Usability of Keyword-Driven Schema-Agnostic Search

A Comparative Study of Keyword Search, Faceted Search, Query Completion and Result Completion

Thanh Tran¹, Tobias Mathäß², and Peter Haase²

¹ Institute AIFB, KIT, Karlsruhe, Germany
ducthanh.tran@kit.edu

² fluid Operations, Walldorf, Germany
{tobias.mathaess,peter.haase}@fluidops.com

Abstract. The increasing amount of data on the Web bears potential for addressing complex information needs more effectively. Instead of keyword search and browsing along links between results, users can specify the needs in terms of complex queries and obtain precise answers right away. However, users might not always know the query language and more importantly, the schema underlying the data. Motivated by the burden facing the data Web search users in specifying complex information needs, we identify a particular class of search approaches that follow a paradigm that we refer to as *schema-agnostic*. Common to these search approaches is that no knowledge about the schema is required to specify complex information needs. We have conducted a systematic study of four popular approaches: (1) simple keyword search, (2) faceted search, (3) result completion, which is based on computing complex answers as candidate results for user provided keywords, and (4) query completion, which is based on computing structured queries as candidate interpretations of user provided keywords. We study these approaches from a process-oriented view to derive the main conceptual steps required for addressing complex information needs. Then, we perform an experimental study based on established conduct of a task-based evaluation to assess the effectiveness, efficiency and usability.

1 Introduction

Motivation. Most of the information needs are nowadays addressed by using Web search engines. Commonly, the type of search supported by these search engines is based on the *keyword search* paradigm. Using keywords, the user finds the resource of interest (informational search) or often, obtains some results as starting points (navigational search) first, from which further information is then discovered and retrieved via additional browsing. This paradigm has proven to be intuitive as well as effective in addressing simple information needs. More complex needs however require users to try out different keywords, and to browse and navigate along the complex Web space.

Recently, large amounts of structured data have been made publicly available. Two examples are data that is associated with Web pages in the form of RDFa¹ and Linked Open Data² (LOD), a large collection of linked datasets available in RDF that ranges

¹ A W3C Recommendation: <http://w3.org/TR/xhtml1-rdfa-primer/>

² <http://linkeddata.org/>

over many different domains. Web data like this is in the order of tens of billions RDF triples and also, it is rapidly growing. This development opens new ways for addressing complex information needs. We illustrate this with the following scenario:

Example 1. Mary is a novice computer science student at KIT. She is eager to learn more about this vast research field and decided to find *information about research work of researchers at AIFB*.

With traditional Web search, Mary searches and browses to find researchers at AIFB first. Then, another round of search and browsing is needed to find information about their work. Browsing is performed purely based on traversal along hyperlinks and keyword-based navigational search. The Web space is complex, containing large amounts of hyperlinks and Web pages that are relevant to a navigational query. Solving this task is thus time consuming. Using the Web of data, this complex information need can be addressed using one single query. For this, Mary specifies the need using a structured query language such as SPARQL and obtains right away the precise answers.

The example illustrates that potentially complex information needs can be addressed more efficiently using Web data. However, it also makes clear that users face the burden of specifying complex needs. Constructing structured queries requires the user to know the language and more importantly, the underlying schema. This cannot be assumed for the Web of data, which contains large amounts of evolving datasets.

Schema-agnostic Search. Usable query interfaces have long been an active field of research. In this work, we identify and study search approaches, which do not require users to know the schema underlying the data. Clearly, *keyword search* is a popular search interface that does not require prior knowledge about the schema. As discussed, this is a simple and intuitive paradigm that is very effective for navigational search, i.e. to obtain some starting points. From these points, users however have to do further navigation and exploration to address complex information needs. More complex needs can be expressed in terms of natural language (NL) questions. This type of interaction is supported by *NL query interfaces* [1]. Besides NL interfaces, the database community has also been investigating the use of keywords for search [2,3,4,5,6,7]. Unlike keyword search used on the Web, which focuses on simple needs, the keyword search elaborated here is used to obtain more complex results. Instead of a single set of resources, the goal is to compute complex sets of resources and their relations. We refer to this work as *result completion*, as it can be implemented to suggest completions in the form of candidate results, given the user provided keywords [8]. As an alternative to computing candidate results, candidate interpretations in the form of structured queries can be computed [9]. Given the provided keywords, completions in the form of queries are presented from which the user chooses the intended one to obtain the results in a subsequent step. We refer to this work as *query completion*. For specific domains and specific needs, experts predefine the types of complex queries that can be asked, and make them usable to lay users via customized forms. These *form-based query interfaces* have proven themselves effective for repetitive queries. They address common needs but fail on ad-hoc information needs. Another search paradigm that gains momentum is *faceted search* [10,11]. In fact, it can be seen as a special kind of form-based search. As opposed to a typical form-based interface, a faceted search interface is not restricted to the fields chosen by programmers but might contain any facets. Typically,

