Vietnam J Comput Sci (2015) 2:191–199
DOI 10.1007/s40595-015-0044-y

REGULAR PAPER

Evaluation of SPARQL-compliant semantic search user interfaces

Adam Styperek[1] · **Michal Ciesielczyk**[1] · **Andrzej Szwabe**[1] · **Pawel Misiorek**[1]

Received: 18 December 2014 / Accepted: 15 May 2015 / Published online: 28 May 2015

Abstract A regular user of a semantic search system frequently posses no knowledge about the SPARQL language nor about the ontology of a given knowledge base, especially when it provides domain-unspecific data obtained from heterogeneous sources. Nevertheless, he/she should be provided with tools enabling both intuitive and effective exploration of RDF-compliant knowledge bases. Natural language querying is one of the solutions that have been proposed so far as means for making knowledge bases more user-friendly. However, the results of natural language querying usually have lower precision and recall than analogical results of graph-based querying. In the paper, we introduce an evaluation methodology based on the 2011 QALD workshop queries that allows to measure the accuracy of a semantic search system as well as the complexity of the query formulation process. The obtained results confirm the intuition that graph-based querying, although assuring comparatively high accuracy of the results, is usually still too difficult for regular users. On the other hand, on the basis of results obtained for an experimental search system referred to as Semantic Focused Crawler, we claim that enhancing a SPARQL-compliant graph-based system by an entity-type recommendation feature may reduce the number of query elements necessary to formulate a query without compromising the quality of the results.

Keywords Semantic Web · Search · RDF · SPARQL · DBpedia

✉ Adam Styperek
adam.styperek@put.poznan.pl

Michal Ciesielczyk
michal.ciesielczyk@put.poznan.pl

Andrzej Szwabe
andrzej.szwabe@put.poznan.pl

Pawel Misiorek
pawel.misiorek@put.poznan.pl

[1] Poznan University of Technology, Piotrowo 3A, 60-965 Poznan, Poland

1 Introduction

The Semantic Web is widely regarded as one of the most important ideas in computer science since Tim Berners-Lee wrote the article [1] to Scientific American in 2001. This idea has become even more popular in the last few years—partly thanks to publicly available knowledge bases like DBpedia [2] and YAGO [12]. As these knowledge bases contain RDF data harvested from Wikipedia, they are heterogeneous from the data provenance perspective.

Since the users of large knowledge bases must be provided with tools that enable to query data in an effective manner, several solutions have been proposed [4,7–10,17]. However, as we show in this paper, despite the growing popularity of knowledge bases, the practical value of applications that enable to explore them is still limited. Such an observation motivates the efforts aimed at developing user interfaces for semantic data stores that are more intuitive but, at the same time, do not compromise the expressiveness of semantic queries.

To the best knowledge of the authors, despite the work on usability studies conducted in [5,14] and work related to methodology for evaluating semantic search tools [16], there is no widespread repeatable methodology allowing to evaluate the semantic search systems with different user interfaces, which does not depend on users' opinions. Therefore, some simple new measures that anyone should be able to apply have been proposed in this paper. It should be also noted that

the system comparison presented herein does not include the assessment of the visual and esthetic quality of graphical user interfaces and concerns only to the capability to express semantic queries. Therefore, the presented analysis is conducted only in terms of the answers' correctness and, in the case of graph-based querying interfaces, also from the perspective of the average number of elements needed to create queries. Finally, it should be also noticed that the proposed user interface evaluation is not meant to be a replacement, but rather a supplement of a usability assessment including tests with real users.

This paper is an extended version of the conference paper [11] presented at ACIIDS 2014. In particular, additional natural language querying systems has been evaluated along with graph-based querying systems. Thus, the related work and evaluation sections have been extended correspondingly. The improved evaluation methodology presented herein allows to quantitatively compare graph-based querying interfaces also with natural languages querying interfaces in terms of their effectiveness. By means of the introduced methodology we claim that the NLP-based systems, in contrast to graph-based systems, are less accurate, and that an graph-based semantic search interface does not have to be significantly more complex than an NLP-based search interface.

2 Related work

The SPARQL, when used directly (i.e., in a command line manner), is quite difficult for regular users [4,7]. Consequently, it seems to be reasonable to assume that a regular user posses no knowledge about the SPARQL or about the ontology of the given knowledge base. It should not be surprising then, that alternative means for the realization of a semantic search engine user interface have been proposed [4,7–10,17]. The leading approaches to the problem are represented by the systems allowing to formulate queries in the natural language or by the systems that enable the users to create SPARQL-like graph queries without requiring any knowledge of the SPARQL syntax.

