An Adaptive Faceted Search Interface for Structured Product Offers on the Web

Alex Stolz and Martin Hepp

Universitaet der Bundeswehr Munich, D-85579 Neubiberg, Germany
{alex.stolz,martin.hepp}@unibw.de

Abstract. In the past few years, a growing amount of e-commerce information has been published online either as Linked Open Data or embedded as Microdata or RDFa markup inside HTML pages. Unfortunately, the usage of such data for product search and comparison is hampered by the products and services being themselves specific and heterogenous with regard to their relevant characteristics, and by the search process that involves learning about the option space. In this paper, we present an adaptive faceted search interface over product offers in RDF. Our search interface is directly based on the popularity of schema elements in the data and does not rely on a rigid conceptual schema with hard-wired product features, thereby being suitable for arbitrary product domains and product evolution. Further it supports learning during the search process. As a proof of concept of our work, we provide two use cases, namely one with product offers from an automobile database, and a second one with real product data collected from the Web.

1 Introduction

Linked Open Data (LOD) has become a popular paradigm for publishing and consuming data on the Web. In the last few years, there has been a complementing trend towards publishing structured e-commerce data as RDFa and Microdata markup embedded in HTML Web pages. Primarily based on the GoodRelations [1] and schema.org[1] vocabularies, this markup can be easily converted into RDF and combined with LOD datasets, which forms a promising data source for novel Web applications and services.

Unfortunately, the usage of such data for product search and comparison remains an open challenge. Because products and services are specific and heterogeneous with regard to their relevant characteristics, the means for exploring the giant RDF graph of online product data are limited. On one hand, the search space is huge because of the multi-dimensionality of products, and on the other hand, the graph of product information is in many branches sparsely populated due to the limited adoption of such data publication practices. For users it is thus very difficult to formulate queries without knowing how well conceptual elements from the schemas are populated and how much they influence the size and characteristics of the result set.

[1] http://schema.org/

Each search process involves a learning effect about the option space [2, p. 9], which should be accounted for by search interfaces. Traditional search models, however, fail to support exploratory searches within the product space: Keyword searches flatten multi-dimensional product descriptions to simple, one-dimensional term matches; query formulation with SPARQL is too complex for the average user and lacks mediation between the conceptual models of the data and the mental models of human users; and other approaches suggested for browsing RDF data (e.g. Tabulator [3]) are very low-level and hence hardly suitable for serious product search.

In this paper, we propose an instance-driven, adaptive faceted search interface to support the incremental nature and the learning aspect of product search over structured e-commerce data on the Web. In a prior publication, we have already evaluated the usability of this kind of search interface with regard to e-commerce [4]. The present work describes the single components of the faceted search interface (i.e. the architecture and implementation of the user interface), and details about the iterative, incremental search strategy applied to the product domain. In this respect, we also introduce a novel, instance-based search filtering approach and highlight the role of user feedback in RDF environments. We then showcase our approach using (a) a homogeneous dataset derived from the automotive domain, and (b) some real e-commerce data that we have collected from the Web of Data.

The rest of this paper is structured as follows: Section 2 describes our faceted search interface over structured product offers; in Sect. 3, we present two use cases; and finally, Sect. 4 concludes our work and discusses future directions.

2 Adaptive Faceted Search Interface for Product Offers

In the following, we describe an adaptive faceted search interface for product offers over RDF data.

2.1 Overview of Faceted Search

Faceted search [5], a special form of exploratory search [6], is a well-established interaction paradigm both recognized in industry (e.g. eBay[2] and Amazon[3]) and academia for guiding users through option spaces [7]. While the term faceted search is sometimes equated with faceted browsing or faceted navigation (e.g. [7] or [2, p. 95]), it is often understood as a combination of faceted navigation and keyword search functionality [5, p. 24]. Substantial research related to faceted search has also been done in the field of *dynamic taxonomies* [8,9].

A faceted search interface is based on facet-value pairs. Facets can be compared to mutually orthogonal categories, whereas facet values (or terms) are instances of these categories. In the context of product search, facets represent product dimensions (e.g. the features "color" and "material"), whereas facet

