping mall has Web Service interfaces and is able to process semantic queries. A more lightweight approach is the shopinfo.xml standard [3]. Shop operators may define a shopinfo.xml that can be downloaded by any shopping portal easily, providing both RESTful Web Services as well as downloading an XML product file for shop federation. Unfortunately, information relevant for buying decisions is not restricted to multiple vendors, but also comprises information on producer websites as well as third-party information, additional data, knowledge or services [4].

The approaches pictured above offer means to consolidate vendor product information, i.e., information provided by online malls like Amazon. Concerning the vendor sources, our approach combines and dilates these works, as we extend the federated product search from integration of different shops with similar interfaces to federation of heterogeneous vendor sources. An adapter-like approach offers the possibility to integrate sources accessible by Web Services as well as Web front ends. Therefore we introduce a generic wrapper adopting specific Web information extraction techniques. Additionally we allow the integration of 3rd-party websites by offering means of information extraction from these sites.

Product information on producer websites often is presented in a semi-structured manner. Extracting and modeling this information is subject to many research approaches as well. Wong et al. [5] describe an unsupervised framework enabling both the extraction and normalization of product attributes from web pages containing semi-structured product information. A number of algorithms are adopted to categorize extracted text fragments and map found attributes to corresponding reference attributes depending on the current product domain. Lermann et al. [6] also picture techniques for extracting semi-structured product information from websites. The developed algorithms are based on AUTOCLASS [7] and ALERGIA [8] and are able to identify page templates and detect table structures to extract information from any semi-structured web page, if only some pages using the same template were given before. Unlike [5], table contents are not directly normalized, that is, the extracted attributes are not matched with reference attributes. Brunner et al. [9] present possibilities to overcome the problem of redundant data management. They describe an architecture for integrating general business object information in ontology entities using different layers of the MOF-model. A highly performant maintenance of ontology information in databases is described as well.

The described approaches offer different possibilities to extract and normalize product information from websites. We introduce alternative methods, especially for the extraction of semi-structured information from producer websites, enabling the extension of the user's product information base with highly relevant and precise information directly from the manufacturer.

The federation of heterogeneous sources using different types of information extraction described in this paper is first presented in [10], providing enrichment of information from online shopping malls with information from product detail pages of producers. We extend this approach to a more generic architecture in this paper. While [10] is restricted to structured vendor information and semi-structured producer information, the work presented in this paper is able to integrate structured as well as semi-structured information from vendors, producers and 3rd-parties in the federated search process.

3 Architecture

Fedseeko's general system architecture is shown in Figure 1. As can be seen on the right side of the figure, Fedseeko is able to query three different types of information sources. These sources are offered by vendors, producers and 3rd-parties. Vendors are online malls like Amazon.com or Buy.com that are able to deliver some basic information about products. Producers are the corresponding manufacturers of the investigated product. They deliver technical specifications of products with a high degree of credibility. 3rd-parties are information providers delivering product information that is often generated by average Internet users. This includes forums, blogs, and test pages. 3rd-party providers also include any dynamic source of product information that is not maintained by vendors or producers, such as search engines. Thus 3rd-parties offer content of varying structure and quality.

