Received 26 March 2021
Revised 26 April 2021
Accepted 26 April 2021

A system for relation-oriented faceted search over knowledge bases

Taro Aso
Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Japan, and
Toshiyuki Amagasa and Hiroyuki Kitagawa
Center for Computational Sciences, University of Tsukuba, Tsukuba, Japan

Abstract

Purpose – The purpose of this paper is to propose a scheme that allows users to interactively explore relations between entities in knowledge bases (KBs). KBs store a wide range of knowledge about real-world entities in a structured form as (subject, predicate, object). Although it is possible to query entities and relations among entities by specifying appropriate query expressions of SPARQL or keyword queries, the structure and the vocabulary are complicated, and it is hard for non-expert users to get the desired information. For this reason, many researchers have proposed faceted search interfaces for KBs. Nevertheless, existing ones are designed for finding entities and are insufficient for finding relations.

Design/methodology/approach – To this problem, the authors propose a novel "relation facet" to find relations between entities. To generate it, they applied clustering on predicates for grouping those predicates that are connected to common objects. Having generated clusters of predicates, the authors generated a facet according to the result. Specifically, they proposed to use a couple of clustering algorithms, namely, agglomerative hierarchical clustering (AHC) and CANDECOMP/PARAFAC (CP) tensor decomposition which is one of the tensor decomposition methods.

Findings – The authors experimentally show test the performance of clustering methods and found that AHC performs better than tensor decomposition. Besides, the authors conducted a user study and show that their proposed scheme performs better than existing ones in the task of searching relations.

Originality/value – The authors propose a relation-oriented faceted search method for KBs that allows users to explore relations between entities. As far as the authors know, this is the first method to focus on the exploration of relations between entities.

Keywords Exploratory search, Knowledge base, Faceted search, Relation clustering

Paper type Research paper

1. Introduction

In recent years, much attention has been paid to knowledge bases (KBs), which store a wide variety of knowledge in a structured form. DBpedia (Auer *et al.*, 2007), Wikidata (Vrandečić and Krötzsch, 2014) and YAGO (Rebele *et al.*, 2016) are popular public KBs

This paper is based on results obtained from a project, JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

International Journal of Web Information Systems
Vol. 17 No. 6, 2021
pp. 698-713
Emerald Publishing Limited
1744-0084
DOI 10.1108/IJWIS-03-2021-0035

that have been used in many applications, such as question answering and knowledge discovery.

When using KBs, users often want to find out the desired information, e.g. interested entities, from the KBs. Meanwhile, typical KBs are stored using the Resource Description Framework (RDF) (Michael *et al.*, 2015), where any knowledge is represented as a set of triples comprising subject, predicate and object. Thus, KBs are regarded as complex knowledge graphs consisting of various entities and different types of relations, which make it difficult to search for the necessary information. More specifically, when using query languages like SPARQL (World Wide Web Consortium and others, 2013), users need to know the structure of the KB graph and the vocabulary as well. Alternatively, using keyword search, it is hard for the users to come up with appropriate keywords to specify the target information.

To help users find information from KBs, many researchers have developed the faceted search methods for KBs (Arenas *et al.*, 2004; Arenas *et al.*, 2014; Bast *et al.*, 2014; Brunk and Heim, 2011; Ferré, 2014; Hahn, 2010; Moreno-Vega and Hogan, 2018; Papadakos and Tzitzikas, 2014; Sherkhonov *et al.*, 2017) that allow users to interactively search over a KB by specifying interested values in predefined facets, thereby browsing entities stored in the KB. These methods are useful in particular for nonexpert users who are not familiar with the structure of the KB.

Notice here that, in many cases, users are interested in finding relations among real-world entities in KBs rather than finding entities themselves. For example, let us consider a user who wants to find information about space projects conducted by astronauts. In this case, one needs to be aware of the relation represented by a predicate. In DBpedia, http://dbpedia.org/ontology/ *mission* (*dbo:mission* for short) is the corresponding predicate and astronauts such as Yuri Gagarin and Neil Armstrong have it like *(dbr:Yuri_Gagarin, dbo:mission, dbr:Vostok_1)* and *(dbr:Neil_Armstrong, dbo:mission, dbr:Apollo_11)* (*dbr:* is a short-expression for http://dbpedia.org/resource/). However, he/she realizes that there are tens of predicates astronauts have. Astronauts have general predicates for humans such as *dbo:nationality* and *dbo:birthPlace* as well as specific predicates for astronauts. In such situations, it is hard for him/her to write an appropriate SPARQL query nor to use existing entity-oriented faceted query interfaces for KBs.

To this problem, we propose a novel *relation facet [1]* based on the relations between entities. One of the challenges in this problem is how to deal with hundreds of predicates used in a KB; for an entity, just listing tens to a hundred of associated predicates does not make sense. One possible way is to exploit the inherent type hierarchy of predicates. However, as we will see later, this does not always help us. To generate it, we apply clustering over predicates to get human-interpretable clusters of predicates. Then, we can generate a facet according to the clustering result.

Our main contributions in this paper can be summarized as follows:

- We propose the novel *relation facet* that allows users, who are interested in searching relations in a KB, in exploring relations between entities.
- To generate a relation facet, we propose to use a clustering method that categorize similar predicates in the same group. A key idea is to find predicates whose subjects and objects are similar. More precisely, we apply a couple of clustering methods, namely, agglomerative hierarchical clustering (AHC) and CP decomposition which is one of the tensor decomposition methods. We experimentally test their performance using a real data set.
- We conduct experiments to show that the proposed scheme performs better than existing faceted search interfaces in the task of searching relations between entities.

