algorithms

MDPI

Article

Property-Based Semantic Similarity Criteria to Evaluate the Overlaps of Schemas

Lan Huang [1], Yuanwei Zhao [2], Bo Wang [1], Dongxu Zhang [2], Rui Zhang [1,*], Subhashis Das [3], Simone Bocca [4] and Fausto Giunchiglia [4]

[1] College of Computer Science and Technology, Jilin University, Changchun 130012, China; huanglan@jlu.edu.cn (L.H.); bowang20@mails.jlu.edu.cn (B.W.)
[2] College of Software, Jilin University, Changchun 130012, China; zhaoyuanwei0410@163.com (Y.Z.); dongxu20@mails.jlu.edu.cn (D.Z.)
[3] CeIC & ADAPT, School of Computing, Dublin City University, 999014 Dublin, Ireland; subhashis.das@dcu.ie
[4] DISI, University of Trento, 38123 Trento, Italy; simone.bocca@unitn.it (S.B.); fausto.giunchiglia@unitn.it (F.G.)
* Correspondence: rui@jlu.edu.cn

Abstract: Knowledge graph-based data integration is a practical methodology for heterogeneous legacy database-integrated service construction. However, it is neither efficient nor economical to build a new cross-domain knowledge graph on top of the schemas of each legacy database for the specific integration application rather than reusing the existing high-quality knowledge graphs. Consequently, a question arises as to whether the existing knowledge graph is compatible with cross-domain queries and with heterogenous schemas of the legacy systems. An effective criterion is urgently needed in order to evaluate such compatibility as it limits the quality upbound of the integration. This research studies the semantic similarity of the schemas from the aspect of properties. It provides a set of in-depth criteria, namely coverage and flexibility, to evaluate the pairwise compatibility between the schemas. It takes advantage of the properties of knowledge graphs to evaluate the overlaps between schemas and defines the weights of entity types in order to perform precise compatibility computation. The effectiveness of the criteria obtained to evaluate the compatibility between knowledge graphs and cross-domain queries is demonstrated using a case study.

Keywords: data integration; knowledge graph; ontology evaluation; schema overlap; semantic similarity

Citation: Huang, L.; Zhao, Y.; Wang, B.; Zhang, D.; Zhang, R.; Das, S.; Bocca, S.; Giunchiglia, F. Property-Based Semantic Similarity Criteria to Evaluate the Overlaps of Schemas. *Algorithms* **2021**, *14*, 241. https://doi.org/10.3390/a14080241

Academic Editors: Haridimos Kondylakis and Nikolaos Papadakis

Received: 23 June 2021
Accepted: 3 August 2021
Published: 17 August 2021

1. Introduction

The rapid emergence of management information systems challenges the classical database techniques to respond to queries across multiple heterogeneous legacy database systems. A mediator serves as middleware between such database systems and the unified query services. It rewrites the queries and unites the answers according to the integrated schemas of these databases. Consequently, an unavoidable problem arises regarding how to choose the candidate databases to fulfill cross-system queries.

For example, suppose that we are querying the profile of a student using the multiple legacy information systems in a smart university scenario: information is collected regarding their grades from the academic affairs office, regarding their activities from the student union, regarding their reading and thesis from the library, their physical records from the hospital, etc. The classical mediator maps the universal query to the local queries for each data source and reformats the answers in the form of a universal query. This solution relies heavily on the universal knowledge graph, which mediates all the schemas of the data sources. A straightforward solution is to construct cross-domain knowledge using all these systems and then build the mediator on top of this. However, this is not always applicable or economical, for example, for extension in the future. A more practical solution is to reuse the state-of-the-art general-purpose knowledge graphs such as DBpedia, schema.org, etc. Consequently, the question becomes how to choose from the existing knowledge graphs.

Different approaches have been proposed, with frameworks from the perspective of engineering demonstrating considerable success, but only a few have considered the evaluation phase of this problem. Furthermore, most of the research involved qualitative discussions rather than computational criteria.

This work is based on the European project 'ISCF HDRUK DIH Sprint Exemplar: Graph-Based Data Federation for Healthcare Data Science (https://gtr.ukri.org/projects?ref=MC_PC_18029 (accessed on 24 December 2019)), which aims to provide a solution to the integration of medical domain heterogeneous legacy systems of Scotland and Italy. On the surface, reusing the data of these legacy systems leads to data integration problems [1], but it is inefficient to consider only how to integrate the data. Therefore, we realized data integration through schema integration. During the project, we encountered problems related to semantic diversity on both the language level and knowledge level [2]. There were mismatches between the terms in the two natural languages, i.e., English and Italian, even in the same domain. Language-level diversity problems were solved by general-purpose dictionaries in a relatively straightforward manner. Knowledge-level diversity is defined as the many-to-many mappings between the **entity types** and the properties [3].

A knowledge graph can be divided into a schema knowledge graph (*hereinafter referred to as* **schema**) at the knowledge level and a data knowledge graph at the individual level. A **schema** is a similar concept to the *database scheme*, and it contains **entity types (similar to a table in a database)** and **properties (similar to attributes in a database)**. A *data knowledge graph* is a similar concept to the *record level* in a database, and it contains entities and property values. Figure 1 shows two different schemas in the same scenario. Individuals are difficult to reuse because of the structural differences, but schemas at the knowledge level can be reused in many general fields. Schema.org and DBpedia [4] contain a large number of high-quality schemas with uniform specifications. When constructing a new knowledge graph, effectively reusing these existing high-quality schemas can save energy and avoid conflicts between new schema and existing schema.

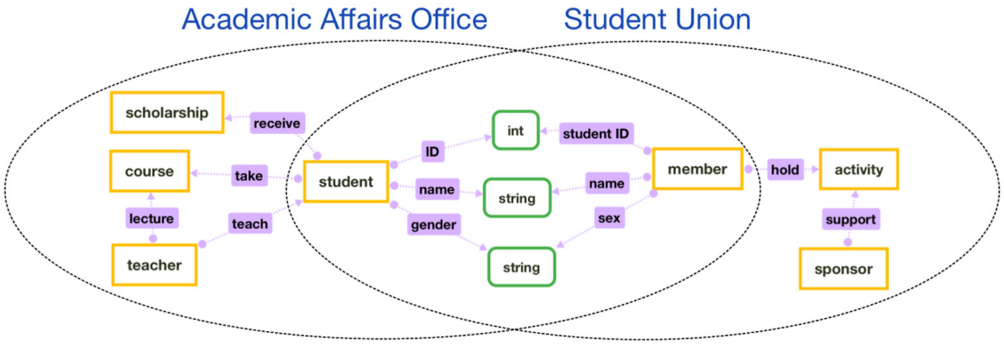

Figure 1. Equivalent entity type student and member in different schemas.