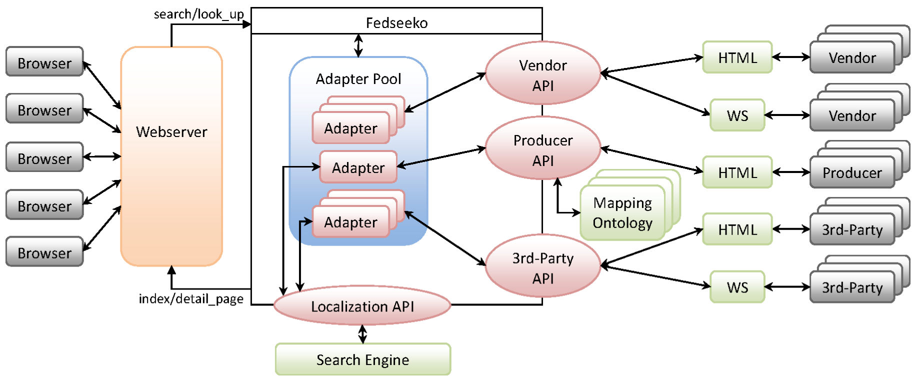


Fig. 1. Architecture of Fedseeko

Consumers access the system through a browser mostly asking for a product list or a product detail page. These requests are executed using the "search" and the "look_up" method respectively. The search method allows providing appropriate parameters, such as a search string, a category, a sort type, etc. The look_up method depends on a corresponding product ID that must be delivered to find the product in the current vendor catalogue. For executing these methods Fedseeko queries one or more adapters that translate the query into a request, which can be understood by the respective vendor.

Using the vendor API the request is sent. When the vendor response arrives, the information is mapped to an internal model and presented to the user. Fedseeko then automatically tries to enrich the vendor information with producer and 3rd-party information. Therefore the Localization API finds out product pages on producer websites using search engines. The information from these pages is extracted, mapped to an internal model using mapping ontologies and added to the user's information base. All tied 3rd-party sources are also queried and available information is presented to the user as well.

The most convenient way for retrieving information is querying Web Services provided by the information sources. As not all sources offer a Web Service, Fedseeko alternatively uses a web scraping wrapper to extract vendor, producer and 3rd-party information. The following chapters will explain the query mechanisms in more detail.

3.1 Integration of Vendor Information

The vendor adapters have to define the before mentioned methods "search" and "look_up" to be included into the system. Then, Fedseeko automatically realizes the integration of a new source and extends the user interface with additional tabs to make the new vendor information accessible. Thus, adding a vendor that provides its product information through a Web Service is made as comfortable as possible, as the adapter is the only thing that has to be created by a user.

An example for a substantial Web Service providing vendor information is the Amazon Associates Web Service (AAWS) [11]. Fedseeko includes the AAWS by providing an adapter that integrates the information through mapping mechanisms allowing the consistent access to product information by controller and model.

Web Scraping Wrapper. For all sources that do not offer Web Services, like Conrad.com or ELV.com, a powerful wrapper was designed that allows Fedseeko to access online malls through web scraping mechanisms. A general purpose adapter for querying these vendor sources offers the functionalities described above. Different parameter values of this special adapter allow the dynamic integration of scraped vendor sources into Fedseeko. These values are saved to the database and consist of details like the structure of the online mall's URL, the parameters available for the product information source and where to add those parameters to the source URL. The database also holds information about the structure of the result page, which is described by regular expressions, that are better suitable for scraping online malls than XPath-queries, as the HTML-code of online malls is not always 100% clean.

Every user should be able to extend the system's vendor source pool by adding additional vendor descriptions. As the users cannot be expected to be well versed in the exposure to URL structures and regular expressions, the web interface offers an easy modality to describe the layout of new vendor URLs and result pages. The process is presented in the following.

To create a generic URL for querying a particular vendor, the user has to provide four different values. The first two values consist of URLs generated by the vendor when querying the corresponding web page for a product with a product name consisting of two words, e.g. "ipod nano". The provided URLs must differ in the page number, e.g. page one and two. The other two values are the used query words, in this case "ipod" and "nano". Then, by comparing the provided URLs and query words, Fedseeko is able to generate a generic request URL with special flags at all points of interest, being the page spot (where to add the page number), the product spot (where to add the product query) and the separator (used for separating query words in the URL). This generic URL is saved to the database for later use.