The rest of this paper is organized as follows. Section 2 introduces the basic interface of the faceted search method. Section 3 reports the related works about the faceted search for KBs and the relation clustering for KBs. Section 4 explains the scheme of the relation facet and the integration to the faceted search system. Section 5 provides our experimentation on the case study and user evaluation. Section 6 concludes this paper.

2. Preliminaries: faceted search

Section 2 describes the faceted search interface, which we use in this work as the method of searching KBs.

Faceted search is one of the exploratory search methods for entities characterized by different properties and has been applied in many systems, such as Amazon.com [2] and Apple iTunes [3]. The interface shows several facets that correspond to the properties of the target entities, along with the actual values and their numbers of occurrences. Users can browse interesting entities by choosing facets and the values of interest, thereby filtering out fewer interesting ones.

Figure 1 is an example of the faceted search interface. According to Tzitzikas *et al.* (2017), users start from the initial state and moves to another state. Each state has an extension that shows the set of items (or entities) being selected; an intention showing the condition/query satisfied by the items of the extension; and several facets and its values as transition markers, each of which leads to another state. Typically, users perform an initial query, like a keyword search, to obtain the initial extension, followed by performing facet selection and cancelation repeatedly to obtain refined search results.

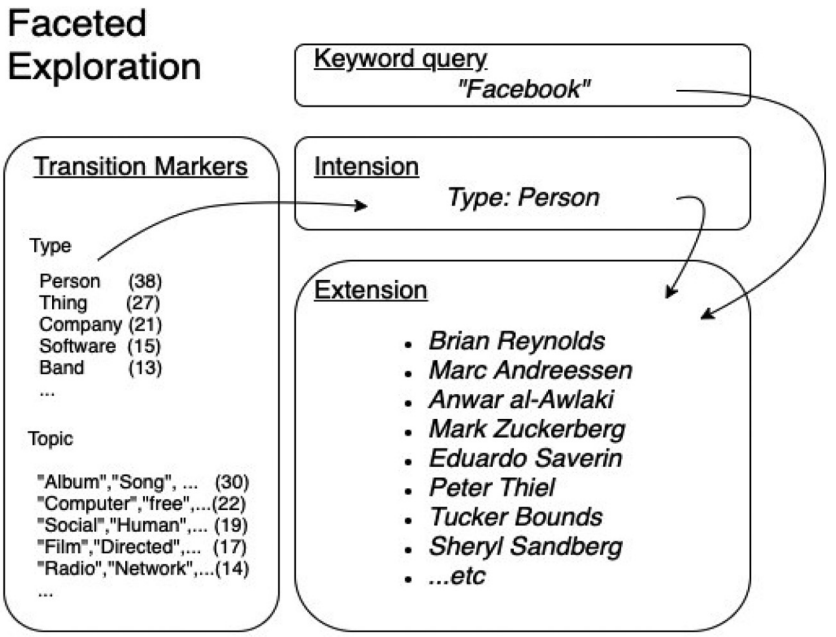

Figure 1.
Example of faceted search interface

3. Related work

3.1 Faceted search for knowledge bases

There have been many studies on faceted search for KBs. Brunk and Heim (2011) have proposed tFacet for DBpedia that uses hierarchical type information as facets by using the ontology of DBpedia. Arenas *et al.* (2014) have proposed SemFacet, which performs searches on YAGO. In SemFacet, properties of entities in RDF data are used for facets, and entities (URIs) are regarded as the search targets. The challenge is how to cope with the sparsity of information about entities, and they addressed this problem by inference using OWL2 ontologies. Papadakos and Tzitzikas (2014) have proposed a system called Hippalus that ranks facets. In Hippalus, one evaluates each facet during the search process and ranks and returns facets based on the facet evaluation, which allows ranking based on users' preference. Bast *et al.* (2014) have proposed a faceted search on Freebase, one of the knowledge bases. The main feature is that they refined the data set to improve the usefulness of the faceted search. Specifically, they removed or integrated redundant entities and properties included in the data set. Likewise, they also refined the taxonomy (classification) of types. Recently, (Moreno-Vega and Hogan, 2018) have proposed GraFa for Wikidata, which aims to speed up queries on large-scale KBs.

The above methods use existing properties of entities to generate facets and are designed to search for entities. On the other hand, we are more interested in finding interesting relations between entities, and the above methods are not sufficient to support such an objective. We will confirm this in the experimental evaluation section below.

3.2 Relation clustering for knowledge bases

Regarding the study of relation clustering, Zheng *et al.* (2016) have proposed the relation clustering for the entity navigation. They apply the information-theoretic co-clustering (Dhillon *et al.*, 2003) to group both of the classes of the object and the predicates simultaneously to present the organized information about the user interested entities. However, they are not considering the application to the faceted search for the exploration of relations but the entity navigation. Besides, to rank subjects and objects of RDF triples according to the characteristics of predicates, Franz *et al.* (2009) have proposed applying the CP decomposition to RDF triples, which is called TripleRank. The method was proven to perform well for ranking entities appearing either in subject or object. However, they did not consider the effectiveness in grouping predicates using CP decomposition. In this paper, we apply the CP decomposition to RDF triples for clustering the predicates and compare the result with ones of other methods.

4. Proposed method

In this section, we propose a novel *relation facet* where various predicates are grouped into a smaller number of groups, thereby allowing users who are interested in searching relations in a KB to browse interesting relations interactively. Then, we propose a relation-oriented faceted search system, RelFacet, where the relation facet is implemented by describing the user interface design and the system overview.