2.1 Natural language querying

Interfaces based on the natural language processing (NLP) tend to be more user-friendly than graph-based query interfaces. Users of NLP-based interfaces obviously need no knowledge about the syntax of RDF, OWL, nor about the syntax of SPARQL queries. Nevertheless, in such a case the system has to map every query term provided by the user to an ontology object. Due to natural language indeterminism, this kind of mapping is error prone. Because every error in such process may cause irrelevant search results, the mapping component is critically important element of an NLP system. What is more, it may be difficult to construct more complex queries. Providing an analytical query using natural language, in such a way that a computer would be able to understand it, could turn out to be very troublesome.

Keyword-based search engines may be regarded as representing a simplified form of NLP systems. For instance, SPARK [17] automatically translates query terms into the corresponding resources from the domain ontology, using a probability ranking model. However, it does not analyze the syntax of the search phrase, thus the outcome of the search is a list of relevant documents rather than a direct answer to the query.

The Google search system may be regarded as a more advanced keyword-based search as it uses NLP tools to answer simple questions such as "how old is Michele Obama". For this reason, Google was evaluated using both simple keywords and natural language queries. Nevertheless, although they use sophisticated algorithms to rank the results (e.g., Google Pagerank), as shown in this paper, they usually provide less relevant results than semantic search engines.

It should be also noted that the Facebook Search is probably the first semantic search engine made available as a component of a large-scale social networking service. Unfortunately, it works only in the Facebook domain, and uses a keyword-based search (Microsoft Bing) outside the domain. For this reason, we were not able to objectively compare Facebook Graph Search with other systems, thus it was omitted in the evaluation.

PANTO [15], a Portable nAtural laNguage inTerface to Ontologies, is a system that takes natural language queries and ontologies as input and returns SPARQL queries. However, since there is no publicly available version of this system we were not able to include it into our evaluation.

PowerAqua [9] is another natural language query interface that is designed to work in any domain and uses many ontologies at the same time. It extracts facts from the user query and searches the ontology to find resources which are relevant to the facts extracted from the query. Because it uses syntactic matching between query elements and ontology resources, the system additionally removes terms having the same syntax but different semantic interpretations.

2.2 Graph-based querying

Using a graph-based interface, similarly as using a natural language query interface, does not require any knowledge about ontologies or formal languages like SPARQL. Moreover, it reduces the problem of the precision of a natural language-based query [7], and allows to save the effort necessary to learn the SPARQL. Each time a user issues a graph-based query, he/she has to specify a set of triple/fact templates that may be seen as simple sen-

Table 1 Semantic search systems comparison

System	User query interface	Domain	Allow keyword search	Type and property recommendations
Facebook Graph Search	Natural language	Facebook	Yes	No
NAGA	Graph-based	YAGO	Yes/No	No
SFC	**Graph-based**	**Any**	**Yes**	**Yes**
GoRelations	Graph-based	DBpedia	No	No
PANTO	Natural language	Any	Yes	No
PowerAqua	Natural language	Any	Yes	No
NITELIGHT	Graph-based	Any	Yes	No
SPARK	Keyword-based	Any	Yes	Yes

The introduced solution is highlighted in bold

tences containing a subject, predicate, and object. Such an approach provides users with the opportunity to query the system more precisely than while using the natural language.

Graph of relations (GoR) [7] is a graph-based DBpedia [2] query interface allowing to directly type in triple query patterns. However, the GoR interface may be considered not user-friendly for several reasons. First of all, the syntax is quite strict and unintuitive. The user has to remember to write each triple in a new line, separate each element of the triple by a comma, and specify the type of each new variable. Moreover, according to these constraints the user cannot create a query about the relations between objects of unspecified type. What is more, GoR does not provide any recommendations about available types and relations. As a consequence, the user has to know what type of subjects and objects are stored in DBpedia and what kind of relations exists between them. The system has to map these types to the DBpedia ontology terms using statistic and semantic similarity components, which, as mentioned before, may result in greater inaccuracy of the results.

YAGO (Yet Another Great Ontology) [12] is a knowledge base that integrates information from Wikipedia and WordNet. The NAGA (Not Another Google Answer) [8] is a search engine which operates on the data from the YAGO knowledge base. It provides users with a web interface [8] in which they can create SPARQL-like queries. Similarly to GoR, NAGA enables users to create more precise queries than in the case of using natural language parsers, however, it provides more functionality. In this interface the user creates a query by completing several (optional) fields: subject, property, object, time, location and keywords.