[2] http://www.ebay.com/

[3] http://www.amazon.com/

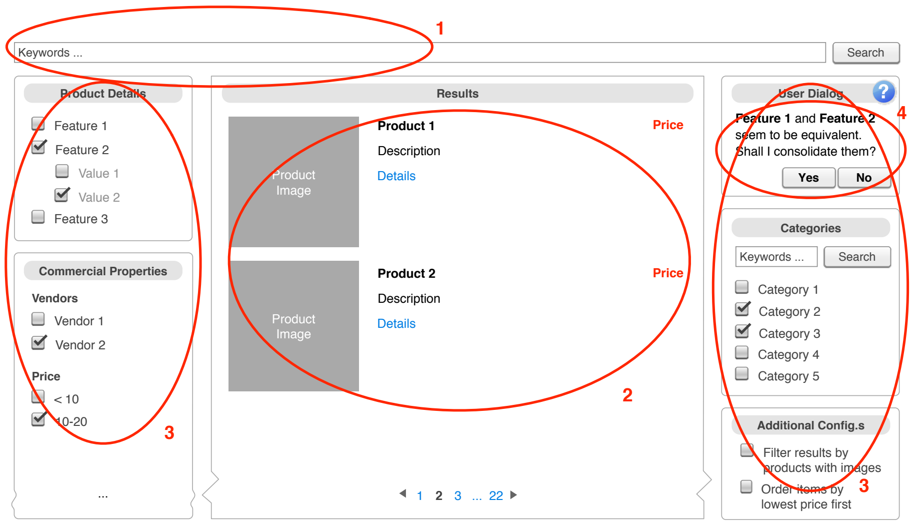


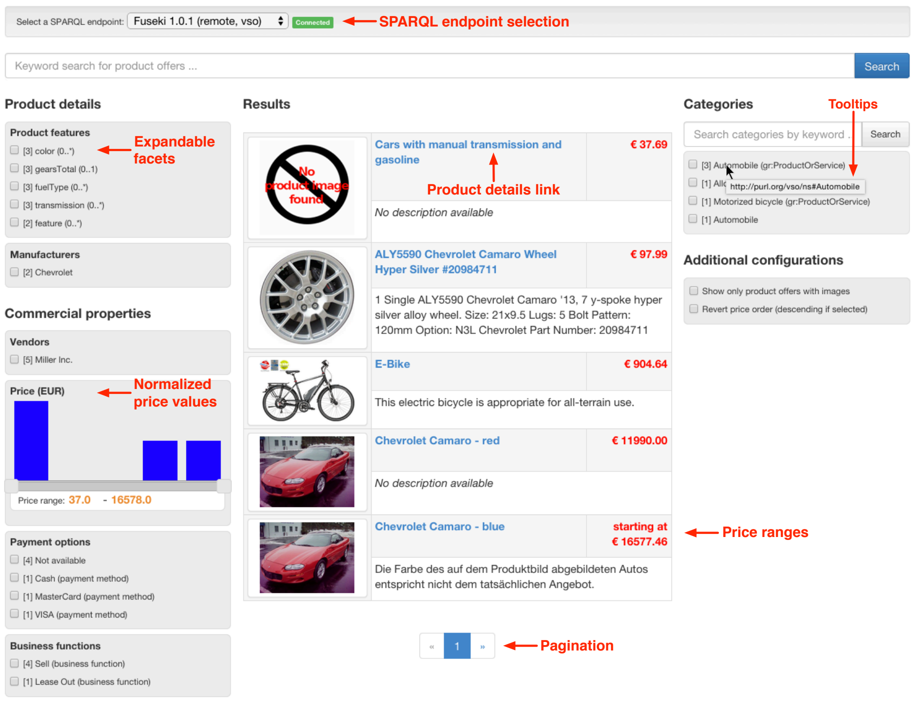
Fig. 1. Mock-up of a faceted search interface for e-commerce

values correspond to instances of these dimensions (e.g. “brown” and “wood”). Products are usually represented by multiple facets and facet values. The selection within facets of faceted search interfaces maps to boolean expressions for the filtering of the option space. While a choice of multiple facets generally leads to their conjunction (e.g. “color” is “brown” *and* “material” is “wood”), multiple facet values are mostly combined using disjunction (e.g. “color” is “brown” *or* “red”) (cf. [10]). In set-theoretic terms, conjunction corresponds to the intersection of items, and disjunction to their union.

Faceted search interfaces dynamically adapt to user interaction. In other words, the facet views update on reducing or expanding the option space. Furthermore, the faceted navigation paradigm does normally not lead to dead ends [11], i.e. an empty result set, because a user is only shown facets for which instance data exists. Unlike parametric searches, where a user is forced into a sequential search order (e.g. first choose the camera type, then the focal length, after that the picture resolution, and finally the color), faceted search allows users to drill down the search space in any preferred order that best suits their individual information needs. Adaptive faceted search interfaces further adjust themselves (i.e. their structure) to the underlying data, sometimes leading to a repeated, dynamic reorganization of the filtering options.

2.2 Faceted Search User Interface

In Fig. 1, we show a sketch of a general faceted search interface over e-commerce data. The faceted search interface combines the two interaction paradigms keyword search and faceted navigation. As illustrated in the graphic, the keyword



Select a SPARQL endpoint: Fuseki 1.0.1 (remote, vso)

Keyword search for product offers ...

Product details

Product features
- [3] color (0..*)
- [3] gearsTotal (0..1)
- [3] fuelType (0..*)
- [3] transmission (0..*)
- [2] feature (0..*)

Manufacturers
- [2] Chevrolet

Commercial properties

Vendors
- [5] Miller Inc.