4.1 Relation facet

In a KB, many predicates connect different entities, but they are relevant to each other; for example, the ones used with entities of specific classes, the ones with a different URI but used in similar ways, the ones used with the specific entities and so on. To generate a useful classification, one possible way is to use RDF schema (Brickley *et al.*, 2014) that provides the definitions of possible predicates by specifying the class, properties, domain and range. Specifically, we can group predicates according to the class of the predicate, the domain and/or range classes. However, as we will see later, this is not always helpful for faceted search systems.

Instead of using the information in RDF schema, we attempt to clusters similar predicates according to their associated entities appearing in subject or object by applying a clustering method. Then, we generate a *relation facet* according to the clustering result, i.e. for each cluster, we generate a value in a facet by generating an appropriate label for the cluster. Specifically, we propose to use two different clustering methods. One is to apply the AHC based on the inherent information that can be observed from how entities are related to each other. Another one is to apply CP decomposition to capture latent relations between predicates and entities, which is inspired by TripleRank (Franz *et al.*, 2009).

4.1.1 Agglomerative hierarchical clustering. In this method, we group similar predicates into a cluster if there is a high degree of similarity that many entities appearing in the subject and/or object positions are common as below example.

Example. In the following triples, *dbo:mission* and *dbo:selection* could be grouped into the same cluster from the subject-based perspective, because both of them have a similar type of entities such as *Neil Armstrong* and *Yuri Gagarin* on the subject position. On the other hand, *dbo:birthPlace* and *dbo:nationality* are naturally related to a similar kind of entities in the object position such as *dbr:Wapakoneta,_Ohio*, *dbr:Klushino*, *dbr:Shropshire* and *dbr:United_Kingdom_of_Great_Britain_and_Ireland*. The two predicates could be members of the same cluster from the object-based perspective.

Triples

- (dbr:Neil_Armstrong, dbo:birthPlace, dbr:Wapakoneta,_Ohio)
- (dbr:Neil_Armstrong, dbo:mission, dbr:Apollo_11)
- (dbr:Neil_Armstrong, dbo:selection, dbr:NASA_Astronaut_Group_2)
- (dbr:Yuri_Gagarin, dbo:birthPlace, dbr:Klushino)
- (dbr:Yuri_Gagarin, dbo:selection, dbr:List_of_astronauts_by_year_of_selection)
- (dbr:Yuri_Gagarin, dbo:mission, dbr:Vostok_1)
- (dbr:Charles_Darwin, dbo:birthPlace, dbr:Shropshire)
- (dbr:Charles_Darwin, dbo:nationality, dbr:United_Kingdom_of_Great_Britain_and_Ireland)
- (dbr:Charles_Darwin, dbo:knownFor, dbr:On_the_Origin_of_Species)

The formulation is as follows. Let D be a KB consisting of triples (s, p, o), where s, p, and o are a subject, a predicate and an object, respectively. Let D^p denote the set of triples in KB D that contain predicate p. Besides, given a set of triples G, let us denote by $S(G) = \{s \mid (s, p, o) \in G\}$, $P(G) = \{p \mid (s, p, o) \in G\}$ and $O(G) = \{o \mid (s, p, o) \in G\}$ the set of subjects, predicates and objects in G, respectively. Then, we calculate the *subject-based similarity* $S_{i,j}$ between predicates p_i and p_j in $P(D)$ by:

$$S_{i,j} = \mathrm{sim}(S(D^{p_i}), S(D^{p_j})),$$

where $sim(\cdot)$ is a similarity measure over sets. In this work, we use the well-known Jaccard coefficient, i.e. $sim(A,B) \equiv \frac{|A \cap B|}{|A \cup B|}$, but other similarity measures can also be used. Likewise, the *object-based similarity* $O_{i,j}$ of predicates can be computed by:

$$O_{i,j} = sim\left(O(D^{p_i}), O(D^{p_j})\right)$$

Having computed two similarity values, namely, subject- and object-based similarity, we integrate them by taking the average.

Based on the similarity among the predicates, we then apply clustering to get a group of similar predicates. In this work, we use the AHC. Then, we make the clusters the relation facet of KB *D*. The following shows the detailed steps:

- We generate a similarity matrix by

$$\frac{S_{i,j} + O_{i,j}}{2}.$$

- Conduct an AHC method using the group average method.
- Generate clusters by setting an appropriate cutoff value.
- Generate a cluster label for each cluster by specifying a representative member of the cluster and selecting its label. Though there are several ways such as the most frequent predicate in each cluster is a representative member. This time, we manually named it. If there is only one member of the cluster, the label of that member is used.

4.1.2 CP decomposition. To extract a group of strongly related predicates, we can use a tensor decomposition against a tensor where a triple is regarded as a tensor. More precisely, a subject, a predicate and an object can be considered as third-order tensors. The tensor decomposition can be applied to approximate them as the product of low-ranked tensors. From this result, we can extract a group of strongly related predicates. In the following, we describe a method using the CP decomposition, which is one of the tensor decompositions.