NITELIGHT [10] enables users to create SPARQL queries using a graphical notation and GUI-based editing actions. However, such an approach limits the usage of the tool to users who already are familiar with SPARQL, thus it was omitted in the evaluation presented herein.

2.3 Qualitative comparison of the selected semantic search systems

Table 1 presents a summary of the key properties of the compared systems such as their domains of use and user query interface types. As it may be seen in Table 1, the system presented in this paper, referred to as Semantic Focused Crawler (SFC), is the only system that provides the user with recommendations of ontology types and relations that occur between objects. Contrarily, the YAGO–NAGA is able to recommend only the resource labels, while GoR does not support any kind of recommendation functionality. The recommendations may be an important feature as they not only enhance the user-friendliness but also improve the accuracy by avoiding mapping terms to ontology objects.

The presented system is the only one of the compared graph-based semantic search systems that allows the user to apply the functionality of full-text search on indexed literals and resource labels. In contrast, YAGO–NAGA allows the user to filter the semantic search results only by using keywords, while GoR does not even support any function of such type [7].

Table 2 shows the level of compliance with SPARQL (the use of aggregation, order by, filter, union, optional functions, supported SPARQL query types and results pagination) observable in the selected search systems.As one can see in Table 2, none of the compared systems support the aggregation or ordering functions (probably due to scalability-related issues). Only YAGO–NAGA and SFC provide the user with results pagination function which may effectively support the user in the search results exploration.

3 Concept of Semantic Focused Crawler

The semantic search system presented in this paper is a component of the system referred to as SFC. SFC features a Web Crawler module that gathers and indexes selected data

Table 2 Compliance with SPARQL—the use of aggregation, order by, filter, union, optional functions, supported SPARQL query types and results pagination

System	Aggregation	SPARQL query types	Order by	Pagination	Optional	Filter	Union
NAGA	No	Select only	No	Yes	No	No	No
SFC	**No**	**Select only**	**No**	**Yes**	**Yes**	**Yes**	**No**
GoR	No	Select only	No	No	No	Yes	No
PANTO	No	Select only	No	No	No	Yes	Yes
NITELIGHT	Yes	Select only	Yes	Yes	Yes	Yes	Yes

The introduced solution is highlighted in bold

from various heterogeneous Internet data sources. Each Web document collected by SFC is converted into an RDF graph consistent with the predefined ontology. The SFC is called semantic, because it is based on RDF, RDFS, OWL standards which form the base of the Semantic Web. Additionally, the presented system is also able to load data from other RDF databases and third-party ontologies like DBpedia [2].

The semantic search system presented in this paper provides means for the user to build semantic queries and search knowledge bases using a graphical user interface. Each query is a list of RDF-like patterns, containing resources, literals along with variable nodes. Because the system is based on the software developed within an open source JENA framework [3], all queries are eventually formulated in SPARQL. Inherently, it is also fully compatible with SPARQL and RDFS W3C recommendations. Additionally, the user is able to enhance the graph queries using keyword-based search (applied to RDF literals). This function, available as a result of the use of the Apache Lucene [13], allows a user to perform free text searches within the semantic queries.

As a result of the use of context-based auto-complete recommendations within the SFC query builder interface, the user benefits from applying the ontology irrespectively from the degree of the knowledge about semantic technologies he/she possesses. In other words, the user definitely does not have to be an OWL expert in order to effectively use the ontology.

Particularly, during the query construction process, the SFC provides the user with features such as:

- recommendation of variables automatically associated with appropriate types allowing to create more precise queries in a simpler manner,
- variable properties enabling to discover relations connecting given objects,
- recommendation of types and relations allowing to avoid error-prone mapping between query terms and ontology types or properties.

3.1 Query construction process

To formulate a query while using the SFC the user has to provide at least one triple. A query triple is in fact an RDF triple

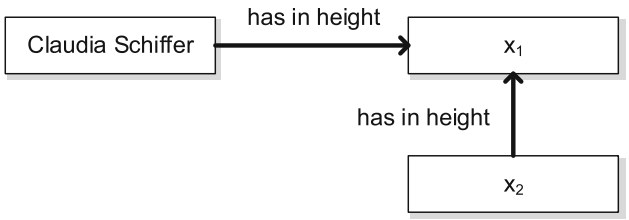

Fig. 1 Query example 1

template having a subject, a property and an object (represented by a variable or a resource). When the user selects an element (a subject, a property, or an object), he/she is provided with recommendations adequate for a specific field. If an element is not selected, it will be automatically set as a variable node. Additionally, the user may specify some advanced search options (e.g., maximum number of returned results).