We use the student profile scenario shown in Figure 1 to motivate our work. There is a table named *student* in the database of the academic affairs office and another table named *member* in the database of the student union. The student *Mary* has records in both. Language-level disambiguation alone does not resolve this kind of mismatch. A general-sense knowledge graph will never assume this semantic equivalence either. Therefore, it is necessary for an effective criterion on the knowledge level to verify this semantic similarity on the overlaps of the two schemas. This paper proposes property-based semantic similarity criteria to evaluate the overlaps of schemas. For example, *name* and *taught-by*, in the table *student*, are properties about the entities, such as *Mary*, of that entity type (in a database table). An entity type is denoted by a feature vector formed by such properties. Coverage and flexibility are used to compute the compatibility between the schemas. Firstly, the vectors are disambiguated by the language-level tools and then matched according to the semantic similarity in three levels: label, property and individual. Secondly, the thresholds for each level are found to be dependent on the domain. Thirdly, the properties are propagated through the *is-a* hierarchy of the entity types and properties to accumulate the weights of those entity types. The weights are exploited to calculate the schema overlaps. Finally, the coverage and flexibility are computed to help choose the schemas.

The contributions of the paper are summarized as the following:

1. A three-fold semantic similarity is proposed to substitute the classical identity to compute schema overlaps;
2. A property-based weight is defined to extend the influence of the entity types to compute the coverage and flexibility;
3. A set of rules is accumulated to apply the thresholds in the practical medical domain.

The rest of the paper is organized as follows: in Section 2, we will introduce the related work. In Section 3, we will introduce the related concepts. In Section 4, we will give the measurement method of semantic equivalence and the formula of quantitative calculation. In Section 5, we will define weight and the calculation method of two evaluation criteria based on weight and we will describe experiments carried out to verify the correctness and effectiveness of the proposed method in Section 6. The last section is Conclusions and Future Work.

2. Related Work

In recent years, the amount of data has exponentially increased but much of this amount exists in the legacy system and does not demonstrate its value. Many people are committed to solving this problem. Alexandrova et al. [5] proposed a solution for the public sector legacy system. Tomak et al. [6] proposed a new method to evaluate the performance of fault management mechanisms in distributed real-time legacy software systems. Golchin et al. [7] proposed a boomerang system which integrates a legacy non-real-time OS that is customized for timing-sensitive tasks. However, they did not pay attention to how legacy data can be used effectively. If we want to make use of legacy data, we should first standardize the chaotic data. The schema proposed in this paper is to meet this exact need. First, the schema level is fused, which is associated with ***ontology integration or knowledge graph fusion.*** Then, the data level is fused. In addition, the methods of ***ontology reuse*** [8] and ***ontology matching*** [9] are used in the process of ontology integration. Next, we will introduce the related work in these areas.

In the past few years, more and more ontologies have been made available with the help of existing tools, such as linked open vocabularies (LOVs) [10], Bioportal (biomedicine) [11], etc. However, when aggregating these ontologies across domains, such as TKM [12], etc., we may need to integrate two different ontologies into the third ontology, and then contradictions may arise [13]. The heterogeneous problems [14] that are difficult to deal with in ontology also need to be solved. There have been some solutions to the normalization of ontology, such as WIDOCO [15], which records the wizard of ontology, and VoCol [16], which supports the integrated environment of version-controlled vocabulary development. Fèrnandez et al. investigated this and found that the heterogeneity between the required conceptualization and the existing ontology is an important obstacle to the development of knowledge integration.

Several semi-automatic methods of ontology integration have been proposed, such as semi-automatic generation of property semantics based on ontology integration [17] and automatic generation of new ontology patterns based on ontology reuse. They include an ontology integration method based on knowledge graphs and machine learning technology, ontology integration [18] based on ontology matching and its related technologies [19], the model ranking of evaluating ontology based on semantic matching [20] and the method of constructing ontology from an ontology pattern [21] and compatibility index for comparing and aggregating ontologies [22]. However, when these methods evaluate ontologies, they all use qualitative methods to measure the relationship. The computability is not considered.

In recent years, related research in the field of knowledge graph has grown rapidly. In many studies [23], a knowledge graph is regarded as a cleaned knowledge base, which is composed of individuals. To a certain extent, we can think that knowledge graph $\approx$ ontology + knowledge base [24]. Now there are many knowledge graphs. It is very feasible to use the schema level of these knowledge graphs.

Next, we will briefly introduce some related work of knowledge graph reuse and knowledge graph fusion. In 2019, He et al. [25] proposed a connectionist framework, which uses a manufacturing knowledge graph to solve the problem of integration. In the same year, Wu et al. [26] tried to realize the construction of multiple online encyclopedias by using knowledge graph fusion and reuse technology. To achieve the purpose of merging multiple encyclopedias into a large knowledge graph. Shen et al. [27] reused the existing knowledge base to build the knowledge graph. They used the incomplete domain-specific knowledge graph and integrated the reusable knowledge base to build a user-friendly medical knowledge graph. However, these methods are for specific functional areas and are not universally applicable. Moreover, when they selected the existing knowledge graph and database, they did not consider the evaluation criteria. This leads to the reuse of inappropriate knowledge graphs and data, resulting in the low performance of later fusion.

In the process of reusing a knowledge graph, it is necessary to use appropriate evaluation criteria. Fan et al. [28] put forward several ontology evaluation criteria, such as semantic intensity, wealth, depth and adaptability. Oh et al. [29] proposed three criteria of ontology modularization and evaluation tools, namely tool performance, data performance and usability. Dastgerdi et al. [30] discussed ontology evaluation criteria, approvals and layers. They introduced the standards mentioned by experts in the field of ontology and the standards proposed by the national ontology research center of the United States. Hoo et al. [31] proposed a framework to guide the selection of appropriate standards for various ontology evaluation levels and methods. Further, they identified overlaps and established the relationships of the various criteria. However, their criteria only focus on ontology itself. These criteria and methods cannot be applied in the schema fusion process. Coverage [32] and flexibility [33] are important criteria to evaluate the overlapping effect between schemas. In 2020, Giunchiglia and Fumagalli carried out a preliminary exploration on quantitative evaluation [34], but the evaluation method regarded the importance of all entity types as the same, without considering the differences among them. In the same year, Park tried to measure the importance of nodes in a knowledge graph [35]. However, this study mainly focused on the influence of inputs from different sources on nodes and did not pay attention to the influence of the relationship between nodes in the knowledge graph.

Our main concern is the overlapping degree in the process of schema fusion. In solving the schema fusion problem, the reuse of high-quality schema is an important issue. When deciding whether a schema is applicable, the quantitative computation of measurement criteria is particularly critical. At present, there is little attention paid to the internal entity of schema. The important contribution of this paper lies in giving a reasonable quantitative computation method of evaluation criteria in the schema fusion process.

3. Problem Definition

As is shown in the related work, it is vital to evaluate the overlaps of schemas to reuse high-quality knowledge graphs. The concepts of coverage and flexibility should be defined in the perspective of properties in a computable way. Since the concept of knowledge graphs came into being and was developed, there have been the emergence of many domain-specific and cross-domain knowledge graphs. However, most of these knowledge graphs are constructed from scratch, with the disadvantage of "built once, used once". On the one hand, this consumes too much time and energy for each case; on the other hand, it leads to high probability of conflicts in the interactions among the multiple applications based on different knowledge graphs. Therefore, it is more practical to reuse than to build a new knowledge graph.

After we recognize the importance of knowledge graph reuse, two important issues are worth thinking about.

- What kind of knowledge graph can I reuse to solve my need? This leads to the introduction of the schema-level knowledge graph.
- How to determine if an existing knowledge graph is suitable for use? This asks for a quantitative evaluation method.