Price (EUR)

Price range: 37.0 - 16578.0

Payment options
- [4] Not available
- [1] Cash (payment method)
- [1] MasterCard (payment method)
- [1] VISA (payment method)

Business functions
- [4] Sell (business function)
- [1] Lease Out (business function)

Results

Cars with manual transmission and gasoline	€ 37.69	No description available
ALY5590 Chevrolet Camaro Wheel Hyper Silver #20984711	€ 97.99	1 Single ALY5590 Chevrolet Camaro '13, 7 y-spoke hyper silver alloy wheel. Size: 21x9.5 Lugs: 5 Bolt Pattern: 120mm Option: N3L Chevrolet Part Number: 20984711
E-Bike	€ 904.64	This electric bicycle is appropriate for all-terrain use.
Chevrolet Camaro - red	€ 11990.00	No description available
Chevrolet Camaro - blue	starting at € 16577.46	Die Farbe des auf dem Produktbild abgebildeten Autos entspricht nicht dem tatsächlichen Angebot.

Categories

Search categories by keyword ...
- [3] Automobile (gr:ProductOrService)
- [1] Allc
- [1] Motorized bicycle (gr:ProductOrService)
- [1] Automobile

http://purl.org/vso/ns#Automobile

Additional configurations
- Show only product offers with images
- Revert price order (descending if selected)

Fig. 2. Screenshot of our faceted product search interface

search field (1) is placed prominently at the top of the search interface, and the boxes surrounding the result list (2) at the center represent the faceted navigation controls (3). The "User Dialog" box (4), displayed on the upper right corner, can further provide a means to unobtrusively incorporate user feedback.

The user interface layout in Fig. 1 served as the baseline for developing the faceted search interface depicted in Fig. 2. It effectively integrates product details, product category information, and commercial properties related to product offers. The results displayed in the provided screenshot are based on toy data modeled using the Vehicle Sales Ontology (VSO)[4]. The most interesting facts such as product image, name, description, and price are summarized in the result list. Nonetheless, for every item in the result list, a link is provided that, when clicked, opens a window where the full product details are shown. If the size of the result set exceeds ten items, then the remaining results are outsourced to other result pages that can be accessed via pagination controls [2, pp. 110-116]. This shall prevent the user from possible information overload. The part left to the result list is dedicated to product-related details and commercial properties. It mainly includes product features and prices, but also manufacturers, vendors, payment options, or business functions. The right part features a category fil-

[4] http://purl.org/vso/

ter, along with additional filter configurations such as to exclude product offers without images or to revert the result order. The search interface further makes heavy use of tooltips that help users at better understanding the option space.

Our search interface is instance-driven and schema-agnostic with respect to the generation of the facet views. Instead of relying on a fixed schema, the data is the first-class citizen that specifies the appearance of the user interface. In other words, the search interface is directly based on the constraints imposed on the underlying RDF data and dynamically adapts with the availability of the data, such as the presence of product features or categories. Accordingly, our approach is able to flexibly cope with structured product data from diverse data sources, as long as it complies with the e-commerce meta-model implied by the GoodRelations vocabulary [1]. This allows for useful guided navigation paths even in the absence of richly axiomatized products.

Keyword Search. We incorporated two forms of keyword searches into our faceted search prototype: A first one for product offers, and a second one for searching within product categories. The keyword searches match terms within textual properties attached to objects, which includes the names and descriptions found in product offers, instances, and models, and the labels and comments for product categories. An autocomplete feature helps the user with term suggestions. It is based on a light-weight SPARQL query executed over the names and labels of products and product categories, respectively.

Simple keyword search functionality can be obtained using the SPARQL CONTAINS function [12], even though such queries are typically very costly for large datasets, as most implementations iterate over all relevant objects. Some SPARQL endpoints thus permit operations over optimized full-text indexes built from textual properties in the available data. Such complementary full-text search engines like Lucene [13] support, in addition to exact term matches, wildcard queries or fuzzy string matches based on given threshold limits [13, pp. 99-101]. Our prototype relies on Lucene, as long as it is supported by the underlying SPARQL endpoint. Otherwise, we fall back to the much slower SPARQL CONTAINS function.

Faceted Navigation. The exploratory search capability is provided through the faceted navigation controls that complement the keyword searches. Faceted navigation uses boolean constraint filtering based on product dimensions, commercial properties of product offers, and product type information.

