IEICE TRANS. INF. & SYST., VOL.E97–D, NO.11 NOVEMBER 2014

PAPER

Automatic Inclusion of Semantics over Keyword-Based Linked Data Retrieval

Md-Mizanur RAHOMAN[†a)], *Nonmember and* **Ryutaro ICHISE**[†,††b)], *Member*

SUMMARY Keyword-based linked data information retrieval is an easy choice for general-purpose users, but the implementation of such an approach is a challenge because mere keywords do not hold semantic information. Some studies have incorporated templates in an effort to bridge this gap, but most such approaches have proven ineffective because of inefficient template management. Because linked data can be presented in a structured format, we can assume that the data's internal statistics can be used to effectively influence template management. In this work, we explore the use of this influence for template creation, ranking, and scaling. Then, we demonstrate how our proposal for automatic linked data information retrieval can be used alongside familiar keyword-based information retrieval methods, and can also be incorporated alongside other techniques, such as ontology inclusion and sophisticated matching, in order to achieve increased levels of performance.

key words: *linked data, keyword, information access, data statistics*

1. Introduction

The Linked Open Data [4] initiative, where data are connected in a network-like structure [7] and which was motivated by the potential for link construction and identification among various data, has opened new worlds in data usage. The concept of this storage paradigm deviates from traditional repository-centric infrastructures to an open publishing model that allows other applications to access and interpret stored data [16]. As of September 2011, 295 knowledge-bases consisting of over 31 billion resource document framework (RDF) *triples* on various domains, have become interlinked via approximately 504 million RDF links*.

Efficient and easy-to-use information access over linked data is now a necessity because these days such linked data hold vast amounts of knowledge. Usually, obtaining information access over a linked data network requires following links [2], [3], [15]. However, simply following links introduces a very basic problem, which is that the use of a network presentation makes it very hard to find endpoints, at least within a reasonable cost [1]. As a result, finding links on linked data is often difficult, especially for general-purpose users who have very little knowledge about the internal structure of linked data, such as schema information or structured query language (SQL) type expressive queries (RQL, RDQL, or SPARQL [24]).

Keyword-based link data access methods are considered easy to use because of their familiarity [18], [23]. However, such data access options are different from other traditional keyword-based data retrieval types because they require adapting keywords to semantics. Since keywords do not contain semantic information (specifically ontology information), a number of researchers have proposed automatic ontology inclusion [28] to bridge that gap. However, automatic ontology inclusion is a challenge because, in such cases, the system itself needs to incorporate ontology that is, as yet, unavailable.

In attempts to resolve the abovementioned problems, such as link finding and keyword semantics inclusion, recently, a number of other researchers have worked to incorporate templates into linked data keyword search schemes [26], [30]. The intuition behind "template" creation is that it would introduce a defined structure that supports finding links and their endpoints, as well as templates that could fill the semantic gaps that result when keywords alone are used. However, current template-based linked data retrieval studies have yet to provide concrete guidelines for template construction, ranking, and merging, all of which are required for effective adaptation of templates to linked data retrieval. In this work, we will propose a guided framework on template construction, ranking, and adaptation for use with keyword-based linked data access that uses internal data statistics for template management.

Our template management technique relies on keyword order. In it, templates are constructed for "adjacent keywords". More specifically, for each keyword, any other keywords that are ordered before and after are considered to be adjacent keywords. For keyword-based information retrieval, our primary assumption is that users will enter keywords in the word order of a natural language sentence. That is, instead of inputting keywords randomly, they will input them in an order that conforms to the natural language structure of a sentence. Michael A. Covington supported this assumption in his work [9] where he showed that major languages such as Chinese, English, and French almost never allow variations in the word orders that make up sentences. A word order-based information retrieval approach is also common in contemporary semantic search research. For example, Unger *et al.* and Yahya *et al.* used a language parser

Manuscript received March 4, 2014.
Manuscript revised July 1, 2014.

[†]The authors are with the Graduate University for Advanced Studies, Tokyo, 101–8430 Japan.
[††]The author is with National Institute of Informatics, Tokyo, 101–8430 Japan.
a) E-mail: mizan@nii.ac.jp
b) E-mail: ichise@nii.ac.jp

DOI: 10.1587/transinf.2014EDP7073

*http://www4.wiwiss.fu-berlin.de/lodcloud/state/

to determine the word order of their natural language questions [30], [34].

Above a keyword-based linked data information retrieval perspective, our proposal contributes in the following ways: i) by considering linked data's data structure and internal statistics, we devise templates into which we can embed linked data's missing semantics, ii) by observing the linked data's internal statistics, we propose a template ranking strategy that can be used to identify a strong potential template (from among other templates) and that can be used to construct a possible SPARQL query, and iii) by introducing a template merging strategy, we guide the scalability of our proposed retrieval framework for any number of keywords. Our proposed framework can automatically create templates from keywords without considering schema or ontology information through the use of an automatic linked data retrieval approach, and we further hypothesize that the inclusion of such information will effectively improve the system's performance.

The remainder of this paper is divided as follows. In Sect. 2 we describe works related to this study, while in Sect. 3 we propose guidelines on template construction and management for query questions that contain two keywords. In Sect. 4, we extend our proposed guidelines for query questions that contain more than two keywords. In Sect. 5 we show the results of implementing our proposal through experimental results and discussion. Section 6 concludes our work.

2. Related Work

Linked data information access is an active research field in the linked data research community. Over the past couple of years, researchers have introduced a range of techniques relating to linked data information retrieval, such as RDF document retrieval using look-up [29], using sophisticated matching and ranking [12], retrieval via automatic ontology inclusion [27], [28], [32], retrieval by incorporating user feedback [19], or retrieval by incorporating natural language [10], [11], [21].

There have also been some approaches where researchers consolidate benefits from two or more techniques like faceted search and explicate query [13], [33], or semi-supervised machine learning techniques that employ user feedback [19]. Additionally, since keywords provide an easy and familiar means for data access [18], [23], some studies have proposed keyword-based linked data retrieval [5], [8], [36], or have advocated utilizing the advantages of keywords [6], [17], [20], [28], [35] when creating linked data access paradigms.

Our work has also been motivated by the possibilities of keyword-based linked data access. Using such an approach, Gheng *et al.* explored the creation of virtual documents that provide query-relevant structures on linked object detection [8]. Zhou *et al.* showed a keyword-based resource mapping and graph construction technique [36], while automatic ontology inclusion to keywords is another technique proposed in [17], [28]. Bicer *et al.* demonstrated keyword-based index creation and query graph construction [6], while Han *et al.* advocated an easy query learning framework where keywords are fitted automatically to query construction using a sophisticated matching technique [14].

