A benchmark for end-user structured data exploration and search user interfaces

A Benchmark for End-User Structured Data User Interfaces

Roberto García[a,*], Rosa Gil[a], Eirik Bakke[b], David R. Karger[b]

[a] *Computer Science and Engineering Department, Universitat de Lleida, Jaume II 69, 25001 Lleida, Spain*
[b] *Computer Science and Artificial Intelligence Laboratory, MIT, 32 Vassar St., Cambridge, MA, USA*

Abstract

During the years, it has been possible to assess significant improvements in the computational efficiency of Semantic Web search and exploration systems. However, it has been much harder to assess how well different semantic systems' user interfaces help their users. One of the key factors facilitating the advancement of research in a particular field is the ability to compare the performance of different approaches. Though there are many such benchmarks in Semantic Web fields that have experienced significant improvements, this is not the case for Semantic Web user interfaces for data exploration. We propose and demonstrate the use of a benchmark for evaluating such user interfaces, which includes a set of typical user tasks and a well-defined procedure for assigning a measure of performance on those tasks to a semantic system. We have applied the benchmark to four such systems. Moreover, all the required resources to apply the benchmark are openly available online. We intend to initiate a community conversation that will lead to a generally accepted framework for comparing systems and for measuring, and thus encouraging, progress towards better semantic search and exploration tools.

Keywords: benchmark, user experience, usability, semantic data, exploration, relational data

*Corresponding author
Email addresses: rgarcia@diei.udl.cat (Roberto García), rgil@diei.udl.cat (Rosa Gil), ebakke@mit.edu (Eirik Bakke), karger@mit.edu (David R. Karger)

1. Introduction

The amount of semantic data is growing, through open data initiatives like the Linked Open Data Cloud [1] or motivated by SEO benefits, like those provided by major search engines for web pages annotated using schema.org [2]. However, this has not noticeably impacted user applications, for instance by the long-sought Killer App for the Semantic Web [3]. One of the main barriers alleged when justifying the lack of the uptake of the Semantic Web is that it has not reached end-users [4].

We might argue that this is, in fact, the desired outcome, that client applications should hide the complexities of semantic technologies and that the benefits should be evident just server-side. For instance, search engines like Google provide better results thanks to semantic annotations, but users do not get aware. This should usually be the desired outcome when trying to satisfy specific user needs: a basic usability rule is that the user should always be provided with the most uncomplicated user experience possible [5].

For known tasks, such as managing a music collection or an address book, the simplest possible experience is often provided by a task-specific application with a task-specific interface. In this case, a familiar user interface will hide any Semantic Web nature of the underlying data. However, there will be other cases where no common application exists to camouflage the underlying Semantic Web data. A user may need to explore a data collection that is too rarely used to have motivated an ad-hoc application or seeking to learn something by combining multiple data collections that are not often combined [6].

For instance, many semantic search tools must work with arbitrary schema and cannot hard-code a particular one into their interfaces. For this task, tech-savvy users can rely on standards like SPARQL to query available data. However, this is beyond the capabilities of most users, and even SPARQL-aware developers have trouble querying unfamiliar data collections [7].

Consequently, we focus on more user-friendly visual query tools. All kinds of users can benefit from tools that make it possible to visually explore semantic data, showing all its richness while providing smooth user experience. It is in this particular scenario we might find the Semantic Web killer app that makes all the power of Web-wide connected data available to ordinary users, so they can even discover unforeseen connections in it.

Proposals are very disparate [8], ranging from Linked Data browsers [9] to Controlled Natural Language query engines [10] or faceted browsers [11].

This heterogeneity makes it difficult to compare them, especially from the user perspective. Therefore, a reference framework for benchmarking is required, as discussions in this research domain have already highlighted [12].

Moreover, it has also been shown that benchmarks help to organize and strengthen research efforts in a particular research area [13]. An example is the Text REtrieval Conference (TREC) benchmarks [14], which have become the de facto standard for evaluating any text document retrieval system. Also, there are success stories in areas related to the Semantic Web like ontology alignment [15, 16].

In the context of semantic data exploration, and thus not including Question Answering [12], there have been just a few efforts and based on quite informal criteria, like the Intelligent Exploration of Semantic Data Challenge[1]. Thus, none of them targets the general user task of semantic data exploration, nor provides a complete benchmark that facilitates comparability and competition in this research topic. On the other hand, there are many benchmarks for performance evaluation from a system perspective, like the Berlin SPARQL Benchmark (BSBM) [17] to evaluate SPARQL query engines, but they do not take into account the end-user perspective.

In this paper, we present a benchmark for semantic data (graphical) user interfaces with a set of user tasks to be completed and metrics to measure the performance of the analyzed interfaces at different levels of granularity. Evaluations based on this benchmark do not require real-users interacting with the application, just experts measuring the number of interactions required to complete each user task. This approach largely reduces the cost of user interface evaluation and produces more objective results that facilitate comparing different user interfaces.

Moreover, we provide a benchmark not just for Semantic Web data exploration, but for structured data more generally. This makes it possible to also compare tools available in more mature domains like relational databases [18]. We also hope to further motivate research in semantic data exploration that goes beyond what is possible with other less rich data models.

In Section 2, we present our approach to providing a benchmark for structured data exploration. Then, in Section 3, we present the benchmark, which was initially put into practice with a couple of faceted browsers, Virtuoso and Rhizomer, as detailed in an earlier publication [19]. In Section 4, this ini-

[1]IESD Challenge, `https://iesd2015.wordpress.com/iesd-challenge-2015`

tial experiment is extended with two additional tools, which now include a tool for relational databases exploration, Sieuferd. This allows a much more sophisticated illustration of the benchmark in action, together with its possibilities and the insights it makes possible by comparing such disparate tools. Conclusions in this regard are presented in Section 5 and future work in Section 6.

2. Approach

To define the benchmark, we first choose the tasks that will be benchmarked. Second, we decide what to measure about the systems as they are used for the chosen tasks. In both parts, our choices influence the fidelity of our benchmark. First, our chosen tasks should be representative of the tasks we expect users to perform. They should cover the common cases, and be neither too hard nor too easy. Second, our performance metrics should provide some suggestion of what real users will experience using the system. At the same time, they will be easier to adopt if at least some measurement can be done analytically, without actual expensive user studies.

These two choices are the "axioms" of our benchmark system; they cannot be proven correct but must instead be justified by experience and argumentation. We will discuss both in detail in the following two sections. For tasks we begin with (then augment) the Berlin SPARQL Benchmark, a set of queries initially intended to serve as a benchmark of computational performance. Our performance measure considers basic user operations such as mouse movements and keyboard clicks under the so-called Keystroke Level Model [20] of user interaction.

In choosing tasks, we want to avoid introducing bias from an a priori conception of the problem or experience developing our own tools. Consequently, we have looked outward to find sets of typical end-user tasks related to structured data exploration.