As the *product features* facet view in Fig. 2 suggests, product features are initially displayed in a compact form and expanded to their corresponding values as the user clicks on them. This mechanism is illustrated in Fig. 3. A selection of multiple product features leads to their conjunction (logical *and*), i.e. items to appear in the result list need to match every selected feature. By contrast, a disjunctive approach is used (logical *or*) among several facet values. For example, imagine that a user adds a second payment option to a payment method that

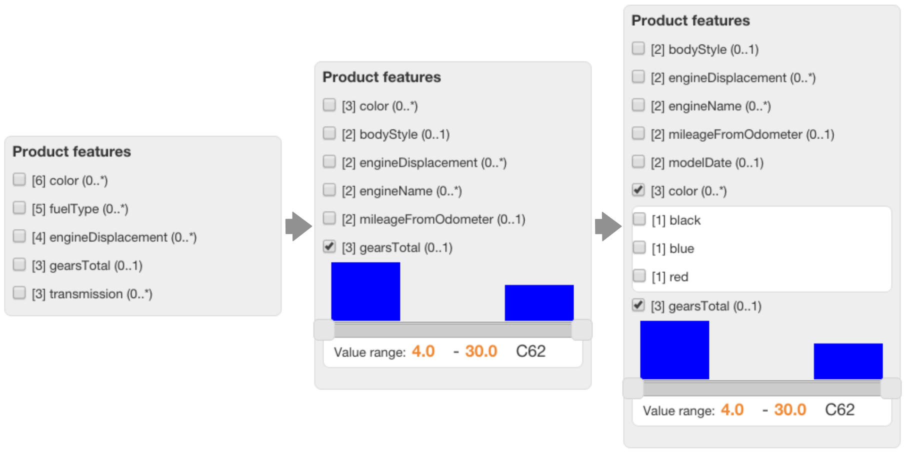


Fig. 3. Iteratively expanded navigation controls

was already selected before. Matching candidate offers then have to accept at least one of the two selected payment options.

The possible facet values correspond to qualitative, quantitative, and datatype properties in OWL. Unlike qualitative or datatype values that may be implemented with checkboxes that a user can click on, quantitative values can cover an indefinite range for which checkboxes would be suboptimal. A popular technique is to group quantitative values into discrete classes given as range intervals (e.g. “$ 0-20”, “$ 20-50”, etc.). Our approach even uses a range slider as demonstrated on the price view in Fig. 2 and the total of gears facet in Fig. 3, respectively. To build up the range slider, we need to generate a useful number of classes with each having the same width. In order to obtain a proper class width, the interval between the minimum and maximum value is divided by the number of classes, which is essentially the instance count with a specified upper limit (e.g. a maximum of 30 classes). The height of the single bars is calculated relative to the class with the highest frequency of instances. The scale of the bars’ height is logarithmic with a fixed maximum height. Thanks to the frequency of values indicated for every class, a user is able to quickly gauge the possible outcomes of his filtering decisions on the result set. While this approach works well for point values, we need to rely on a heuristic for range intervals: For closed intervals, we use the lower boundary for the classifying of numeric values into classes, whereas for open intervals, we take the actual value that is available.

2.3 Implementation

From a conceptual point of view, the front-end of the application is backed by data from an RDF store accessible through a SPARQL endpoint, which can be configured as per the endpoint selection dropdown menu in Fig. 2.

The front-end of the application was realized using HTML 5 for static content, JavaScript and JQuery for user interaction, and Twitter Bootstrap for page responsiveness. Wherever possible, human-readable labels are shown in the user interface instead of long and cryptic URIs for classes, instances, and properties. Provided that they have been loaded into the RDF store before, labels can be extracted from product vocabularies and instance data. The back-end is based on the Python programming language and the Jinja 2 templating engine for generating custom-tailored SPARQL queries, that are then submitted to the SPARQL endpoint. The application can be run on any general-purpose Apache Web server instance where the *mod_python* and *mod_wsgi* modules are active. If supported by the SPARQL endpoint, keyword searches execute over a full-text search engine such as Lucene [13].

Every facet view of the user interface is generated by executing its own, unique SPARQL SELECT query, which has to take into account the current set of constraints. It was not feasible to formulate one single, comprehensive query that would encompass the data to create all facet views. First, due to the fact that each facet view uses different aggregate functions over the data (e.g. group results by manufacturers vs. vendors). And second, because large and complex queries are very costly. Facet-value pairs of faceted search interfaces are represented in RDF by properties and instances (or values). Similarly, conjunction of facets and disjunction of facet values within a facet are realized in SPARQL using a sequence of triple patterns and UNION statements, respectively. In Listing 1.1, we outline a simplified SPARQL SELECT query to generate the facet view for product prices. Let us suppose that a user has chosen to select products that are manufactured either by Apple Inc. (e.g. URI `http://www.apple.com/#company`) or by Microsoft Corporation (e.g. URI `http://www.microsoft.com/#company`), i.e. he has checked these both organizations in the facet view for manufacturers. The resulting filter constraint is a UNION clause in SPARQL. Please note also that the number of affected instances is counted in the projection part of the query, which is further displayed to the user through the search interface.

As of writing this paper, we had successfully tested our prototype with three different SPARQL endpoints compliant with the W3C SPARQL 1.1 standard, namely Virtuoso Open Source Edition[5], the Jena-TDB-based Fuseki SPARQL server[6], and Stardog[7].