Thus, it can be seen that the linked data research community has been working to create easy and effective linked data search techniques, and that many of their proposals incorporate keywords as search options. However, keyword-based linked data or semantic data retrieval is different from other traditional keyword-based data retrieval techniques because it requires the adaptation of semantics to keywords. Some researchers have previously proposed the inclusion of templates based on the idea that they could be used to provide structured frameworks that bridge the gap between keywords and linked data semantics [30]. However, those studies failed to create concrete guidelines for template construction, ranking, and merging, all of which are necessary for the effective adaptation of templates to linked data retrieval. For example, Unger *et al.* proposed a natural language based question answering (QA) system for linked data with *Pythia* [31]. However, this system is considered inefficient for template creation because it employs natural language (NL) tools, which sometime leads to incorrect template construction [30]. Furthermore, *Pythia* itself is subject to scalability issues that makes this system very specific to some particular data. In contrast, Shekarpour *et al.* proposed a template construction approach that is similar to the method discussed in this study, but their method requires knowledge of schema information, such as instance or class type information, which they provide manually [26]. Furthermore, their proposed method is only able to handle queries with two keywords.

3. Two-Keyword-Based Retrieval

In this section, we describe our proposed retrieval framework, primarily by focusing on how we construct a template using two adjacent keywords, and then identify the best template from among many such templates. In normal use, templates are predefined structures that are used to perform tasks by setting position holders to specific task parameters. From a linked data perspective, such position holders provide a predefined structure in the form of ontology (or semantics) that can be used to identify links and locate endpoints. We describe our approach as a *binary progressive approach*, which means that queries are constructed with resources from two adjacent keywords, and can then be extended to use more than two keywords. Template management for queries with more than two keywords is described in Sect. 4.

Figure 1 shows the two-keyword-based template selection process flow. It takes two keywords k_1 and k_2. Then, in the next step, the *resource manager* process collects and manages resources that are related to those keywords. Next, the *template constructor* process constructs a number of templates and identifies the best among them. Each of these

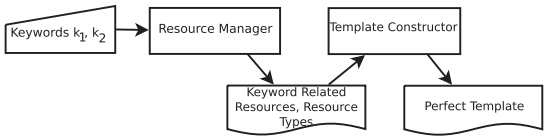


Fig. 1 Template selection process flow for two-keyword query questions.

processes will be described in greater detail below.

3.1 Resource Manager

The resource manager process collects and manages resources that are related to the keywords. It starts by taking two adjacent keywords and then produces *keyword-related resources* with their classifications. In the first step, for each keyword k, it extracts the keyword-related resources ($RR(k)$) from the underlying knowledge-base (KB). For linked data, the KB is constructed from the set of resource description framework (RDF) triples that store data using the form $< s, p, o >$ where s, p, and o are respectively considered as the *subject*, *predicate*, and *object* component elements in a RDF triple.

In the second step, for each keyword-related resource (r), the resource manager calculates three *positional frequencies* ($PF_s(r), PF_p(r)$, and $PF_o(r)$). Then, in the third step, resource manager selects a resource-type ($rType(r)$). The resource-type provides the basis of the template construction while the positional frequencies of the resource guides the process that leads to the identification of the best template. We will discuss the steps in detail below.

1. Keyword-related resource ($RR(k)$) for keyword k is a set of linked data resources that represent keyword k.

$$RR(k) = \{r \mid \exists < r, p, o > \in KB \wedge p \in rtag \wedge (m(o, k) \geq \alpha)\}$$

   where *rtag* is a set of resource-representing tags such as label, name title, prefLabel, etc. and $m(o, k)$ is a string-similarity function used to select the resource r that corresponds the keyword k. String-similarity is calculated between the object o of the RDF triple $< r, p, o >$ and the keyword k. r is selected for a particular similarity-threshold value α.

2. The positional frequencies ($PF_s(r), PF_p(r)$ and $PF_o(r)$) for the resource r are three different frequencies of r in the KB. Since r can be any of the three *subject, predicate* and *object* component elements in a RDF triple, subject, predicate, and object positional frequencies for r are respectively defined as follows:

$$\begin{aligned} PF_s(r) &= |\{< r, p, o > | \exists < r, p, o > \in KB\}| \\ PF_p(r) &= |\{< s, r, o > | \exists < s, r, o > \in KB\}| \\ PF_o(r) &= |\{< s, p, r > | \exists < s, p, r > \in KB\}| \end{aligned}$$

3. The resource-type ($rType(r)$) for resource r is a classification that classifies r as either a predicate-type (PR) or non-predicate-type (NP) resource.

$$rType(r) = \begin{cases} PR & \text{iff}(PF_p(r) > PF_s(r)) \\ & \wedge (PF_p(r) > PF_o(r)) \\ NP & \text{otherwise} \end{cases}$$

3.2 Template Constructor

The template constructor process constructs templates and identifies the best template. It constructs templates for two adjacent keyword-related resources on the basis of their resource-types and then identifies the most suitable template from among all those constructed.

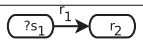
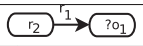
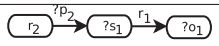
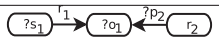
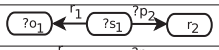
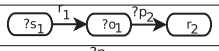
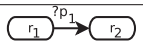
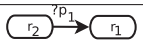
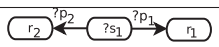
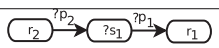
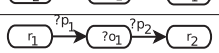
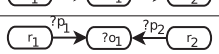
Before defining a template, we will introduce the term "triple-pattern" ($tp(r_1, r_2)$), which is a pattern constructed for two keyword-related resources. The third column of Table 1 contains triple-patterns. A triple pattern picks triples from the KB. In a triple pattern, a variable resource, which starts with a question mark (i.e., ?), matches any resource in the KB and selects the matched triples. Template ($tmp(r_1, r_2)$) is set of triple-patterns i.e., $tmp(r_1, r_2) = \{tp_1(r_1, r_2), tp_2(r_1, r_2), \ldots\}$.