Although our main interest is semantic technologies, we prefer a benchmark that can also be applied to relational database tools, so we can compare them with semantic tools and highlight the pros and cons between them. Visual query tools will insulate the user from details of the underlying storage representation, meaning RDF or relational databases could equally be used as back-ends.

From existing benchmarks with user tasks, a clear candidate emerged: the Berlin SPARQL Benchmark (BSBM). Although this benchmark is in-

tended for measuring the computational performance of semantic and relational database query engines, it is based on a set of realistic queries inspired by common information needs in these domains. We can, therefore, leverage the same queries to measure the user interaction performance of visual query systems. Moreover, BSBM is based on a synthetic dataset generated by a tool given a target dataset size and output format, SQL or RDF, which facilitates the distribution of the benchmark.

All the user tasks are accompanied by both the SPARQL and SQL query to satisfy them. From the perspective of a user experience benchmark, these queries are technological details that might not be relevant because users can satisfy the tasks by generating different queries. However, they might be helpful to verify the outcomes of users' tasks and check they are getting the intended result.

Therefore, we adopted the proposed user tasks that motivate the actual SPARQL and SQL queries that conform the Berlin SPARQL Benchmark. The tasks are contextualized in an e-commerce scenario, where different vendors offer a set of products and different consumers have posted reviews about these products.

There are three different sets of tasks in the BSBM, depending on task types. The BSBM Explore set of tasks are directly connected to the proposed benchmark aim. There is a second set of Business Intelligence tasks, which are too complex to be considered in the context of data exploration tasks for the moment. Finally, there are Update tasks, which in the future, we hope to use to define a benchmark for users editing, rather than searching semantic data.

Consequently, the data exploration tasks in BSBM have been used as the starting point for the proposed structured data exploration benchmark from a user experience perspective. These are 12 tasks that illustrate the user experience of a user looking for a product. The tasks are presented in the following subsection.

Note that our goal is not to evaluate e-commerce tools specifically. The intended targets are exploration tools for arbitrary structured data, so they cannot have any e-commerce features hardcoded into them. However, e-commerce provides a convenient and intuitive domain in which to define queries we expect users to want to carry out. We are interested in general operations, such as combining two constraints, but for concreteness, we provide tasks in our benchmark in e-commerce language.

Our benchmark does not aim to assess discoverability/learnability. We

posit a user who is already familiar with the tool being evaluated who knows where to access available operations and how to invoke them. To conclude this section, and before starting to describe each task in detail, it is important to note that the SPARQL and SQL queries associated to each task are not included in this paper due to space constraints but are available from the benchmark repository[2].

3. Structured Data Exploration Benchmark

The proposed benchmark currently consists of 12 end-user tasks to be completed with the evaluated tool, listed in Section 3.1. For each task we detail the information need and provide some context. Then, we give a sample query based on the sample dataset accompanying the benchmark together with the expected outcome.

The proposed benchmark also includes a set of metrics to measure the effectiveness and efficiency of the evaluated tool when performing each of the proposed tasks. These metrics yield numbers that can be used to compare the performance of structured data exploration tools, as detailed in Section 3.2.

3.1. End-User Tasks

The following subsections introduce each of the 12 end-user tasks. All but one of them are directly adopted from the Berlin SPARQL Benchmark (BSBM). One additional task, Task 2, has been added as a variation of Task 1 to cover a gap in the original benchmark (OR versus AND operations for combining subqueries. Full details are available from the benchmark site[3].

Although the BSBM presents a particular e-commerce schema, we hold that a true semantic web query tool cannot make assumptions about the schema of the data it is to query. It should operate equally well on any data schema it encounters. A tool that hardwires the BSBM schema into its interface will be useless on a different data set and thus is not a true semantic web tool. The BSBM instantiates one arbitrary schema to let us talk about our queries concretely, but the tool being analyzed should not be permitted advance knowledge of this particular instantiation.

[2]BESDUI, `http://w3id.org/BESDUI`

[3]`https://github.com/rhizomik/BESDUI/tree/master/Benchmark`

Task 1. Find products for a given set of combined features:
A consumer seeks a product that present a specific set of features. The corresponding information need for the benchmark dataset specifies a product type from the product hierarchy (one level above leaf level), two different product features that correspond to the chosen product type and that should be present simultaneously and a number between 1 and 500 for a numeric property. For instance:

> *"Look for products of type* ***sheeny*** *with product features* ***stroboscopes*** *AND* ***gadgeteers****, and a* ***productPropertyNumeric1*** *greater than* ***450****".*

For the previous query, and considering the sample BSBM 1000 Products dataset[4], the product labels the user should obtain are:

> *"auditoriums reducing pappies" and "driveled".*

Task 2. Find products for a given set of alternative features:
A consumer is seeking a product with a general idea about some alternative features of what he wants. This task has been added beyond those provided by BSBM. It makes Task 1 to less specific by considering feature alternatives; the user is interested in any product that presents at least one of them. This benchmarks how exploration tools lets users define OR operations. A sample query for this task might be:

> *"List products of type* ***sheeny*** *with product features* ***stroboscopes*** *OR* ***gadgeteers****, and a* ***productPropertyNumeric1*** *greater than* ***450****".*

For the previous query, and considering the sample dataset, the product labels the user should obtain if restricted to the first 5 ordered alphabetically are:

> *"aliter tiredest", "auditoriums reducing pappies", "boozed", "by-play", "closely jerries".*

[4]urlhttps://github.com/rhizomik/BESDUI/blob/master/Datasets/bsbm-1000products.ttl.tgz

Task 3. Retrieve basic information about a specific product for display purposes:
The consumer wants to view basic information about a specific product. For instance:

> *"Get details about product **boozed**".*

From the entry page, and considering the synthetic dataset generated using the BSMB tool, the response should include the following properties for the selected product with their corresponding values, which are omitted due to space restrictions but avail-able from the benchmark repository[5]:

> *"label", "comment", "producer", "productFeature", "propertyTextual1", "propertyTextual2", "propertyTextual3", "propertyNumeric1", "propertyNu-meric2", "propertyTextual4", "propertyTextual5", "propertyNumeric4".*

Task 4. Find products having some specific features and not having one feature:
After looking at information about some products, the consumer has a more specific idea what she wants, features the products should have and others that should not. The main feature of this task is the use of negation. A sample query for this task is:

> *"Look for products of type **sheeny** with product features **stroboscopes** but NOT **gadgeteers**, and **productPropertyNumeric1** value greater than **300** and **productPropertyNumeric3** smaller than **400**".*

For this query and the BSBM 1000 dataset, the the user should obtain:

> *"boozed", "elatedly fidelis release" and "learnable onomatopoeically".*

Task 5. Find products matching two different sets of features:
After looking at information about some products, the consumer has a more specific idea what he wants. Therefore, he asks for products matching either one set of features or another set. The complexity in this case is the union of the sets of products selected by two different patterns. For instance:

[5] `https://github.com/rhizomik/BESDUI/blob/master/Benchmarks/3.md`

*"Look for products of type **sheeny** with product features **stroboscopes** and **gadgeteers** and a **productPropertyNumeric1** value greater than **300** plus those of the same product type with product features **stroboscopes** and **rotifers** and a **productPropertyNumeric2** greater than **400**".*

For the previous query, and the sample dataset, the product labels the user should obtain if restricted to the first 5 ordered alphabetically are:

"auditoriums reducing pappies", "boozed", "driveled", "elatedly fidelis release", "zellations".

