Interactive Relationship Discovery via the Semantic Web

Philipp Heim[1], Steffen Lohmann[2], and Timo Stegemann[3]

[1] Visualization and Interactive Systems Group, University of Stuttgart, Germany
philipp.heim@vis.uni-stuttgart.de
[2] DEI Laboratory, Carlos III University of Madrid, Spain
slohmann@inf.uc3m.es
[3] Interactive Systems and Interaction Design, University of Duisburg-Essen, Germany
timo.stegemann@uni-due.de

Abstract. This paper presents an approach for the interactive discovery of relationships between selected elements via the Semantic Web. It emphasizes the human aspect of relationship discovery by offering sophisticated interaction support. Selected elements are first semi-automatically mapped to unique objects of Semantic Web datasets. These datasets are then crawled for relationships which are presented in detail and overview. Interactive features and visual clues allow for a sophisticated exploration of the found relationships. The general process is described and the RelFinder tool as a concrete implementation and proof-of-concept is presented and evaluated in a user study. The application potentials are illustrated by a scenario that uses the RelFinder and DBpedia to assist a business analyst in decision-making. Main contributions compared to previous and related work are data aggregations on several dimensions, a graph visualization that displays and connects relationships also between more than two given objects, and an advanced implementation that is highly configurable and applicable to arbitrary RDF datasets.

1 Introduction

Today's world is complex, and so are the relationships within most knowledge domains. Even experts in a field are often not aware of all relationships that exist between certain elements of interest. However, overlooking relevant relationships can have fatal consequences in many situations. To name just two examples of the financial and medical domains: Not considering crucial relationships when developing a portfolio of investments might be disastrous for a broker. Even more fatal could be the result if a physician overlooks negative effects caused by a combination of several drugs.

Typical reasons for overlooking and therefore not considering relevant relationships in decision-making and related activities are:

1. A large number of relationships that cannot all be cognitively grasped.
2. Relationships that do not follow logical human thinking.
3. Indirect relationships that are hard to derive by purely cognitive reasoning.

Preprint of a paper to appear in the proceedings of the
7th Extended Semantic Web Conference (ESWC 2010)
published by Springer in LNCS vol. 6088, pp. 303-317
(http://dx.doi.org/10.1007/978-3-642-13486-9_21)

The Semantic Web offers suitable opportunities to assist humans in getting an overview on existing relationships. Since the information objects of the Semantic Web are interlinked via their properties, they altogether describe a "giant global graph" [3] that can be crawled for relationships[4] by appropriate algorithms [12]. That way, it is possible to find relationships that, in particular, deal with the above-mentioned challenges and therefore ideally supplement the relationships that are considered by human thinking.

However, since the information objects in the Semantic Web are ideally strongly interlinked, simply crawling for relationships and listing them is not sufficient in most cases. Instead, user-centered processes and interactive tools are needed that provide comprehensive assistance in the discovery of even very large numbers of relationships. Users must be able to efficiently get an overview on found relationships, to interactively explore them and to easily spot and separate relationships that are of relevance in a certain situation.

In this paper, we therefore define relationship discovery via the Semantic Web as a highly user-centered process. Even though the relationships are found automatically in the datasets, all steps of the process can be controlled and monitored by the user, allowing for a powerful combination of interactive and automatic mechanisms. The process is defined on an abstract level making it applicable in a wide range of domains. It consists of four steps: object mapping, relationship search, visualization, and interactive exploration.

With the RelFinder, we present a concrete implementation of this process. Each step is supported by sophisticated visualization and interaction techniques that aim to move the process on to the subsequent step. The mapping of selected elements to unique objects in the Semantic Web is supported by auto-completion and semi-automatic disambiguation features. Starting from these objects, an algorithm then automatically searches for relationships by following links in the dataset. After relationships are found, they get presented to the user by both a detailed view that shows a limited set of relationships in a graph visualization and an overview that shows the whole set of found relationships aggregated according to topological and semantic dimensions in lists. The aggregated representations can be used to determine what relationships should be shown in the graph visualization and to highlight nodes and edges according to their properties in the different dimensions. In addition, nodes in the graph visualization can be selected to get further information about them.

In the following sections we first define the process for interactive relationship discovery via the Semantic Web in general and present the RelFinder as a concrete implementation of it. With the RelFinder we aim to proof the applicability of our process definition and therewith also the potential of the general approach. We use a scenario to illustrate the benefits of using the RelFinder and evaluate it against existing approaches from the common Web. We conclude with a discussion and future work.