Usually, the more complex a query is, the more query triples it has. Let X be the set of variables, matching any object or property in a query, $|X| = n$ and $x \in X$. For instance, to find names of the all models which are exactly as tall as *Claudia Schiffer*, the query should include three query triples:

Claudia Schiffer has in height x_1.
x_2 *is a Model*.
x_2 *has in height* x_1.

In practice, the user of SFC does not have to create all three query triples but only two. It is possible because the user is allowed to define the variables' types while creating the following query triple: x_2 *has in height* x_1. Figure 1 presents the described query in the SFC system. It contains two variables, x_1 (without a type) and x_2 with a defined type *Model*.

3.2 Recommendation of variable names

In SFC, the user is provided with some additional support while specifying a variable name allowing to automatically provide its type. In particular, while looking for an unspecified model, the SFC user may select the automatically

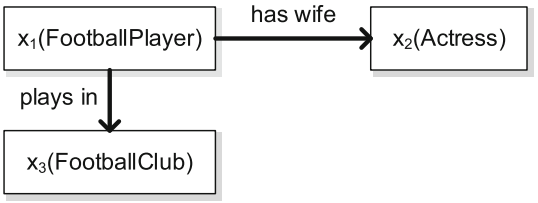

Fig. 2 Query example 2

suggested variable *?Model #1*, and the type will be properly complemented with an ontology class. The character '#' (a special sign in the variable name) and the number after it are automatically appended, and are mandatory to enable the user to create other variables with the name starting with the same prefix.

This feature is useful when a user wants to create a query which contains more than one query triple in SPARQL, e.g., when he/she wants to find a football player (along with his club) whose wife is an actress, as in Fig. 2. In fact, in the underlying query model, there are still five triples in the query but the user does not have to add all of them separately. In particular, the query may be expressed in the following form:

x_1 *is a FootballPlayer*. x_2 *is a Actress*.
x_1 *has wife* x_2 . x_1 *plays in* x_3.
x_3 *is a FootballClub.*

The variable recommendation feature enables one to avoid error-prone mapping of the query terms to ontology classes and properties, which could result in irrelevant answers to the query.

3.3 Variable properties

What is more, the user may also use a variable property, which may be helpful if the user wants to find all relations between two or more objects. A query about relations between Ryan Giggs and David Beckham may be example of a query which in SPARQL contains more than one query triple. In this case, if the user queries DBpedia using such a query he/she will receive an empty set as a result. There are no results for this query, because there is no direct relation between Ryan Giggs and David Beckham in the knowledge base. Nevertheless, if the user knows that both these football players played together in Manchester United, then he/she may create another query like the one shown in the Fig. 3, to receive a relevant answer. This query consists of two triples having one common variable (x_2).

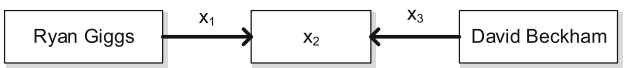

Fig. 3 Query example 3

In SFC, all the fields are optional, but at least one of them should be filled. In other words, the user does not have to complete all the fields every time. For example, if the user types only the subject, then the system automatically creates variables for the property and the object. If the user wants to create a query similar to the example shown in Fig. 3, then he/she does not have to input the predicate variable. If he/she leaves the property field empty the system will automatically add a new variable property.

3.4 Recommendation of object types and relations

All class types that may be recommended to the user are derived from the ontology. For example, if the user creates a new variable and he/she wants to specify its type, the system automatically provides applicable (i.e., relevant to the typed query) ontology type suggestions. For example, if the user types "*po*" the system will display a list which contains types like: *polo league, poker player, political party*. This solution allows to avoid mapping (which may always introduce an error) terms from a user query to ontology classes. All the recommended types come from the ontology, and all recommended properties and class instances come from the data store (in explicit, from an RDF graph). As a result, the user is only allowed to select properties and objects that exist in the data store. Additionally, if the user initially defines the subject and the subject type, then he/she is provided with list containing only the properties that are present in the data store—each occurring as a relation between the subject of this type and some object.