Task 6. Find product that are similar to a given product:
The consumer has found a product that fulfills his requirements. She now wants to find products with similar features. The corresponding query starts from a product and looks for all other products with at least one common feature and a wider range of values for two of its numeric properties. For instance:

*"Look for products similar to **boozed**, with at least one feature in common, and a **productPropertyNumeric1** value between **427** and **627** and a **productPropertyNumeric2** value between **595** and **895** (150 more or less than its value for boozed, 745)".*

For the previous query, and considering the sample dataset, the product labels the user should obtain if restricted to the first 5 ordered alphabetically are:

"debouches orangs unethically", "dirk professionalize", "grappled", "im-posed", "pepperiness gothically shiner".

Task 7. Find products having a name that contains some text:
The consumer remembers parts of a product name from former searches. She wants to find the product again by searching for the parts of the name that she remembers. The corresponding query is just one of the words from the list of words[6] that were used during dataset generation by the BSBM Data Generator[7]. For instance:

[6] `https://github.com/rhizomik/BESDUI/blob/master/Datasets/titlewords.txt`
[7] `http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/spec/BenchmarkRules/index.html#datagenerator`

*"Search products whose name contains **ales**".*

For the previous query, and considering the sample dataset, the product labels the user should obtain if restricted to the first 5 ordered alphabetically are:

"cogitations centralest recasting", "overapprehensively dales ventless", "skidooed finales noisemaker" and "unwed convalescents".

Task 8. Retrieve in-depth information about a specific product including offers and reviews:
The consumer has found a product which fulfills his requirements. Now he wants in-depth information about this product including offers from German vendors and product reviews if existent. The corresponding query refers to a selected product and defines a current date within the "valid from" and "valid to" range of the offers. Compared to previous tasks, this one introduces being able to pose restrictions to different model entities that are interrelated, in this case vendors and reviews that are interrelated with products and offers. For instance:

*"For the product **waterskiing sharpness horseshoes** list details for all its offers by German vendors and still valid by **2008-05-28** plus details for all reviews for this product, including values for **rating1** and **rating2** if available".*

Considering the benchmark sample dataset, the user should get access to the details about the following offers and reviews:

"Offer10801", "Offer5335", "Offer10597", "Review5481", "Review7546", "Review2669", "Review5731", "Review8494".

Task 9. Give me recent reviews in English for a specific product:
The consumer wants to read the 20 most recent English language reviews about a specific product. The corresponding query refers to a selected product. This task required being able to filter literals language and ordering by date. For instance:

*"For the product **waterskiing sharpness horseshoes** list the 20 more recent reviews in **English**".*

Given the sample dataset, the user should obtain the details for the following reviews in the order they are listed:

"Review5481", "Review8494" and "Review2669".

Task 10. Get Information about a reviewer:
In order to decide whether to trust a review, the consumer asks for any kind of information that is available about the reviewer. The corresponding query refers to a selected product. This tasks explore how easy it is to reach the information about a resource from a related one. For instance:

"Get all available information about ***Reviewer11****".*

For the sample dataset, the user obtains all the details about the following reviewer:

"Reviewer1".

Task 11. Get offers for a given product which fulfill specific requirements:
The consumer wants to buy from a vendor in the United States that is able to deliver within 3 days and is looking for the cheapest offer that fulfills these requirements. The corresponding query refers to a selected product and defines a current date within the "valid from" and "valid to" range of the offers. For instance:

> *"Look for the cheapest and still valid by* ***2008-06-01*** *offer for the product* ***water-skiing sharpness horseshoes*** *by a* ***US*** *vendor that is able to deliver within* ***3*** *days".*

Considering the sample dataset, the user interface should get as a response the following offers:

"Offer3499", "Offer11865" and "Offer15103".

Task 12. Export the chosen offer into another information system which uses a different schema:
After deciding on a specific offer, the consumer wants to save information about this offer on his local machine using a different schema. The corresponding query refers to a selected offer, or the one considered by the previous task.

*"Save in the local computer the information about the **vendor** for **Offer3499**, this is **half** the task. To complete it, restrict the output to just **label**, **homepage** and **country** and map them to **schema.org** terms: **name**, **url** and **nationality**".*

3.2. Metrics

Our benchmark gives a number of generic yet typical information-seeking tasks to be measured. We now ask the following three increasingly detailed questions to measure the effectiveness and efficiency of the tool on these tasks. Additionally, there is a combined effectiveness/efficiency fourth metric:

1. **Capability (effectiveness)**. Is performing a task possible with the given system?
   For individual tasks, the values are **0** if not possible or **1** otherwise. However, some tasks are divided into subtasks and state other potential values. For instance, **0.5** if just the first subtask of Task 12 is completed. Usually, this metric is displayed as a percentage, thus **0%...100%**. For a given task, the Capability (C) is computed as:

$$C_{Task_i} = \begin{cases} 1 & \text{if task completed in whole} \\ (0,1) & \text{if task partially completed, e.g. 0.5 if just half} \\ 0 & \text{otherwise} \end{cases}$$

   For the whole benchmark, it is the percentage of all 12 tasks completed and computed as:

$$C = \frac{\sum_{i=1}^{12} C_{Task_i}}{12}$$

2. **Operator Count (efficiency)**. How many basic interaction steps (mouse clicks, keyboard entries, scrolling,...) must be performed to carry out a task?
   The Operator Count (OC) for a particular task is computed as the count of all operations performed to complete it, where what constitutes an interaction operation (o) is detailed later in this section:

$$OC_{Task_i} = \begin{cases} \sum[o \in Task_i] & \text{if } C_{Task_i} > 0 \\ 0 & \text{otherwise} \end{cases}$$

For the whole benchmark, it is the mean of the Operator Counts for all tasks that can be completed at least in part:

$$OC = \sum_{i=1}^{12} OC_{Task_i} / \sum_{i=1}^{12} [C_{Task_i} > 0]$$

3. **Time (efficiency)**. How quickly can these interaction steps be executed to carry out a task?
   This metric is based on a function *time()* that maps interaction operations (o) to the amount of time required to perform it, in seconds. This function is detailed next in this section. For a given task, the Time (T) is computed as:

$$T_{Task_i} = \begin{cases} \sum_{o \in Task_i} time(o) & \text{if } C_{Task_i} > 0 \\ 0 & \text{otherwise} \end{cases}$$