A knowledge graph can be regarded from the schema level (with the classes and properties) and the data level (with the individuals). Once we have constructed a schema for a knowledge graph, it can be populated via different sets of individuals. Such knowledge graphs with individuals are difficult to reuse [36], but the schema itself can be reused rather easily. Common sense knowledge graphs such as Schema.org, DBpedia, etc. contain many high-quality schemas with uniform specifications. When constructing a new knowledge graph, it will save energy and avoid conflicts if we can reuse these existing high-quality schemas effectively. Now the question is to determine whether an existing schema is suitable to reuse. The first step is to determine the entity types and the properties in the **query** (short for competency query [37]) provided by users. The next step is to choose an existing schema that can cover most of the entity types and properties. In this process, the key is to measure overlaps of the schema onto the query to determine whether the schema selected is suitable.

Coverage and flexibility are evaluation criteria used to measure the overlaps of schemas. Coverage focuses on the overlaps of one schema on another, while flexibility focuses on the redundancy after one schema covers another. Suppose there are two schemas, A and B. The calculation formula defined in the form of a set in Formulas (1) and (2) is abbreviated as $Cov(A,B)$ and $Flx(A,B)$. Note that the two parameters A, B are ordered. It means that the coverage degree of schema A over B is not equal to the coverage degree of schema B over A. In the experiment in Section 5, we use these two criteria to measure the overlapping effect between the query [38], the schema and the datasets.

At present, the definitions are based on set theory. $Cov(A,B)$ is the ratio of the part of schema A that covers B $(A \cap B)$ divided by schema B. $Flx(A,B)$ is the ratio of the part of schema A that does not cover B $(A - B)$ divided by schema A. Neither criterion is symmetric.

$$Cov(A,B) = \frac{A \cap B}{B} \tag{1}$$

$$Flx(A,B) = \frac{A - B}{A} \tag{2}$$

In Formula (1), "$A \cap B$" represents the "same" part of A and B and "$A - B$" in Formula (2) represents the part of A that is different from B, that is, the part of A not covered by B. However, it is not obvious that A and B are schemas that are not classical sets. To compute quantitatively the two criteria, the equivalence of entity types in the schema should be well defined.

4. Semantic Equivalence

The evaluation criteria defined in the form of a set are introduced in the previous section. The next step is to compute them quantitatively. Taking $Cov(A,B)$ as an example, in Formula (1), A and B are schemas, and $A \cap B$ refers to the "equivalent" entity types. This section mainly describes the semantic equivalence of the entity types. Let us imagine several scenarios. Firstly, if all properties of two entity types are the same (semantically equivalent), then the two entity types are very likely to represent the same thing in the real world. Secondly, if two entity types in different schemas have the same individuals, they are similar in a certain way. The second situation is very common in life, as a person is likely to leave the same record in different databases. Additionally, these two individuals instantiate different entity types in the different schemas. Although we name the two entity types differently (in different schemas), the meanings of the two entity types are similar, at least for this person.

It is rare to have two entity types to be 100% the same in the two schemas. If the similarity is high to a certain extent, they can be taken as equivalent entity types. The similarity of entity types lies in three levels.

- Label level, such as polysemy, can be calculated by similarity at the language level. If the similarity between two entity types at the label level reaches a prescribed threshold, they can be directly considered as equivalent.
- Property level, such as entity types with similar properties, for example, the entity types *student* and *member* in Figure 1. Although these two entity types are in different schemas, they have very similar properties, which shows that these two entity types are similar at the property level. Similarly, we should judge the similarity of different properties at the label level, such as properties *gender* and *sex*, *ID* and *student ID* in Figure 1 as representing the same meaning. If the similarity between two entity types at the property level reaches a prescribed threshold, it can be judged that the two entity types are equivalent.
- Individual level, from the bottom up. Suppose *Mary* is an individual of entity type *student* both in the database of the academic affairs office and an individual of entity type student in the database of the student union, as shown in Figure 1. It means that two different entity types correspond to the same individual, which shows that the entity type *student* and the entity type *member* are potentially similar (at least related). It might be rational to view the entity type *student* and the entity type *member* as the same from the different schemas.

The following is the formal definition of entity type equivalence in the three aspects.

Definition 1. *Semantic Equivalence:*

The semantic similarity of two entity types U, V can be calculated with Formula (3).

$$Sim_s(U,\ V) = \alpha Sim_L(U,\ V) + \beta Sim_P(U,\ V) + \gamma Sim_I(U,\ V) \tag{3}$$

where

$$Sim_L(U,\ V) = |\overline{U}, \overline{V}|$$

In which $\overline{P}$ is the vector formed by the natural language labels of P and $|\overline{U}, \overline{V}|$ returns the semantic distance [39] of the two vectors;

$$Sim_P(U,\ V) = \sum\nolimits_{i,j} Sim_L(P_{u,i},\ P_{v,j})$$

where $P_{u,i}$ and $P_{v,j}$ are properties of U and V, respectively,

$$Sim_I(U,\ V) = \sum\nolimits_{i,j}^{m,n} e(I_{u,i},\ I_{v,j}) / Min(m,\ n)$$

where $I_{u,i},\ I_{v,j}$ are among the m, n individuals of U, V, respectively, and

$$e(I_u,\ I_v) = \begin{cases} 1, \text{if } I_u \text{ refers to the same individual as } I_v, \\ 0, \text{otherwise.} \end{cases}$$

Moreover, to compute Formula (3), the three factors take effects in the following conditions.

$$(\alpha,\ \beta,\ \gamma) = \begin{cases} (1,0,0),\ if\ Sim_L(U,\ V) > T_L \\ (1,1,0), if\ Sim_L(U,\ V) \le T_L\ and\ Sim_P > T_P \\ (1,1,1),\ otherwise. \end{cases}$$

Here, T_L, T_P are the predefined thresholds for the similarity on the label level and property level, respectively. Furthermore, an overall threshold T_S $(T_S < T_L \; and \; T_S < T_P)$ is defined with the empirical value from an expert to simulate the semantic equivalence. For example, for the entity types *student* in the schema of the education affairs office and *member* in the schema of the student union as shown in Figure 1, if $Sim_s\ (student,\ member) > T_s$, the two entity types are considered semantically equivalent.

Theorem 1. *The method proposed in Definition 1 can measure the semantic equivalence of two entity types.*

Proof. First, if two entity types $U,\ V$ reach a given threshold at the label level, there is

$$Sim_L(U,\ V) > T_L$$

then in Formula (3), $(\alpha,\ \beta,\ \gamma) = (1, 0, 0)$, there is

$$Sim_s(U,\ V) = Sim_L(U,\ V) > T_L$$

According to the limitation of $T_S,\ T_S < T_L$, we can deduce

$$Sim_s(U,\ V) > T_S$$

We can conclude that the two entity types $U,\ V$ are regarded as semantically equivalent. When $Sim_L(U,\ V) < T_L$ but $Sim_P > T_P$, then $(\alpha,\ \beta,\ \gamma) = (1, 1, 0)$, and there is

$$Sim_s(U,\ V) = Sim_L(U,\ V) + Sim_P(U,\ V)$$

$$Sim_s(U,\ V) > Sim_L(U,\ V) + T_P > T_P$$

and T_S meets $T_S < T_P$, so

$$Sim_s(U,\ V) > T_S$$

which indicates that the two entity types $U,\ V$ are regarded as semantically equivalent.