4 Evaluation

The comparison presented herein focuses on publicly available graph-based querying systems, but also contains results of NLP querying systems like Google and PowerAqua, provided to show their precision limitations. The comparison shows which of the evaluated systems returns more relevant results, and which of the evaluated graph-based querying systems requires the smaller number of elements in query in the construction process. This section presents the methodology of the evaluation reported in the paper, including the presentation of the methodology assumptions, the description of the data set used for the comparison, the description of the evaluation measures, as well as additional issues related to the measurement process—especially those dependent on the characteristics of the systems being compared.

4.1 Evaluation methodology assumptions

Evaluation methodology presented in this paper is based on a few assumptions. Each of these assumptions is described

in detail in the following sections. First, it has been assumed that the evaluation is based on the use of a subset of the widely referenced QALD Query Set[1] — a subset consisting of those QALD queries which may be represented in all the different input data representation forms that are acceptable for the semantic search systems under comparison (i.e., written in natural language, the SPARQL language, and in the form of keywords); the set is presented in Sect. 4.2. Second, it has been specified what results of the querying provided by the evaluated systems (i.e., the 'output') are considered as correct ones in accordance to the proposed methodology. Such a specification is necessary, because of the heterogeneity of the systems seen from the perspective of the form that querying results provided to the user have (a graph, a list of Web links enriched with descriptions in snippets, or a natural language string). This issue is presented in detail in Sect. 4.3, with a special attention paid to the problem of the multiplicity of the correct answers that may correspond to a given single query, and the problem of realistic estimation of the number of answers that are practically useful for a user issuing a query to the Google question answering (QA) system. Third, it is assumed that the binary relevance quality evaluation presented in this paper is quantitative and performed with the use of precision, recall and F-measure—the most widely used measures of binary classification quality [6]. Section 4.3 elaborates the way these measures have been used in the experiments reported in this paper. Last, but not least, Sect. 4.4 explains the approach to the query complexity measurement that has been followed in the evaluation efforts presented in this paper to complement the primary, accuracy-centric evaluation.

4.2 QALD query set

All the evaluated systems featuring natural language-based and graph-based querying interfaces have been compared using the QALD-1 trainset queries set which contains queries to DBpedia knowledge base [2]. DBpedia is a dataset that, in the newest version, describes 4.58 million entities and includes 583 million RDF triples extracted from the English edition of Wikipedia.[2] The QALD query set is a set of 100 questions and relevant answers for these questions. In most cases, the answers are represented as URIs of DBpedia resources or as a string.

All QALD queries have been formulated in natural language, SPARQL, and in the form of keywords. In this paper, every natural language query have been considered as having the same semantics as the corresponding SPARQL query. The Google search engine has been evaluated both using keyword and natural language queries.

[1] http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/.

[2] http://wiki.dbpedia.org/Datasets.

Some of the evaluated systems do not support aggregation, filter and ordering functions and are not able to represent some of the relations from the QALD query set. For these reasons, in our experiments, we have used only the questions that may be represented in all the semantic search systems. For example, SFC, NAGA and GoR do not support aggregation functions, so these systems are unable to answer about "*How many films did Leonardo DiCaprio star in?*". Finally, we selected a subset containing 28 queries for evaluation purposes. Identifiers of these queries are: 2, 3, 6, 14, 15, 18, 22, 23, 25, 28, 29, 31, 35, 36, 38, 39, 42, 43, 44, 48, 54, 55, 56, 62, 73, 85, 87, 90. These queries subset contains simple queries like "*Give me the official website of Tom Cruise*" and more complex queries like "*Which monarchs of the United Kingdom were married to a German?*".

4.3 Accuracy measurement

It is very difficult to compare search systems with substantially different interfaces (i.e., natural language-based, graph-based, and Google with keyword-based querying interface) in a fully objective manner. However, a search result is always a list containing both results that are relevant or irrelevant to the query, thus it is possible to equivalently measure their accuracy. Some papers [8] present only the final results of the relevance assessment made by humans. Obviously, due to the human factor, it is not possible to exactly repeat such an evaluation.