The result page is described by regular expressions. To create the regular expressions, the user has to provide some attribute values. These values consist of the attribute name (e.g. "price"), a unique string in front of the attribute value (e.g. "<td class=\"imageColumn\"

width=\"123\">") and a unique string behind the attribute value (e.g. "<span class=\"aliasName\">"). Additionally the block containing the total set of attributes for one result product has to be described in the same way. Holding these values, Fedseeko creates a set of regular expressions that enable the system to extract all product information from result pages as long as the vendor's layout is not changed. However, studies showed that the description of vendor layouts by regular expressions makes the system more resistant to layout variations than for instance XPath queries would.

3.2 Integration of Producer Information

After fetching the vendor information from an arbitrary information source, the product can be presented to the user. As the vendor information only consists of few details, Fedseeko is able to enrich this data with details from producer websites.

For dynamically locating additional information sources like product information pages on the websites of corresponding producers, the following algorithm was implemented in the Localization API:

```
producer_page = query(producer_name + " site:com")[0].root_domain
product_page = nil
while(!product_page) do
  product_page = query(product_name + " site:" + producer_page)[0]
  product_name.vary
end
```

In the first step a search engine is queried for the producer's website by using its name, which is known from the vendor information, and a restriction to the domain "com". For instance, if searching for a Nikon camera, the query would look like "Nikon site:com". The first result in this case is "http://www.nikonusa.com". The system then removes all parts of the URL except the root domain, resulting in "nikonusa.com". In the second step the search engine is queried for the product name using the previously retrieved producer website to make a hard restriction on the search results. For instance a query for a Nikon camera could look like this: "Nikon D40 6.1MP Digital SLR Camera Kit with 18-55mm site:nikonusa.com". By this means the system finds the product website among all the producer's websites. As the product title may not be spelled correctly, Fedseeko tries out different variations of the title elements until it discovers the right page. The producer homepage as well as the product presentation page from the producer are delivered to the user interface.

Now the producer adapter tries to extract product information from a list or table in the product page using the scRUBYt!-API [12]. For being able to do this, corresponding XPath-queries are required. The query set consists of an absolute base query for locating the table in the page and several relative queries for each column of the table. Most producers present their products in a uniform manner. Thus a set of queries is only required once per producer. To generate the XPaths, Fedseeko demands one set of example data for a random product of this producer from the user. For example, if the user examines a Nikon camera, he follows the product page link provided by Fedseeko to go to Nikon's detail page. There he extracts the items "Image Sensor