   For the whole benchmark, it is the mean of the Times for all tasks that can be completed at least in part:

$$T = \sum_{i=1}^{12} T_{Task_i} / \sum_{i=1}^{12} [C_{Task_i} > 0]$$

4. **Task Efficiency (effectiveness/efficiency)**: measured as a ratio of **Capability** to **Time** in terms of "*goals per minute*".
   For an individual task, Task Efficiency (TE) is computed as follows (times 60 because it is measured in minutes instead of seconds):

$$TE_{Task_i} = \frac{C_{Task_i}}{T_{Task_i}} \times 60$$

   For the whole benchmark, TE is computed as the ration between the overall Capability (K) and the average Time of all the tasks that have been completed at least in part, i.e. with Time greater than zero. This way of computing TE favours more versatile tools with a higher K and penalizes those very specialized in just a small set of tasks:

$$TE = \frac{K}{T} \times 60$$

Presumably, a system can be judged superior to another if it can be used to perform more of the tasks, with fewer basic steps that take less time. Our general target is graphical user interfaces for querying structured data. For contrast, if we consider for example a SPARQL command line, a suitably trained user would be able to perform all benchmark tasks with just a single primitive operation (typing the SPARQL query) in a very small amount of time (leaving out designing and debugging the SPARQL query). But most users don't have the training or understanding necessary to use such a tool. Instead, some type of GUI is the norm, and it is such systems we aim to evaluate.

The first two questions, of Capability and Operator Count, can be answered entirely analytically. They simply require identifying and counting up the sequence of operators that complete each task. Ideally, the third question would be answered by a timed user study. However, conducing user studies is a very time consuming activity, especially because it involves recruiting users. To facilitate the application of the benchmark, our proposed metric relies on past HCI research that offers a way to answer the time question analytically as well, by applying known, analytic timing models for primitive actions (keyboard and mouse operations) in the identified sequence.

In particular, the Keystroke Level Model (KLM) [20] gave experimentally derived timings for basic operations such as typing a key, pointing on the screen with the mouse, moving hands back to the keyboard, and so forth. Given a sequence of these basic operations, we can total up their timings to yield an overall predicted execution time for the task. Our proposal is to use the main interaction operators proposed by KLM and their mapping to time to define the Operator Count and Time metrics. The first one does not distinguish among operations so it is computed as the sum of the counts of all operations, though we will also keep the totals per kind of KLM operators to facilitate computing the Time metric. The considered operators and their mappings to time in seconds used to analytically compute the Time metric are shown in Table 1.

4. Benchmark Evaluation

To facilitate the adoption of the benchmark, and to evaluate its applicability, we have tested it with three faceted browsers for semantic data: Rhizomer [11], Virtuoso Facets [21] and PepeSearch [22]. The first two are more sophisticated and provide features like pivoting, while the last does not

operator (o)	operator description	**time(o)** (seconds)
K	Button press (including mouse's) or keystroke (keys, not characters, so shift-C counts as two)	0.2
P	Pointing to a target on a display, e.g. with a mouse (time differs depending on target distance and size, but held constant for simplicity)	1.1
H	Homing the hand(s) on the keyboard or other device (this includes movement between any two devices)	0.4

Table 1: Interaction operators considered and their corresponding time based on the Keystroke Level Model (KLM)

though its simplicity makes it easier for users. We have also evaluated a very sophisticated tool not based on semantic but relational data, Sieuferd [23]. This is a query construction tool through direct manipulation of nested relational results.

This way we try illustrate how the benchmark works for quite different tools and it is capable of capturing their different capabilities, ranging from simpler tools that are more efficient but do not support all tasks to sophisticated ones more effective but less efficient.

Moreover, we have set a GitHub repository for the benchmark that can be forked to contribute results for additional tools, which can be then incorporated into the reference repository through a pull request. Additional details about how to contribute to the benchmark are available from the repository. For the moment, the whole set of results for Sieuferd, PepeSearch, Virtuoso Facets and Rhizomer are available. Next, an overview of the results for the first three tasks is included in this paper.

Task 1 Results. Find products for a given set of combined features This task can be completed just with Virtuoso Facets and Sieuferd. Rhizomer and PepeSearch do not support this kind of query because when defining the values for a particular facet, like "stroboscopes" and "gadgeteers" for "feature", it is not possible to specify that both should be available for the same product simultaneously. The Capability metric value is then 0% for both, as shown in Table 3.

Sieuferd and Virtuoso can complete this task and the outcome is the expected considering the sample dataset, the products "driveled" and "au-

ditoriums reducing pappies", as it is shown for Virtuoso in Figure 1. To complete this task with Virtuoso, the interaction steps and corresponding KLM Operators are listed in Table 2, while those required to complete the task using Sieufred are available from the benchmark repository[8]. The metrics for all four tools are shown in Table 3. As it can be observed, Virtuoso needs less KLM operators to complete the tasks. The required time is also smaller, computed using the mappings from KLM operators to time available in Table 1:

$$\text{Virtuoso } T_{Task_1} : 28 \times 0.2 + 18 \times 1.1 + 5 \times 0.4 = 27.4$$

This makes Virtuoso slightly more efficient than Sieuferd, also regarding the Task Efficiency metric as both tools attain the same Capacity, 2.2 completed $Task_1$ per minute for Virtuoso and 2.0 for Sieuferd, while Task Efficiency is 0 for Rhizomer and PepeSearch.

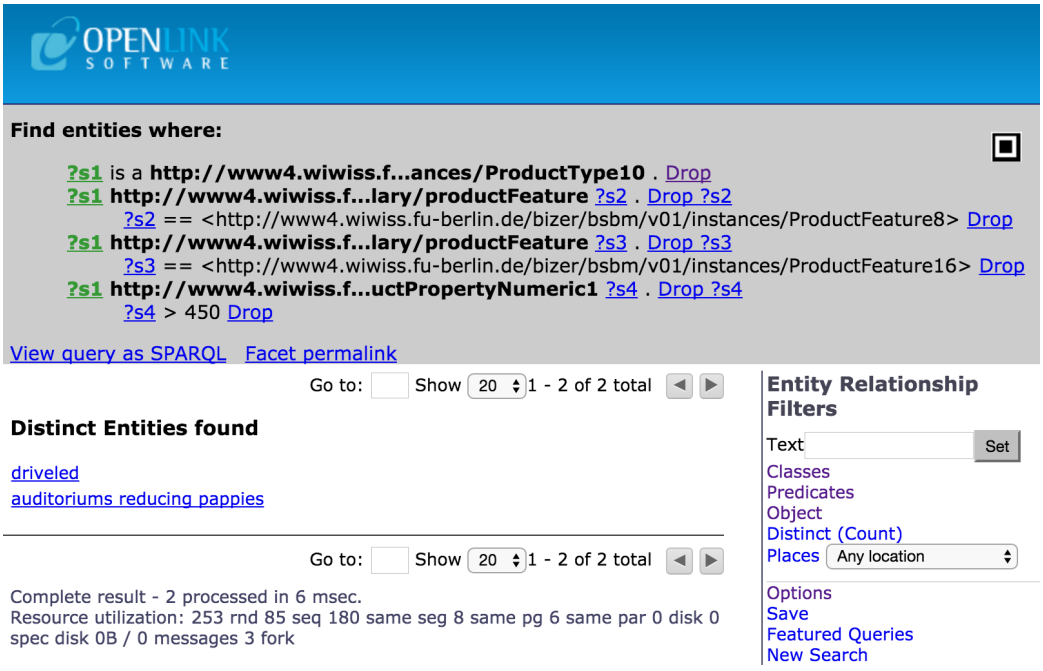

Figure 1: Using Virtuoso Facets to complete Task 1

[8] https://github.com/rhizomik/BESDUI/tree/master/Results/Sieuferd#results-per-task