Finally, in the third case when $Sim_L(U,\ V) \leq T_L \; and \; Sim_P \leq\ T_P$, there is $Sim_s(U,\ V) = Sim_L(U,\ V)\ + Sim_P(U,\ V) + Sim_I(U,\ V)$. If $Sim_L(U,\ V) + Sim_P(U,\ V) + Sim_I(U,\ V) > T_S$, we can deduce $Sim_s(U,\ V) > T_S$. □

In conclusion, when the calculated results of two entity types at the level of label and property reach the given threshold, the two entity types can be regarded as directly equivalent. The three levels label, property and individual can comprehensively evaluate the equivalence of the two entity types from multiple perspectives.

The classical equivalence of two entity types lies only on the label level. Potentially equivalent entity types are usually neglected because of different labels. Such different labeling happens frequently for entity types from different schemas. This exaggerates the difference between the entity types, which in turn reduces the precision of the criteria to evaluate the overlaps between schemas. We consider the particularity of entity type in schemas and enhance the measurement of entity types. Our method not only considers the label level, which is compatible with the previous calculation methods, but also considers the property level and individual level. This method can find more equivalent entity types.

5. Computation of Coverage and Flexibility

With the semantic equivalence calculation method of entity types, $A \cap B$ is transformed into the equivalent entity types both in schema A and schema B. $Cov(A, B)$ and $Flx(A, B)$ can be calculated quantitatively. In this section, we introduce the weight of entity types based on properties. Then, we introduce the quantitative calculation formula of $Cov(A, B)$ and $Flx(A, B)$ based on the weight.

5.1. Weight

Before the definition of weight, let us take the schema (Figure 2) extracted in Figure 1 as an example (hereinafter referred to as **schema A**).

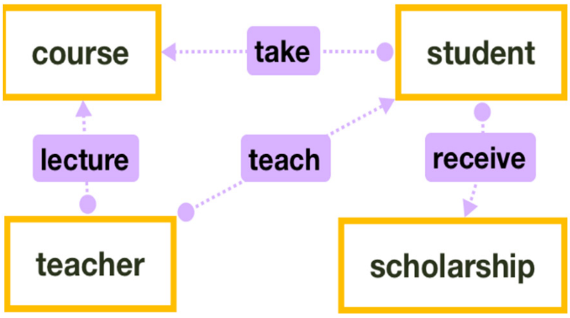

Figure 2. Schema A extracted from Figure 1.

There are four entity types in schema A {*student, teacher, course, scholarship*} and four object properties {*receive, take, teach, lecture*} between these four entity types. Among these four entity types, *student* is associated with the other three entity types, while *scholarship* is associated with *student* only. Suppose there is another schema referred to as B containing entity type {*teacher, course, scholarship*} without entity type *student*. Entity type *student* in schema A is not covered by schema B. The consequences brought by this mismatch are relatively serious because the absence of entity type *student* will lead to a lack of connection between entity type *student* and other entity types, which will easily lead to the incomplete and inaccurate phenomenon when we use schema B to cover schema A. Semantically, if entity type *student* is missing in schema B, $Cov(A, B)$ should be lower in this case.

In another case, if schema B contains the entity type {*teacher, course, student*} without entity type *scholarship*, it means that schema B can cover all entity types in schema A but entity type *scholarship*. This kind of incompleteness will not affect the relationship between other entity types because *scholarship* is associated with *student* only. That is to say, the negative impact is relatively small, so the computation result of $Cov(A, B)$ should be greater than in the previous situation.

Similarly, suppose there is a schema C containing entity type {*teacher, course, scholarship*} without entity type *student*. Now we need to calculate the redundancy degree of schema A after it covers schema C, that is, $Flx(A, C)$. There is no entity type *student* in schema C while entity type *student* is very important in schema A, which means that entity type *student* in schema A is very redundant. That is, schema A has serious redundancy. Therefore, $Flx(A, C)$ is large. On the contrary, if schema C contains entity type {*teacher, course, student*} without entity type *scholarship*, the redundancy degree will be relatively low. Additionally, $Flx(A, C)$ should be smaller than in the previous situation because redundant entity type A in schema C is not as important in schema C.

According to the example above, the intuition is that different entity types should be associated with different weights according to their relationship with other entity types. In a schema, the relationship between entity types is related to object properties. We use "property" for "object property" if no ambiguity exists. Next, we introduce the definition and quantitative calculation method of entity type weight based on property.

Definition 2. *Weight of Entity type*

In a schema, the weight of entity type E is

$$weight(E) = \frac{|L_E|}{2 * |L|} \tag{4}$$

where $|L|$ is the total number of all properties between all entity types in the schema. L_E is defined as follows:

$$L_E = \{\mathrm{p} | E \in domain(p)\ or\ E \in range(p)\}$$

$|L_E\ |$ is the cardinality of L_E.

There are two exceptions to mention. Firstly, the entity type without any property is difficult to express in a schema, so we do not consider these isolated entity types. Secondly, in the denominator of weight definition $|L|$ needs to be multiplied by 2 because the property p is calculated twice in the definition L_E. This process ensures the normalization of weights, that is, the weight of each entity type is less than 1 and the weight sum of all entity types is 1, $\sum_{i=1}^{n} |L_{E_i}| = 2 * |L|$ (n is the total number of entity types in a schema).

In order to make this formula easier to understand, we use the entity types in schema A in Figure 2 as an example. We give the weight of each entity type based on property according to Formula (4). There are four entity types in schema A {*student, teacher, course, scholarship*} and four properties {*receive, take, teach, lecture*}. $|L| = 4$ because there are four properties in total in schema A. For entity type $student$, $student \in domain(take), domain(receive)\ and\ range(teach)$. Therefore, $L_{student}\ = \{take, receive, teach\}$, $|L_{student}| = 3$, $weight(student) = |L_{student}|/(2 * |L|) = 3/8$. Similarly, $L_{scholarship} = \{receive\}$, $weight(scholarship) = 1/8$. $L_{course} = \{take, lecture\}$, $weight(course) = 1/4$. $L_{teacher} = \{teach, lecture\}$, $weight(teacher) = 1/4$. The above is a simple example of calculating the weight of entity type with Formula (4).

5.2. Handling of Is-a Relationship

An is-a relationship exists between entity types and between properties. The relationship between entity types implies that subclasses inherit the property of their superclass. In computation, we transfer the property influencing the superclass to subclasses. At the same time, if all subclasses of the same superclass are associated with another entity type A, we think that the superclass will also be associated with entity type A due to the closed world assumption. We preprocess the schema before weight computation based on the above semantic analysis of the is-a relationship between entity types. As the is-a relationship corresponds to the semantics of superclass to subclass and subclass to superclass, we preprocess these two situations separately.

5.2.1. From Superclass to Subclass

In a schema, if the superclass has any property, then the subclass should inherit the same property, which does not exist explicitly in the schema. This should be considered when calculating the weight of entity types. The specific method is to traverse all the properties of the superclass and add them to each subclass entity type. Quantitative calculation is $|L_{subclassEntityType}| + 1$ and $|L| + 1$ in Formula (4). The weight of the subclass entity type will be larger.