2.4 Incremental Search Strategy

We understand product search as an iterative process (cf. [4]). On every search step, the user is presented with dynamic facet and result views that represent the search space pruned according to the current selection. Product search supposedly ends as the user is satisfied with the results or not able to find a path leading him to better search results. Until then, the search process may move across multiple search paradigms. Based on the quality of the results, several

[5] `http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/`
[6] `http://jena.apache.org/documentation/serving_data/`
[7] `http://stardog.com/`

```
PREFIX gr: <http://purl.org/goodrelations/v1#>

SELECT (COUNT(DISTINCT ?offer) AS ?cnt) ?pspec ?price ?code
WHERE {
  # price-specific graph pattern
  ?offer a gr:Offering ;
    gr:includes ?product ;
    gr:hasPriceSpecification ?pspec .
  ?pspec a gr:UnitPriceSpecification ;
    gr:hasCurrencyValue ?price ;
    gr:hasCurrency ?code .
  # the next part of the graph pattern is common to all facet views
  ## filter constraint on manufacturer
  { ?product gr:hasManufacturer <http://www.apple.com/#company> } UNION
  { ?product gr:hasManufacturer <http://www.microsoft.com/#company> }
  ## additional filter constraints ...
}
GROUP BY ?pspec ?price ?code
ORDER BY ?price
```

Listing 1.1. Simplified SPARQL SELECT query to generate the facet view for prices with a constraint on the manufacturer

iterations with possibly different search paradigms might be necessary. As illustrated in Fig. 4, the user might start with keyword search, faceted navigation, or directly head to the product details page of an item from the initial result set. The search interface may also accept user feedback that is incorporated into the graph with product data, as we will explain in Sect. 2.6.

The task of presenting optimal facets to users is a key challenge towards increasing the search efficiency in an incremental search. A user would ideally be presented, in every search step, the options (facets and facet values) that yield the best possible partition of the search space (and create the highest utility to the user). At the time of writing this paper, the splitting strategy employed for our prototype was relying on presenting to the user the five most frequent facets and facet values in the data (cf. [14]). Nevertheless, the authors are aware of several weaknesses that this methodology is suffering from. E.g., if all or only very few items exhibit a certain feature or feature-value pair, then the current algorithm tends to over- or underrate their value for the search progress. Similarly, the algorithm is misguided as multiple instances of the same feature belong to a single item. Superior approaches have been suggested in literature, e.g. [14] mention some popular facet-pair suggestions strategies, namely relying on frequency, probability, and the information gain. The authors in [15] further give an overview over different metrics appropriate for product search to help decide which facets shall be presented to the user. Investigating these alternative presentation strategies is planned as future work for our prototype.

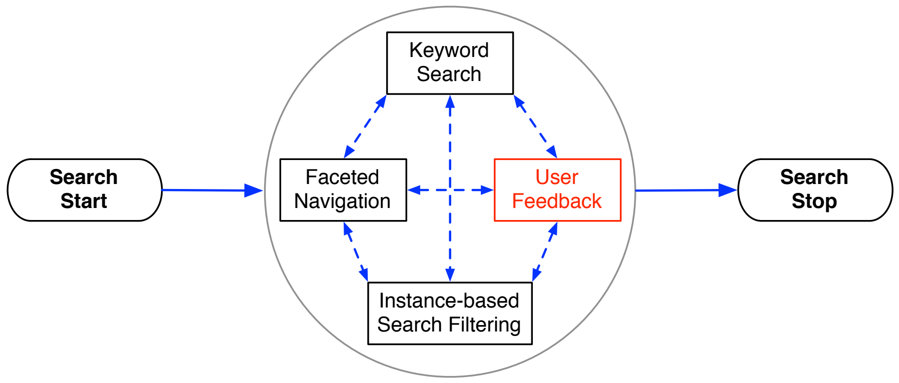


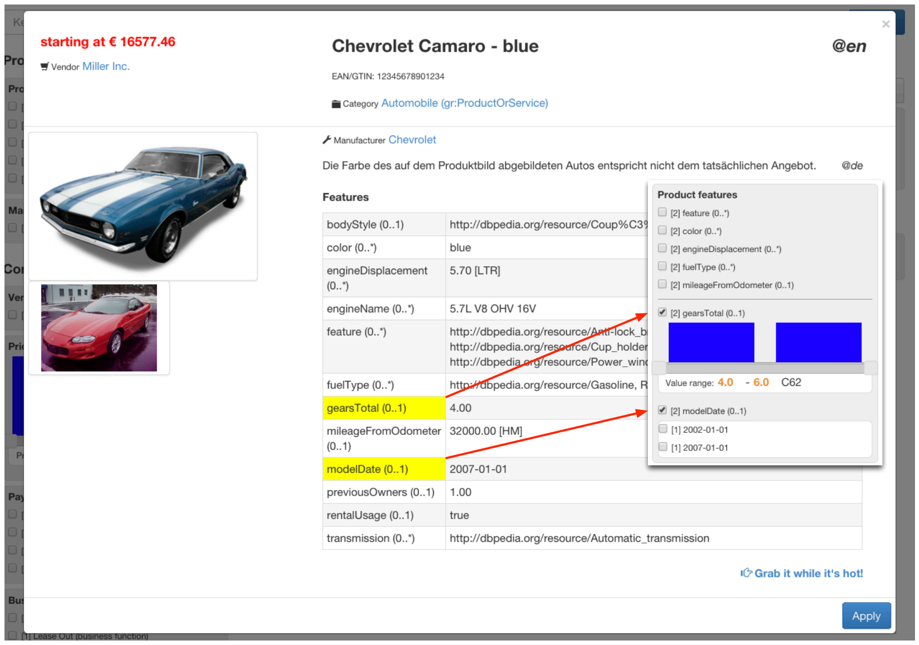
Fig. 4. Iterative product search across multiple search paradigms