Type “sheeny” and “Enter”, then click “ProductType10”.	9K, 2P, 3H
Click “Go” for “Start New Facet” , then click “Options”.	2K, 2P
For “Interence Rule” Click and Select rules graph then “Apply”.	2K, 2P
Click “Attributes” , then “productFeature” and “stroboscopes”.	3K, 3P
Click “Attributes” , then “productFeature” and “gadgeteers”.	3K, 3P
Click “Attributes” and “productPropertyNumeric1”.	2K, 2P
Click “Add condition: None” and select “>”.	2K, 2P
Type “450” and click “Set Condition”.	5K, 2P, 2H
Total	28K, 18P, 5H

Table 2: Interaction steps and corresponding KLM operators to complete Task 1 using Virtuoso

	$\mathbf{C}_{Task_1}$	$\mathbf{OC}_{Task_1}$	$\mathbf{T}_{Task_1}$	$\mathbf{TE}_{Task_1}$
Rhizomer	0%	-	-	0
Virtuoso Facets	**100%**	**51** (28K, 18P, 5H)	**27.4**	**2.2**
Sieuferd	**100%**	67 (47K, 19P, 1H)	30.7	2.0
PepeSearch	0%	-	-	0

Table 3: BESDUI metrics for Task 1 (best results in bold)

Task 2 Results. Find products for a given set of alternative features

Rhizomer, as shown in Figure 4, supports this task because its facets can be used to select more than one of their values as alternatives, as illustrated in Table 2 where the interaction steps and KLM Operators required to complete this tasks using Rhizomer are presented. Virtuoso Facets and Sieufred can also complete this task, but not PepeSearch. The interaction steps required to complete the task using Virtuoso are available from the benchmark repository[9], just like those for Sieufred.

The metrics for all four tools are presented in Table 5. In this case Rhizomer is the more efficient tool in terms of operations and time to complete the task, just 12 seconds. The same holds for Task Efficiency, with 5 $Task_2$

[9] `https://github.com/rhizomik/BESDUI/tree/master/Results/Virtuoso#results-per-task`

per minute for Rhizomer. On the other hand, Virtuoso and Sieuferd have almost identical efficiency metrics for this task.

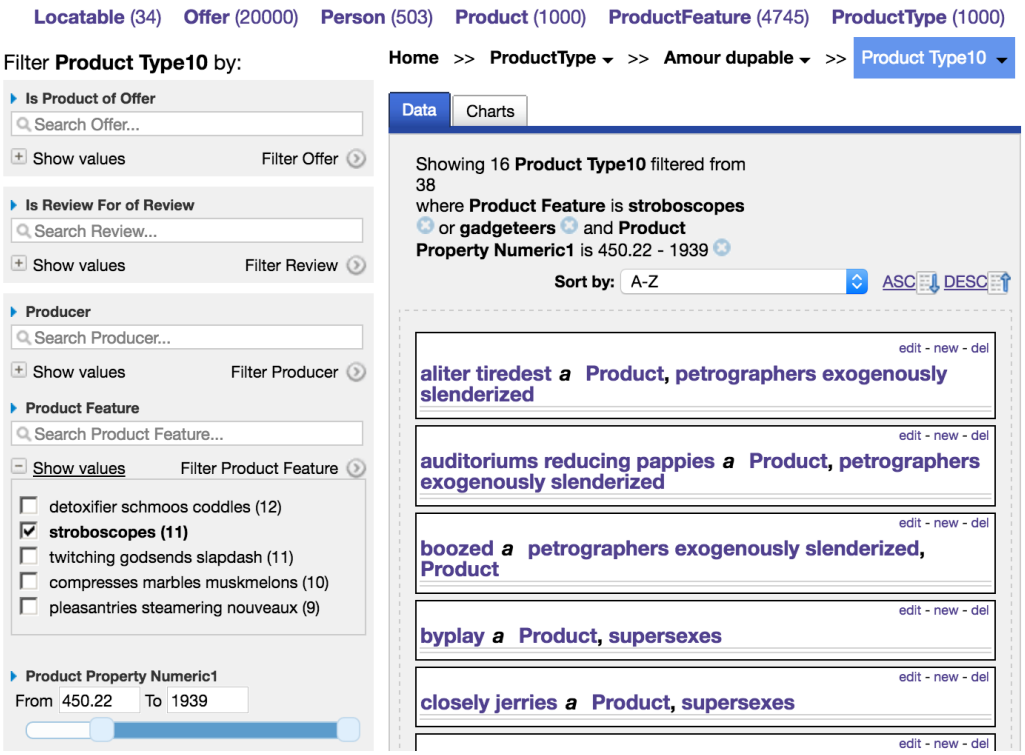

Figure 2: Using Rhizomer to complete Task 2

Task 3 Results. Retrieve basic information about a specific product for display purposes

All tools, including PepeSearch as shown in Figure 3, support this task. The interaction steps and KLM Operators followed with PepeSearch are presented in Table 6. The interaction steps required to complete the task using Rhizomer are available from the benchmark repository[10], just like those for Sieufred and Virtuoso.

The metrics for all four tools are presented in Table 7. They show that Rhizomer is the most efficient tool and that, though Virtuoso requires more

[10] https://github.com/rhizomik/BESDUI/tree/master/Results/Rhizomer#results-per-task

Click menu ProductType and then Sheeny submenu.	2K, 2P, 1H
Click Show values for facet Product Feature.	1K, 1P
Click facet value stroboscopes.	1K, 1P
Type in input Search Product Feature gad....	4K, 1P, 1H
Select gadgeteers from autocomplete.	1K, 1P, 1H
Set left side of Product Property Numeric1slider to 450.	1K, 2P
Total	10K, 8P, 3H

Table 4: Interaction steps and corresponding KLM operators to complete Task 2 using Rhizomer

	$\mathbf{C}_{Task_2}$	$\mathbf{OC}_{Task_2}$	$\mathbf{T}_{Task_2}$	$\mathbf{TE}_{Task_2}$
Rhizomer	**100%**	**21** (10K, 8P, 3H)	**12.0**	**5.0**
Virtuoso Facets	**100%**	53 (29K, 19P, 5H)	28.7	2.1
Sieuferd	**100%**	67 (38K, 19P, 1H)	28.9	2.1
PepeSearch	0%	-	-	0

Table 5: BESDUI metrics for Task 2 (best results in bold)

operators than PepeSearch, it is more time efficient. This is because it requires less Pointing (P) operators, the most time-expensive one. Observing the interaction steps and operators, it can be observed that PepeSearch is being penalized by the fact that there is not a global search form, like in the case of Rhizomer or Virtuoso, and the user needs to get first to the "Product" type before being able to search for the one labelled "boozed". It can be also observed that, though both feature a global search form, Rhizomer is more efficient than Virtuoso thanks to its autocomplete feature.