The CP decomposition factorizes a tensor into a sum of component rank-one tensors. Rank-one tensos can be written as the outer product of N vectors, i.e. $X = a^{(1)\circ}\, a^{(2)\circ} \cdots a^{(N)}$. If so, An N-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is rank one (Kolda and Bader, 2009). For example, given a third-order tensor $\chi \in \mathbb{R}^{I \times J \times K}$, CP decomposition factorizes it as:

$$\mathcal{X} = \sum_{r=1}^{R} \lambda_r \quad \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r,$$

where R is a positive integer, λ_r is a *r-th* principal factor (pf) and $\boldsymbol{a}_r \in \mathbb{R}^I$, $\boldsymbol{b}_r \in \mathbb{R}^J$, $c_r \in \mathbb{R}^K$ for $r = 1, \ldots, R$. Elementwise,

$$x_{ijk} \approx \sum_{r=1}^{R} \lambda_r a_{ir} b_{jr} c_{kr} \; for \; i = 1, \ \ldots, I, j = 1, \ \ldots, J, \; k = 1, \ \ldots, K.$$

Matrices $\boldsymbol{A} \in \mathbb{R}^{I \times R}$, $\boldsymbol{B} \in \mathbb{R}^{J \times R}$, $\boldsymbol{C} \in \mathbb{R}^{K \times R}$ can be interpreted as low-dimensional representations of the decomposed tensor χ for each mode.

We make a relation facet following the below steps:

- We convert a triple set *G* composed of a subject set *S(G)*, an object set *O(G)* and a predicate set *P(G)* into tensors. Each subject, object and predicate is assigned an ID started from a number of 1. For example, a triple (dbr:Neil_Armstrong, dbo:mission, dbr:Apollo_11) is converted to $x_{123,12,1234} = 1$ (IDs here are just examples).The size of tensors is $|S(G)| \times |P(G)| \times |O(G)|$.
- Conduct the CP decomposition to the tensors specifying a parameter *R* indicating low-rank levels.

- Select principal factor λ_r and generate a cluster including the top-k predicates with high value.
- Generate a cluster label for each cluster by top-n predicates with high value, where n is smaller than k.

In this procedure, the parameters R, n and k are specified based on the derived result.

4.2 User interface

Figure 2 shows the user interface of the proposed RelFacet where the relation facet is integrated into the faceted search interface for KBs. Each component is described below:

- *Keyword query*: users input keywords in this field and choose either subject or object to specify the target to which the condition is applied.
- *Transition Markers*: users can select interesting value appearing in the facets, i.e., subject, predicate and/or object to refine the result shown in the extension.
- *Intension*: users can check the search intention by the information.
- *Extension*: it shows the current results whereby users can check whether or not there are any interesting results according to the current search intention.

4.3 Overview of the system

Figure 3 illustrates the system architecture. In response to the operation by the user, the system processes the request in the following way:

- First, the user inputs keywords as the initial input to qualify the entity appearing either in subject or object. RelFacet then retrieves the set of ranked triples that contain the user specified keywords in their text and shows them in the extension. It also shows the facets of the subjects, predicates (or relation) and objects according to the current set of triples.

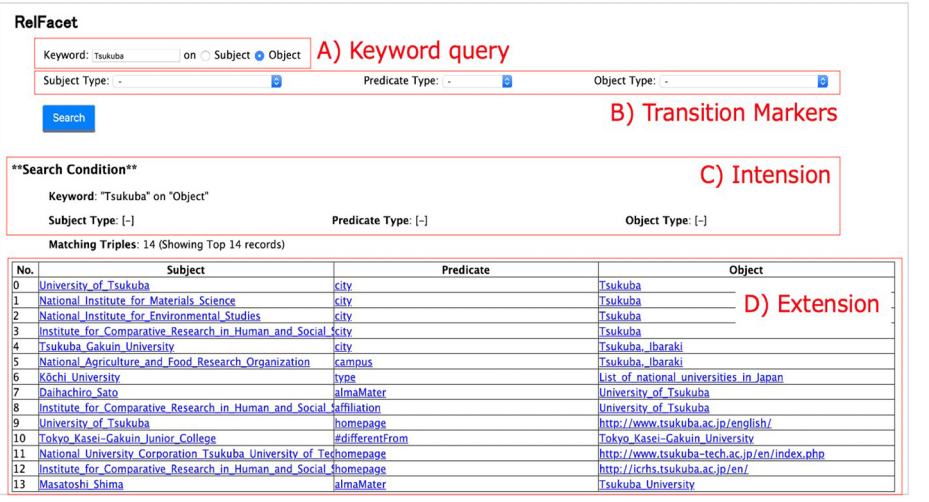

Figure 2.
User interface of RelFacet

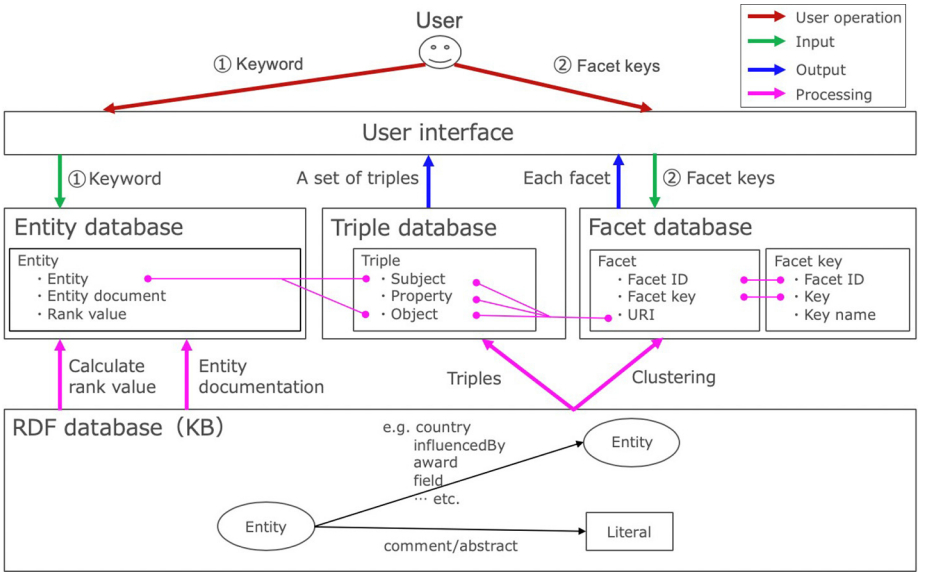

Figure 3.
Overview of the proposed system

- To further narrow down the current result, the user may select one of the keys in a facet based on his/her interest. RelFacet refines the current results and updates the facet as well.