[4] Between two objects o_0 and o_n exists a relationship $o_0\ l_1\ o_1\ l_2\ o_2$, ... $l_{n-1}\ o_{n-1}\ l_n\ o_n$ $(n > 0)$ if for each link and the two neighboring objects $o_{i-1}\ l_i\ o_i$ $(0 < i \leq n)$ in the relationship, either $o_{i-1}\ l_i\ o_i$ or $o_i\ l_i\ o_{i-1}$ is a triple in the dataset (cp. [7]).

2 Process Definition

In this section we define the general process for our approach of interactive relationship discovery via the Semantic Web. We describe the process on an abstract level, independently from any concrete implementation or domain. We call it the ORVI process according to the initial letters of its four sequential steps (Object mapping, Relationship search, Visualization, and Interactive exploration). It can be used in any situation, in which relationships between certain elements are of interest. Knowing the labels of at least two elements is sufficient in order to trigger the process.[5] The selection of these elements is not part of the process as it can be realized in multiple ways. For instance, elements of interest can be manually chosen, the result of natural language processing or any other manual, automatic, or semi-automatic selection process.

In addition, it must be decided on which dataset the process is executed. This decision is also not part of the general process since it depends strongly on the domain in focus and the user goals. It might, for instance, be a large public dataset which covers general knowledge (such as DBpedia [4] or the LOD cloud[6]) or a domain-specific dataset that is privately hosted and maintained by a company.[7]

After the elements have been selected and the decision for a dataset has been made, the four steps of the ORVI process are sequentially executed. They are summarized in the following (cp. Fig. 1):

1. **Object Mapping**: As a common requirement of the Semantic Web, the selected elements must first be mapped to unique objects of the datasets. In order to minimize effort, manual disambiguation should only be required in cases where a unique automatic mapping is not feasible (e.g., in case of ambiguity). In addition, all manual disambiguation should be accompanied by appropriate user support (e.g., auto-completion).
2. **Relationship Search**: After all selected elements have been mapped to unique objects, the dataset is automatically crawled for relationships between these objects by appropriate algorithms. Since it is hard to generally determine which relationships are of relevance in a certain situation and which are not [1], filtering should at best only be used to clean the search results in this process step (e.g., suppressing cycles[8]). Thus, the general goal

[5] In contrast to related work, the process does explicitly not limit the elements between which relationships are to be found to only two but supports relationship discovery also between more than two elements.

[6] `http://linkeddata.org/`

[7] Since we defined the process with the average user in mind who has usually little or no expertise in Semantic Web topics, a dataset might be preselected in most cases that is appropriate for the user's tasks (i.e., contains interrelated objects that the user is interested in).

[8] "Suppressing cycles" means that the same object occurs at most once in the same relationship [9].

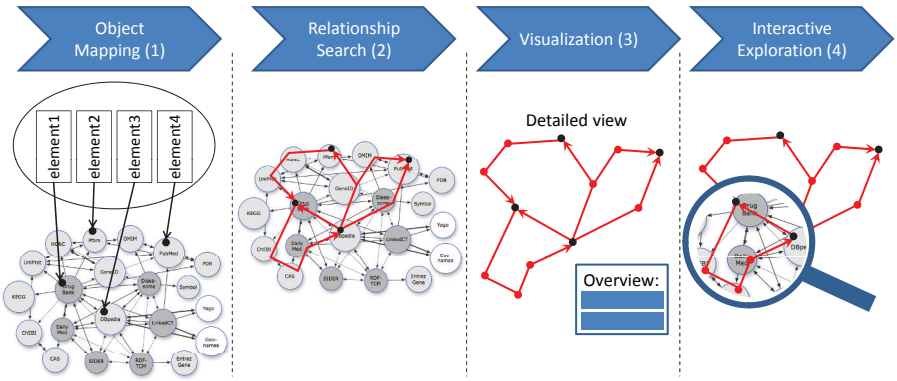

Fig. 1. The ORVI process: Selected elements are mapped to objects in the dataset (1) which are then used as starting objects to find relationships between them (2). The found relationships are visualized (3) and can be explored by the user (4).

in this step is to find as many relationships as possible since each single relationship could be valuable in a certain situation.[9]

3. **Visualization**: The found relationships are then presented to the user. The visualization must be capable to handle even large numbers of relationships. In many cases, only a small subset of the found relationships can be presented in a detailed view due to spatial limitations of the display area (usually, screen size is limited). Therefore, an overview is required that can aggregate the found relationships according to different dimensions (e.g., statistical, topological, or semantic ones) and thus facilitate users' understanding of the whole result set.
4. **Interactive Exploration**: The last step of the process is the interactive exploration of the found relationships. The main goal is the discovery of relationships that are of relevance in a certain situation. In order to reach this goal, interactive features (e.g., dynamic filtering) and visual clues (e.g., highlighting) are needed that enable a sophisticated exploration on the different views of the visualization.