Select "Product" in the list of concepts	1K, 1P, 1H
Click on the product label field and type "bo"	3K, 1P, 1H
Select boozed from autocomplete	2K
Click on the product link "boozed"	1K, 1P, 1H
Total	7K, 3P, 3H

Table 6: Interaction steps and corresponding KLM operators to complete Task 3 using PepeSearch

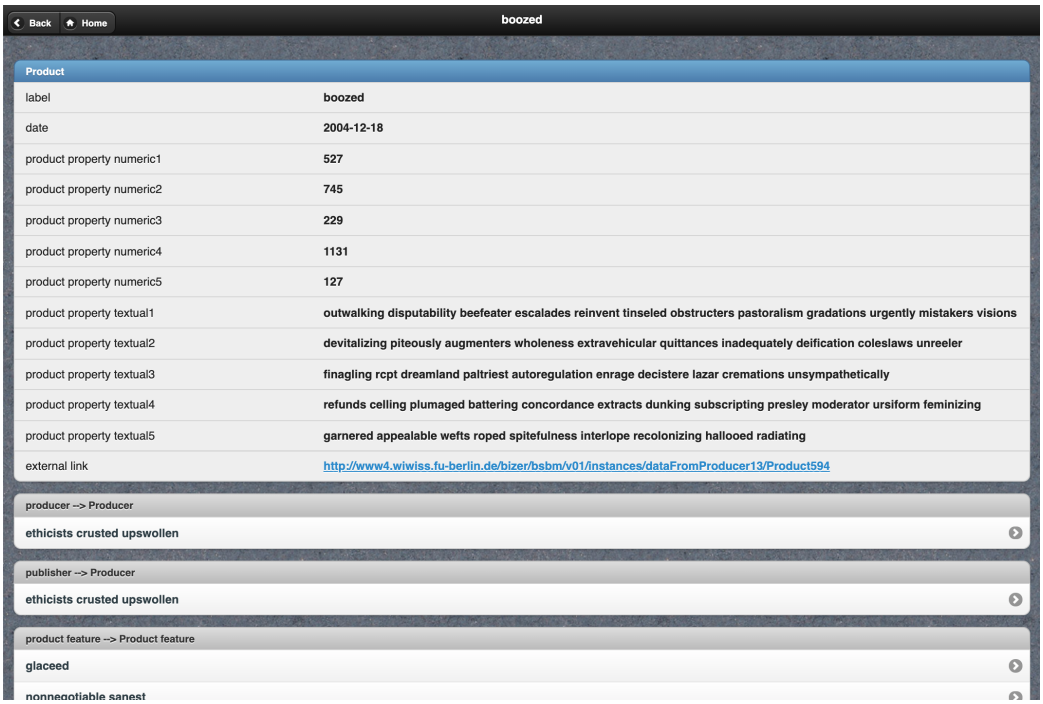

Figure 3: Using PepeSearch to complete Task 3

	$\mathbf{C}_{Task_3}$	$\mathbf{OC}_{Task_3}$	$\mathbf{T}_{Task_3}$	$\mathbf{TE}_{Task_3}$
Rhizomer	**100%**	**11** (6K, 2P, 3H)	**4.6**	**13.0**
Virtuoso Facets	**100%**	14 (9K, 2P, 3H)	5.2	11.5
Sieuferd	**100%**	27 (23K, 3P, 1H)	8.3	7.2
PepeSearch	**100%**	13 (7K, 3P, 3H)	5.9	10.2

Table 7: BESDUI metrics for Task 3 (best results in bold)

Task 4 Results. Find products having some specific features and not having one feature
Just with Sieuferd, shown in Figure 4, a user is capable of completing this task. The interaction steps and KLM Operators followed with Sieuferd are presented in Table 8. The rest of the tools do not offer a mechanism for users to specify that there are certain product features that should not be present. Consequently, the user cannot build the proper underlying query to retrieve the expected products. The Capacity is then 0% except for Sieuferd, as shown together with the rest of metrics for this task in Table 9.

4.1. Overall Results

The detailed outcomes for the rest of the tasks are not included in this paper to keep it focused. After illustrating how the 4 first tasks have been completed, each one with one of the 4 tools evaluated, the mechanics of the benchmark are already clear. In any case, the results for all four tools and the 12 taks are available from the benchmark repository[11].

Now, from the results obtained for each task, it is possible to compute the metrics for the whole benchmark as it was detailed in Section 3.2. Table 10 presents the values for the whole benchmark for all tools, which are discussed next in the paper conclusions, Section 5.

5. Conclusions

As already shown in other research domains and discussed in Section 1, the existence of benchmarks that facilitate comparing contributions related

[11] `http://w3id.org/BESDUI`

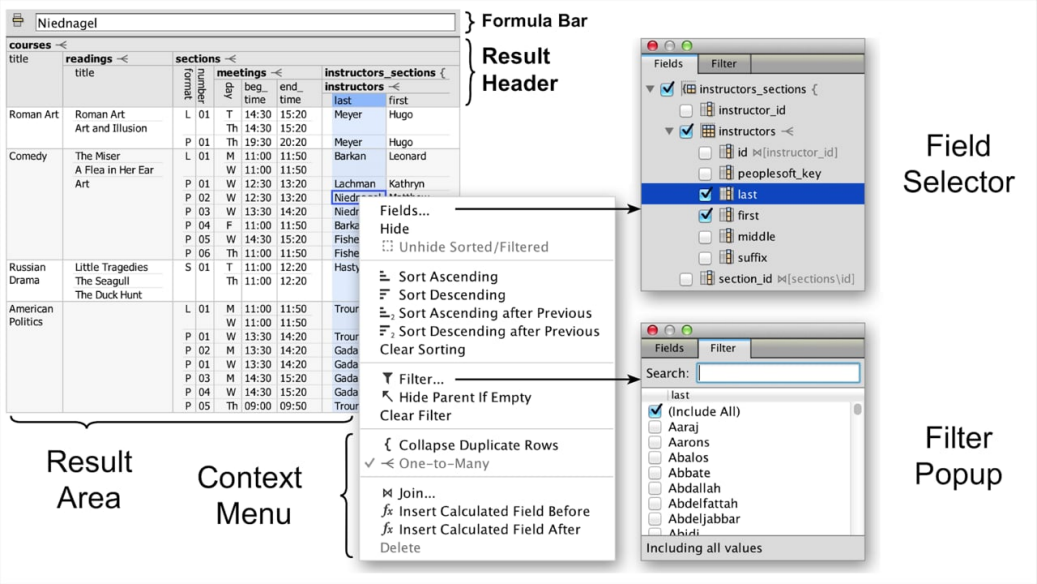

Figure 4: Components of the Sieuferd user interface.

to a specific research challenge helps foster efforts in that particular domain and clarify the scope of the contributions. In the domain of semantic data exploration, there are many proposed tools and surveys but there is not a benchmark to easily and objectively compare them from a user experience perspective.