2.5 Instance-based Search Filtering

As a complement to faceted search filters over aggregated data, we herein present a novel idea for supporting the incremental search process, namely *instance-based search filtering* (see Fig. 5). Occasionally, a user might be looking at the details of a product offer and discover features that he intends to consider for the next filtering steps in the search process. Now, by going back to the result list, the user would lose track of the respective features and only find them by chance in the list of displayed product features, namely if they are ranked high with respect to the current items in the option space. Of course, the same holds true for feature values and individuals. From a user interaction point of view, we can prevent this level of indirection by letting the user apply filters directly from the product details page. As a nice side effect, this solution facilitates the comparison among similar products via navigating across a collection of items that have some product features in common. Even though not entirely the same, this approach is in a way similar to the pivoting operation in user interfaces (e.g. [16, pp. 83-84], [17]), where a concept can be found through shared features with another concept. In order for this to work well, the properties in the RDF graph need to be consolidated first, which is a research problem on its own that we are not going to further discuss in this paper.

2.6 User Feedback with Conceptual Consolidation

For product search with incremental learning, it is not only vital to assist in navigating and pruning the option space, but also to actively engage the user in the search process, to incorporate user feedback, and to exploit user intelligence. As already depicted in the mock-up in Fig. 1, a viable approach is to integrate a user dialog in the search interface. This approach goes one step beyond the traditional relevance feedback method (e.g. [18, pp. 178-179], [10]), where systems



starting at € 16577.46

Vendor Miller Inc.

Chevrolet Camaro - blue

@en

EAN/GTIN: 12345678901234

Category Automobile (gr:ProductOrService)

Manufacturer Chevrolet

Die Farbe des auf dem Produktbild abgebildeten Autos entspricht nicht dem tatsächlichen Angebot. @de

Features

bodyStyle (0..1)	http://dbpedia.org/resource/Coup%C3%
color (0..*)	blue
engineDisplacement (0..*)	5.70 [LTR]
engineName (0..*)	5.7L V8 OHV 16V
feature (0..*)	http://dbpedia.org/resource/Anti-lock_b http://dbpedia.org/resource/Cup_holder http://dbpedia.org/resource/Power_wind
fuelType (0..*)	http://dbpedia.org/resource/Gasoline, R
gearsTotal (0..1)	4.00
mileageFromOdometer (0..1)	32000.00 [HM]
modelDate (0..1)	2007-01-01
previousOwners (0..1)	1.00
rentalUsage (0..1)	true
transmission (0..*)	http://dbpedia.org/resource/Automatic_transmission

Product features

- [2] feature (0..*)
- [2] color (0..*)
- [2] engineDisplacement (0..*)
- [2] fuelType (0..*)
- [2] mileageFromOdometer (0..1)
- ✓ [2] gearsTotal (0..1)

Value range: 4.0 - 6.0 C62

- ✓ [2] modelDate (0..1)
  - [1] 2002-01-01
  - [1] 2007-01-01

Fig. 5. Product details window with instance-based search filtering

explicitly, or implicitly via user behavior, collect information about the relevance of the results presented to the user.

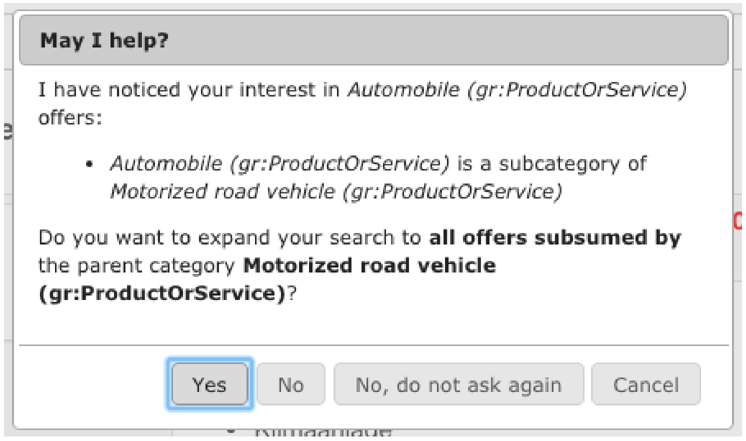
It is outside the scope of this paper to mention here all possibilities of providing explicit user feedback in search systems. Yet, a user interface could ask the user for help in consolidating conceptual correspondences, e.g. to approve a possible match of the two features *gr:hasGTIN-14* and *ex:productId* (possibly by way of presenting an instance for each of the two properties), leading to the following equivalence axiom:

```
gr:hasGTIN-14 owl:equivalentProperty ex:productId .
```