Precision, Recall, and F-score measures, widely used in the area of Information Retrieval [6], have been used to compare the systems. Herein, a query result is considered relevant only when it is equal to one of the results presented in the QALD answers. In the case of Google, only a result which contains relevant keywords or link in the snippet is considered relevant. For example, if the user, after querying Google about the creators of Wikipedia, obtains results containing the names of Jimmy Walles and Larry Sanger and not containing other people's names in the snippets, such a result is treated as relevant. Similarly, if the result contains the *Jimmy Walles* keyword in the snippet (while not containing other names), it is still regarded relevant. Obviously, in such a case the Precision will not change, but the Recall will be lower (as the system has not returned all of the correct answers). In other compared systems, the result is considered as a relevant if it is equal to the result in QALD. For example, if in QALD the correct results are *Jimmy Walles* with URL http://dbpedia.org/resource/Jimmy_Walles and *Larry Sanger* with URL http://dbpedia.org/resource/Larry_Sanger and evaluated system returns string or URI which is equal to QALD correct results, such a result will be evaluated as relevant.

Another problem is how to decide how many of the results should be taken into consideration. In particular, in many cases, Google may return few millions results. For example,

if the user queries the Google search engine about the Tom Cruise official website, he/she will obtain around 86,000,000 results. To estimate the precision of the returned results, one cannot simply consider all of them. To resolve this problem, in this paper, it is assumed that the precision will be calculated for the first n results, where n is equal to a number of the relevant answers from the QALD. Consequently, if there is only one relevant answer to the query "What is the official website of Tom Cruise" in the QALD, only the first result is considered. In practice, in our experiments, the n value varies from 1 to 11 (as it may be maximum 11 correct answers).

When performing the evaluation one should also consider the Google QA system, which tries to display the direct answer for the query. For example, if the user queries Google about "How tall is Claudia Schiffer", the correct answer is displayed above the returned links. In our experiments, for queries for which Google has been able to use the QA system (i.e., 11 out of 28 queries), only the direct answer has been taken into consideration. In other words, answers returned by Google in the traditional results' list are not considered as potentially relevant when a direct answer returned by the Google QA system is displayed above them.

4.4 Query complexity measurement

In this paper, it is assumed that an interface may be considered more user-friendly when the user is forced to fill in less query elements. In the case of graph-based search applications the query elements are: resources, variables, and literals. Obviously, the query complexity formulation is not the only factor influencing the user-friendliness. On the other hand, this feature may be used as the basis for an objective evaluation methodology.

In our research, we focused on evaluation methods that are easily repeatable. User studies, due to the participation of the human factor, are obviously not possible to be repeated exactly. Therefore, we have not included the human subjective opinions in the evaluation process.

Table 3 presents an example query "*In which films directed by Garry Marshall was Julia Roberts starring?*" written in the form that is appropriate for each of the compared systems. The user of GoR has to specify types of all variables and resources and to type in each triple element. In contrast, in the same query scenario, the user of SFC is not required to manually add the subject variable (x_1) in the first query line and to specify all the object types. Consequently, he/she has to add only five elements instead of nine, while in the case of using NAGA the user has to add the variable x_1 in the both query lines.

5 Evaluation results

The averaged results of the individual Precision and Recall measurement performed for each query are presented in Table 4. It should be noted that there was a relatively large spread in the obtained values. Specifically, in most cases a system either returned a correct answer for a query or it was unable to return any relevant result at all. This could be due to relatively low n used for precision and recall measurements in combination with the errors caused during the mapping of the query terms onto the ontology objects (which is especially prominent in NLP-based systems).

It should be also noted that one of the key factors affecting the search results is the use of different knowledge bases. However, the comparison may still be regarded as objective for two reasons. First, all the systems have access to all the data that is necessary to generate all relevant answers for each of the selected queries. Second, all the compared systems allow the user to create all the tested queries and to issue them to the system.

PowerAqua is one of the natural language-based semantic querying systems that have been evaluated in the test. As it can be seen in Table 4, PowerAqua has achieved lower result than other systems. Moreover, Google, queried both by keyword and natural language queries, has achieved lower result than SFC, YAGO–NAGA, and GoR. It is also worth mentioning that, as it can be seen in Table 4, Google returns more relevant results, when it is queried in natural language rather than when solely keyword-based search is used. These results confirm the intuition that a system with the natural language-based querying interface, in contrast to other semantic search systems, is usually unable to find correct answers for a significant number of questions.