Our proposal is based on a set of user tasks, most of them borrowed from the Berlin SPARQL Benchmark (BSBM), to be completed using the evaluated tools. Though these tasks are originally conceived to test SPARQL engines' performance, they are very well contextualized in the e-commerce domain, cover a wide range of information needs and are accompanied by a synthetic dataset generator that facilitates the distribution of the benchmark and its deployment.

Though the dataset is synthetic and, for instance, many resources present funny names like "waterskiing sharpness horseshoes", it is important to note that this does not introduce any significant effect from the user experience perspective when measured using the Keystroke Level Model. Moreover, it is also important to note that, as testing with real users is very costly and time-consuming, the benchmark is based on analytical methods and therefore require only the involvement of a researcher experienced in semantic data

Create and open a perspective from the “product” table	4K, 3P
Make the “nr”, “propertyNum1”, “propertyNum3”, “productfeatureproduct.productfeature.label”, and “producttypeproduct.producttype.label” fields visible	11K, 9P
Filter to show only products for which “producttypeproduct.producttype.label” is “sheeny”	4K, 3P
Do a custom join operation to join in another instance of the productfeatureproduct table	5K, 5P
Make the “productfeatureproduct2.productfeature.label” field visible	4K, 3P
Filter for “stroboscopes” in “productfeatureproduct”	4K, 3P
Filter for “NOT gadgeteers” in “productfeatureproduct2”	5K, 4P
Insert a calculated field next to “product.label”	2K, 2P
Key the cursor to the calculated column and type the formula “=[propertyNum1]>300 and [propertyNum3]<400” (field references inserted by arrow key presses)	17K, 1H
Filter to show only products for which the formula is “true” (use keyboard shortcut since hand is already on keyboard)	6K
Total	62K, 32P, 1H

Table 8: Interaction steps and corresponding KLM operators to complete Task 4 using Sieuferd

exploration tools, ideally one of the tool developers.

The metrics are **Capability**, an effectiveness metric that measures if a task can be completed or not using an evaluated tool, **Operator Count**, an efficiency metric counting how many KLM Operators are required to complete the task, and **Time**, another efficiency metric that translates the KLM Operators required to complete the task into an approximate amount of time using the experimentally derived timings for these operators. The operators and timings are: K for each keystroke or button press amounting 0.2 seconds, P for pointing to a target with the mouse corresponding to 1.1 seconds and H for homing the hands on the keyboard or other devices, 0.4 seconds.

Finally, there is a combined effectiveness/efficiency fourth metric, **Task Efficiency**, computed as the ration of Capability to Time. This metric provides an overall measure of *how good* a tool is in terms of the amount of “goals per minute” it potentially allows its users to complete and favours more versatile tools with a higher Capability while penalizes those very specialized in just a small set of tasks though they have a smaller Time metric.

	$\mathbf{C}_{Task_4}$	$\mathbf{OC}_{Task_4}$	$\mathbf{T}_{Task_4}$	$\mathbf{TE}_{Task_4}$
Rhizomer	0%	-	-	0
Virtuoso Facets	0%	-	-	0
Sieuferd	**100%**	**95** (62K, 32P, 1H)	**48.0**	**1.2**
PepeSearch	0%	-	-	0

Table 9: BESDUI metrics for Task 4 (best results in bold)

	C	**OC**	**T**	**TE**
Rhizomer	58%	29 (15.6K, 10.9P, 2.6H)	16.1	**2.2**
Virtuoso Facets	54%	36.1 (20.4K, 12.7P, 3H)	19.3	1.7
Sieuferd	**96%**	71.3 (48.7K, 19.8P, 2.9H)	32.6	1.8
PepeSearch	25%	**21** (10.3K, 5.3P, 5.3H)	**10.1**	1.5

Table 10: BESDUI metrics values for the whole benchmark, i.e. the 12 tasks (best results in bold)

Based on the 12 proposed tasks and the 4 metrics, the benchmark has been applied to four structured data exploration tools. This has been done without having to recruit test users, experienced users capable of using the tool have been enough, usually involving the tool developers. The experts recorded the interaction steps and then translated them to KLM Operators, from which all the metrics were computed.

Three of the analyzed tools are based on semantic data (Rhizomer, Virtuoso and PepeSearch) and one on relational (Sieuferd). The full results are available from a GitHub repository[12] intended for maintaining the benchmark, keeping track of evaluations and organizing contributions. In addition to showing benchmark applicability, this experience has provided interesting insights that can be derived from the overall results for the metrics presented in Table 10, which highlights in bold the best results for each metric.

First of all, Sieuferd is the most effective tool, capable of performing all

[12] `http://w3id.org/BESDUI`

tasks except for Tasks 12 that is completed just in part. However, quite logically, this power comes with a more sophisticated user interface that provides the lowest efficiency: more than 70 operators and 30 seconds on average to complete these tasks. On the other hand, PepeSearch allows completing just 3 tasks due to its much simpler user interface, though its simplicity makes it very efficient, requiring 21 operators and just 10 seconds on average to complete these tasks. Rhizomer and Virtuoso lay on the middle, support slightly more than half of the tasks while showing more efficiency than Sieuferd and less than PepeSearch. Finally, there is Task Efficiency that allows getting a more balanced score between effectiveness and efficiency. Thus, Rhizomer gets the best mark because it provides almost 60% Capability with half the time when compared to Sieufred, which amount to a Task Efficiency of 2.2 goals per minute. This last metric tries to balance effectiveness versus efficiency but penalizing tools that are too specialized in just some tasks. Thus, Sieuferd, though not very efficient, gets the second-best Task Efficiency followed by Virtuoso, while PepeSearch gets the worst score due to its overspecialization.

6. Future Work

The main objective of this contribution is to foster the formation of a community around the evaluation and comparison of tools for structured data exploration. Consequently, we have prepared a GitHub repository where all the required elements to conduct an evaluation are available. This includes a sample dataset, the descriptions of the tasks, reference SPARQL and SQL queries to test expected responses, descriptions of the metrics and templates to report results.

We have added contributing instructions based on the common practice in GitHub that encourages forking the repository, making contributions like new evaluation results based on the templates and then making a pull request to incorporate them in the reference repository. We also expect contributions like additional tasks or metrics, which will also be considered for inclusion.