To enable timely interaction against large KB and to support auxiliary facets that do not exist in the KB, we use a relational database. In Figure 3, there are three databases, namely, Entity database, Triple database and Facet database.

4.4 Supporting databases

4.4.1 Resource description framework database (knowledge base). The RDF database is used to store the triples of the target KB. More precisely, it stores two types of triples: the triples with literal values, such as comment and abstract, that describe entities and the triples representing relations between entities. The former is used to extract entities and their descriptions in the text (called *entity document*). The latter is used to generate relation facets and search results as well.

4.4.2 Entity database. The entity database stores information related to entities in terms of the entity's URI (universal resource identifier), the text description of the entity, which is the concatenation of literal values in the comment and the abstract and the entity's ranking value. As for the ranking value, we use the well-known PageRank algorithm (Page *et al.*, 1999) on the graph consisting of the entities and the relations among them. As PageRank cannot deal with different types of edges, we just ignored the labels of predicates bridging entities.

4.4.3 Triple database. The triple database stores the triples, i.e. subject, predicate (or relation) and object and is used to generate the search result shown in the extension space.

4.4.4 Facet database. The facet database uses two types of tables to maintain the information related to facet. A facet table corresponds to a facet associated with either entities or relations in terms of facet ID, facet key and URI of entity or relation.

A facet key table stores the correspondence between the name of a facet and their possible key values in terms of facet ID, facet key and key value.

5. Evaluation

In this section, we evaluate the proposed relation facet. First, we conduct an example-based case study using a real data set to compare the clustering results by the two proposed methods and a baseline method. Next, we conduct a user study to compare the performance of the proposed method with a couple of existing methods using a prototype system.

5.1 Experiment 1

We examine the relation facet generated by the two proposed methods and a baseline method to see whether the generated relation facet is useful. To implement the clustering by AHC in a proposed method, we used *SciPy* (Virtanen *et al.*, 2020) in Python 3.7, and we set the cutoff threshold for generating clusters 0.990 according to the dendrogram. To implement the clustering by CP decomposition in another proposed method, we used MATLAB R2020a and MATLAB Toolbox.

5.1.1 Baseline method. As the baseline method, we used a method based on the RDF schema, where predicates are classified according to the predicate hierarchy. Specifically, we group the predicates according to their super predicates. To generate a relation facet, we made each cluster a value in the relation facet, and we used the name of the super property as the label of the cluster.

5.1.2 Data set. We made the experimental data set as follows. From DBpedia 2016–10 (Freudenberg, 2017), we extracted all entities of the following classes: university, company, scientist, politician and astronaut and extracted the triples that describe the extracted entities from the Mappingbased Objects. The reason why we chose such five classes were related to the task design policy below for user study, i.e. we intended to homogenize the difficulty level among different tasks. For this reason, we carefully chose entity types such that entities in different classes were mutually related. Table 1 shows the statistical information of the data set.

5.1.3 Results.

Baseline method. Table 2 depicts the clusters of the relation facet generated by the baseline method. With the baseline method, we grouped 101 predicates into 12 clusters according to "rdfs:subPropertyOf." As we can see, some clusters involve relatively a large number of predicates, while others do not. Table 3 shows the list of predicates involved in the largest cluster "sameSettingAs" and according to "rdfs:comment" (www.ontologydesignpatterns.org/ont/dul/DUL.owl), it is defined as "A relation between two entities participating in a same Situation, e.g. "Our company provides an antivenom service (the situation is the service, the two entities are the company and the antivenom)." Although it is useful in many applications, we think that just showing its sub-predicates as a group to the end-user is not helpful to them, in particular when searching for interesting relations.

Table 1. Statistical information of the data set

	Astronaut	Company	Politician	Scientist	University	Total
# of Subjects	635	54,570	16,986	23,423	20,214	115,828
# of Predicates	11	29	54	31	28	101
# of Objects	1,536	117,567	27,829	51,067	39,461	219,968
# of Triples	5,157	349,860	78,733	172,382	113,587	719,719

Clusters	# of Properties
www.ontologydesignpatterns.org/ont/dul/DUL.owl#sameSettingAs	32
www.ontologydesignpatterns.org/ont/dul/DUL.owl#coparticipatesWith	26
www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasLocation	17
www.ontologydesignpatterns.org/ont/dul/DUL.owl#isMemberOf	7
www.ontologydesignpatterns.org/ont/dul/DUL.owl#isClassifiedBy	3
www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasRole	3
http://dbpedia.org/ontology/location	2
www.ontologydesignpatterns.org/ont/dul/DUL.owl#isDescribedBy	2
www.ontologydesignpatterns.org/ont/dul/DUL.owl#isPartOf	2
www.ontologydesignpatterns.org/ont/dul/DUL.owl#isParticipantIn	2
www.ontologydesignpatterns.org/ont/dul/DUL.owl#hasSetting	1
www.ontologydesignpatterns.org/ont/dul/DUL.owl#conceptualizes	1

Table 2. Result of the baseline method

appointer	monarch	predecessor	runningMate
athletics	nominee	president	service
child	owner	primeMinister	spouse
citizenship	owningCompany	principal	subsidiary
education	parent	provost	successor
incumbent	parentCompany	rector	superintendent
influencedBy	parentOrganisation	relation	viceChancellor
keyPerson	partner	relative	vicePresident

Table 3. List of predicates belonging to "sameSettingAs"

Agglomerative hierarchical clustering (AHC). On the other hand, in the proposed method of AHC, we can observe that 101 predicates are categorized into 23 clusters as shown in Table 4. We found that the members of each cluster are of similar entities in certain domains. Table 5 shows an example of two lists of predicates from the university and the academics clusters and the former is relevant to entities in the domain of university, while the latter is relevant to academic domains. We believe that such classification is more interpretable and informative to users than the baseline method which is based on RDF schema.