5.2.2. From Subclass to Superclass

As shown in Figure 3, it is assumed that entity types $\{A_1, A_2 \ldots A_k\}$ are all subclasses of entity type A in a schema. If entity types $\{A_1, A_2 \ldots A_k\}$ have properties $p_i(i = 1, \ldots, k)$ pointing to entity type B, then we create a property p pointing from entity type A to entity type B. Here, p is only used to calculate weights, so the specific meaning of p is not considered. However, in theory, the common parent property of the original k edges can be extracted. In addition, we assume that these k properties are different, otherwise they should be refined into properties of super classes entity type A to entity type B before building a schema.

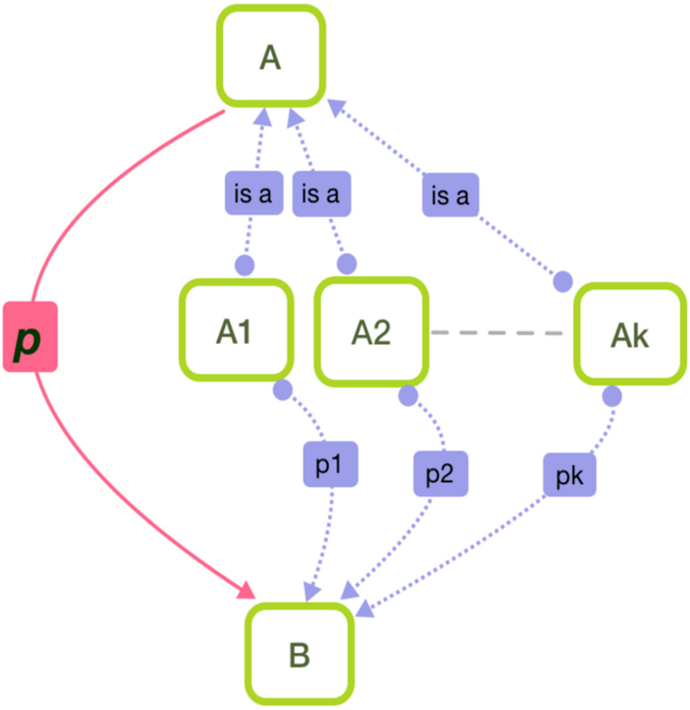

Figure 3. Convert subclass property to superclass.

On the semantic level, this operation is reasonable. If every subclass of entity type A has a relationship with entity type B, it means that entity type A and entity type B also have a certain relationship. When calculating the weight of an entity type A, the influence of entity type B should be considered together. In the computation, if each subclass of entity type A has a property pointing to entity type B, then $|L_A| + 1$ *and* $|\mathrm{L}| + 1$ in Formula (4). The weight of entity type A will be larger.

In the process of migrating a property from superclass to subclass, there are two special cases that need to be dealt with. As shown in Figures 4 and 5, suppose that in a schema, entity type A_1 is a subclass of entity type A, there is a property p_1 between entity type A and entity type B and entity type B_1 (only in Figure 5) is a subclass of entity type B.

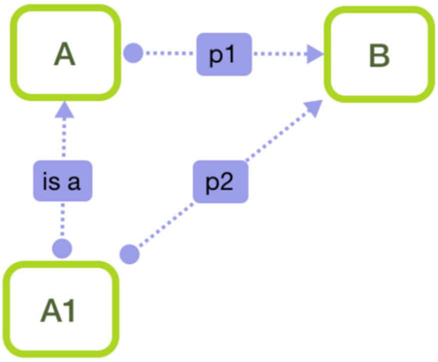

Figure 4. Special case 1 of is-a relation processing.

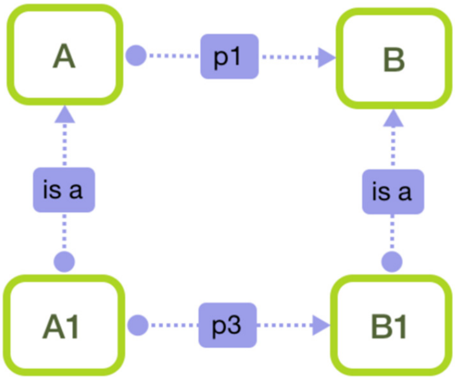

Figure 5. Special case 2 of is-a relation processing.

1. As is shown in Figure 4, if there is property p_2 between A_1 and B and p_2 is a subclass of p_1, there is no need to create a new property from A_1 to B. In other cases, we create a new property p from A_1 to B;
2. As is shown in Figure 5, if there is property p_3 between A_1 and B_1, and p_3 is a subclass of p_1, there is no need to create a new property p from A to B. In other cases, we create a new property from A to B.

After preprocessing, if the is-a relationship in the original schema is well handled, then the weight is calculated according to Definition 2.

5.2.3. Handling of Restriction Relationship

After dealing with the is-a relationship, next step is to deal with the restriction relationship in OWL. Protégé-OWL API makes a clear distinction between named class and anonymous class. Named classes are used to create instances, and anonymous classes are used to explain in detail the logical characteristics of named classes. We can classify all individuals with the same properties into an anonymous class, which is called a *Restriction*.

To give an example, *teacher*, an entity type, has a restriction "teaching people", so theoretically speaking, "teaching people" is a superclass of teacher. Our method does not consider the weight of an anonymous superclass, because its function is only to restrict subclasses. The properties between an anonymous superclass and another entity type are still inherited by subclasses according to the method in Section 5.2.1.

5.3. Computation of Coverage and Flexibility

After defining the weights, we can define the computation formulas of coverage and flexibility:

Definition 3. *Coverage and Flexibility.*

Given a schema X,

$$X = \{A_1 : k_1, \ldots, A_n : k_n, B_1 : k_{n+1}, \ldots, B_m : k_{n+m}\}$$

$A_i : k_i$ means that for entity type A_i in schema X, $k_i = weight(A_i)$, and given a schema Y,

$$\mathrm{Y} = \{A_1 : j_1, \ldots, A_n : j_n, C_1 : j_{n+1}, \ldots, C_l : j_{n+l}\}$$

The coverage and flexibility of schema X to schema Y are given by:

$$Cov(X, Y) = \textstyle\sum_{i=1}^{n} j_i \tag{5}$$

$$Flx(X, Y) = \textstyle\sum_{i=n+1}^{n+l} k_i \tag{6}$$

Some explanations of the formula:

1. The semantic equivalent entity type between schema X and schema Y is $\{A_1, \ldots, A_n\}$;
2. The positions of the two independent variables in (5) and (6) cannot be exchanged, that is, the independent variables do not have symmetry.