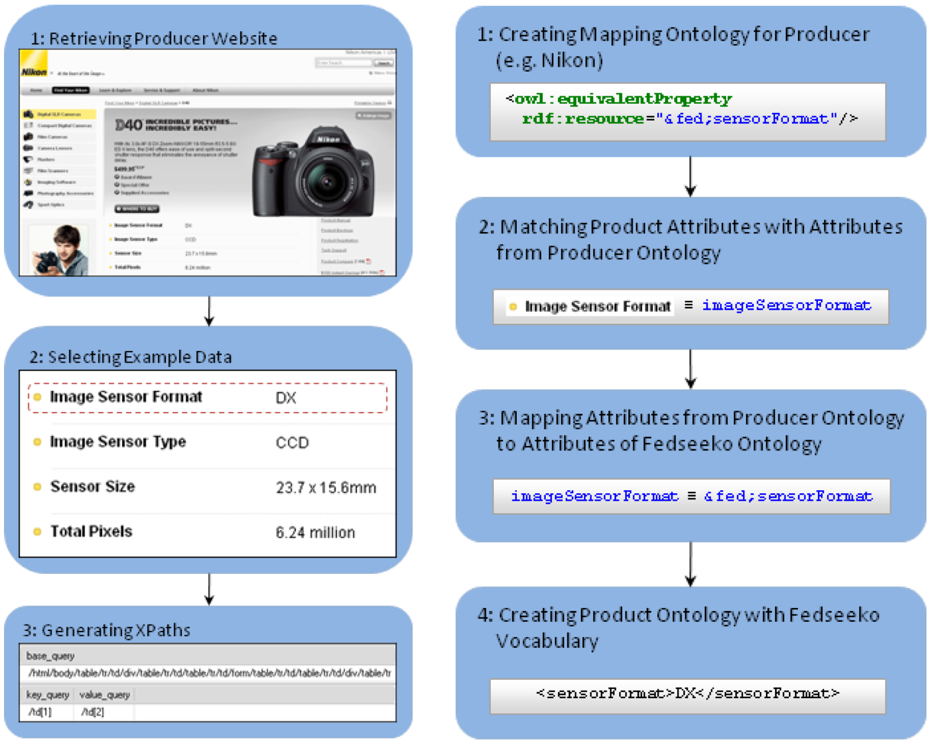

1: Retrieving Producer Website
2: Selecting Example Data
3: Generating XPaths
1: Creating Mapping Ontology for Producer (e.g. Nikon)
2: Matching Product Attributes with Attributes from Producer Ontology
3: Mapping Attributes from Producer Ontology to Attributes of Fedseeko Ontology
4: Creating Product Ontology with Fedseeko Vocabulary

Image Sensor Format	DX
Image Sensor Type	CCD
Sensor Size	23.7 x 15.6mm
Total Pixels	6.24 million

base_query	
/html/body/table/tr/td/div/table/tr/td/table/tr/td/form/table/tr/td/table/tr/td/div/table/tr	
key_query	**value_query**
/td[1]	/td[2]

```
<owl:equivalentProperty
  rdf:resource="&fed;sensorFormat"/>
```

```
Image Sensor Format ≡ imageSensorFormat
```

```
imageSensorFormat ≡ &fed;sensorFormat
```

```
<sensorFormat>DX</sensorFormat>
```

Fig. 2. Extracting and mapping product information

Format" and "DX" from the website and posts this information together with the address of the producer's mapping ontology into a form provided by Fedseeko. Then the system analyzes the page and finds the provided strings.

To avoid problems of different encodings between example data in the website and Fedseeko, the Levenshtein distance [13] is used to compare the extracted data with the different elements of the examined website. After the example data set was found, Fedseeko calculates the corresponding set of XPath-queries and saves them to the local database.

The next time a user investigates a product from Nikon, Fedseeko automatically finds the table information and extracts it. If no table is found (structured product information is often presented in an extra tab of the product page) the system is able to follow some links from the found product page to increase its hit rate. The procedure is displayed on the left side of Figure 2.

For having a consistent view on all product features from different producers, a mapping ontology is used to map Nikon's terminology to Fedseeko's terminology. The ontology holding this mapping information can reside anywhere on the Internet, preferably on the respective producer website. It consists of a taxonomy of product types with corresponding attribute names. Every attribute is extended with a "owl:equivalentProperty"-clause that describes how to map found attribute names to the internal terminology defined by Fedseeko's product ontology. After providing the address of the mapping ontology to the program, Fedseeko translates the producer's attributes into ontology-compliant terms. In a third step the attributes are translated into Fedseeko's terminology and finally used for creating an ontology containing all extracted product information in a consistent format. The described algorithm is shown on the right side of Figure 2. The created product ontology is stored in a public folder to be used by external information systems such as search engines. It is also analyzed by Fedseeko to include the generated product information in the web interface (Figure 3).