Clusters	# of Predicates	Clusters	# of Predicates
company	20	director	2
academics	15	ethnicity	1
university	11	deathCause	1
professionals	11	appointer	1
personalInformation	7	language	1
military	6	dean	1
politics	4	provost	1
governance	4	vicePresident	1
relationship	4	officerInCharge	1
organization	3	depiction	1
soundRecording	2	partner	1
principal	2		

Table 4. Result of the proposed method (AHC)

Predicates in "University" cluster	Predicates in "Academics" cluster
affiliation	academicAdvisor
athletics	almaMater
campus	award
chancellor	birthPlace
city	citizenship
country	deathPlace
head	doctoralAdvisor
sport	doctoralStudent
state	field
viceChancellor	influenced
differentFrom	influencedBy
	knownFor
	nationality
	notableStudent
	residence

Table 5.
Predicates of the "university" cluster and the "academics" cluster

CP decomposition. In this experiment, the original tensor (115,828; 219,968; 101) was approximated by 300 dimensions, but the result was not good (Table 6).shows the result indicating the clusters from 1st principal factor to 10th principal factor including top-5 predicates with high score. The results of the predicates are almost the same, and it is difficult to understand the characteristics of the cluster. Therefore, we can observe that CP decomposition performs worse than AHC in generating a relation facet.

To examine the reason, we investigated the fitness between the original tensor and the approximate tensor obtained by CP decomposition, which is defined by $1 - norm(X - M)/norm(X)$), where X is the original tensor and M is the approximated tensor by CP decomposition. The degree of the fitness was about 0.11 when CP decomposition was performed in 300 dimensions. We tried it with other dimensions, but it was about 0.07 in 100 dimensions and about 0.13 in 500 dimensions. It is considered that the low fit is owing to the sparsity of data.

5.2 Experiment 2: user study

Next, we conduct a user study to evaluate the performance of our method by comparing it with two existing faceted search systems, i.e. Precision Search and Find (http://live.dbpedia.org/fct/) and GraFa (http://grafa.dcc.uchile.cl/?lang=en).

Clusters (Principle factors)	Top 5 members
1	dbo:nationality,dbo:location,dbo:country,dbo:birthPlace,dbo:deathPlace
2	dbo:location,dbo:country,dbo:birthPlace,dbo:type,dbo:city
3	dbo:location,dbo:country,dbo:birthPlace,dbo:regionServed,dbo:locationCity
4	dbo:location,dbo:birthPlace,dbo:foundationPlace,dbo:residence,dbo:locationCity
5	dbo:location,dbo:locationCountry,dbo:nationality,dbo:birthPlace,dbo:residence
6	dbo:location,dbo:birthPlace,dbo:country,dbo:deathPlace,dbo:locationCountry
7	dbo:location,dbo:deathPlace,dbo:birthPlace,dbo:type,dbo:nationality
8	dbo:type,dbo:locationCountry,dbo:industry,dbo:foundationPlace,dbo:keyPerson
9	dbo:birthPlace,dbo:residence,dbo:country,dbo:deathPlace,dbo:locationCountry
10	dbo:location,dbo:locationCountry,dbo:type,dbo:regionServed,dbo:birthPlace

Table 6.
Result of the proposed method (CP decomposition)

5.2.1 Implementation of our system. Our proposed system RelFacet was implemented in Node.js and PostgreSQL 12.1 and executed on 3.3 GHz dual-core Intel Core i7, macOS Catalina with 16GB RAM.

5.2.2 Comparative systems. We chose Precision Search and Find and GraFa as the comparative systems. These user interfaces are shown in Figure 4. Precision Search and Find is developed by OpenLink and is available at DBpedia as a default faceted search system. It is a popular commercial system installed in various data sets. GraFa is a faceted search system for Wikidata and can be considered as one of the state-of-the-art systems. Notice that the data set of GraFa is not DBpedia, but it is still useful when considering the objective of our user study. Notice also that these systems are available on the Web.

5.2.3 Tasks. Unlike ordinary entity-centric query tasks, we have designed the tasks to search for such relationships strongly associated with a class of entities. Specifically, we have designed user tasks according to two types of scenarios, i.e. Explore and Find. In Explore tasks, a user is given a keyword, and he/she tries to understand what kind of information is stored. Because of the nature of this task, there is no predefined answer to each task. Meanwhile, in Find tasks, a user is given a question about a relationship regarding the given keyword, and he/she tries to find its answer from the KB using the interface. As an example, the task regarding NASA is shown in Table 7. Overall, we made five tasks in total about both kinds of tasks and selected three as the experimental tasks. This selective procedure is based on the perspective that there is a difference regarding the data set stored in the comparative systems. So, we carefully chose the tasks so that the task's level of difficulty is not affected by the system (and the data set).