3 RelFinder – An Implementation of the ORVI Process

We developed the RelFinder, a tool that demonstrates the applicability of the ORVI process and illustrates the support it can provide in real-world-contexts.[10] It is implemented in Adobe Flex[11] and gets compiled to a Flash movie which

[9] Though filtering should only be carefully applied in this process step, it might, however, in practice be necessary due to performance issues and/or resource limitations.

[10] A demo installation of the RelFinder can be tested online at: `http://relfinder.semanticweb.org`

[11] `http://www.adobe.com/products/flex`

runs in all Web browsers with installed Flash Player. SPARQL queries are generated on the client side and can be sent to any SPARQL endpoint that is available either locally or via the Web. Thus, the RelFinder can be used to find relationships in datasets of various domains without a need of modifying them.

In order to illustrate how the RelFinder implements the ORVI process and to show its application potentials, we present a complete walkthrough and describe the implemented features by means of a scenario. In this scenario, we simulate the situation of a business analyst who uses the RelFinder to explore a large number of relationships that were found between companies she is interested in. We assume the analyst to hold shares from the following German automotive companies: BMW, Porsche, Volkswagen, and MAN SE. Against the background of recent developments in the German automotive sector[12], the analyst needs a deeper understanding of the relationships between the four companies to establish an optimal trading strategy. She decides to use the RelFinder to get supported in these tasks.

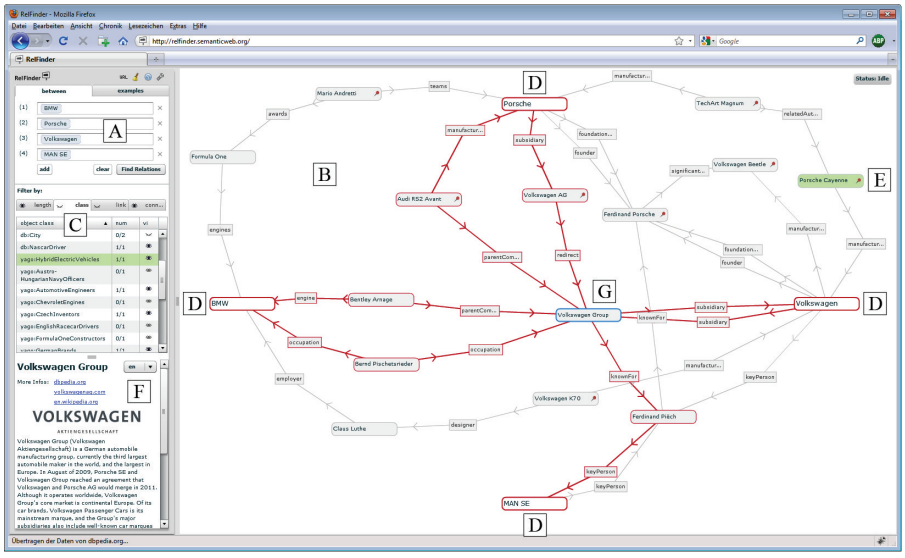

Fig. 2. Screenshot of the RelFinder with relationships between BMW, Porsche, Volkswagen and MAN SE found in the DBpedia dataset.

3.1 Object Mapping

The labels of the elements are entered as strings in separate input fields in the upper left corner of the user interface (Fig. 2, A). Initially, two input fields are

[12] VW to buy half of Porsche by 2010. BBC News. 2009-10-20. `http://news.bbc.co.uk/2/hi/business/8317010.stm`. Retrieved 2009-12-13.

presented but additional ones can easily be added. The business analyst from the scenario, for instance, adds two more input fields, since she is interested in relationships between four elements ("BMW", "Porsche", "Volkswagen" and "MAN SE"). As detailed in the following, the RelFinder assists the mapping of entered strings to objects in the dataset in various ways.

While the user is entering the element labels, possibly matching objects are displayed in a drop-down list below each input field. These suggestions are retrieved by SPARQL queries against selected properties of the dataset[13] and are ranked according to their popularity, if supported[14] (else according to string similarity using the Jaro-Winkler distance measure).

The user is not forced to use the auto-completion feature while she inputs her element labels.[15] Hence, all elements that have not been disambiguated via auto-completion need to get mapped to objects of the dataset before the relationship search can be performed. The RelFinder implements a semi-automatic mapping strategy that aims to maintain an optimal trade-off between user effort and false mappings (which should both be minimized). Automatic mapping is triggered in two cases: 1) if the user input matches exactly the property string of an object and the object's ranking value is above a certain threshold compared to the top-ranked object[16] , 2) if the ranking value of the top-ranked object is above a certain threshold compared to the object that is ranked second. Hence, both thresholds are relative; their values are configurable so that they can be optimized for each dataset separately.