In addition to this expected community-building efforts and results, our plans also include concrete tools to be evaluated, metrics and tasks to consider. First of all, we are currently exploring the tasks also proposed in the BSBM in the Business Intelligence scenario. Though these tasks are much more complicated than the Explore tasks, some of them might be interesting to test with tools featuring visualizations.

We also plan to test more tools using the benchmark, ranging from semantic data tools providing direct manipulation like Explorator, to tools that facilitate building queries interactively like YASGUI or relational data exploration tools like Cipher.

Our performance metrics currently emphasize basic low-level operations such as keystrokes and mouse clicks. These can be refined. For example, Fitts's law has related the time to execute a mouse operation to the target regions size and from it is the mouse starting point; this can be incorporated into our timing analysis. At a higher level, our benchmark currently does not capture user effort. An interface that requires the user to think hard about which operation to perform next (and how to do it) will be more taxing and take more time.

As discussed previously, and extreme model of this is the SPARQL command line, which is extremely efficient in the KLM because all the work is mental, figuring out what SPARQL query to time. Similarly, our benchmark favours complicated UI layouts where all actions are "one click away", neglecting the fact that Fitts' Law indicates that selecting these actions becomes slower. The KLM model does not capture this, but there are so-called GOMS [24] models that try to.

To fully evaluate the usefulness of the proposed efficiency metrics, we will also test the systems using user experience evaluation techniques that involve real users and include measuring the actual time users need to complete the tasks.

References

[1] R. Cyganiak, A. Jentzsch, The linking open data cloud diagram, 2014. URL: `http://lod-cloud.net`.

[2] R. V. Guha, D. Brickley, S. Macbeth, Schema.org: Evolution of structured data on the web, Communications of the ACM 59 (2016) 44–51. doi:`10.1145/2844544`.

[3] H. Alani, Y. Kalfoglou, K. OHara, N. Shadbolt, Towards a Killer App for the Semantic Web, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2005, p. 829843. doi:`10.1007/11574620`.

[4] N. Shadbolt, W. Hall, T. Berners-Lee, The semantic web revisited, Intelligent Systems 21 (2006) 96–101. doi:`10.1109/MIS.2006.62`.

[5] S. Krug, Dont Make Me Think, Revisited: A Common Sense Approach to Web Usability, 3rd edition ed., New Riders, 2014.

[6] V. Tamma, Semantic web support for intelligent search and retrieval of business knowledge, IEEE Intelligent Systems 25 (2010) 84–88. doi:10.1109/MIS.2010.25.

[7] A. Freitas, E. Curry, J. G. Oliveira, S. ORiain, Querying heterogeneous datasets on the linked data web: Challenges, approaches, and trends, IEEE Internet Computing 16 (2012) 24–33. doi:10.1109/MIC.2011.141.

[8] A.-S. Dadzie, M. Rowe, Approaches to visualising linked data: A survey, Semantic Web 2 (2011) 89–124. doi:10.3233/SW-2011-0037.

[9] T. Berners-Lee, J. Hollenbach, K. Lu, J. Presbrey, E. Pru d'ommeaux, m. c. schraefel, Tabulator redux: Browsing and writing linked data, in: Proceedings of the Linked Data on the Web Workshop (LDOW08), volume 369, CEUR Workshop Proceedings, 2008, pp. 1–8.

[10] E. Kaufmann, A. Bernstein, Evaluating the usability of natural language query languages and interfaces to semantic web knowledge bases, Web Semantics: Science, Services and Agents on the World Wide Web 8 (2010) 377–393. doi:10.1016/j.websem.2010.06.001.

[11] J. M. Brunetti, R. García, S. Auer, From overview to facets and pivoting for interactive exploration of semantic web data, International Journal on Semantic Web and Information Systems 9 (2013) 1–20. doi:10.4018/jswis.2013010101.

[12] R. Usbeck, M. Röder, M. Hoffmann, F. Conrads, J. Huthmann, A.-C. Ngonga-Ngomo, C. Demmler, C. Unger, Benchmarking question answering systems, Semantic Web 10 (2019) 293–304. doi:10.3233/SW-180312.

[13] S. E. Sim, S. Easterbrook, R. C. Holt, Using benchmarking to advance research: A challenge to software engineering, in: Proceedings of the 25th International Conference on Software Engineering, ICSE 03, IEEE Computer Society, 2003, pp. 74–83.

[14] E. M. Voorhees, D. K. Harman, TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing), The MIT Press, 2005.

[15] J. Euzenat, C. Meilicke, H. Stuckenschmidt, P. Shvaiko, C. Trojahn, Ontology Alignment Evaluation Initiative: Six Years of Experience, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 158–192. doi:10.1007/978-3-642-22630-4_6.

[16] P. Shvaiko, J. Euzenat, E. Jimnez-Ruiz, M. Cheatham, O. Hassanzadeh, Proceedings of the 13th International Workshop on Ontology Matching, CEUR Workshop Proceedings, CEUR Workshop Proceedings, 2018. URL: http://ceur-ws.org/Vol-2288/.

[17] C. Bizer, A. Schultz, The berlin sparql benchmark, International Journal on Semantic Web and Information Systems (IJSWIS) 5 (2009) 1–24. doi:10.4018/jswis.2009040101.

[18] T. Catarci, M. F. Costabile, S. Levialdi, C. Batini, Visual query systems for databases: A survey, Journal of Visual Languages & Computing 8 (1997) 215–260. doi:10.1006/jvlc.1997.0037.

[19] R. García, R. Gil, J. M. Gimeno, E. Bakke, D. R. Karger, Besdui: A benchmark for end-user structured data user interfaces, in: The Semantic Web ISWC 2016, Lecture Notes in Computer Science, Springer, Cham, 2016, pp. 65–79. doi:10.1007/978-3-319-46547-0.

[20] S. K. Card, T. P. Moran, A. Newell, The keystroke-level model for user performance time with interactive systems, Commun. ACM 23 (1980) 396–410. doi:10.1145/358886.358895.

[21] O. Erling, I. Mikhailov, RDF Support in the Virtuoso DBMS, volume 221 of *Studies in Computational Intelligence*, Springer Berlin Heidelberg, 2009, pp. 7–24. doi:10.1007/978-3-642-02184-8_2.

[22] G. Vega-Gorgojo, M. Giese, S. Heggestyl, A. Soylu, A. Waaler, Pepesearch: Semantic data for the masses, PLOS ONE 11 (2016). doi:10.1371/journal.pone.0151573.

[23] E. Bakke, D. R. Karger, Expressive query construction through direct manipulation of nested relational results, in: Proceedings of the 2016

International Conference on Management of Data, SIGMOD 16, ACM, 2016, pp. 1377–1392. doi:10.1145/2882903.2915210.

[24] B. E. John, D. E. Kieras, The goms family of user interface analysis techniques: Comparison and contrast, ACM Trans. Comput.-Hum. Interact. 3 (1996) 320–351. doi:10.1145/235833.236054.