facets are computed on-line for the given result set. Based on these facets, the user explores and iteratively refines the result sets. Finally, less popular but also worth mentioning are graphical query interfaces [12]. The main contributions of this paper are as follows:

- Motivated by the burden facing the data Web search users in specifying complex information needs, we identify a particular class of search paradigms that we refer to as schema-agnostic approaches.
- We conduct a systematic study of four widely used approaches representing the class of keyword-driven schema-agnostic search. In particular, we focus on keyword search, query completion, result completion and faceted search, which all make use of an initial set of user keywords. We take a process-oriented view to investigate what conceptual steps these approaches require to address complex information needs.
- In the experiment, we carry out a task-based evaluation based on a large dataset and 19 participants. Based on the results, we derived the conclusions that keyword search and faceted search are effective and efficient for tasks which involve simple information needs. For complex needs, users proved to be more efficient with completion-based approaches and also, preferred them in terms of usability.

The paper is structured as follows: In Section 2 we explain the data and query model we assume for our work and define the problem of specifying queries in this context. In Section 3, we introduce a generic search process, based on which we analyze the different schema-agnostic search paradigms. We describe our experimental study in Section 4 and conclude in Section 5.

2 Problem Definition – Specifying Complex Information Needs

In this section we present the data and query model, in terms of which we describe the problem of specifying complex information needs.

We consider data represented using a general graph-structured data model. We distinguish two types of vertices in the graph: vertices representing entities and vertices representing data values. Edges may represent relations (connecting entities) or attributes (connecting entities with data values). We consider a predefined edge *type*, which denotes the class membership of an entity. Note that the data model is similar to that of RDF³, but general enough to capture also web documents, XML as well as relational data.

Additionally, a data graph can be described with a schema that defines the semantics of classes and their relations. The schema itself can be represented as a graph, with classes, data types, relations and attributes as vertices. Special edges, including *subclassof*, *domain*, *range*, are used to represent class hierarchies, the domains and ranges of relations and attributes, etc.

The information needs of users we are dealing with in this work can be represented as conjunctive queries. A query of this type is basically a conjunction of query atoms,

³ The intuitive mapping from RDF to this is, resources correspond to entities, properties to either relations or attributes and literals to data values.

which are constructed using binary predicate symbols, constants and variables. The predicate symbols are drawn from labels of relations and attributes specified in the data schema. Conjunctive queries represent a relevant fragment of first-order queries that is capable of expressing a large portion of relational queries (relational algebra). The vast majority of languages in practice falls into this fragment, including large parts of SQL, i.e. the select-from-where pattern, and SPARQL, i.e. the basic graph matching pattern. We now discuss common information needs that can be expressed with this model:

- **Entity Search.** In the IR community, this is also commonly known as navigational search. It is typically used as an entry point to the system, which is an entity such as a product, a Web page or a document in the collection. The user already knows the existence of the entity and uses the search function as a shortcut to this. Since the result is more or less known, the information need is expressed rather precisely, often with constants to refer to the particular entity, e.g. to use $name(x, AIFB)$ to search for the entity with the name *AIFB*.
- **Fact Search.** This is used in situations where the user is interested in a certain fact, like a phone number of a friend or the current temperature in San Francisco. While entity search involves one or several entities as results, this kind of search produces facts in the form of specific attribute values. Also, it is different to entity search in that it is not navigational known-item search, but rather, the purpose is to find unknown information. Thus, it is also referred to as informational search. For instance, the fact query $name(x, AIFB) \wedge location(x, y)$ searches for the specific location of *AIFB*.
- **Relation Search.** This is another type of informational search, where the goal is to gather not only information about a specific entity, but to find a complex set of entities, and especially, how they are related. In terms of conjunctive queries, this type of search allows to retrieve n-ary tuple sets as results. The running example is of the type relation search which can be expressed as $type(y, researcher) \wedge worksAt(y, x) \wedge name(x, AIFB) \wedge researchWork(y, z)$.