Following this implementation, the user inputs "BMW" and "Porsche" can be automatically mapped to corresponding objects of the DBpedia dataset in the scenario, since they meet the first case (exact match and top-ranked object). "Volkswagen" is ranked second but can also be automatically mapped since it is an exact match and its ranking value is above the defined threshold compared to the top-ranked object ("Volkswagen Group").[17] The user input "MAN" cannot be automatically mapped as it has no exact match with one of the `label` properties of DBpedia and as the ranking value of the top-ranked object is not above the defined threshold.

If no automatic mapping is feasible, manual disambiguation is supported by an enhanced suggestion list displayed in a pop-up dialog. This dialog provides additional features such as links to the URIs of the suggested objects or a possi-

[13] Configuration allows to freely define the queried dataset and the property types that are included in the search for suggestions (by default, the `label` property of the RDF Schema vocabulary is used).

[14] The RelFinder uses the `count` command for popularity ranking that is, however, not (yet) part of the SPARQL standard but was proposed as extension.

[15] Demanding a manual disambiguation for all elements would be against the ORVI principle of minimizing user effort.

[16] This case is only applicable for the popularity ranking based on the `count` value.

[17] The suggestions were ranked according to their `count` value (not string similarity) in the scenario. This results in 'Volkswagen Group' being ranked higher than 'Volkswagen'.

bility to directly input and validate object URIs.[18] The dialog can also be used to correct false disambiguations. Thus, the RelFinder provides twofold support for manual disambiguation with lightweight auto-completion during user input and more advanced functionalities in unclear cases (i.e., if no automatic mapping is possible or if entered strings match with more than one object). Ultimately, it ensures that each user input gets mapped to a unique object of the dataset with minimized effort for manual disambiguation. These objects serve as *starting objects* in the subsequent relationship search.

3.2 Relationship Search

Relationship search can be realized in various ways. The RelFinder uses an algorithm that was first introduced in [9] and further developed in [6]. It defines several parameters that can be configured to adapt relationship search and that are also used in related algorithms [1,7], such as *length*[19] and *directionality*[20].

As defined by the ORVI process, the general goal of this process step should be the extraction of as many relationships as possible since each single relationship can be valuable depending on the situation. Therefore, the parameters of the relationship search should generally be configured with this goal in mind. However, this idealized vision cannot be fully realized in many situations due to performance reasons or resource limitations. Thus, the RelFinder allows a configuration of the *length* and *directionality* parameters. In addition, it allows the definition of properties that should be ignored as they provide no benefits for relationship discovery (e.g., contain no valuable semantics).

The RelFinder installation of the scenario has been configured to search for all relationships that have a maximum length of two and contain no more than one change in the link directions. Furthermore, it has been defined to ignore the `wikilink` property of the DBpedia ontology [4] since it simply represents hyperlinks and provides therefore no valuable semantics for relationship discovery. In addition, all links of types `subject` from the SKOS vocabulary[21] and `type` from the RDF Schema vocabulary are ignored. Overall, this results in 64 relationships that were found between the four starting objects by the algorithm (26 of length 1 and 38 of length 2), with 16 object classes belonging to the YAGO ontology[22],

[18] If the URIs of the objects are known in advance, they can also be directly entered in the input fields.

[19] We define the "length" of a relationship by the number of intermediate objects that are contained within this relationship. According to this definition, direct relationships (i.e., the starting objects are directly linked without intermediate objects) are of length 0 and indirect relationships that contain $x + 1$ links and x intermediate objects are of length x.

[20] The "directionality" expresses the number of changes in the directions of the links that a relationship consists of. See [7] for more information about this concept.

[21] `http://www.w3.org/2004/02/skos/`

[22] `http://www.mpi-inf.mpg.de/yago-naga/yago/`

four to OpenCyc[23] and three to DBpedia, and with 37 different link types.[24] The next challenge addressed by the RelFinder in accordance with the ORVI process is the visualization of these relationships in a user-centered way.

3.3 Visualization

The visualization of relationships in the RelFinder is organized in two separate parts, a detailed view of a limited number of relationships in the main area (Fig. 2, B) and an overview of the whole set in the sidebar on the left (Fig. 2, C).

Detailed View: The detailed view consists of a graph visualization that represents objects as nodes and links as directed labeled edges between them.[25] The nodes representing starting objects (all objects the selected elements got mapped to) are marked by a stronger border (Fig. 2, D) and are elliptically arranged in an oval on fixed positions on the screen. The found objects and links that are contained in the relationships between the starting objects span a graph between these fixed nodes and get automatically arranged by a force-directed layout [5].