In the current state of our implementation, we accept user feedback in the form of a dialog window that pops up as there exist possibly interesting super-concepts with respect to the concept in regard (see Fig. 6). On this account, a user is able to expand the search scope, e.g. going from an automobile (*vso:Automobile*) to its super-concept motorized road vehicle (*vso:MotorizedRoadVehicle*).

```
:automobile_instance1 a gr:MotorizedRoadVehicle .
:automobile_instance2 a gr:MotorizedRoadVehicle .
```

Technically, this method offers a guided way to materialize inference rules.


May I help?

I have noticed your interest in *Automobile (gr:ProductOrService)* offers:

- *Automobile (gr:ProductOrService)* is a subcategory of *Motorized road vehicle (gr:ProductOrService)*

Do you want to expand your search to **all offers subsumed by** the parent category **Motorized road vehicle (gr:ProductOrService)**?

Fig. 6. User feedback dialog window

In an RDF environment, corresponding axioms can be easily added to the existing data as named RDF graphs – potentially managed on a per-user basis. The named graphs can then persist in the RDF store to reflect past user experiences, or otherwise be cleared before a user starts a new search.

3 Use Cases

During the development of our adaptive faceted search interface, we set up two slightly different use cases. The first one uses product offers and properties derived from a car listing Web site, and the second one is based on real e-commerce data from the Web. Both use cases have been published on the Web in a combined demo application[8]. We further made the respective (raw and random sampled) datasets available online[9]. A thorough user study of our application has already been conducted in a prior publication yielding promising results [4].

3.1 Use Case with Product Data from a Car Listing Web Site

Our first use case serves to illustrate how the search interface works with densely populated product data and a homogeneous dataset, as it is mostly the case for vertical search engines. More precisely, we have crawled a random sample of automobile offers from the car listing Web site *mobile.de* (i.e. as much as we could collect from the visible part of the Web pages). Our random sample originally consisted of 875 automobile offers, but due to overly complex SPARQL queries for the restricted hardware at hand (dual core processor clocked at 2.6 GHz and with 16 MB cache size, 4 GB RAM), we had to reduce it to 25 instances. In

[8] http://www.ebusiness-unibw.org/tools/product-search/
[9] http://www.ebusiness-unibw.org/tools/product-search/datasets/

particular, we selected 50 result pages at random (based on a URL template with a random page number, a lower bound, and an upper bound for the price) with 20 results each, summing up to 1000 automobile offers. Effectively, we could obtain 875 car offers. Of these 875 car offers we again picked a random sample of 25 car offers that we loaded into our demonstrator. Quite clearly, there exist more appropriate technologies (e.g. relational databases) than the Semantic Web to implement this specific use case in terms of efficiency and ease of use, but they complicate data integration (e.g. include information from DBPedia) and oppose schema flexibility during run-time. The most obvious finding in this use case is that due to the high data quality of such a Web site (i.e. almost every car offer is populated exhaustively), drilling down the option space based on feature values works quite well. However, as almost every automobile offer uses the same features, the discriminatory value of product features is low. The present use case can be accessed from the online demo by selecting the SPARQL endpoint with the "automobile dataset".

3.2 Use Case with Real Product Data From the Web

We set up another use case that operates over real e-commerce data from the Web. We collected product offer data by crawling Web shops with household appliances. More precisely, we selected shops from the Rakuten Deutschland platform[10] that were classified into categories related to the general topic *household*. By that we could obtain 23 Web shops, from which 17 shops contained GoodRelations [1] markup. Thanks to earlier research [19], we have already been in the possession of high-quality, structured product model data by BSH[11] in the form of a BMEcat catalog. We used that one to augment the product offer data from the Web crawl with product features to enable deep product comparison. Furthermore, we executed cleansing and consolidation rules on the data such as to convert all price specifications to a common currency "Euros". For the purposes of this paper, we abstract from possible challenges related to this task. The interesting aspect of this use case is the integration of various, heterogeneous data sources in a single faceted search interface for products. Because the product features are mainly coming from a single BMEcat product catalog, the product features are homogeneous as well. However, because the product types are rather diversified (e.g. coffee machine vs. vacuum cleaner), the variety in product features is higher than in the previous use case. A live demonstration of this use case with real product data is available via the provided online tool by selecting the SPARQL endpoint that contains the "household crawl data".