The examples demonstrate that different types of information needs can be represented. However, particularly for complex relation search, query construction is a difficult task. Users are required to know not only the query language but also the underlying schema, i.e. to know which predicate symbols to be used for query construction. In the context of querying Web data, this is too large a burden for the users. The data as well as the schema are evolving, making it difficult for users to keep track of the changes and to be always knowledgeable. Moreover, search on the Web often involves unknown resources. In this case, the assumption should rather be that users have no schema knowledge.

3 Keyword-Driven Schema-Agnostic Search

In this section, we identify and study the category of keyword-driven schema-agnostic search paradigms. As discussed, common to schema-agnostic approaches is that no knowledge about the schema is required to specify complex information needs. Work extensively studied and also applied in practice that falls into this category includes

keyword search, NL search, form-based search, faceted search, graphical query interfaces, and query as well as result completion. The last two paradigms can be regarded as extensions of auto-completion. However, auto-completion known from standard search interfaces operates at the level of words. Basically, a dictionary of terms is used to suggest candidate word completions as the user types. The completion approaches investigated here operate at a higher level, i.e. at the level of queries and results. In this paper, we focus on the study of schema-agnostic paradigms that are based on the use of keywords, i.e. keyword-driven schema-agnostic search. In particular, we will discuss keyword search, query completion, result completion as well as faceted search. We will show that, from a process-oriented point of view, also faceted search relies on the use of keywords to obtain an initial result set and thus, falls into this category.

3.1 Process of Specifying Complex Information Needs

We will firstly investigate the search process. The goal is to identify the main conceptual steps each approach requires to address complex information needs. In Figs. 1 and 2, we illustrate these steps for the approaches under investigation. Common to the approaches are the phases (1) specifying needs using keywords, (2) inspecting initial results and (3) further browsing, analysis and retrieval of the final results. With keyword search, there is not much help from the system to retrieve complex results. The user needs to analyze and browse at the level of resources to collect the items of interest. Faceted search supports users in inspecting, analyzing and browsing at the level of facets. With completion-based approaches, users might retrieve complex answers right away – in the form of tuples that contain resources and their relations. We will now discuss state of the art techniques and the individual steps involved in each approach.

3.2 Keyword Search and Resource-Based Browsing

State of the art. Keyword search is a paradigm commonly used by Web search engines to enable the retrieval of documents. Recently, a number of Semantic Web search engines such as Hermes [13], FalconS [14] and Sindice [15] have been developed, which also primarily rely on keyword search. Instead of documents, semantic entities (i.e. RDF resources) are returned. All these search systems are built upon the same basic concepts: (1) term-based representation of resources and queries (also called bag-of-words) and

Fig. 1. Process of Addressing Complex Information Needs

Fig. 2. Process for the Running Example

(2) term-based matching of queries against resources. Different matching and ranking techniques are employed. The ones frequently used are based on (an adopted version of) extensively studied information retrieval (IR) models such as the vector space model [16] and the probabilistic models [17]. The ranking schemes employed by these models leverage the discriminative quality of terms such as TFIDF and authority of resources derived via PageRank [18]. Besides, also structure information might be used for ranking. In BM25F [19] for instance, different weights are used to associate fields of documents (or generally speaking: properties of resources) with varying degrees of importance. This is to implement the intuition that certain properties are more discriminative and thus, more important than others, e.g. terms matching *name* shall yield higher scores than terms matching *comment*. Besides the class of probabilistic models such as Okapi BM25 and its extensions, current state of the art IR approaches have embraced the use of language modelling [20]. This approach assumes that resources and expressions of information needs are objects of the same type, and assesses their match by adopting techniques of language modeling from speech and natural language processing.

In the actual implementation, the bag-of-words representation of documents are stored in an inverted index (along with scores). While commercial solutions rely on their own infrastructures, Lucene has been widely used for the implementation of Semantic Web search engines, Sindice, Hermes and FalconS in particular.

Process. Using a keyword search system, the user starts by entering a list of keywords. The system matches this keyword query against resources to return a ranked list of results. This might contain the immediate items of interest, which is often the case with entity and fact search. However, most keyword search performed on the Web is navigational, i.e. to obtain initial results that are then used as the starting point for further browsing. In particular for relation search (i.e. for retrieving complex information), it is necessary that the user starts with navigational search. Then, the user inspects the result set, chooses the relevant resource(s) from which further navigation is performed to collect all the items of interest. The navigation is resource-based, i.e. is performed by following links between resources.