Theorem 2. *Formulas (5) and (6) conform to the semantics set of Formula (1) and (2), that is, the computation method proposed in Definition 3 is correct.*

Proof. $Cov(X, Y)$ is defined as

$$Cov(X, Y) = \frac{X \cap\ Y}{Y}$$

Combined with Definition 2, for each entity type E_i in schema Y, there is

$$weight(E_i) = \frac{|L_{E_i}|}{2 * |L_Y|}$$

where $|L_{E_i}|$ is the number of all properties with entity type E_i as the range or domain, and $|L_Y|$ is the number of all properties in schema Y. According to Definition 3, there is

$$j_i = weight(E_i) = \frac{|L_{E_i}|}{2 * |L_Y|}$$

thus

$$Cov(X, Y) = \sum_{i=1}^{n} j_i = \sum_{i=1}^{n} \frac{|L_{E_i}|}{2 * |L_Y|} = \frac{\sum_{i=1}^{n} |L_{E_i}|}{2 * |L_Y|}$$

$X \cap\ Y$ are the entity types in two schemas, namely entity type $A_1 - A_n$, and the number of properties of this part is $\sum_{i=1}^{n} |L_{E_i}|$. The total number of properties in schema Y is $|L_Y|$, as shown in Figure 2, each property links two entity types and both entity types count this property when calculating their weights. Therefore, each property is counted twice, so $2 * |L_Y|$ represents the number of times involved in the calculation.

Therefore, $\sum_{i=1}^{n} |L_{e_i}|/2 * |L_Y|$ represents the meaning of $X \cap\ Y/Y$, and the computation method in Definition 3 is correct. The correctness of $Flx(X, Y)$ can be proved in the same way. □

The application scenarios of these two formulas are widely used. When we need to fuse schemas, we need to measure their overlapping effect. In the past, it was mostly decided by domain experts whether the two schemas could be fused, which is not scientific and efficient. This kind of quantitative evaluation can assist domain experts to judge. Moreover, when the number of schemas that need to be evaluated is too large, it is unrealistic to rely only on domain experts. Then, automation or semi-automation can be designed according to the metrics above. It can increase efficiency, save labor costs and avoid instability of subjective factors. In a word, the computability of evaluation criteria is very important. The calculation method in Formulas (5) and (6) contributes to the evaluation of schema fusion and the evaluation process.

6. Experiment

6.1. Computation of Coverage and Flexibility

Our experiment is mainly compared with the method in [37]. For the neutrality of the experimental results, we select the data of a business practice project of the University of Trento in Italy as the data source. We compared with the original method in [37] which does not consider the weight of entity type in the calculation. In addition, under the premise of considering the weight, we carry out experiments to compare the results of whether to deal with the is-a relationship. The experimental results can verify the effectiveness of the method. The experiment is carried out on a desktop PC with a Intel i7-6700hq processor, 8GB DDR4 1333 Ram and SSD hard disk of 900GB.

There are three steps in this experiment. First, we extract entity type and properties from *query, datasets* and *existing universal schema* to form three schemas. For the convenience of description, these three schemas are referred to as **Query, Datasets and Schema, respectively**. The second step is to determine semantically equivalent entity types between the 3 schemas pairwise according to the definitions in Section 4. To simplify the problem, we assume that the semantic equivalence of entity types is symmetric and transitive across schemas. Therefore, we only need to calculate the semantic equivalence of Query-Schema and Datasets-Schema, respectively. Next, we explain the methods of extracting entity types and properties from different schemas.

Query is presented in natural language in the project. We manually extracted the entity types contained in Query with the help of the project document. The project document provides some lexical alignment, and because of this, most of the entity types can be judged as semantically equivalent on the label level. However, there are some exceptions. For example, entity type *Company* extracted from Query and entity type *Business Organization* from Schema are neither semantically similar nor synonymous in WordNet, but they have the same properties at the property level, so they are computed to be semantically equivalent according to the definition in Section 4. To facilitate the next step of our experiment, standardize all equivalent entity types between Query and Schema.

As for Datasets, it is a bit more complicated, because the equivalence of entity types contained in different tables of Datasets must be determined before the comparison with Schema. Therefore, individual-level judgement in Section 4 works. For example, entity type *Company Category* and entity type *Industry* from 2 different tables have the same individuals and these two entity types are semantically equivalent according to the definition in Section 4.

After entity type equivalence judgment and standardization, the numeric relationship of Schema, Query and Datasets in our experiment is found, as shown in Figure 6. The overlapping parts represent the equivalent entity types. For example, the number "9" in the blue part in Figure 6 indicates that nine equivalent entity types can be found in Query, Schema and Datasets, and the "8" in the yellow part indicates that there are 8 equivalent entity types in both Query and Schema.

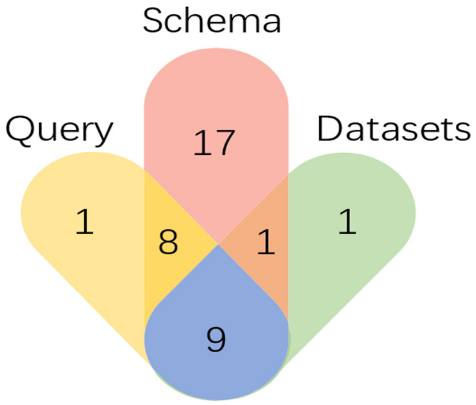

Figure 6. Equivalent entity types across schemas.

The third step of our experiment is to calculate the weights of all entity types in Datasets and Schema by the method proposed in Section 5. Limited by space, we cannot list all the results. Figure 7 shows the distribution of entity type weights. We divide these entity types into six important degrees according to a step size of 0.02 and use them as independent variables of the experiment.

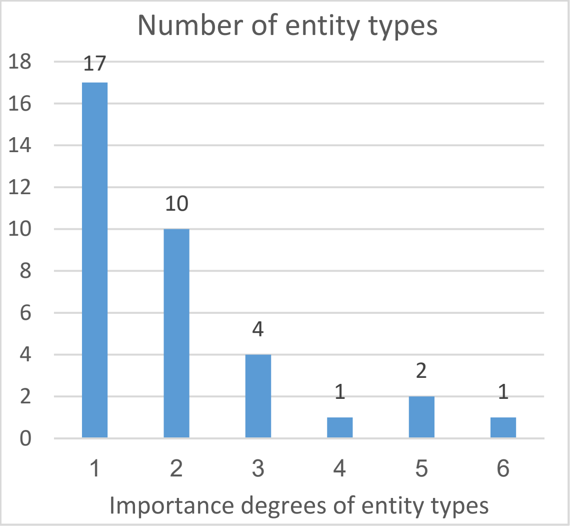

Figure 7. Number of entity types in different importance degrees.

It can be seen from Figure 7 that most entity types in Schema are of low importance. Only a few important entity types have high weights. Three computation methods were involved in our experiment:

1. The method which does not consider property-based weight computation (classical method in [37]);
2. The method which considers weight computation but does not consider is-a relationship handling in Section 5.2;
3. The method which both considers weight computation and is-a relationship handling proposed in Section 5.

They are abbreviated as Method 1, 2 and 3 below.

For each method, the coverage and flexibility of the schema to Query and Schema to Datasets are calculated. Note that $\boldsymbol{Cov(A,B)} = 1 - \boldsymbol{Flx(B,A)}$. Therefore, we can calculate $\boldsymbol{Cov(A,B)}$ and get the value of $\boldsymbol{Flx(B,A)}$. As there is no is-a relationship in Query or Datasets, the weights of entity types in Query and Datasets are not affected by preprocessing the is-a relationship or not. For the above reasons, there is no difference between Method 2 and Method 3 when calculating $\boldsymbol{Cov(Schema,Query)}$. Therefore, we only compare Method 1 and Method 2 in Section 6.2.1. During the experiment, we remove one entity type belonging to a certain degree of importance in Schema at a time, and the changes in coverage and flexibility are observed. The average value is taken when entity types belonging to the same degree of importance are removed.