The fourth column of Table 1 shows a graphical illustration of each triple-pattern. A template accumulates a set of triple-patterns. Triple-patterns are constructed by considering resource types of participating resources. The second column of Table 1 shows such consideration by the name "condition". Here, it can be seen that templates are constructed by maintaining two set of conditions i.e., i) $\text{rType}(r_1) = \text{PR} \wedge \text{rType}(r_2) = \text{NP}$, and ii) $\text{rType}(r_1) = \text{NP} \wedge \text{rType}(r_2) = \text{NP}$. However, for other two possible conditions sets, such as, for $\text{rType}(r_1) = \text{NP} \wedge \text{rType}(r_2) = \text{PR}$, we construct templates by swapping r_1 and r_2, and for $\text{rType}(r_1) = \text{PR} \wedge \text{rType}(r_2) = \text{PR}$, we use *modified template*s (described in Sect. 4.3). The fifth column of Table 1 shows *closeness* of a triple-pattern which indicates how closely r_1 and r_2 are attached.

Since a template (resp. triple-pattern) is constructed according to the structure of an RDF triple, it is assumed that the template incorporates linked data semantics and can able to retrieve the required information. For example, for keywords k_1 = spouse and k_2 = Barack Obama, it is assumed $r_1 \in RR(spouse)$ and $r_2 \in RR(\text{Barack Obama})$, and $rType(r_1)$ = PR and $rType(r_2)$ = NP, which fits into the triple-pattern $<?s_1, r_1, r_2 >$ and retrieve the required information.

Each template holds multiple triple-patterns and each keyword generates multiple keyword-related resources. Therefore, to determine the best template (which we call a *perfect template* ($pefTmp(k_1, k_2)$)) for each two adjacent keywords k_1 and k_2, we need to identify the best triple-patterns among the templates that can incorporate the largest amount of the linked data semantics. By calculating the *relatedness* of each triple-pattern towards the KB, we can then pick the best triple-pattern and select it as the perfect template.

Table 1 Templates for $r_1 \in RR(k_1)$ and $r_2 \in RR(k_2)$.

	Condition	Triple pattern $tp(r_1, r_2)$	Graphical illustration	Closeness of triple-pattern
$tmp(r_1, r_2)$	rType(r_1) = PR ∧ rType(r_2) = NP	$<?s_1, r_1, r_2>$		1
		$< r_2, r_1, ?o_1 >$		1
		$<?s_1, r_1, ?o_1 >< r_2, ?p_2, ?s_1 >$		2
		$<?s_1, r_1, ?o_1 >< r_2, ?p_2, ?o_1 >$		2
		$<?s_1, r_1, ?o_1 ><?s_1, ?p_2, r_2 >$		2
		$<?s_1, r_1, ?o_1 ><?o_1, ?p_2, r_2 >$		2
$tmp(r_1, r_2)$	rType(r_1) = NP ∧ rType(r_2) = NP	$< r_1, ?p_1, r_2 >$		1
		$< r_2, ?p_1, r_1 >$		1
		$<?s_1, ?p_1, r_1 ><?s_1, ?p_2, r_2 >$		2
		$<?s_1, ?p_1, r_1 >< r_2, ?p_2, ?s_1 >$		2
		$< r_1, ?p_1, ?o_1 ><?o_1, ?p_2, r_2 >$		2
		$< r_1, ?p_1, ?o_1 >< r_2, ?p_2, ?o_1 >$		2

3.2.1 Selection Criterion of Triple-Pattern Relatedness

Resource frequency plays a prime role in measuring triple-pattern relatedness. We are motivated by this frequency-based approach from the classical *term frequency* (TF) calculation. From a document-based information retrieval perspective, the TF of a term measures its importance over a particular document. From a linked data information retrieval perspective, we replace terms with keyword-related resources. Therefore, to conform a TF-like triple-pattern relatedness calculation, we hypothesize that the more frequently keyword-related resources appear in an underlying dataset, the greater its potential for use in retrieval [22].

In a triple-pattern relatedness calculation, we consider the following: i) how similar triple-pattern's keyword-related resources are in representing the keywords, ii) how frequent a triple-pattern is, and iii) how frequently the triple-pattern's keyword-related resources appear in the triple-patterns.

Therefore, the *relatedness* value of each triple-pattern towards the KB is calculated using following statistics:

1. **String-similarity value** $m(o, k)$ **between the resource** r **and the given keyword** k: this r is the subject component element of $< r, p, o >$ where $p \in rtag$.
2. **Frequency of triple-pattern** $fqTP(tp(r_1, r_2))$: this counts the number of RDF triples associated with the $tp(r_1, r_2)$.
3. **Frequency of resource** r **w.r.t. triple-pattern** $fqR(r, tp(r_1, r_2))$: this is the positional frequency of the resource r in the KB where the position (*subject, predicate* or *object*) of r is guided by $tp(r_1, r_2)$.

$$fqR(r, tp(r_1, r_2)) = \begin{cases} PF_s(r) & \text{if } r \text{ is on } subject \text{ in } tp(r_1, r_2) \\ PF_p(r) & \text{if } r \text{ is on } predicate \text{ in } tp(r_1, r_2) \\ PF_o(r) & \text{if } r \text{ is on } object \text{ in } tp(r_1, r_2) \end{cases}$$

Final relatedness $fRel(tp(r', r''))$ of triple-pattern $tp(r', r'')$ is defined as

$$\begin{aligned} fRel(tp(r', r'')) = {} & m(o', k_1) * m(o'', k_2) \\ & * fqTP(tp(r', r'')) * fqR(r', tp(r', r'')) \\ & * fqR(r'', tp(r', r'')) \end{aligned}$$

3.2.2 Selection of the Perfect Template

We select the perfect template (from the all triple-patterns of all possible templates) by considering two parameters: closeness and relatedness. We first select closeness value 1 type triple-patterns with relatedness values greater than zero, sort them by their relatedness values, and then choose the highest related triple-pattern as the perfect template. If no closeness value 1 type triple-patterns with relatedness values greater than zero can be identified, we then consider the highest relatedness valued closeness 2 type triple-pattern to be the best triple-pattern, and select it as the perfect template. In any case, if we identify several best possible triple-patterns (because of identical relatedness and closeness values), we pick one at random as the perfect template. Considering the above, it can be seen that the template selector process can provide a perfect template for two given keywords. For a two-keyword query question, this perfect template related SPARQL query is then used to identify our intended result.