Since the found energy level is not necessarily a global minimum, i.e. the graph layout is probably suboptimal, or the user might want to arrange nodes according to individual preferences, single nodes can be picked by the user and pinned at fixed positions on the screen decoupling them from the automatic layout.[26] Whether a node is pinned or not is indicated by a needle symbol (Fig. 2, E). By clicking the needle symbol of an already pinned node, this node releases its fixed position and gets again coupled to the automatic layout.

The graph building is animated incrementally by displaying the found relationships one after the other. This has three advantages: 1) An early impression of found relationships: Even though the search is still in progress, first results are already displayed; 2) Time for cognitive processing: Having a delay between each newly added relationship gives users time to adjust to the graph visualization (especially new users without any familiarity with graphs are often overstrained by their complexity); 3) An easy traceability of relationships: Since relationships are stepwise added to the graph, the layout needs to integrate only a few nodes and edges at a time and thus the nodes are changing their positions rather smoothly allowing the eyes to keep track of them.

Overview: In the overview, the found relationships get aggregated according to two topological and two semantic dimensions. The two topological dimensions are:

[23] `http://www.cyc.com/opencyc/`

[24] These were the results at time of writing.

[25] Actually, edge labels are also handled as nodes in the graph layout for readability reasons.

[26] We call the manual adaption of the positions of single nodes in the graph *pick-and-pin* operation.

– *Relationship lengths* (see Section 3.2 and Footnote 19).
– *Connectivity levels* of found objects contained in the relationships.[27]

The two semantic dimensions are:

– *Link types* contained in the relationships.
– *Classes* of found objects contained in the relationships.

For each of the four dimensions, all aggregated properties of the relationships are organized in lists that can be accessed via a tab menu in the sidebar (Fig. 2, C). Each row in the lists stands for a distinct property and contains its label, the numbers of aggregated relationships that are visible in the detailed view, the total number of aggregated relationships, and information whether the property is generally set visible or not (by an eye symbol).

If the number of found relationships exceeds a certain threshold, which defines the maximum number of relationships that can clearly be arranged in the graph visualization, not all of them get visualized in the detailed view initially. In order to decide which relationships should initially be visible in the graph visualization, we define a *measure of importance*. The questions are: In which relationships are the users most likely interested in? And even more important: What properties can be used to calculate the importance of relationships automatically?

Unfortunately, the definition of the importance of relationships seems to change profoundly with the current user task. Nevertheless, since we do not ask for the current user task in advance, at least in the current RelFinder implementation, we chose the relationship length as a general property to automatically measure the importance of relationships (cp. [1]). Thus, direct relationships or relationships with few intermediate objects are more likely to be shown in the initial graph visualization than relationships with more intermediate objects. If a property, as for example a certain relationship length, is set invisible, this is indicated in the corresponding row of the list in the overview by a closed eye symbol.

3.4 Interactive Exploration

The found relationships in the RelFinder can be explored by interacting with both the properties in the overview and the nodes and edges in the detailed view. The user can control what relationships are shown in the graph visualization, can highlight nodes, edges and relationships, and can get additional information about selected objects.

By interacting with the properties in the overview, relationships can be shown or hidden in the detailed view and nodes and edges can be highlighted. Clicking on the open eye symbol of a certain relationship length removes all relationships of this length from the graph visualization (e.g., all direct relationships).

[27] We define the connectivity level of a found object by the number of distinct starting objects it connects in the graph. Each found object connects at least 2 and at most all starting objects.

Clicking on the open eye symbol of a certain link type removes all relationships that contain links of this type and thus allows users to ignore links that are possibly irrelevant for the current analysis (e.g., "companyType"). Clicking on the open eye symbol of a certain connectivity level or ontology class removes all relationships that contain corresponding objects. This facilitates focusing on relationships via objects that belong to certain classes (Fig. 2, C) or that connect a certain number of starting objects (e.g., all starting objects). Selecting one of the properties in any of the four aggregation dimensions highlights all corresponding nodes and edges in the detailed view (e.g., selecting the class "HybridElectricVehicles" of the YAGO ontology highlights the "Porsche Cayenne" node in Fig. 2, E).

In the detailed view, nodes can be selected in order to highlight related information in both the detailed view and the overview. First, the properties of the selected object are highlighted in the aggregated lists to provide information about its class and connectivity level. Second, objects in the detailed view that share the same properties as the selected one are highlighted to provide some awareness of similar information in the graph. Third, all relationships that contain the selected object are highlighted as "red threads" in the detailed view helping the user to visually track relationships within the graph (e.g., all relationships that contain "Volkswagen Group" in Fig. 2, G). Furthermore, additional information about the selected node, such as an image and a short description, is shown in the sidebar (Fig. 2, F) to help users to interpret possibly unknown objects that were found between the starting objects and thus to better understand the found relationships in general.