6.2. Results and Analysis

6.2.1. Coverage

When calculating $Cov(Schema,Query)$, according to Formula (5), we need to calculate the weight of the entity types in Query. Then, the overlap degree of Schema over Query is calculated. The greater the total weight of the entity types that cover query, the higher the degree of overlap and the greater the result of $Cov(Schema,Query)$. The calculation method of $Cov(Schema,Datasets)$ is similar.

As shown in Figure 8, when no entity types are removed, the coverage value calculated by **Method 2** is higher than that calculated by **Method 1**, because the importance of entity types not included in Schema but in Query is very small. This also shows that **Method 2** can measure the coverage more accurately. With the importance of removed entity types increasing from 1 to 3, the coverage value of **Method 2** falls faster and faster, because the absence of entity types with high importance in Schema will greatly reduce the overlapping effect between Schema and Query, while the decline of **Method 1** has no obvious change, indicating that the computation results of **Method 2** can better reflect the change in overlaps.

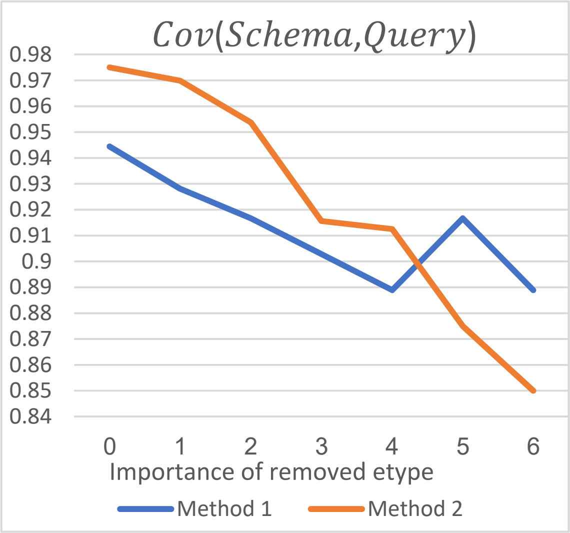

Figure 8. Coverage of Schema to Query.

In particular, when removing entity types with an importance degree of 4, the decreasing amplitude of **Method 2** becomes smaller, because there is only one entity type, *Patent*, with an importance degree of 4, and the weight of this entity type in Query is not high. Therefore, it presents abnormal points, as shown in Figure 8. When removing entity types with importance degrees of 5 and 6, the coverage value calculated by **Method 2** is lower than that calculated by **Method 1**, because if the most important entity types are not included in Schema, the overlap degree of Query to Schema will decrease greatly, while the calculated value of **Method 1** increases instead because it does not consider the difference in entity type weight.

As for coverage of Schema to Datasets, the initial situation is like Query-Schema, as shown in Figure 9. However, when removing entity type with an importance of 4, the calculated values of the two methods both rise, which is because the entity type *Patent* is not included in Datasets of this project, leading to an increase in the schema's overlap degree to Datasets after removing Patent from the schema. When removing entity types with the highest degrees of importance, the calculated value of **Method 2** drops significantly, because the weights of the most important entity types in Schema are also very high in Datasets. Removing these entity types will lead to a significant decrease in the overlap degree of Schema to Datasets, which is not reflected by the trend of **Method 1**.

To sum up, when calculating the coverage of Schema to Query and Schema to Datasets, the calculation results of **Method 2**, which considers weight, can reflect the difference in importance of entity types better than **Method 1**.

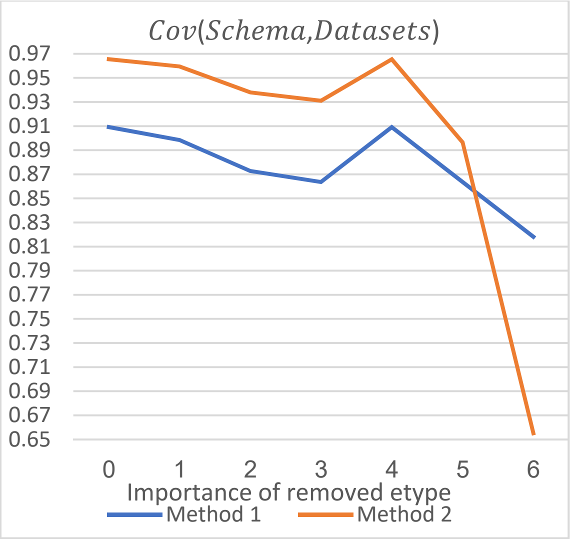

Figure 9. Coverage of Schema to Datasets.

6.2.2. Flexibility

After verifying the effectiveness of **Method 2** for computing coverage, we verify the effectiveness of **Methods 2** and **3** for computing flexibility. When calculating $Flx(Schema, Query)$ according to Formula (6), we need to calculate the weight of the entity types in Schema. Then, the redundancy degree of Schema after covering query can be calculated. The greater the weight of the entity types that do not cover Query, the higher the degree of redundancy and the greater the result of $Flx(Schema, Query)$. The calculation method of $Flx(Schema, Datasets)$ is similar.

As shown in Figure 10, among the three computation methods, Method 3 is most in line with this trend. The trend of Method 1 is irregular. Although Method 2 also has an increasing trend, the flexibility values are all greater than the values in Method 3. This is because Method 2 does not deal with the is-a relationship. The calculated weight values of entity types which have high importance degrees are less prominent, resulting in the larger redundancy. Additionally, it is also the reason why there is a drop at degree 6 in Method 2. For flexibility of Schema to Datasets in Figure 11, the situation is roughly the same as flexibility of Schema to Query.

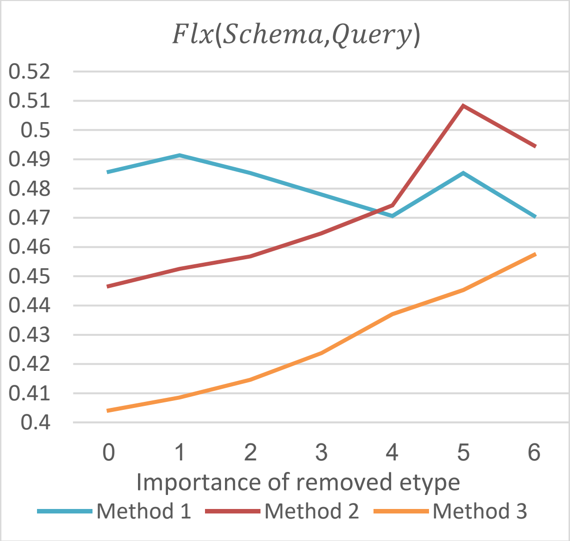

Figure 10. Flexibility of Schema to Query.

To sum up, when calculating the flexibility of Schema to query and Schema to Datasets, Method 3, which utilizes every property-based metric in Section 5, is better than Methods 1 and 2. By comparing the experimental results, the applicability and effectiveness of Methods 2 and 3 for computing flexibility are verified.