4. More-than-Two-Keyword-Based Retrieval

In this section, we will describe how we extend previous

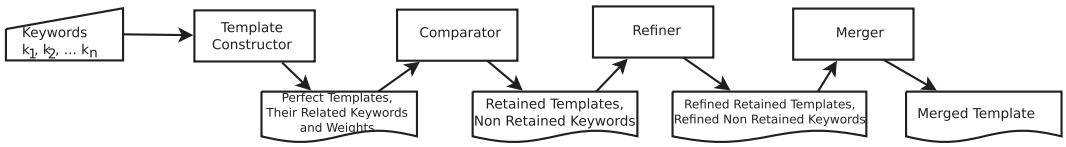


Fig. 2 Template selection process flow for query question with more than two keywords.

two-keyword-based template management to handle queries with more than two keywords and show how, we use pair of keywords and pair of templates to find the merged template. Figure 2 shows this merged template selection procedure. It starts with the *template constructor* generating perfect templates for each two adjacent keywords. Next, the *comparator* process takes each two adjacent perfect templates and determines whether it should retain them by designating one as the *retained template (RT)* and finding a *not retained keyword (NRK)* from the other perfect template. The *refiner* process then selects some RTs as refined-RTs and generates modified templates from the NRKs. Finally, the *merger* template merging process produces the *merged template*. We will describe each process in detail below.

Throughout our description, we will use a question *"In which films directed by Garry Marshall was Julia Roberts starring?"*, for which we devise keywords $\{k_1$ *=Film,* k_2 *=director,* k_3 *=Garry Marshall,* k_4 *=Julia Roberts,* k_5 *=starring*$\}$ as a running exemplary question and execute our proposed framework over DBpedia[†] KB.

4.1 Template Constructor

This process selects all perfect templates for each two adjacent keywords and stores them along with their keywords and weights. For i number of keywords, we get $i-1$ number of perfect templates for each two adjacent keywords. For example, for the running exemplary question, we get four perfect templates – the first perfect template ($pefTmp_1$) for $\{k_1, k_2\}$, the second perfect template ($pefTmp_2$) for $\{k_2, k_3\}$, and so on. In the perfect template selection, if we are unable to find any triple-pattern with relatedness value greater than zero, any triple-pattern identified will be designated as the perfect template. For the running exemplary question, we get four adjacent best possible perfect templates as follows:[††].

- $pefTmp_1$ = <?s_1, o:director, ?o_1 ><?o_1, ?p_2,o:Film>
- $pefTmp_2$ = <?s_1, o:director, r:Garry_Marshall >,
- $pefTmp_3$ = <?s_1, ?p_1, r:Garry_Marshall><?s_1, ?p_2, r:Julia_Roberts >,
- $pefTmp_4$ = <?s_1, o:starring, r:Julia_Roberts >

4.2 Comparator

This process compares each two adjacent perfect templates using their closeness values and relatedness values. Using this process, we designate one perfect template as the RT and select one keyword from the other perfect template as the NRK. The RT is selected by the lower depth and higher relatedness valued perfect templates between the participating perfect templates. In contrast, NRK is found by the exclusively associated keyword held by a perfect template other than RT.

For the running exemplary question, between $pefTmp_1$ and $pefTmp_2$, $pefTmp_2$ carries lower closeness and higher relatedness values than $pefTmp_1$, we consider $pefTmp_2$ as the first RT (say rt_1) and k_1 (i.e., Film) as the first NRK (e.g., nrk_1) because k_1 is an exclusively associated keyword in $pefTmp_1$.

Since we follow a *binary progressive approach*, the comparator process is executed for the adjacent pairs perfect templates. Therefore, if the number of perfect templates is $i-1$ (see Sect. 4.1), we compare pairs for $pefTmp_j$ and $pefTmp_{(j+1)}$, where $1 \leq j \leq i-2$. This comparison gives all RTs and NRKs. However, in some cases, RTs and NRKs might share *common keywords*. For example, for the running exemplary question, $pefTmp_2$ associates the keyword Garry Marshall, which also appears as an NRK. Furthermore, $pefTmp_2$ appears twice as RTs which also share common keywords between them. Since we only construct templates for each keyword once, it is necessary to refine common keyword-related RTs and NRKs.

4.3 Refiner

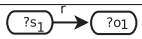
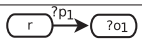
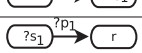
This process refines previously generated RTs and NRKs so that the final template will only use each keyword once. The *refiner* process eliminate redundancies using two operations: i) eliminating redundant RTs, and ii) eliminating redundant NRKs. We will discuss the operations in details in the below:

i) As the first operation, we take a set of RTs along with their generation order, i.e., rt_1 appears first, after which the second one appears, and so on, from which we generate a set of refined-RTs. We eliminate redundancy between the RTs in order to make each RT unique. For the running exemplary question, since there are two $pefTmp_2$s RTs, we first eliminate one $pefTmp_2$ and then identify the unique RTs as $\{pefTmp_2, pefTmp_4\}$. After finding a set of unique RTs, we determine whether the two unique RTs share common keywords and eliminate any such identified. For such cases, if there is a shared common keyword between two unique RTs, we store the most

[†]http://dbpedia.org/

[††]o: is prefix for http://dbpedia.org/ontology/ and r: is prefix for http://dbpedia.org/resource/

Table 2 Modified templates for single resource $r \in$RR(k).

	Condition	modified- $tp(r)$	Graphical illustration	Closeness of modified-$tp(r)$
modified-$tmp(r)$	rType(r)=PR	$<?s_1, r, ?o_1>$		1
modified-$tmp(r)$	rType(r)=NP	$< r, ?p_1, ?o1 >$		1
		$<?s_1, ?p_1, r >$		1

recently generated one and eliminate the other one. For the running exemplary question, the unique RTs $\{pefTmp_2, pefTmp_4\}$ do not share a common keyword, so we do not need to eliminate any of them. At the end of the first operation, the unique RTs that remain are considered to be refined-RTs. Therefore, for the running exemplary question, the refined-RTs are $\{pefTmp_2, pefTmp_4\}$.

ii) For the second operation, we take a set of NRKs and refined-RTs and generate a set of refined-NRKs. Each NRK is checked to determine whether it is already associated with any of the refined-RTs. Those that are not are considered refined-NRKs. For example, for the running exemplary question, we get {Film} as a refined-NRK as "Film" is not associated with any of the refined-RTs.

Since it is necessary to use a template for refined-NRK, we convert each refined-NRK k to its perfect template, which is called a modified perfect template (modified-perTmp(k)). Our earlier template generation was intended for two keywords, but in this step we modify template generation for a single keyword (i.e., for each refined-NRK). To accomplish this, we find keyword-related resources for each refined-NRK along with their type classifications, and then construct single-resource-based triple-patterns (modified-$tp(r)$s) and modified templates (modified-$tmp(r)$s), as can be seen in Table 2.

For each refined-NRK k, we identify the best modified triple-pattern as the modified perfect template from among all modified templates that possesses maximum relatedness value towards the KB. The relatedness of a modified triple-pattern is counted by considering how many RDF triples are associated in the KB.