Example 2. Mary enters *researchers AIFB* to obtain the corresponding list of researchers. For every researcher, Mary follows the links to research work such as *publications*.

3.3 Keyword Search and Facet-Based Browsing/Search

State of the art. Browsing through the complex space of Web resources (e.g. documents, data) is a difficult task. The user might lose orientation along the way. Faceted search has become a popular paradigm as it helps to address this problem by providing facets for the user to inspect and navigate through the resources. Basically, facets correspond to properties, i.e. attributes of the underlying resources or relations between them. Faceted search is widely used in commercial systems such as on-line shops. In that setting, faceted search is conceptually similar to form-based search. The programmers predefine a fixed set of facets based on which the user can define (refine) the items of interest in terms of specific facets and facet values. Recently, advanced faceted search systems have been developed to search generic sets of resources, such as Freebase Parallax⁴ for dealing with domain-independent datasets. Based on the resource schema, the system automatically computes facets for a given set of resources. Faceted search of this kind has been proposed for searching documents [21,10], for databases [22,11], as well as for RDF data [23,24].

Currently, research in this area is concerned with the aspects of efficiency and effectiveness. For instance, an efficient implementation of faceted search has been proposed based on the inverted index [25]. The Lucene index is extended to store not only terms but also the facets to which they belong. Since the number of facets that can be derived for a result set might be large, the ranking of facets has attracted interest. Widely used in faceted search systems is frequency-based ranking, which is based on the count of values that are associated with a facet [22]. Based on the similar idea, set-cover ranking has been suggested to maximize the number of distinct objects that are accessible from the top-k ranked facets [22]. A more elaborated metric is proposed by Dash et al. to incorporate the notion of interestingness [10]. Basically, it ranks those facets high which lead to surprising items, given a certain expectation. A different direction is to minimize the cost for the user finding some items of interest, where cost is derived from the facet hierarchy [11], or more specifically, the operations the user performed on it [22].

Process. In order to retrieve complex result sets, the user browses the set of resources obtained from keyword search. However, instead of operating at the level of resources, the user firstly obtains an overview of the resources in the form of facets. The user inspects the facets and then, navigates along these high-level resource descriptions. By adding or removing facets, the user refines and expands the current result set. This is performed until finding and collecting all relevant items of interest. Since facet search should be regarded as an additional feature, the navigation at the resource level is still possible, and might be required in some cases.

Example 3. After obtaining the list of resources for *researchers AIFB*, the user inspects the facets describing these results. This includes *name*, *address*, *affiliation* as well as *research work*. The last facet can be further decomposed into *publication* and

⁴ www.freebase.com/labs/parallax/

project. The user navigates along these facets, and chooses to add the facet *publication*. The resulting list of resources now contains all publications from researchers at AIFB.

3.4 Keyword Search and Result Completion

State of the art. With databases, the standard use case is to obtain complex results in the form of tuples instead of documents. Also here, keyword search is recognized as an intuitive paradigm that is more assessable to lay users. The underlying technique to support retrieval of tuples using keywords leverages some concepts widely used in the IR community. In fact, IR-style ranking schemes based on TFIDF as well as an adopted version of PageRank have been applied [4,5,9]. Relations are represented using the bag-of-words model and in some systems, are also stored in an inverted index [9]. However, keywords are not matched against documents but against values (i.e. treated as terms) located in columns that are part of some database relations. More importantly, keywords of a single query might match values of a number of tuples that might be located in several relations. Thus, the matching technique employed here goes beyond simple IR-style matching. It involves additional processing to obtain sets of connected tuples, which contain all the query keywords. In particular, tuples containing some of the query keywords have to be joined such that the task amounts to finding join graphs of tuples that contain all the keywords. Each join graph is a complex tuple (that might combine results from different relations), representing the answer to the keyword query. State of the art techniques in this area can be distinguished into two classes, i.e. (1) schema-based approaches and (2) schema-agnostic approaches that directly operate at the data level. Example systems of the first type are mostly implemented as database extensions, such as DBXplorer [7] and Discover [6]. These systems translate keywords to candidate networks, which are essentially join expressions constructed using information given in the schema. These networks are then used to instantiate a number of SQL queries, which are finally executed using the underlying query engines. The results are returned as answers to the keyword query. The schema-agnostic approaches use customized indexes to match query terms against data tuples. As opposed to the schema-based approaches, the data that has to be explored for join graphs of tuples is potentially large. Thus, building the appropriate indexes to leverage materialized paths is one direction of addressing the efficiency issue [2]. Another main direction of research is to investigate efficient procedures for top-k join graph [9] (or Steiner-trees [3]) computation. Instead of returning all the results after the user finished typing the entire query, Li et al. [8] suggest to return top ranked candidate answers as completions for the keywords the user has finished typing. Thus, we refer to this kind of approaches as keyword-driven result completion.