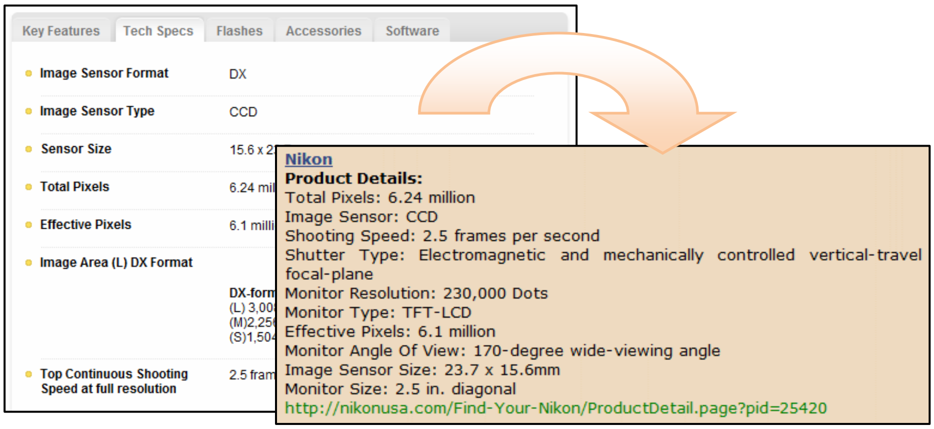


Image Sensor Format	DX
Image Sensor Type	CCD
Sensor Size	15.6 x 2
Total Pixels	6.24 mil
Effective Pixels	6.1 milli
Image Area (L) DX Format	DX-form (L) 3,00 (M)2,25 (S)1,50
Top Continuous Shooting Speed at full resolution	2.5 fram

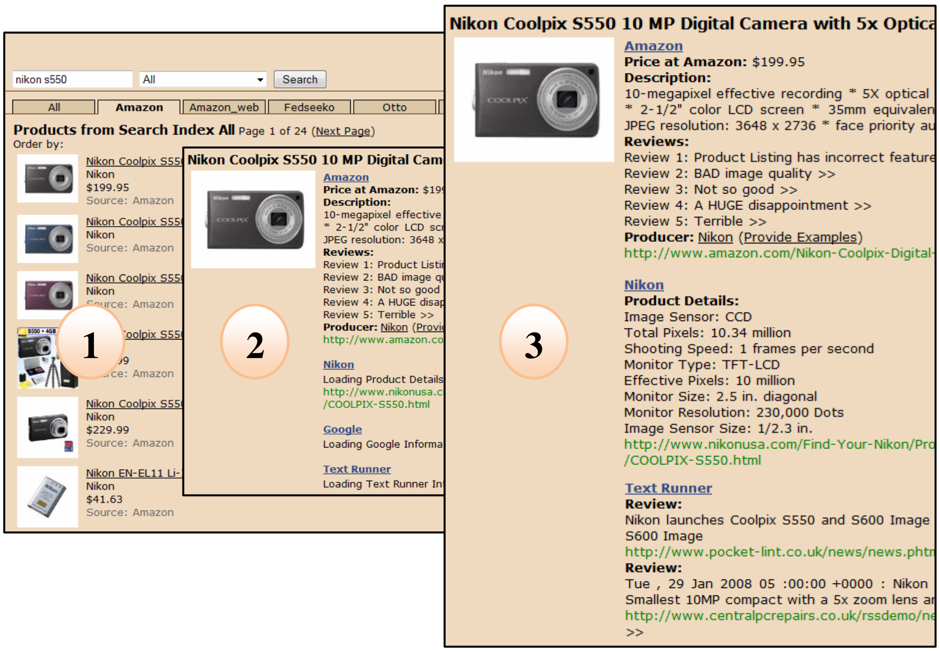
Nikon
Product Details:
Total Pixels: 6.24 million
Image Sensor: CCD
Shooting Speed: 2.5 frames per second
Shutter Type: Electromagnetic and mechanically controlled vertical-travel focal-plane
Monitor Resolution: 230,000 Dots
Monitor Type: TFT-LCD
Effective Pixels: 6.1 million
Monitor Angle Of View: 170-degree wide-viewing angle
Image Sensor Size: 23.7 x 15.6mm
Monitor Size: 2.5 in. diagonal
http://nikonusa.com/Find-Your-Nikon/ProductDetail.page?pid=25420