For the running exemplary question, we get $<?s_1, ?p_1$, o:Film> as modified-pefTmp(Film). This single-keyword-based template also supports template construction when two adjacent keyword-related resources appear as predicate type resources (see Sect. 3.2). In such cases, for each predicate type keyword-related resource, we construct a modified triple-pattern.

As a result, the refiner process generates modified perfect templates for all refined-NRKs, and then forwards all refined-RTs and modified perfect templates to the next process.

4.4 Merger

This process, which produces a merged template for all query keywords, begins after the generation of all refined-RTs and modified perfect templates. Here, we merge refined-RTs and modified perfect templates according to the given keyword order. Template merging can be considered triple-pattern merging because each individual refined-RT and individual modified perfect template are nothing more than single triple-patterns. Therefore, in the following paragraphs, template merging is described as triple-pattern merging.

We merge triple-patterns by introducing *connectors*. Connectors are merging points where triple-patterns are merged with one another. We use a triple-pattern's variable resource holding subject and object component elements as connectors. For example, for a triple-pattern $<?s_1, r_1, ?o_1>$ $<r_2, ?p_2, ?s_1>$, $?s_1$ and $?o_1$ are connectors. Since each triple-pattern holds multiple connectors, we greedily try to merge them until we find a valid merged triple-pattern. The validity of the merged triple-pattern is checked by its SPARQL query, which determines whether the merged triple-pattern corresponding to the SPARQL query generates any non-empty output. To serve this greedy approach, we assign priorities to the triple-pattern connectors and then merge triple-patterns according those priorities.

4.4.1 Triple-Pattern Connector Priorities

The priorities of triple-pattern connectors will vary depending on how many connectors each triple-pattern holds. Below is the calculation used to assign priority:

i) For a two-connector based triple-pattern, a connector that contributes as the RDF triple component element with a later order keyword-related resource is considered the *higher priority connector*, while the other connector is considered the *lower priority connector*. If both connectors appear in the same RDF triple component, the connector that contributes as the subject component element of the RDF triple is considered to be the higher priority connector. For example, for a triple-pattern $<?s_1, r_1, ?o_1 >< r_2, ?p_2, ?s_1 >$ which is constructed for two resources of two orderly given keywords k_1 and k_2, $?s_1$ is the higher priority connector while $?o_1$ is the lower priority connector because s_1 is associated with the later keyword k_2 (through the r_2). In contrast, for a triple-pattern like $<?s_2, r, ?o_2 >$ where both connectors appear in the same RDF triple component, $?s_2$ holds a higher priority than $?o_2$.

ii) For one connector based triple-pattern, the connector is simultaneously considered both the higher priority

connector and lower priority connector.

4.4.2 Merging of Triple-Patterns

In the binary progressive approach, we start triple-pattern merging for the first two triple-patterns by attempting to merge higher priority and lower priority connectors. If we can obtain a valid merged triple-pattern, we then consider this as an intermediate merged triple-pattern. Then, we merge the next triple-pattern to the connectors of the intermediate merged triple-pattern (i.e., come from its constituent triple-patterns).

Such iterative triple-pattern merging is continued until we merge the last triple-pattern. However, in intermediate merged triple-pattern generation, the connector priorities get updated during each valid merged triple-pattern finding. Below we describe the priority update process for the connectors of each intermediate merged triple-pattern.

i) Connectors that belong to the lastly merged triple-pattern hold higher priorities. If the triple-pattern merged last holds two connectors, the highest priority goes to the connector that can generate a valid merged triple-pattern followed by the second connector.

ii) The connectors that remain follow the updated priorities.

For example, for an intermediate merged triple-pattern $<?s_1, r_1, ?o_1 ><?o_1, r_2, ?o_2 >$ where $?o_1$ holds the (so far) highest priority followed by $?o_2$ and $?s_1$, and a next triple-pattern $<?s_2, r_3, r_4 >$ attempting to merge at the connector $?o_1$ but are unable to generate a valid merged triple-pattern, but do get merged at the connector $?o_2$ where they are able to generate a valid intermediate merged triple-pattern $<?s_1, r_1, ?o_1 ><?o_1, r_2, ?o_2 ><?o_2, r_3, r_4 >$, then, in the new intermediate merged triple-pattern, $?o_2$ holds the (new) highest priority followed by $?o_1$ and $?s_1$. In this manner, priority-based merging reduces merging complexity.

Therefore, for the running exemplary question, we obtain a valid merged triple-pattern (or template) as

$<?s_1, ?p_1,$ o:Film>.
$<?s_1,$ o:director, r:Garry_Marshall>.
$<?s_1,$ o:starring, r:Julia_Roberts >.

5. Experiment

In this experiment, we use question answering over linked data 1 and 2 (i.e., QALD-1[†] and QALD-2[††]) open challenge test question sets. Both challenges include the same type of natural language training and test question sets from DBpedia and MusicBrainz[†††].

[†]http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=home&q=1

[††]http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=home&q=2

[†††]http://musicbrainz.org/

Exemplary question *"In which films directed by Garry Marshall was Julia Roberts starring?"*, is a QALD-1 test question (Q#29). Here underlining is used to show the keywords of the question. The order of the keywords is as the keywords are appeared in the question.

The DBpedia dataset used in our experiment comprises more than 30 million instances, 288 thousand classes, and almost 50 thousand properties, while the MusicBrainz dataset comprises almost 4 million instances, 31 classes, and 125 properties. A resource is a class that appears with [††††]type. We call our system, **B**inary Pr**o**gressive **T**emplate Paradigm over **L**inked Data **Ret**rieval or **BoTLRet**.

We discarded a few questions, such as questions that require *Boolean* type answers, aggregation functions, temporal precedence (such as *latest*, *past five years*, etc.) understanding, and questions for which resources are not found in the KB, because they are out of the scope of our research. We used 78.12% of the QALD-2 DBpedia test questions and 74% of the QALD-2 MusicBrainz test questions.