Process. Instead of a list of resources, the user obtains complex results for the entered keywords in the form of tuples. In particular, the user obtains candidate results upon entering every keyword of the query. The user inspects the result tuples. When the system correctly interprets the information need as expressed in terms of keywords, the process finishes after the user has entered the entire query. The need has been satisfied right away such that no additional effort is needed for browsing and collecting results. Otherwise, the user might choose some resources in the result set, and continues with resource-based navigation and retrieval of additional results, just like with standard keyword search.

Example 4. Upon the user entering every keyword of the query *publication researchers AIFB*, the system suggests different candidate results, i.e. top ranked list of publications after the first keyword *publication* has been entered, then all publications and associated researchers after the user finished adding *researchers* to the query and finally, after the user added *AIFB*, top ranked tuples containing publications of researchers from AIFB are presented. The process stops here as the need has been satisfied.

3.5 Keyword Search and Query Completion

State of the art. Recognizing that returning queries instead of answers might improve the type of expressive keyword search discussed previously, an approach we refer to as keyword-driven query completion has been proposed [9]. The queries computed represent possible interpretations of the keywords. The idea is to present a list of candidate interpretations, from which the user can choose the intended one. The main advantages of this approach over result completion are that (1) queries can be seen as descriptions that might facilitate users in inspecting and understanding the results, (2) they enable more effective refinement (compared to operating at the level of results) and also, (3) irrelevant results can be reduced because the system computes only answers to the intended query. The technique behind this approach is similar to the one used by schema-based result completion [7,6] discussed previously. Schema information is used to search for join expressions, which can meaningfully connect elements that match the keywords. These join expressions are finally mapped to queries, representing possible interpretations of the keywords.

Process. Upon the user entering a keyword, the system suggests queries that represent interpretations of the keywords entered so far. The user continues typing keywords and inspecting queries, until finding a query completion that matches the intended meaning. The user poses this query against the engine to obtain complex result tuples. In cases the computed interpretations do not perfectly match the intended meaning, the user might choose to inspect specific resources and to navigate along their links, just like with standard keyword search.

Example 5. After entering *research work*, the system suggests (among others), a query that retrieves all research work. As the user finishes typing *researchers*, the system suggests a query that retrieves all research work along with the associated researchers. Finally, after the entire query *research work researchers AIFB* has been finished, one query returned by the system represents the intended meaning, i.e. to retrieve all research work from researchers at AIFB. In this example, the process also stops here.

4 Experimental Study

In this section we present a comparison of the different search paradigms presented before using an experimental study. Because of the type of complex information needs we are dealing with (relation search in particular), the standard evaluation based on the metrics of precision and recall is too limited. Given complex needs, we aim to assess whether users are able to address them and how much effort they have to invest.

Table 1. Tasks Group A

Task 1	Find Tom Hanks (the actor)
Task 2	Find The Beatles (the band)
Task 3	Find Boston (the place)
Task 4	Find the occupation of Barack Obama
Task 5	Find the hometown of Metallica
Task 6	Find the birthplace of Jesus
Task 7	Find people with birthplace Karlsruhe
Task 8	Find the names of the spouses of all directors of Rambo movies
Task 9	Find all people whose birthplace is Albany, together with their deathplace

Thus, we conduct a task-based evaluation [26] which has gained acceptance in the IR community – especially for dealing with search solutions that go beyond the standard document- and keyword-centric IR paradigm. We compare the paradigms in terms of effectiveness and efficiency, and finally, we discuss some initial usability results.

4.1 Description of the Experiment

Tasks. Every participant had to use every one of the search paradigms to solve nine tasks of varying complexity. The nine tasks involved the types of search discussed before, i.e. three of them rely on entity search, another three rely on fact search, and the remaining three involve relation search. Table 1 shows the tasks for one of the groups of the user study. Every one of the tasks was in principle solvable with either one of the search paradigms. The process of task execution was monitored by an expert. We considered a task as correctly solved as soon as the answer to the task was displayed on the screen, and the participant was able to identify this answer. For every task, the participant had the possibility to decline it as not solvable with an arguable effort. If a user spent more than three minutes trying to solve a task, we aborted this task and considered it not solved.