Table 3 Example query representations with the number of corresponding elements the user has to type in (italic in every query)

System	Semantic representation of the query	Number of elements
GoR	*Julia Roberts*/*Actor*, *starredIn*, x_1/*Movie*	9
	Garry Marshall/*Director*, *directed*, x_1	
SFC	*Garry Marshall*, *directed*, x_1	**5**
	Julia Roberts, *starredIn*, x_1	
NAGA	*Julia Roberts*, *actedIn*, x_1	6
	Garry Marshall, *directed*, x_1	

The best result is highlighted in bold

Table 4 Systems accuracy comparison results

System	Precision	Recall	*F*-score
SFC	**0.961**	**0.906**	**0.927**
GoR	0.693	0.672	0.682
NAGA	0.661	0.639	0.649
Google (Natural Language)	0.620	0.632	0.626
PowerAqua (DBpedia only)	0.512	0.481	0.494
Google (Keywords)	0.422	0.425	0.423

The best results are highlighted in bold

Table 5 Average number of elements required to create each query

System	SFC	GoR	NAGA
Elements per query	**2.786**	5.714	3.571

The best result is highlighted in bold

SFC failed on queries with identifiers: 23, 28, 85, 2, 42, 35. In most cases the Precision of returned answers was equal to 1 but the Recall was less than 1. The correct answer for question 35 is not clear because NBCUniversals is an owner of Universal Studio but Universal Studio is a part of Comcast (which is the answer returned by SFC). In the case of question 2, SFC returned a list of results containing one irrelevant result, so the precision was less than 1.

SFC achieved the best results in the conducted experiments. The system returned the most relevant results with the highest precision. SFC enabled to achieve higher precision and recall mainly due to the fact that it avoids mapping of the query terms to ontology classes and properties (which tend to be an error-prone process causing irrelevant results). Additionally, the feature of more advanced full-text search on indexed literals introduced in SFC has also some impact on the accuracy of the returned results.

Table 5 presents the average number of elements per query that the user has to type to formulate the query in each system. According to the experimental results presented herein, a user of SFC does not have to type in as much query elements as the users of other systems. On average, there are approximately three elements per query. Consequently, the user may create queries without the need for typing redundant information, such as types of query variables. SFC follows such an approach by automatically filling the variable type, based on the variable name. This feature, unique among other semantic search systems [7,8], enables to significantly reduce the amount of data that the user has to type in to construct a query.

The presented results show that a system featuring a graph-based, user-friendly querying interface, may still be very useful in terms of semantic expressiveness of queries. As shown in the experiments, the proposed system provides the user with more useful responses to queries than the other compared systems do, and features the querying interface requiring the user to type a smaller number of query elements.

6 Conclusions

One problem, which has been addressed in result of developing the semantic search user interface presented in this paper is the simplicity and user-friendliness of the interface that does not compromise the system usability. Many of existing semantic search systems provide graph-based querying interfaces. However, these user interfaces are still not very self-explaining and simple to use. On the other hand, some of the systems with natural language querying interfaces feature user-friendly interfaces, but their practical usability is usually significantly lower than that of graph-based querying systems. As shown in the experimental results, at least one of the systems presented in this paper, namely SFC, may be regarded as effectively combining graph-based querying systems usability with the user-friendliness.

6.1 Main results

The main objective of the software development activities presented in the paper is to accompany the SFC system with a user-friendly search interface. As shown in the previous sections, SFC allows the user to create graph-based queries in a simpler way than it is done in the other graph-based querying systems. The key features of the system include:

- Keyword-based search as a function of the graph-based search interface.
- Intuitive variable naming convention which allows the user to creates triples which are similar to simple sentences.
- Query triples construction process, which is simpler than in semantic search systems presented in the literature [7,8].
- Automatic type and property suggestions that the user may use in the query construction process.

6.2 Key findings

It is difficult to evaluate semantic search systems in an objective manner as well as to use measures that give fully repeatable results. One of the related findings is that, due to

participation of human subjective opinions in the evaluation process, some of the semantic search systems tests in the literature [8] are difficult or even impossible to repeat. At the same time, in this paper the analyzed systems were compared using methods that are easy to repeat by anyone.

Our experimental results confirm that NLP-based systems, in contrast to graph-based systems, due to their simplicity are inherently inaccurate [7]. Assuming that semantic search interfaces are not much more complex than Google keywords queries, the semantic search systems may give the user more satisfying results, than Google does (see [8] and Sect. 5), without requiring much more effort on the query construction.

The presence of contextual recommendations of the element's types and the relations that may occur between the elements of given types (i.e., defined in the ontology) is an important feature of a user-friendly semantic search system. This solution is a countermeasure to the problem of ontology types and object properties heterogeneity—the problem appearing as a natural consequence of the heterogeneity of data sources being used to develop a domain-generic knowledge base such as DBpedia [7]. Moreover, as shown in our research, variables with an unspecified type may be automatically added by the system to improve the process of query construction.