The QALD-2 test questions were used for our detailed experimental evaluation. In contrast, QALD-1 questions were used to compare other keyword-based data retrieval initiative. In the comparison between BoTLRet and other keyword-based system, we were able to compare 66% of the QALD-1 DBpedia test questions because the comparison was performed for questions that are executed by both BoTLRet and other systems. When the other systems did not execute MusicBrainz questions, only the DBpedia dataset questions were used. To select keyword-related resources via the resource manager process, we use similarity-threshold value $\alpha = 1$. For DBpedia and MusicBrainz datasets, we manually define *rtag* as {[†††††]label, [††††††]title}. We then implement BoTLRet using the Java Jena (version 2.6.4) framework. The BoTLRet hardware specifications are as follows:

Intel®Core™i7-4770K central processing unit (CPU) 3.50 GHz based system with 16 GB memory. We loaded DBpedia and MusicBrainz KBs in Virtuoso (version 06.01.3127) triple-store, which was maintained in a network server. To evaluate BoTLRet, we performed three experiments and analyzed their results. These experiments will be described in detail below.

5.1 Experiment 1

The first experiment was performed to evaluate how the BoTLRet performs for two-keyword-based retrieval and more-than-two-keyword-based retrieval. Therefore, we will report BoTLRet performance according to the keyword group, i.e., number of keywords each question holds. Keyword groups are separated by the number of keywords each question can hold. For example, the exemplary question

[††††]http://www.w3.org/1999/02/22-rdf-syntax-ns#

[†††††]http://www.w3.org/2000/01/rdf-schema#

[††††††]http://purl.org/dc/elements/1.1/

Table 3 BoTLRet Recall, precision, and F1 measure grouped by number of keywords for the DBpedia and MusicBrainz questions.

	DBpedia				MusicBrainz			
	No of Qs	Recall (avg)	Precision (avg)	F1 Measure (avg)	No of Qs	Recall (avg)	Precision (avg)	F1 Measure (avg)
2	51	0.961	0.961	0.961	7	1.000	1.000	1.000
3	13	0.923	0.852	0.857	8	1.000	1.000	1.000
4	6	0.833	0.833	0.833	16	0.875	0.875	0.875
5	5	1.000	1.000	1.000	5	0.800	0.800	0.800
6	-	-	-	-	1	1.000	1.000	1.000
	Average	0.946	0.943	**0.944**	Average	0.918	0.918	**0.918**

shown in Sect. 5 falls into a "five keyword group" question, because it holds five keywords. We executed the BoTLRet system for each group of questions and evaluated their results in terms of average recall, average precision, and average F1 measure. Based on the given answers of QALD-2 test questions, recall is the fraction of relevant answers that the BoTLRet can retrieve. Precision is the fraction of retrieved answers that are relevant, and the F1 measure is the harmonic mean of precision and recall. For each of the keyword-group questions, we calculated the average precision and average recall by summing up the individual recall and individual precision, and then dividing them by the number of questions for each group. However, it was impossible to calculate the average F1 measure using the same method because the individual F1 measure cannot be calculated if the recall of that individual question is zero. In such cases, we put the individual question's F1 measure at zero as well, and then calculated the average F1 measure for each group of questions.

Additionally, for the DBpedia dataset, the average execution costs for two, three, four, and five keyword group questions were measured as 41.00, 91.74, 134.27, and 164.02 seconds respectively – which is a linearly increased trend.

Table 3 shows our keyword-group-wise result analysis for recall, precision, and F1 measure. The bottom of the table shows averages for the recall, precision, and F1 measure for both set of questions. As you can see, the performance of the "two keyword group" questions indicates that our template selection proposal works well.

This also ensures usefulness of triple-pattern relatedness calculation i.e., $fRel(tp(r', r''))$ (shown in Sect. 3.2.1). The performance of the questions for more than two keywords also validates our template merging policy. Therefore, we conclude that internal structure of the linked data and their statistics have more significant impact on template construction, which can be used potentially over keyword-based linked data information retrieval.

5.2 Experiment 2

The second experiment was performed to evaluate the effectiveness of resource classification of our proposal when selecting valid triple-patterns. To accomplish this, we investigated the triple-pattern generation approach of BoTLRet with an exhaustive system (referred to hereafter as a maximum triple-pattern system (MTS)), which uses an exhaustive triple-pattern generation approach to compare the performance of BoTLRet and the performance of MTS in result generation. We then report the primacy of one system over the other.

Table 4 Comparison between MTS and BoTLRet systems in terms of number of triple-patterns used and computational cost.

case	Used No of Triple-patterns		Computational Cost by BoTLRet w.r.t. MTS
	MTS	BoTLRet	
PR-NP	49	7	0.142
NP-NP	49	8	0.163
TOT	49	15	0.306

Table 5 Completeness comparison between MTS and BoTLRet.

	MTS			BoTLRet		
	Recall (avg)	Precession (avg)	F1 Measure* (avg)	Recall (avg)	Precession (avg)	F1 Measure (avg)
DBpedia	0.959	0.956	0.957	0.946	0.943	0.944
MusicBrainz	0.918	0.918	0.918	0.918	0.918	0.918

In principle, the framework details of both BoTLRet and MTS are nearly identical. However, in the BoTLRet, we construct triple-patterns by considering the classifications of keyword-related resources, while in the MTS, we construct triple-patterns without considering the classification. Therefore, the MTS considers all possible combinations for keyword-related resources when constructing triple-patterns and can be considered to be an exhaustive version of the BoTLRet.

Next, we compared comparative computational cost requirements between the two systems. Triple-pattern usage by the BoTLRet system can be divided into three cases:

- PR-NP: Triple-patterns that hold one predicate type keyword-related resource. Seven triple-patterns belong to this case.
- NP-NP: Triple-patterns that hold two non-predicate type keyword-related resources. Eight triple-patterns belong to this case.
- TOT: Triple-patterns that hold both PR-NP and NP-NP cases. Fifteen triple-patterns belong to this case.

Since MTS always uses 49 triple-patterns, it is clear that the BoTLRet system requires less execution time. Table 4 shows an efficiency comparison by dividing them into three triple-pattern cases, i.e., PR-NP, NP-NP, and TOT, between the two systems. As the number of triple-patterns used is less for BoTLRet in all the three cases, it is clear that the computational costs of our method are significantly lower than that of MTS. We then tested MTS and BoTLRet for recall, precision, and F1 measure while checking for performance deterioration. Table 5 shows the result of this comparison for BDPedia and MusicBrainz questions. Despite significant reductions in the required number of RDF triple searches by BoTLRet, we found that the BoTLRet performance was roughly equivalent to that of MTS.

Table 6 Performance comparison between QALD-2 challenge participant systems and BoTLRet for DBpedia test questions.