Participants. We conducted the experiment with 19 volunteers at the age of 21 to 37 years. Nine of them were software developers, two were computer science researchers and the rest of the participants were non-technical users. Fig. 3 shows statistics about the self-assessment of the participants. Most of them had experience with query languages like SQL or XQuery. The experience with standard Semantic Web Technologies like RDF or SPARQL was almost equally distributed from very much experience to no experience at all. We divided the participants in two groups (A and B) who had to solve different tasks that are of the same types.

Systems. We implemented the search paradigms and integrated them as separate search modules into a demonstrator system of the Information Workbench⁵ that has been developed as a showcase for interaction with the Web of data. In particular, keyword search is implemented according to the design and technologies employed by standard Semantic Web search engines. Like Sindice and FalconS, we use an inverted index to

⁵ <http://iwb.fluidops.com/main.jsp>

Fig. 3. a) Skills in using search engines and b) experience with different technologies

store and retrieve RDF resources based on terms. Also using the inverted index, faceted search is implemented based on the techniques discussed in [25]. Result completion is based on recent work discussed for the TASTIER system [8]. For computing join graphs, we use the top-k procedure elaborated in [9]. This technique is also used for computing top-k interpretations, i.e. to support query completion. We choose to display the top-6 queries and the top-25 results respectively.

Dataset. For the evaluation, we used DBpedia, a dataset covering a large amount of broad-ranging knowledge [27] together with YAGO [28]. It allowed us to design evaluation tasks that are not domain specific.

The experiments were conducted using commodity PCs (i.e. Intel Pentium Core2Duo with 2 x 2Ghz CPU and 3 GB RAM). All actions taken by the participants and the system responses were recorded using a screencast software. The experimental study was based on both the analysis of these screencasts as well as a questionnaire every participant had to answer after the experiment. Additional information such as this questionnaire and the handouts with a list of the tasks provided to participants can be found in the report [29].

4.2 Evaluation Results

In this section we firstly present the effectiveness and efficiency results. Then, we discuss usability issues based on results of the user questionnaire.

Effectiveness. In our context, we define the effectiveness as the fraction of the tasks which were solved correctly. The percentage of unsolved tasks grouped according to the classification introduced in Section 2 is shown in Fig. 4. We will now discuss the results with respect to the types of search used to solve the tasks.

The results show that the paradigms do not differ for simple entity search. Every participant was able to solve every entity search task with every paradigm. This is not very surprising, since entity search can mostly be performed by simply typing the name of the entity. Then, the user conducts keyword search directly or obtains a translated query and evaluates this query. Either way, it was straightforward for participants to accomplish all tasks.

For fact search, we noticed slight differences between the paradigms. About 2 percent of the fact search tasks were not solved with keyword search and faceted search.

Fig. 4. Percentage of unsolved tasks

This means that one of the participants did not accomplish one task. We observed more failures with result completion: 9 percent of the fact queries were not solved with this paradigm. A problematic task was for instance *Find the birthplace of Jesus*. In this case, the result set is very large such that the item of interest (i.e. the entity Jesus) is not part of the top-25 results. However, more tasks were solved with query completion. More of the item of interests were covered by the top queries than the top results.

The differences in effectiveness between the paradigms became obvious for relation search: while only one candidate (about 5%) had problems obtaining the answers for relation search using query completion, we noted the percentage of tasks not solved using result completion to be 25 percent. The tasks that were difficult are for instance *Find people whose birthplace is Washington, together with their death place*. Since this involves a very special information need it was necessary to specify very precise keywords. Otherwise, the number of results computed as candidate answers is high – and in this case, does not contain the relevant answer. We found that with relation search, the percentage of unsolved talks is even higher for keyword search. For instance, only one candidate was able to accomplish searching for people born in Washington and their death place. To do this, users have to find the town (i.e. Washington) first. This is easily done by submitting the keyword *Washington*. Then, users have to inspect the information contained in the resource page. In particular, they have to find all the people born in this city. This is also relatively straightforward because in this case, this information was simply given in a list. However, much effort is required in the final step, where users have to visit the pages of every person in order to find out their death place. Along the way, most participants recognized this problem and did not complete or refused to complete the task because the given time of three minutes is too limited.