Acknowledgments The presented paper has been supported by European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreement No. 218086—Project INDECT, and by the Polish National Science Center under grant DEC-2011/01/D/ST6/06788.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Sci. Am. **284**(5), 34–43 (2001)
2. Bizer, C., Auer, S., Kobilarov, G., Lehmann, J., Becker, C., Hellmann, S.: DBpedia—querying wikipedia like a database and an interlinking-hub in the web of data. In: Querying Wikipedia Like a Database (4/4/2009) FU Berlin, Universitt Leipzig (2009)
3. Carroll, J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: implementing the semantic web recommendations. Tech. rep., Hewlett Packard (2003)
4. Damljanovic, D., Agatonovic, M., Cunningham, H.: Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In: Aroyo, L., Antoniou, G., Hyvnen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC (1). Lecture Notes in Computer Science, vol. 6088, pp. 106–120. Springer, Berlin (2010)
5. Elbedweihy, K., Wrigley, S.N., Ciravegna, F.: Evaluating semantic search query approaches with expert and casual users. In: Proceedings of the 11th International Conference on The Semantic Web (ISWC'12), vol. Part II, pp. 274–286. Springer, Berlin (2012)
6. Goutte, C., Gaussier, E.: A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In: Losada, D.E., Fernndez-Luna, J.M. (eds.) ECIR. Lecture Notes in Computer Science, vol. 3408, pp. 345–359. Springer, Berlin (2005)
7. Han, L., Finin, T., Joshi, A.: GoRelations: an intuitive query system for DBpedia. In: Pan, J.Z., Chen, H., Kim, H.G., Li, J., Wu, Z., Horrocks, I., Mizoguchi, R., Wu, Z. (eds.) JIST. Lecture Notes in Computer Science, vol. 7185, pp. 334–341. Springer, Berlin (2011)
8. Kasneci, G., Suchanek, F.M., Ifrim, G., Ramanath, M., Weikum, G.: NAGA: searching and ranking knowledge. In: 24th International Conference on Data Engineering (ICDE'08). IEEE, IEEE Press, Cancun (2008)
9. Lopez, V., Fernndez, M., Motta, E., Stieler, N.: Poweraqua: supporting users in querying and exploring the semantic web. Semantic Web **3**(3), 249–265 (2012)
10. Russell, A., Smart, P.R., Braines, D., Shadbolt, N.R.: Nitelight: A graphical tool for semantic query construction. In: Semantic Web User Interaction Workshop (SWUI'08) (2008), event Dates: 5th April 2008
11. Styperek, A., Ciesielczyk, M., Szwabe, A.: Sparql compliant semantic search engine with an intuitive user interface. In: Nguyen, N., Attachoo, B., Trawiski, B., Somboonviwat, K. (eds.) Intelligent Information and Database Systems. Lecture Notes in Computer Science, vol. 8397, pp. 201–210. Springer International Publishing, NY (2014)
12. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: a core of semantic knowledge unifying WordNet and Wikipedia. In: Proceedings of the 16th International World Wide Web Conference (WWW'07), pp. 697–706. Banff (2007)
13. The Apache Software Foundation: Apache Lucene Core. http://lucene.apache.org/core/ (2014). Accessed 5 Apr 2015
14. Tran, T., Math, T., Haase, P.: Usability of keyword-driven schema-agnostic search. In: Aroyo, L., Antoniou, G., Hyvnen, E., Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) The Semantic Web: Research and Applications. Lecture Notes in Computer Science, vol. 6089, pp. 349–364. Springer, Berlin (2010)
15. Wang, C., Xiong, M., Zhou, Q., Yu, Y.: Panto: a portable natural language interface to ontologies. In: Franconi, E., Kifer, M., May, W. (eds.) ESWC. Lecture Notes in Computer Science, vol. 4519, pp. 473–487. Springer, Berlin (2007)
16. Wrigley, S.N., Reinhard, D., Elbedweihy, K., Bernstein, A., Ciravegna, F.: Methodology and campaign design for the evaluation of semantic search tools. In: Proceedings of the 3rd International Semantic Search Workshop (SEMSEARCH'10), pp. 10:1–10:10. ACM, New York (2010)
17. Zhou, Q., Wang, C., Xiong, M., Wang, H., Yu, Y.: Spark: adapting keyword query to semantic search. In: Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudr-Mauroux, P. (eds.) The Semantic Web. Lecture Notes in Computer Science, vol. 4825, pp. 694–707. Springer, Berlin (2007)