	Total answered questions	Recall	Precision	F1 Measure
SemSek	80	0.48	0.44	0.46
Alexandria	25	0.46	0.43	0.45
MHE	97	0.40	0.36	0.38
QAKiS	35	0.37	0.39	0.38
BoTLRet	75	0.94	0.94	0.94

In BoTLRet, selection of a triple-pattern case (i.e., PR-NP, NP-NP, or TOT) could be considered as an overhead because, in such a selection, BoTLRet requires positional frequencies, i.e., $PF_s(r), PF_p(r)$, and $PF_o(r)$ (shown in Sect. 3.1), for each of the keyword-related resources. However, even with this overhead, the computational cost of BoTLRet is lower. For example, for a two keyword group query, if we have m and n number of keyword-related resources, BoTLRet requires a $3 * m * n$ unit overhead computational cost to decide a triple-pattern case. However, this helps BoTLRet to reduce the required number of triple-pattern constructions to either $7 * m * n$, $8 * m * n$, or $15 * m * n$, while, for the same setting, MTS always constructs a $49 * m * n$ number of triple-patterns. Furthermore, this overhead computational cost can be avoided by introducing pre-calculated positional frequencies.

Based on the results shown in Tables 4 and 5, we conclude that, even with BoTLRet's quite low computational cost, the system shows almost the same level of performance as an exhaustive system such as MTS. Therefore, it can be said that BoTLRet fulfills the completeness competency requirement considering current proposal of template constructions and their merging.

5.3 Experiment 3

The third experiment was performed to evaluate the performance with other systems. Firstly, we evaluated the performance comparison between BoTLRet and QALD-2 challenge participant systems, specifically SemSek, Alexandria, MHE, and QAKiS. For BoTLRet, the answered questions were 75 DBpedia questions that had also been used in Experiment 1. Table 6 shows a performance comparison between the QALD-2 challenge participant systems and BoTLRet. In the evaluation report[†], the challenge participant systems reported on how many questions each system answered. Next, for the answered questions, each system reported its average recall, average precision, and average F1 measure. Table 6 columns two, three, four, and five, respectively, show these performance levels. The results are shown for DBpedia test questions because of their availability.

It can be seen that BoTLRet performs far better than the other systems. However, it is necessary to mention that the systems were not fully comparable, because BoTLRet is a keyword-based system, while the others are natural language based systems. However, we present this performance comparison based on the assumption that if the required keywords are given to BoTLRet, BoTLRet will work in a very sophisticated manner. We also acknowledge that automatic identification of such keywords will further increase complexities. This point will need to be investigated in our future work.

[†]http:/greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=home&q=2

Table 7 Performance comparison between GoRelations and our system for QALD-1 DBPedia test questions.

	Recall	Precision	F1 Measure
GoRalations [14]	0.722	0.687	0.704
BoTLRet	0.825	0.793	**0.801**

Next, we sought to compare BoTLRet with other keyword-based system. However, we were unable to find any keyword-based system had been evaluated using QALD-2, although we did find a system called GoRelations [14] which is, in some sense, a keyword-based system and which used the QALD-1 question set in its evaluation. Therefore, we compared BoTLRet and GoRalations using the QALD-1 test question set. Of the 50 questions in the set, we found that both GoRalations and BoTLRet were able to execute 33 questions in common, which were then compared for average recall, average precision, and average F1 measure. Table 7 shows that BoTLRet outperformed GoRelations in all three areas.

It was also found that the BoTLRet system performed well in comparison with other state linked data information retrieval systems. This indicates, for linked data information retrieval systems, the necessity of harnessing the internal structures and statistics of linked data.

6. Conclusions

Because the use of keywords is a comfortable choice for data retrieval, numerous researchers have worked to adapt keywords for use in linked data access. However, because keywords do not hold required linked data semantics, which are mandatory when performing linked data retrieval, a number of researchers have worked on template-based retrieval techniques that have the potential to bridge the gap between keywords and semantics. Until now, however, such initiatives have suffered from a lack of robust template establishment guidelines.

In this study, we proposed an outline for template construction, ranking, and merging that can be used for automatic keyword-based linked data retrieval. This method utilizes the internal statistics of the data during the template construction ranking processes. We have also introduced a template merging technique that makes multi-depth query construction possible. Because our method relies on the internal statistics of data, which are calculated automatically, we believe BoTLRet provides a very promising tool for use in achieving fully automatic information access of linked data. We have also introduced a binary progressive processing paradigm that is scalable for any number of query keywords.

Other than defining resource-representing tags, i.e., *rtag* (see Sect. 3.1), no special customization is required to adapt our proposal to a linked dataset. Furthermore, the systems *rtag* include labels, titles, etc., which are quite generic among datasets. Experiments conducted using our proposed system have shown generally positive outcomes, which indicates that the system provides a functional technique for use in linked data access.

We presume that our template creation technique could benefit from use in conjunction with other linked data access approaches, such as automatic ontology inclusion, feedback incorporation, and other sophisticated keyword matching techniques that can provide more appropriate templates, and hope to explore these possibilities in our future work. As our work depends on various statistical parameters, we also presume that the incorporation of off-line processing could increase the system's performance. This, we feel, is another promising area for our future investigations.