5.2.4 Methodology. We asked eight volunteers to contribute to this experiment. According to our assumption in this research, we assumed end-users who were not familiar with both RDF and the KB being queried. First, we explained to them the overview of the

Figure 4.
User interfaces of comparative systems

Note: The left image is GraFa and the right one is Precision Search and Find

	Tasks
Explore	Search by a keyword NASA and check what information is available regarding NASA
Find	Astronauts are engaged in some missions. Find a property that describes the mission and answer who performed what mission.

Table 7.
Example of tasks about NASA

experiment, the structure of the KB, and the user interface and its operation of the experimental systems. Second, each user used the three experimental systems in random order and did the three tasks, e.g. Table 7. Notice that each user used either of the systems to perform a task without any overlap. Also, we set the time limit to each task, i.e. 2 min for each Explore task and 3 min for each Find task. We recorded the users' answers and their responses as operation logs. After the completion of all tasks, we asked the examinees to answer a questionnaire shown in Table 8 containing a five-point Likert scale ranging from 1 (Strongly Disagree) to 5 (Strongly Agree), as well as comments on the usability of the systems. We expected the participants would answer strongly agree (or strongly disagree) if the statement applied (or did not apply) at a high level. We thought the scale was popular and, in our experiment, we used the scale provided by Google Forms.

5.2.5 Results and discussions.

User evaluations. Figure 5 shows the evaluations w.r.t. the user experiences. The results show a similar tendency, i.e. the proposed RelFacet is ranked higher than the comparative systems, followed by GraFa and Precision Search and Find being ranked 2nd and 3rd, respectively. The difference of the scores is largest at Q4 ("How well did you find out which entity has a relation to which entity from the search result you get?"). For this question, RelFacet got the highest score among all questions, while others showed lower scores. We think that it shows the difference between the characteristics of each system. As Figure 4 shows, the comparative systems are oriented to search entities so that only entities are

Table 8. Questionnaire

	Questions
Q1	How easy were the search operations?
Q2	How useful was the information (facets) used to refine the search results to help search?
Q3	How easy was the property you need (which is the relationship between entities) to find?
Q4	How well did you find out which entity has a relation to which entity from the search result you get?
Q5	How useful are these systems?

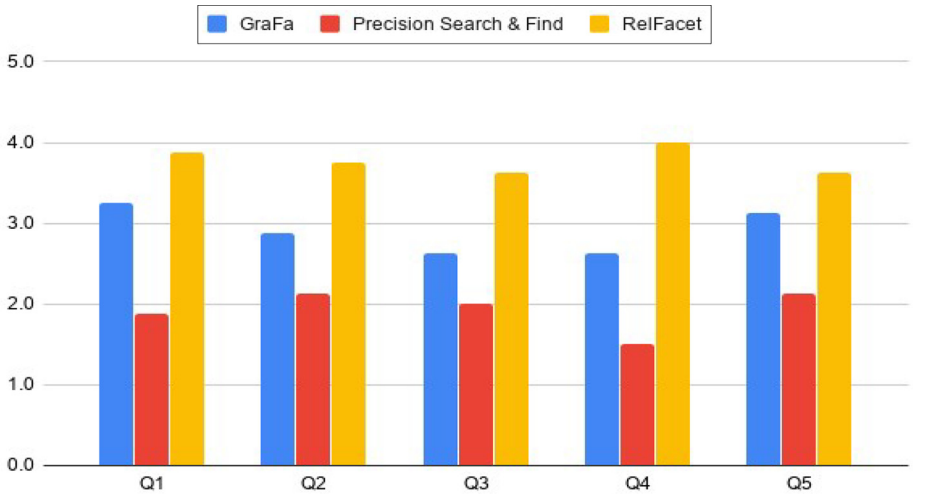

Figure 5. Results of questionnaire

presented in the extension's space. Predicates are just used as facets of entities, and the relations between entities are not so clear at a glance. However, our system is oriented to search relations between entities. Predicates are considered as relations between entities, and the search result is the relations between entities in the extension's space. While checking the relations in that space, users can explore the relations using a relation facet. It is useful in searching for relations between entities.

Success rates of Find tasks. Table 9 shows the success rates in the Find tasks for the three systems. RelFacet shows the highest scores, while GraFa shows low scores. One of the comments in the questionnaire says "GraFa is easy to understand the operation but hard to know where to find it because there are so many choices regarding facets (or properties)." We think that this is one of the major problems of GraFa being low in the success rate.

Number of operations in a find task. Table 10 shows the number of operations in a Find task. GraFa allows users to search with a little smaller number of operations than RelFacet. On the other hand, Precision Search and Find requires a lot of user interaction. One of the examinees made the following comment "GraFa and RelFacet were easier and simpler to use, where Precision Search and Find was not only slow but more difficult to navigate."

6. Conclusion

In this paper, we have proposed a relation-oriented faceted search method for KBs that allows users to explore relations between entities. For this method, we proposed a novel "relation facet" to find relations between entities. To generate it, we have applied two clustering methods over predicates, namely, AHC and CP decomposition. We experimentally showed that the clustering method based on AHC was better than the CP decomposition. Besides, the results of the user study have demonstrated that our proposed scheme's performance is better than those of existing methods when searching relations among entities.

Our future work includes comparing the data set to investigate what kind of data is appropriate for what kind of clustering method. Specifically, we have found the clustering method based on the Jaccard coefficient derives a good result for the dataset this time. We would try it and CP decomposition for different RDF data sets to see their effectiveness in detail. Besides, we plan to apply the proposed method to more massive data sets.