Fig. 3. Product information on nikonusa.com and Fedseeko

3.3 Integration of 3rd-Party Information

The 3rd-party API enables Fedseeko to scrape static websites as well as querying and scraping dynamic websites. Similar to the adapter pattern for vendor information providers, a 3rd-party information adapter can be added for every source to query. The adapter has to provide a method called "query" that accepts a product name, a producer name and a category name, which then can be used in the adapter's sole discretion. The adapter should finally return a list of hashes with the query results that is provided to the view. The view tries to extract a hash value with the key "url" for every list element, which is visually put in the end of each data set. All other hash keys are used as titles for their corresponding values. 3rd-party information adapters can make use of the web form poster, which is a component to automatize the retrieval of information from dynamic web pages. Like the producer information adapter, the web form poster uses scRUBYt! to enable efficient web scraping.

As an example, TextRunner [14] was tied to Fedseeko. TextRunner is a facts-based search engine that is able to deliver assertions stated on different websites all over the Web. These assertions often belong to field reports submitted by users that already possess the product of interest. Thus, Fedseeko uses the adapter to provide the current product's name to TextRunner, which then generates a corresponding list of assertions related to the product. The assertions as well as their source URLs are extracted



Products from Search Index All Page 1 of 24 (Next Page)
Order by:

Nikon Coolpix S55
Nikon
$199.95
Source: Amazon

Nikon Coolpix S55
Nikon
Source: Amazon

Nikon Coolpix S55
Nikon
Source: Amazon

oolpix S55
9
ce: Amazon

Nikon Coolpix S55
Nikon
$229.99
Source: Amazon

Nikon EN-EL11 Li-
Nikon
$41.63
Source: Amazon

Nikon Coolpix S550 10 MP Digital Camera with 5x Optica

Amazon
Price at Amazon: $199.95
Description:
10-megapixel effective recording * 5X optical
* 2-1/2" color LCD screen * 35mm equivalen
JPEG resolution: 3648 x 2736 * face priority au
Reviews:
Review 1: Product Listing has incorrect feature
Review 2: BAD image quality >>
Review 3: Not so good >>
Review 4: A HUGE disappointment >>
Review 5: Terrible >>
Producer: Nikon (Provide Examples)
http://www.amazon.com/Nikon-Coolpix-Digital-

Nikon
Loading Product Details
http://www.nikonusa.c
/COOLPIX-S550.html

Google
Loading Google Informa

Text Runner
Loading Text Runner In

Nikon
Product Details:
Image Sensor: CCD
Total Pixels: 10.34 million
Shooting Speed: 1 frames per second
Monitor Type: TFT-LCD
Effective Pixels: 10 million
Monitor Size: 2.5 in. diagonal
Monitor Resolution: 230,000 Dots
Image Sensor Size: 1/2.3 in.
http://www.nikonusa.com/Find-Your-Nikon/Pro
/COOLPIX-S550.html

Text Runner
Review:
Nikon launches Coolpix S550 and S600 Image
S600 Image
http://www.pocket-lint.co.uk/news/news.phtn
Review:
Tue , 29 Jan 2008 05 :00:00 +0000 : Nikon
Smallest 10MP compact with a 5x zoom lens ar
http://www.centralpcrepairs.co.uk/rssdemo/ne
>>

Fig. 4. Screenshot of Fedseeko

and put into the results list. As the TextRunner adapter defines that every contained hash has the two elements “review” and “url”, one data set presented in the view consists of a description and a specifically marked source URL (Figure 4).

3.4 Incremental Page Reproduction

Fedseeko was implemented as a prototype providing all features described in this paper. A screenshot of the current system is shown in Figure 4. The figure demonstrates that the different steps of information consolidation are noticeable to the user, because intermediate data is directly inserted into the web interface using Ajax technologies. This way a consumer does not need to wait for the completion of all tasks before examining the additional information.