Efficiency. For assessing the efficiency, we identified all basic user interactions with the systems. These interactions correspond to the steps as discussed previously for the search process. We consider the operations of entering a keyword, executing the keyword query (time needed for the keyword search step in keyword search and faceted search), completing the keyword query (time needed for the keyword search step in query completion and result completion), visiting a resource (resource inspection and

Fig. 5. a) Number of basic operations, b) average time users needed to solve a task (in seconds)

resource-based navigation), inspecting complex results (query set inspection, result tuple set inspection), facet-based result inspection and the operations for facet-based expansion and refinement. We measured the average times to conduct every one of these interactions. We took only the times for correctly solved tasks into consideration.

The efficiency results for the different paradigms are shown in Fig. 5. The results suggest the time needed for keyword search and faceted search increases rapidly with the complexity of the information needs. This is clearly illustrated in Fig. 5(b). While keyword search and faceted search were the most efficient paradigms for entity queries, the time almost doubled for fact queries. This trend continues as we turn to the even more complex conjunctive queries. We observed another large increase of 40 to 60 percent. The increase in time invested by the user is in fact larger for faceted search than for keyword search. Fig. 5(a) delivers an explanation for this: The number of basic operations performed by participants using faceted search is by far the highest. Although each can be performed efficiently, the overall high number of operations lead to a large amount of time in total.

In contrast to these two paradigms, we did not observe a significant increase of time for the completion-based paradigms. In fact, the time invested by participants is almost the same for all types of information needs, as illustrated in Fig. 5(b). Fig. 5(a) also shows that the number of basic operations is independent of the complexity of the information needs. The search process consists of the same steps for every query: enter a list of keywords, translate those keywords, and, when using query completion, choose the correct query, and finally, find the answer in the result set. In the cases users did not choose the keywords precisely enough, they had to repeat the entire process.

Under the aspects of efficiency, we observed only minimal differences between result and query completion. Result completion seems to be faster in some cases.

User Questionnaire. After the experiment, we conducted an interview with all of the candidates to assess the usability of the search paradigms. Fig. 6(a) shows that 12 out of 19 candidates (63%) liked the query completion most. The preferences of the remaining candidates were equally distributed among the other three paradigms. Further details on the comparison of the two paradigms with best evaluation results so far systems are shown in Fig. 6(b+c). It seems that most users did not face any difficulties with choosing the correct results or the correct queries while using the completion-based paradigms. Query completion seems to be slightly more usable than result completion.

Fig. 6. a) most popular paradigm, b) ease of choosing correct result (result completion), c) ease of choosing correct query (query completion)

5 Conclusions

The increasing availability of structured data on the Web bears potential for addressing complex information needs more effectively. A primary challenge lies in enabling the users to specify complex queries without the user knowing details of the internal data and query model, and more importantly, the underlying schema of the data. In this paper, we identified and analyzed schema-agnostic search paradigms which address this challenge. We have conducted a systematic study of four popular keyword-driven approaches that rely on the use of keyword queries: (1) simple keyword search, (2) faceted search, (3) result completion, and (4) query completion. We have studied these approaches from a process-oriented view and then preformed a controlled user study to compare them.

From our experimental study we can draw the following conclusions: Not surprisingly, simple keyword search turned out to be sufficient for simple information needs. In fact, the results show that the effectiveness of the paradigms does not differ for simple entity search. The advantages of more advanced search paradigms were apparent for more complex information needs. For the most complex information needs under consideration (relation search), it turned out that only with query completion the users were effectively able to answer most queries.

Directions for future work thus include the question of how the paradigms can be combined in a useful way such that the user can easily resort to the most adequate paradigm for a given information need. In this context, the personalization of search interfaces is relevant, as preferences for one or the other paradigm might be subjective.

Acknowledgements. Research reported in this paper was supported by the German Federal Ministry of Education and Research (BMBF) under the iGreen project (grant 01IA08005K).