4 Conclusions and Future Work

In this paper, we have proposed a data-driven, adaptive faceted search interface as a means to navigate the giant but often sparse graph of e-commerce data

[10] `http://www.rakuten.de/`, a hosting platform for online shops.

[11] *Bosch und Siemens Haushaltsgeräte GmbH*, a manufacturer specializing on household appliances.

on the Web. Our prototype enables incremental, multi-parametric searches over RDF data based on distinguishing properties and attributes of product items, provides explicit support for user learning about the option space, and allows to enhance the conceptual consolidation during the search process. As the search interface does not rely on a rigid conceptual schema with hard-wired product features, it is suitable for arbitrary product domains and product evolution. We have demonstrated the viability of our approach by setting up two use cases with (a) data derived from a car listing Web site, and (b) real product data collected from the Web.

As future work, we envision to enhance our work by more accurate and context-sensitive user dialogs, personalization and diversification of facet and result views, and in particular to utilize metrics for improving the efficiency of the search process, i.e. to propose features and values that best possible partition the search space and promise the highest utility to a given information need. We will also try to incorporate machine-learning components.

Acknowledgments. The authors would like to thank Florian Ott for the kind support in setting up and hosting the infrastructure of our search prototype.

References

1. Hepp, M.: GoodRelations: An Ontology for Describing Products and Services Offers on the Web. In: Proceedings of the 16th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2008), Acritezza, Italy, Springer Berlin Heidelberg (2008) 329–346
2. Morville, P., Callender, J.: Search Patterns. O'Reilly Media (2010)
3. Berners-Lee, T., Chen, Y., Chilton, L., Connolly, D., Dhanaraj, R., Hollenbach, J., Lerer, A., Sheets, D.: Tabulator: Exploring and Analyzing Linked Data on the Semantic Web. In: Proceedings of the 3rd International Semantic Web User Interaction Workshop (SWUI 2006), Athens, GA, USA (2006)
4. Stolz, A., Hepp, M.: Adaptive Faceted Search for Product Comparison on the Web of Data. In: Proceedings of the 15th International Conference on Web Engineering (ICWE 2015), Rotterdam, The Netherlands, Springer Berlin Heidelberg (2015) 420–429
5. Tunkelang, D.: Faceted Search. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool (2009)
6. Marchionini, G.: Exploratory Search: From Finding to Understanding. Communications of the ACM **49**(4) (2006) 41–46
7. Wei, B., Liu, J., Zheng, Q., Zhang, W., Fu, X., Feng, B.: A Survey of Faceted Search. Journal of Web Engineering **12**(1) (2013) 41–64
8. Sacco, G.M.: The Intelligent E-Store: Easy Interactive Product Selection and Comparison. In: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC 2005), Munich, Germany, IEEE Computer Society (2005) 240–248
9. Sacco, G.M., Tzitzikas, Y.: Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience. Springer Berlin Heidelberg (2009)

10. Hearst, M.A., Elliott, A., English, J., Sinha, R., Searingen, K., Yee, K.P.: Finding the Flow in Web Site Search. Communications of the ACM **45**(9) (2002) 42–49
11. Ferré, S., Hermann, A.: Semantic Search: Reconciling Expressive Querying and Exploratory Search. In: Proceedings of the 10th International Semantic Web Conference (ISWC 2011), Bonn, Germany, Springer Berlin Heidelberg (2011) 177–192
12. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language. http://www.w3.org/TR/2013/REC-sparql11-query-20130321/ (2013)
13. McCandless, M., Hatcher, E., Gospodnetic, O.: Lucene in Action. 2 edn. Manning Publications Co. (2010)
14. Koren, J., Zhang, Y., Liu, X.: Personalized Interactive Faceted Search. In: Proceedings of the 17th International World Wide Web Conference (WWW 2008), Beijing, China, ACM (2008) 477–485
15. Vandic, D., Frasincar, F., Kaymak, U.: Facet Selection Algorithms for Web Product Search. In: Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM 2013), San Francisco, CA, USA, ACM (2013) 2327–2332
16. Stefaner, M., Ferré, S., Perugini, S., Koren, J., Zhang, Y.: User Interface Design. In Sacco, G.M., Tzitzikas, Y., eds.: Dynamic Taxonomies and Faceted Search. Springer Berlin Heidelberg (2009) 75–112
17. Brunetti, J.M., García, R., Auer, S.: From Overview to Facets and Pivoting for Interactive Exploration of Semantic Web Data. International Journal on Semantic Web and Information Systems (IJSWIS) **9**(1) (2013) 1–20
18. Baeza-Yates, R.A., Ribeiro-Neto, B.A.: Modern Information Retrieval: The Concepts and Technology behind Search. 2 edn. Addison-Wesley (2011)
19. Stolz, A., Rodriguez-Castro, B., Hepp, M.: Using BMEcat Catalogs as a Lever for Product Master Data on the Semantic Web. In: Proceedings of the 10th Extended Semantic Web Conference (ESWC 2013), Montpellier, France (2013) 623–638