The business analyst in the scenario interactively explores the found relationships mainly by using the semantic dimensions "class" and "link type". Since relationships based on spatial correlations (e.g., companies located in the same city) or on similar organizational structures (e.g., companies with a chief executive officer) are not of much interest to her, she sets all relationships that contain objects like "cities" or "countries" and links like "companyType" and "deathPlace" invisible in the detailed view. Thus, the graph visualization shows only a limited but possibly valuable set of found relationships like those containing persons (e.g., "Bernd Pischetsrieder"), products (e.g., "Porsche Cayenne"), or organizations (e.g., "Volkswagen Group"). Such relationships can be indicators for personal, structural, or financial connections and dependencies between the four automotive companies and are thus of interest to the business analyst. A very prominent insight that can be gained by studying the graph visualization in Fig. 2 is the important role of the "Volkswagen Group" as a connector of all four starting objects. Dependencies based on this object should be considered very carefully when developing a trading strategy.

4 Evaluation

We performed a user study in which we compared relationship discovery via the Semantic Web according to the ORVI with relationship discovery as it can

be commonly performed by Web users nowadays. We used the RelFinder as implementation of the ORVI process and decided for Google and Wikipedia as reference applications for relationship finding in the "common" Web.[28]

4.1 Study Design

The user study consisted of three tasks that had to be accomplished with all three applications (Google, Wikipedia, and RelFinder). One relationship had to be found in the first task and three relationships in the second. The third task also asked for three relationships but additionally defined an object class that should be included in the found relationship (e.g., person, location, etc.). The relationships had always to be found between two given elements that varied across the applications.[29] Likewise, the class varied that was asked for in the third task.

The elements were selected from three thematic categories – persons, locations, and culture –, whereas each study participant received elements from each category. We run a pre-test to ensure that relationships between the given elements could be found in all three applications with relatively small effort.[30] For example, one text for the third task in the category *culture* was: "Name three movies that relate Quentin Tarantino and Samuel L. Jackson. Give also the kind of relationship" (with "Quentin Tarantino" and "Samuel L. Jackson" as given elements and "movie" as given class).

The time limit for each task was three minutes. If a study participant was not able to solve a task within this time, its execution was aborted and it was continued with the next task. The study participants were advised to disregard previous knowledge as much as possible. They were only allowed to name relationships they evidently found with the applications. Correspondingly, they were not permitted to use further terms for search that are based on their own knowledge. The DBpedia dataset was used for the RelFinder in the user study. We ensured that the study participants did not access Web pages from Wikipedia via Google. Further restrictions were not made so that participants could use their own preferred strategies to solve the tasks in each application.[31]

[28] We selected Google and Wikipedia for the following reasons: 1) These two applications were mentioned by most people in an informal poll when we asked for Web applications they would use for relationship discovery, 2) They are very popular representatives for the application classes search engine and Wiki, 3) We expected a high familiarity with these applications among the study participants lowering the need for explanations or trainings.

[29] We restricted relationship finding to only two elements in the user study since it is highly challenging to find relationships between more than two elements via Google and Wikipedia. Even more challenging is the selection of examples that work in all three applications if more than two given elements are considered.

[30] For instance, we ensured in the case of Google that relationships were preferable displayed on the first, at least on the second result page.

[31] The high diversity of the datasets accessed by the applications (DBpedia, Wikipedia, Google-indexed WWW) could not be avoided but was also not considered as a

4.2 Procedure

Twelve participants, mainly students, took part in the study. Their familiarity with Google and Wikipedia was high[32] (Google: M=9, SD=1.2; Wikipedia: M=8.8, SD=1.0)[33].

The study began with a short introduction and explanation of the three tasks. Subsequently, the three applications (Google, Wikipedia, and RelFinder) were presented in systematically varying order. The tasks had to be accomplished with all three applications.[34] For each task, the participants had to rate their satisfaction with the found relationships.[35] In addition, after the presentation of all three tasks, they had to rate how much the respective application helped in the discovery of relationships.[36]

4.3 Results

Fig. 3 shows the average number of tasks that were solved with each application (upper left diagram, separately listed for each task type). In this direct comparison, the RelFinder provides generally the best support. A similar result is shown by the satisfaction values that were measured after each task's solution (Fig. 3, lower left diagram). These results indicate that the ORVI process as implemented in the RelFinder is generally preferred to relationship discovery as it is currently possible in common Web applications such as Google and Wikipedia. This is also reflected by the answers on the statement: "If I would need to search for relationships between two elements, I would use the following tool". Eight participants decided for the RelFinder and only two for Google and two for Wikipedia.[37]