References

[1] C.C. Aggarwal and H. Wang, "Graph data management and mining: A survey of algorithms and applications," in Managing and Mining Graph Data, pp.13–68, Springer 2010.
[2] Z. Akar, T.G. Halaç, E.E. Ekinci, and O. Dikenelli, "Querying the web of interlinked datasets using void descriptions," Proceedings of the 21st World Wide Web Conference Workshop on Linked Data on the Web, pp.133–138, 2012.
[3] K. Alexander, M. Hausenblas, and J. Zhaox, "Describing linked datasets - on the design and usage of voiD, the 'vocabulary of interlinked datasets'," Proceedings of the 18th International World Wide Web Conference Workshop on Linked Data on the Web, 2009.
[4] T. Berners-Lee, "Linked Data - Design Issues," http://www.w3.org/DesignIssues/LinkedData.html, 2006.
[5] R. Bhagdev, S. Chapman, F. Ciravegna, V. Lanfranchi, and D. Petrelli, "Hybrid search: Effectively combining keywords and semantic searches," Proceedings of the 5th Extended Semantic Web Conference, pp.554–568, 2008.
[6] V. Bicer, T. Tran, A. Abecker, and R. Nedkov, "KOIOS: Utilizing semantic search for easy-access and visualization of structured environmental data," Proceedings of the 10th International Semantic Web Conference, pp.1–16, 2011.
[7] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data: The story so far," International Journal of Semantic Web Information System, vol.5, no.3, pp.1–22, 2009.
[8] G. Cheng and Y. Qu, "Searching linked objects with Falcons: Approach, implementation and evaluation," International Journal on Semantic Web Information System, vol.5, no.3, pp.49–70, 2009.
[9] M.A. Covington, "A dependency parser for variable-word-order languages," Derohanes (eds.), Computer assisted modeling on the IBM, vol.3090, pp.799–845, 1992.
[10] D. Damljanovic, M. Agatonovic, and H. Cunningham, "Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction," Proceedings of the 7th International Conference on The Semantic Web, pp.106–120, 2010.
[11] D. Damljanovic, M. Agatonovic, and H. Cunningham, "FREyA: An interactive way of querying linked data using natural language," Proceedings of the 8th International Conference on The Semantic Web, pp.125–138, 2011.
[12] L. Ding, T.W. Finin, A. Joshi, R. Pan, R.S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, "Swoogle: A search and metadata engine for the semantic web," Proceedings of the 13th ACM Conference on Information and Knowledge Management, pp.652–659, 2004.
[13] S. Ferré and A. Hermann, "Semantic search: Reconciling expressive querying and exploratory search," Proceedings of the 10th International Semantic Web Conference, pp.177–192, 2011.
[14] L. Han, T. Finin, and A. Joshi, "GoRelations: An intuitive query system for DBpedia," Proceedings of the 1st Joint International Semantic Technology Conference, pp.334–341, 2011.
[15] O. Hartig, C. Bizer, and J.C. Freytag, "Executing SPARQL queries over the web of linked data," Proceedings of the 10th International Semantic Web Conference, pp.293–309, 2009.
[16] T. Heath and C. Bizer, Linked Data: Evolving the Web into a Global Data Space, Morgan & Claypool Publishers, 2011.
[17] D.M. Herzig and T. Tran, "Heterogeneous web data search using relevance-based on-the-fly data integration," Proceedings of the 21st World Wide Web Conference, pp.141–150, 2012.
[18] M. Reichert, S. Linckels, C. Meinel, and T. Engel, "Studentś perception of a semantic search engine," Proceedings of the IADIS International Conference Cognition and Exploratory Learning in Digital Age, pp.139–147, 2005.
[19] J. Lehmann and L. Bühmann, "AutoSPARQL: Let users query your knowledge base," Proceedings of the 8th Extended Semantic Web Conference, pp.63–79, 2011.
[20] Y. Lei, V. Uren, and E. Motta, "Semsearch: A search engine for the semantic web," Proceedings of the 15th the International Conference on Knowledge Engineering and Knowledge Management, pp.238–245, 2006.
[21] V. Lopez, E. Motta, and V.S. Uren, "Poweraqua: Fishing the semantic web," Proceedings of the 4th Extended Semantic Web Conference, pp.393–410, 2006.
[22] C.D. Manning, P. Raghavan, and H. Schütze, An Introduction to Information Retrieval, Cambridge University Press, 2009.
[23] X. Ning, H. Jin, W. Jia, and P. Yuan, "Practical and effective IR-style keyword search over semantic web," Journal of Information Processing and Management, vol.45, no.2, pp.263–271, 2009.
[24] E. Prud'hommeaux and A. Seaborne, "SPARQL query language for RDF, W3C recommendation, W3C," http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115, Jan. 2008.
[25] M.-M. Rahoman and R. Ichise, "An automated template selection framework for keyword query over linked data," Proceedings of the 2nd Joint International Semantic Technology Conference, pp.175–190, 2012.
[26] S. Shekarpour, S. Auer, A.-C.N. Ngomo, D. Gerber, S. Hellmann, and C. Stadler, "Keyword-driven SPARQL query generation leveraging background knowledge," Proceedings of the 10th International Conference on Web Intelligence, pp.203–210, 2011.
[27] T. Tran, P. Cimiano, S. Rudolph, and R. Studer, "Ontology-based interpretation of keywords for semantic search," Proceedings of the 6th International Semantic Web Conference, pp.523–536, 2007.
[28] T. Tran, H. Wang, and P. Haase, "Hermes: Data web search on a pay-as-you-go integration infrastructure," Journal of Web Semantics, vol.7, no.3, pp.189–203, 2009.
[29] G. Tummarello, R. Delbru, and E. Oren, "Sindice.com: Weaving the open linked data," Proceedings of the 6th International Semantic Web Conference, pp.552–565, 2007.
[30] C. Unger, L. Bühmann, J. Lehmann, A.-C.N. Ngomo, D. Gerber, and P. Cimiano, "Template-based question answering over RDF data," Proceedings of the 21st World Wide Web Conference, ACM, pp.639–648, 2012.
[31] C. Unger and P. Cimiano, "Pythia: Compositional meaning construction for ontology-based question answering on the semantic web," Proceedings of the 16th International Conference on Applications of Natural Language to Information Systems, pp.153–160, 2011.
[32] D. Vallet, M. Fernández, and P. Castells, "An ontology-based information retrieval model," Proceedings of the 2nd European Semantic Web Conference, pp.455–470, 2005.
[33] H. Wang, Q. Liu, T. Penin, L. Fu, L. Zhang, T. Tran, Y. Yu, and Y.

Pan, "Semplore: A scalable IR approach to search the web of data," Journal of Web Semantics, vol.7, no.3, pp.177–188, 2009.

[34] M. Yahya, K. Berberich, S. Elbassuoni, and G. Weikum, "Robust question answering over the web of linked data," Proceedings of the 22nd ACM International Conference on Conference on Information, ACM, pp.1107–1116, 2013.

[35] G. Zenz, X. Zhou, E. Minack, W. Siberski, and W. Nejdl, "From keywords to semantic queries-incremental query construction on the semantic web," Journal of Web Semantics, vol.7, no.3, pp.166–176, 2009.

[36] Q. Zhou, C. Wang, M. Xiong, H. Wang, and Y. Yu, "SPARK: Adapting keyword query to semantic search," Proceedings of the 6th International Semantic Web Conference, pp.694–707, 2007.

Md-Mizanur Rahoman is a Ph.D. candidate in the Department of Informatics, The Graduate University for Advanced Studies, Tokyo, Japan. He also works as a lecturer (currently with study leave) in the Department of Computer Science and Engineering, Begum Rokeya University, Rangpur, Bangladesh. His research interests include intelligent information retrieval, semantic webs, and machine learning.

Ryutaro Ichise received his Ph.D. degree in computer science from Tokyo Institute of Technology, Tokyo, Japan, in 2000. From 2001 to 2002, he was a visiting scholar at Stanford University. He is currently an associate professor in the Principles of Informatics Research Division of the National Institute of Informatics, Japan. His research interests include semantic webs, machine learning, and data mining.