References

1. Androutsopoulos, L.: Natural language interfaces to databases - an introduction. *Journal of Natural Language Engineering* 1, 29–81 (1995)
2. He, H., Wang, H., Yang, J., Yu, P.S.: Blinks: ranked keyword searches on graphs. In: *SIGMOD Conference*, pp. 305–316 (2007)

3. Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., Karambelkar, H.: Bidirectional expansion for keyword search on graph databases. In: VLDB, pp. 505–516 (2005)
4. Hristidis, V., Gravano, L., Papakonstantinou, Y.: Efficient ir-style keyword search over relational databases. In: VLDB, pp. 850–861 (2003)
5. Liu, F., Yu, C.T., Meng, W., Chowdhury, A.: Effective keyword search in relational databases. In: SIGMOD Conference, pp. 563–574 (2006)
6. Hristidis, V., Papakonstantinou, Y.: Discover: Keyword search in relational databases. In: VLDB, pp. 670–681 (2002)
7. Agrawal, S., Chaudhuri, S., Das, G.: Dbxplorer: enabling keyword search over relational databases. In: SIGMOD Conference, p. 627 (2002)
8. Li, G., Ji, S., Li, C., Feng, J.: Efficient type-ahead search on relational data: a tastier approach. In: SIGMOD Conference, pp. 695–706 (2009)
9. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In: ICDE, pp. 405–416 (2009)
10. Dash, D., Rao, J., Megiddo, N., Ailamaki, A., Lohman, G.M.: Dynamic faceted search for discovery-driven analysis. In: CIKM, pp. 3–12 (2008)
11. Roy, S.B., Wang, H., Das, G., Nambiar, U., Mohania, M.K.: Minimum-effort driven dynamic faceted search in structured databases. In: CIKM, pp. 13–22 (2008)
12. Fogg, D.: Lessons from a “living in a database” graphical query interface. In: SIGMOD Conference, pp. 100–106 (1984)
13. Tran, T., Wang, H., Haase, P.: Hermes: Data web search on a pay-as-you-go integration infrastructure. *J. Web Sem.* 7(3), 189–203 (2009)
14. Cheng, G., Qu, Y.: Searching linked objects with falcons: Approach, implementation and evaluation. *Int. J. Semantic Web Inf. Syst.* 5(3), 49–70 (2009)
15. Tummarello, G., Delbru, R., Oren, E.: Sindice.com: Weaving the open linked data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 552–565. Springer, Heidelberg (2007)
16. Wong, S.K.M., Ziarko, W., Wong, P.C.N.: Generalized vector space model in information retrieval. In: SIGIR, pp. 18–25 (1985)
17. Robertson, S.E., Maron, M.E., Cooper, W.S.: The unified probabilistic model for ir. In: SIGIR, pp. 108–117 (1982)
18. Richardson, M., Domingos, P.: The intelligent surfer: Probabilistic combination of link and content information in pagerank. In: NIPS, pp. 1441–1448 (2001)
19. Robertson, S.E., Zaragoza, H., Taylor, M.J.: Simple bm25 extension to multiple weighted fields. In: CIKM, pp. 42–49 (2004)
20. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: SIGIR, pp. 275–281 (1998)
21. Hearst, M., Swearingen, K., Li, K., Yee, K.P.: Faceted metadata for image search and browsing. In: CHI 2003: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 401–408. ACM, New York (2003)
22. Dakka, W., Ipeirotis, P.G., Wood, K.R.: Automatic construction of multifaceted browsing interfaces. In: CIKM 2005: Proceedings of the 14th ACM International Conference on Information and Knowledge Management, pp. 768–775. ACM, New York (2005)
23. Schraefel, W.M., Russell, A., Smith, D.A.: mspace: improving information access to multimedia domains with multimodal exploratory search. *Commun. ACM* 49(4), 47–49 (2006)
24. Hyvönen, E., Mäkelä, E., Salminen, M., Valo, A., Viljanen, K., Saarela, S., Junnila, M., Kettula, S.: Museumfinland – finnish museums on the semantic web. *Journal of Web Semantics* 3(2), 25 (2005)
25. Ben-Yitzhak, O., Golbandi, N., Har’El, N., Lempel, R., Neumann, A., Ofek-Koifman, S., Sheinwald, D., Shekita, E.J., Sznajder, B., Yagev, S.: Beyond basic faceted search. In: WSDM, pp. 33–44 (2008)

26. Elsweiler, D., Ruthven, I.: Towards task-based personal information management evaluations. In: SIGIR, pp. 23–30 (2007)
27. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia - a crystallization point for the web of data. *J. Web Sem.* 7(3), 154–165 (2009)
28. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A large ontology from wikipedia and wordnet. *J. Web Sem.* 6(3), 203–217 (2008)
29. Mathaess, T.: Semantische suchsysteme, Master’s thesis, Karlsruhe Institute of Technology (2009), <http://www.stud.uni-karlsruhe.de/~usauk/studium/diplomathesis/damathaess.pdf>