The diagram on the right of Fig. 3 shows the ratings of the participants on the dimensions *efficiency*, *satisfaction*, and *control* that we collected directly after the presentation of each application via the twelve Likert-scaled items. The relationship discovery support as implemented in the RelFinder reached once again the highest satisfaction values on average. The ratings on the dimension *control* were nearly the same for all three applications. This result indicates

problem since the pre-tests ensured that all tasks could principally be solved with all applications and within the given time frame of three minutes.

[32] Familiarity was measured on a scale of 1 to 10 (with 1 = "unfamiliar" and 10 = "very familiar").

[33] M = mean; SD = standard deviation

[34] Since the given elements in the tasks were presented in the same order each time, they also varied systematically across the applications.

[35] Satisfaction was measured on a scale of 1 to 10 (with 1 = "not satisfied at all" and 10 = "totally satisfied").

[36] Support was measured according to twelve pre-defined items on a five-point Likert scale (with 1 = "not agree" and 5 = "fully agree"). The items were then mapped to aggregated measures on the dimensions *efficiency*, *satisfaction*, and *control*.

[37] The decision for Google and Wikipedia was mainly motivated with a higher familiarity with these applications.

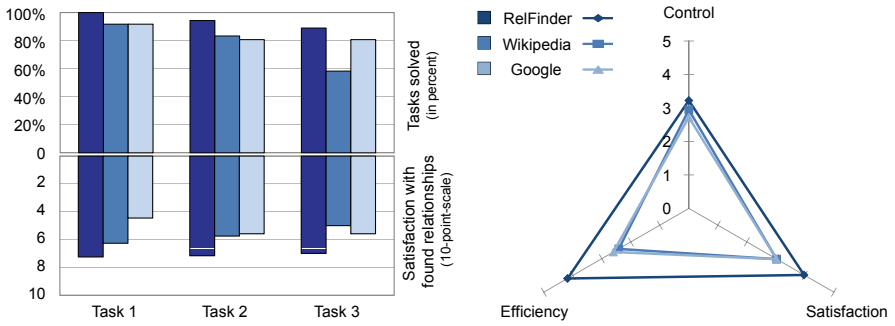

Fig. 3. Left: Average number of tasks solved (in percent) and average value of satisfaction with the task solutions (on scale of 1 to 10) for each application and task type; Right: User ratings for the three applications on the dimensions control, satisfaction, and efficiency (measured by 12 items on a five-point Likert-scale).

a successful interaction design of the RelFinder since familiarity with Google and Wikipedia was much higher among the participants. The *satisfaction* values support this interpretation and the results given above. The largest differences shows the *efficiency* dimension: The RelFinder has foremost been considered as much more efficient than the other two applications.

5 Discussion

Overall, the RelFinder performed very well in the user study. This is not surprising since it has been developed exactly for the kind of tasks that were investigated in the study. Nevertheless, the results demonstrate the high applicability of this approach and the general potentials and benefits it offers for relationship discovery compared to common Web applications. Interestingly, the RelFinder was considered as very efficient, although the manual disambiguation and relationship search took longer time than entering and searching for terms in Google and Wikipedia. It seems as if this additional effort pays off and is ultimately appreciated by the users when they recognize its benefit. These results argue for the generally high suitability of the Semantic Web to support relationship discovery. They also argue for viewing relationship discovery as a highly user-oriented process that consists of several interactive steps, as proposed by our approach.

5.1 Related Work

This view on relationship discovery as an iterative and highly interactive process is a main difference to related work which mainly investigates querying strategies and algorithms for relationship search in the Semantic Web. Considerable

research in this area has been conducted in the SemDis project[38]. A specific focus of this project was on the development of ranking measures for found relationships. The authors distinguish between semantic and statistical metrics and define a SemRank value that combines these metrics in different configurations [2]. In contrast to our topological and semantic classification dimensions, their ranking criteria do not include aggregations. However, our dimensions showed to provide very powerful assistance in the interactive exploration of the found relationships. This limitation holds also for the "DBpedia Relationship finder" [9] that has largely inspired our work and provided the basic algorithms for relationship search. The found relationships are simply listed as text strings in this tool but are neither aggregated nor extended by interactive functionalities.