In Figure 4 five vendor adapters are plugged into Fedseeko, which caused the system to generate five tabs for accessing each vendor respectively. Additionally the all-tab offers the possibility to query all vendor information providers simultaneously. The final product detail page shows producer information, which was extracted from Nikon’s website using the generated XPath queries. Below are additional information snippets from 3rd-party sources. Like the vendor adapters, Fedseeko automatically detected all 3rd-party adapters and queried them for product information.

4 Evaluation

To get an idea of the system performance, the success rate of the information extraction from producer websites was evaluated (Table 2). To generate significant test results, Fedseeko was used to query over one hundred products from the Amazon catalogue. A gold standard was created for the whole product set to be able to benchmark the results generated by the system. The gold standard consisted of sets each containing a product name, the corresponding producer name, the address of the producer's website, the websites presenting the product (if available) and a flag describing the existence of a semi-structured element on this website presenting product details. For instance, if talking about digital cameras, a gold standard set consists of the product name "D40", the producer name "Nikon", Nikon's website "http://www.nikonusa.com", the product detail page address "http://www.nikonusa.com/Find-Your-Nikon/ProductDetail.page?pid=25420" and a flag set to "true", as the page contains a table presenting technical information about the D40. The categories analyzed in this evaluation were "Technical" (e.g. Digital Cameras), "Leisure" (e.g. Bicycles), "Body Care" (e.g. Shampoos) and "Media" (e.g. Books). All information was gathered by hand. Then we queried Fedseeko for each of the products and checked, if the gold standard could be fulfilled. The implemented algorithms showed good results, as 71% of existing product information tables could be found and analyzed. Details of the evaluation are presented below.

Table 2. System performance concerning product information

	a_{page}	$r_{localize}$	a_{table}	$r_{extract}$		r_{total}	$r_{localize}$*$r_{extract}$
Technical	93%	82%	89%	84%		58%	69%
Leisure	83%	76%	68%	94%		41%	71%
Body Care	99%	70%	40%	100%		28%	70%
Media	75%	87%	40%	100%		26%	87%
All	88%	78%	64%	91%		40%	71%

Following data was evaluated: the product page availability a_{page}, which describes how many products have a presentation page; the localization recall $r_{localize}$, which describes how many of the existing product pages were found by Fedseeko; the product table availability a_{table}, which describes how many of the existing product pages own semi-structured information; the extraction recall $r_{extract}$, which describes how many of the existing information tables the system was able to analyze.

For system users it is interesting to know, in how many cases the system is actually able to enhance the product information basis with information extracted from producer pages. Thus, the total recall r_{total} of the extraction procedure was calculated using the following formula:

$$r_{total} = a_{page} * r_{localize} * a_{table} * r_{extract} \, . \tag{1}$$

Obviously technical products are most appropriate for extending their information base with the Fedseeko system (r_{total} = 58%), while only 40% of randomly chosen products can be enriched with producer information. This is mainly rooted in the high amount of available product pages and semi-structured data for products of this kind. Hence the system performance would receive a strong boost, if more producers offered product pages and product information tables. This is proven by the value $r_{localize}*r_{extract}$, which describes the system performance independent from the immutable values a_{page} and a_{table}. Here Fedseeko shows a noticeable performance, as more than $^2/_3$ (71%) of available producer information is found and extracted.

The remaining errors (29%) are caused by the system's algorithms. For example, the product's producer website is not always localized correctly. This especially happens when retrieving web pages of relatively unknown producers or companies with ambiguous names. In this case the product page cannot be localized as well. If the producer's website could be localized correctly, sometimes the localization of the product page still fails, as other websites on the producer's domain may contain the product title and are ranked higher by the queried search engine. Improvements in the localization algorithm would meliorate the success rates considerably as the extraction process already shows excellent results.

Nonetheless the evaluation shows that the overall recall is high enough to offer valuable information to the user. Fedseeko is able to facilitate the creation of an all-embracing view on a product of interest and thus supports the user in taking his buying decision.