Table 9. Success rates of find tasks

	Rate
GraFa	0.38
Precision Search and Find	0.50
RelFacet	0.75

Table 10. Number of operations in a find task

	Median	Average
GraFa	9.0	9.6
Precision Search and Find	15.5	23.3
RelFacet	10.5	11.8

Notes

1. This work is an extension of the paper (Aso *et al.*, 2020).
2. www.amazon.com/
3. www.apple.com/itunes/

References

Arenas, M., Cuenca Grau, B., Kharlamov, E., Marciuska, S. and Zheleznyakov, D. (2004), "Faceted search over ontology enhanced RDF data", *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, New York, NY, pp. 939-948.

Arenas, M., Cuenca Grau, B., Kharlamov, E., Marciuska, S., Zheleznyakov, D. and Jimenez-Ruiz, E. (2014), "SemFacet: semantic faceted search over yago", *Proceedings of the 23rd International Conference on World Wide Web (WWW '14 Companion)*, New York, NY, pp. 123-126.

Aso, T., Amagasa, T. and Kitagawa, H. (2020), "Relation-oriented faceted search method for knowledge base", *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications and Services (iiWAS '20)*, pp. 192-199.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. and Ives, Z. (2007), "Dbpedia: a nucleus for a web of open data", *The Semantic Web*, Springer, Berlin, Heidelberg, pp. 722-735.

Bast, H., Bäurle, F., Buchhold, B. and Haußmann, E. (2014), "Easy access to the. Freebase dataset", *Proceedings of the 23rd International Conference on World Wide Web (WWW '14 Companion)*. New York, NY, pp. 95-98.

Brickley, D., Guha, R.V. and McBride, B. (2014), "Rdf schema 1.1. W3C. recommendation", *World Wide Web Consortium*, Vol. 2.

Brunk, S. and Heim, P. (2011), "tFacet: hierarchical faceted exploration of. Semantic data using well-known interaction concepts", *Proceedings of the International Workshop on Data-Centric Interactions on the Web*, p. 31.

Dhillon, I.S., Mallela, S. and Modha, D.S. (2003), "Information-theoretic co-clustering", *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, pp. 89-98.

Ferré, S. (2014), "Expressive and scalable query-based faceted search over. SPARQL endpoints", *International Semantic Web Conference*, Springer, pp. 438-453.

Franz, T., Schultz, A., Sizov, S. and Staab, S. (2009), "TripleRank: Ranking. Semantic web data by tensor decomposition", in Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta and Krishnaprasad Thirunarayan (Eds), *The Semantic Web ISWC*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 213-228.

Freudenberg, M. (2017), "ANN: DBpedia version 2016-10 released", *Listhub. w3. org*, Vol. 4.

Hahn, R., Bizer, C., Sahnwaldt, C., Herta, C., Robinson, S., Bürgle, M., Düwiger, H. and Scheel, U. (2010), "Faceted wikipedia search", *International Conference on Business Information Systems*, Springer, Berlin, Heidelberg, pp. 1-11.

Kolda, T.G. and Bader, B.W. (2009), "Tensor decompositions and applications", *SIAM Review*, Vol. 51 No. 3, pp. 455-500.

Michael, F., Basil, E., Carsten, M. and Achim, R. (2015), "A comparative survey. of dbpedia, freebase, opencyc, wikidata, and yago", *Semantic Web Journal*, Vol. 1 No. 1, pp. 1-5.

Moreno-Vega, J. and Hogan, A., (2018), "GraFa: Scalable faceted browsing for. RDF graphs", in Denny Vrandečić, K., Bontcheva, M.C., and Suárez-Figueroa, V. Presutti, Irene Celino, Marta Sabou, Lucie-AiméeKaffee and ElenaSimperl (Eds), *The Semantic Web – ISWC*, Springer International Publishing, Cham, pp. 301-317.

Page, L., Brin, S., Motwani, R. and Winograd, T. (1999), "The PageRank. Citation ranking: bringing order to the web", *Technical Report*, Stanford InfoLab. Stanford.

Papadakos, P. and Tzitzikas, Y. (2014), "Hippalus: preference-enriched faceted. Exploration", *EDBT/ICDT Workshops*, Vol. 172.

Rebele, T., Suchanek, F., Hoffart, J., Biega, J., Kuzey, E. and Weikum, G. (2016), "YAGO: a multilingual knowledge base from wikipedia, wordnet, and geonames", *International Semantic Web Conference*, Springer, Cham, pp. 177-185.

Sherkhonov, E., Grau, B.C., Kharlamov, E. and Kostylev, E.V. (2017), "Semantic faceted search with aggregation and recursion", *International Semantic Web Conference*, Springer, Cham, pp. 594-610.

Tzitzikas, Y., Manolis, N. and Papadakos, P. (2017), "Faceted exploration of. RDF/S datasets: a survey", *Journal of Intelligent Information Systems*, Vol. 48 No. 2, pp. 329-364.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., . . . van Mulbregt, P. (2020), "SciPy 1.0: fundamental algorithms for scientific computing in python", *Nature Methods*, Vol. 17 No. 3, pp. 261-272.

Vrandečić, D. and Krötzsch, M. (2014), "Wikidata: a free collaborative. knowledgebase", *Communications of the ACM*, Vol. 57 No. 10, pp. 78-85.

World Wide Web Consortium and others (2013), *SPARQL 1.1 Overview*, World Wide Web Consortium.

Zheng, L., Xu, J., Jiang, J., Qu, Y. and Cheng, G. (2016), "Iterative entity. navigation via co-clustering semantic links and entity classes", *European Semantic Web Conference*, Springer, Cham, pp. 168-181.

Corresponding author
Taro Aso can be contacted at: aso@kde.cs.tsukuba.ac.jp