Another related approach is SPARQLeR [7] that defines additional language constructs for SPARQL designed for relationship search. Unfortunately, SPARQLeR is not supported by common endpoints and could therefore not be used in the RelFinder. Further related work can be found in the research areas of ontology matching, learning, and enrichment. However, these approaches are rather interested in relationships on the conceptual level [11]. More distantly related research on the identification of semantic relationships in other data sources (e.g., text documents [10]) provides only few valuable input to the topic of interactive relationship discovery via the Semantic Web.[39] Though information seeking processes are generally a topic in related work [8], to the best of our knowledge it exists no approach for relationship discovery via the Semantic Web that covers the whole process, from object mapping to interactive exploration. Our approach aims to close this gap and, with the RelFinder, proposes an implementation that applies interactive relationship discovery via the Semantic Web to real-world-contexts.

5.2 Conclusion and Future Work

The most notable difference of our approach compared to related work is the emphasis on the *interactive* aspect of relationship discovery. In this understanding, "real" discovery is only possible with a human involved, since only the user can ultimately decide if a found relationship is relevant in a certain situation or not. However, this notion of "discovery" does explicitly not exclude any pre-selection and ranking of relationships as long as these are meant to support the interaction and do not restrict search results. An improved pre-selection of search results that are initially presented in the detailed view is therefore a main topic for future work. This will require some kind of context- or situation-awareness in order to adapt the pre-selection criteria to the users' information needs. Future work includes also the application of the RelFinder to use cases of different domains (e.g., health care, eLearning, etc.) and on datasets of different size (e.g., LODD[40], Semantic Wikis datasets, etc.) in order to get an improved understanding of the application potentials, benefits, and scalability of our approach.

[38] `http://lsdis.cs.uga.edu/projects/semdis/`
[39] Refer to [12] for an overview on different types of semantic relationships in the Web.
[40] `http://esw.w3.org/topic/HCLSIG/LODD`

Acknowledgments

We are grateful to Sebastian Hellmann and Jens Lehmann for the fruitful discussions and their contributions to the RelFinder implementation. We also thank Jürgen Ziegler for his valuable input and Lena Tetzlaff for assisting us with the user study. Last but not least, we thank Jörg Schüppel, Jens Lehmann, and Sören Auer for the development of the "DBpedia Relationship Finder" [9] which has largely inspired this work and provided basic algorithms for relationship search.

References

1. Boanerges Aleman-Meza, Christian Halaschek-Wiener, I. Budak Arpinar, Cartic Ramakrishnan, and Amit P. Sheth. Ranking complex relationships on the semantic web. *IEEE Internet Computing*, 9(3):37–44, 2005.
2. Kemafor Anyanwu, Angela Maduko, and Amit P. Sheth. SemRank: ranking complex relationship search results on the semantic web. In *Proc. of the 14th International World Wide Web Conference (WWW '05)*, pages 117–127, 2005.
3. Tim Berners-Lee. Giant Global Graph [last access: 2009/12/20]. `http://dig.csail.mit.edu/breadcrumbs/node/215`, 2007.
4. Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. DBpedia – a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009.
5. Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21(11):1129–1164, 1991.
6. Philipp Heim, Sebastian Hellmann, Jens Lehmann, Steffen Lohmann, and Timo Stegemann. Relfinder: Revealing relationships in RDF knowledge bases. In *Proc. of the 4th International Conference on Semantic and Digital Media Technologies (SAMT'09)*, volume 5887 of *LNCS*, pages 182–187. Springer, 2009.
7. Krys Kochut and Maciej Janik. SPARQLeR: Extended SPARQL for semantic association discovery. In *Proc. of the 4th European Semantic Web Conference (ESWC '07)*, volume 4519 of *LNCS*, pages 145–159. Springer, 2007.
8. Carol Collier Kuhlthau. *Seeking meaning: a process approach to library and information services*. Libraries Unlimited, 1993.
9. Jens Lehmann, Jörg Schüppel, and Sören Auer. Discovering unknown connections – the DBpedia relationship finder. In *Proc. of the 1st Conference on Social Semantic Web (CSSW'07)*, 2007.
10. Cartic Ramakrishnan, Krys Kochut, and Amit P. Sheth. A framework for schema-driven relationship discovery from unstructured text. In *Proc. of the 5th International Semantic Web Conference (ISWC '06)*, pages 583–596, 2006.
11. Marta Sabou, Mathieu d'Aquin, and Enrico Motta. Relation discovery from the semantic web. In *Proceedings of the Poster and Demonstration Session at the 7th International Semantic Web Conference (ISWC '08)*, volume 408 of *CEUR*. CEUR-WS.org, 2008.
12. Amit P. Sheth and Cartic Ramakrishnan. Relationship web: Blazing semantic trails between web resources. *IEEE Internet Computing*, 11(4):77–81, 2007.