5 Conclusions

We have investigated an approach for federating multiple resources of disparately structured types for consolidating product information from all over the Web. Design patterns for integrating vendor information sources of different kinds were shown as well as methods for the dynamic extension of this information by finding and querying information sources from producers and 3rd-parties at runtime to create an all-embracing view for the user. Evaluation showed the success of the approach.

Considering the criteria for an ideal information source mentioned in the beginning of this paper, we have facilitated a noticeable improvement in the field of product information search. The system is not yet delivering complete information about any product of interest, but it collects a high amount of information from different sources. The collected information can be seen as correct in the sense that enough sources are queried to enable the user of the system to compare the retrieved information and filter out conflicts. The information is fresh, as all sources are queried at runtime. Still caching functionalities are envisioned for future versions to allow a better performance when many users access the system simultaneously. The information in its whole is neutral, as information snippets from different sources can be compared with each other. The information is goal-oriented, because no advertisement is included in the view. The comparison of viewed products with automatically retrieved alternative products is not possible yet, as the system needs to be able to identify exact product types for offering this feature. Verification in turn is given, because every information snippet is delivered along with its source URL.

Future works should concentrate on the improvement of the localization algorithm to gain a higher recall concerning the producer information. Additionally, the algorithm for extracting information from product pages should be further automatized. Another important feature would be the personalization of Fedseeko to increase the usability of the system. At the moment an arbitrary amount of information sources can be offered to the user, which could overcharge his receptivity. Instead, the system should suggest some relevant information sources to first-time users and provide the possibility to add and remove information sources in a simple manner for registered users. A fair enhancement would also be the automatic adoption of source suggestions according to recorded usage statistics.

References

1. Fasli, M.: Shopbots: A Syntactic Present, A Semantic Future. IEEE Internet Computing 10(6), 69–75 (2006)
2. Kim, W., Choi, D., Park, S.: Intelligent Product Information Search Framework Based on the Semantic Web. In: 3rd ISWC, Hiroshima, Japan (2004)
3. Wolter, T., Schumacher, J., Matschi, M., et al.: Der shopinfo. XML-Standard. XML-Tage, Berlin (2006)
4. Hepp, M.: The True Complexity of Product Representation in the Semantic Web. In: 14th ECIS, Göteborg, Sweden (2006)
5. Wong, T., Lam, W., Wong, T.: An Unsupervised Framework for Extracting and Normalizing Product Attributes from Multiple Web Sites. In: 31st SIGIR, Singapore City (2008)
6. Lermann, K., Knoblock, C., Minton, S.: Automatic Data Extraction from Lists and Tables in Web Sources. In: IJCAI, Seattle, USA (2001)
7. Cheeseman, P., Stutz, J.: Bayesian Classification (AUTOCLASS): Theory and results. In: Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press, Cambridge (1996)
8. Carrasco, R., Oncina, J.: Learning Stochastic Regular Grammars by Means of a State Merging Method. In: Carrasco, R.C., Oncina, J. (eds.) ICGI 1994. LNCS, vol. 862. Springer, Heidelberg (1994)
9. Brunner, J., Ma, L., Wang, C., et al.: Explorations in the Use if Semantic Web Technologies for Product Information Management. In: WWW, Banff, Canada (2007)
10. Schuster, D., Walther, M., Braun, I.: Towards Federated Product Search From Heterogeneous Sources. In: WWW/Internet, Freiburg, Germany (2008)
11. Amazon: Associates Web Service (2008), `http://www.amazon.com/E-Commerce-Service-AWS-home-page/b?ie=UTF8&node=12738641`
12. Szinek, P.: scRUBYt! - A Simple to Learn and Use, yet Powerful Web Scraping Toolkit Written in Ruby (2008), `http://scrubyt.org`
13. Sulzberger, C.: The Levenshtein-Algorithm (2008), `http://www.levenshtein.net`
14. Banko, M., Cafarella, M., Soderland, S., et al.: Open Information Extraction from the Web. In: IJCAI, Hyderabad, India (2007)