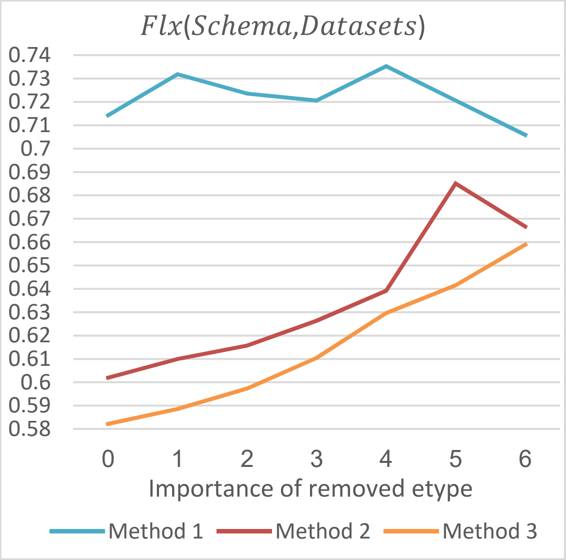

Figure 11. Flexibility of Schema to Datasets.

7. Conclusions and Future Work

This research studies the quantitative evaluation method of semantic equivalence of entity types and overlapping effect between schemas. On the one hand, we give a method to judge the semantic equivalence between entity types. In the past, most of the equivalence judgments were based on the label of entity type, and did not make effective use of the particularity of entity type in a schema. Our method considers the influence of properties and individuals on entity type, which makes the calculation method of semantic equivalence of entity type in a schema more scientific and reasonable. Our work can give some inspiration in the sense that it can be considered as judging the equivalence of entity type from different perspectives, rather than just from the single perspective of label. In future work, we will refine the calculation method and explore more clear and scientific quantitative means. On the other hand, we calculate the weight of entity types in a schema based on properties. We effectively consider the influence between entity types and quantify the influence. On this basis, we define the calculation methods of coverage and flexibility which contribute to ontology evaluation in the process of scheme fusion. It is vital to define the fine-grained weights of entity types, for example, to compute differences in the weights of object/data properties from type to type. In the future, the context of the entity type will be considered in the calculation of the weight.

Author Contributions: Conceptualization, Y.Z., B.W., R.Z. and F.G.; Data curation, Y.Z. and B.W.; Formal analysis, B.W.; Funding acquisition, L.H. and R.Z.; Investigation, Y.Z.; Methodology, Y.Z., R.Z., F.G., S.B. and S.D.; Project administration, Y.Z. and R.Z.; Resources, L.H. and R.Z.; Software, B.W.; Supervision, L.H. and R.Z.; Writing—original draft, Y.Z.; Writing—review and editing, L.H., B.W., D.Z. and R.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: https://github.com/UniTN-KDILab/Business-2018-19 (accessed on 12 March 2021).

Acknowledgments: National Key Research and Development Project (2018YFC2001302).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhao, L.; Ichise, R. Ontology Integration for Linked Data. *J. Data Semant.* **2014**, *3*, 237–254. [CrossRef]
2. Giunchiglia, F.; Fumagalli, M. Proceedings of the 2019 Joint Ontology Workshops (JOWO). In Proceedings of the WOMoCoE 2518 (CEUR-WS: 2019), Graz, Austria, 23–25 September 2019.
3. Giunchiglia, F.; Fumagalli, M. Teleologies: Objects, Actions and Functions. In Proceedings of the International Conference on Conceptual Modeling, ER 2017, Xi'an, China, 22–25 October 2017. [CrossRef]
4. Ngomo, J.G.N.; Lopes, G.R.; Campos, M.L.M.; Cavalcanti, M.C.R. An Approach for Improving DBpedia as a Research Data Hub. In Proceedings of the WebMedia 20: Brazillian Symposium on Multimedia and the Web, São Luís, Brazil, 30 November–4 December 2020.
5. Alexandrova, A.; Rapanotti, L. Requirements analysis gamification in legacy system replacement projects. *Requir. Eng.* **2020**, *25*, 131–151. [CrossRef]
6. Tomak, J.; Gorlatch, S. Measuring Performance of Fault Management in a Legacy System: An Alarm System Study. In *Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*; Springer: Cham, Switzerland, 2021.
7. Golchin, A.; Sinha, S.; West, R. Boomerang: Real-Time I/O Meets Legacy Systems. In Proceedings of the 2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), Sydney, Australia, 21–24 April 2020; IEEE: Piscataway, NJ, USA; pp. 390–402.
8. Lonsdale, D.; Embley, D.W.; Ding, Y.; Xu, L.; Hepp, M. Reusing ontologies and language components for ontology generation. *Data Knowl. Eng.* **2010**, *69*, 318–330. [CrossRef]
9. Euzenat, J.; Shvaiko, P. *Ontology Matching*; Springer: Berlin/Heidelberg, Germany, 2013. [CrossRef]
10. Vandenbussche, P.Y.; Atemezing, G.A.; Poveda-Villalón, M.; Vatant, B. Linked Open Vocabularies (LOV): A gateway to reusable semantic vocabularies on the Web. *Semant. Web* **2016**, *8*, 437–452. [CrossRef]
11. Whetzel, P.L.; Noy, N.F.; Shah, N.H.; Alexander, P.R.; Nyulas, C.; Tudorache, T.; Musen, M.A. BioPortal: Enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications. *Nucleic Acids Res.* **2011**, *39*, W541–W545. [CrossRef] [PubMed]
12. Selvaraj, S.; Choi, E. TKM Ontology Integration and Visualization. In Proceedings of the ICSIM 20: The 3rd International Conference on Software Engineering and Information Management, Sydney, Australia, 12–15 January 2020.
13. Fernández-López, M.; Villalon, M.; Suárez-Figueroa, M.; Gomez-Perez, A. Why are ontologies not reused across the same domain? *J. Web Semant.* **2019**, *57*. [CrossRef]
14. Horvat, M.; Dunđer, I.; Lugović, S. Ontological heterogeneity as an obstacle for knowledge integration in the Semantic Web. *Polytech. Des.* **2016**. [CrossRef]
15. Garijo, D. WIDOCO: A Wizard for Documenting Ontologies. In Proceedings of the International Semantic Web Conference, Vienna, Austria, 21–25 October 2017; pp. 94–102.
16. Halilaj, L.; Petersen, N.; Grangel-González, I.; Lange, C.; Auer, S.; Coskun, G.; Lohmann, S. *VoCol: An Integrated Environment to Support Version-Controlled Vocabulary Development*; Springer International Publishing: Cham, Switzerland, 2016; pp. 303–319. [CrossRef]
17. Hnatkowska, B.; Kozierkiewicz, A.; Pietranik, M. Semi-automatic definition of attribute semantics for the purpose of ontology integration. *IEEE Access* **2020**, *8*, 1. [CrossRef]
18. Euzenat, J.; Meilicke, C.; Stuckenschmidt, H.; Shvaiko, P.; Trojahn, C. *Ontology Alignment Evaluation Initiative: Six Years of Experience*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 158–192. [CrossRef]
19. Shvaiko, P.; Euzenat, J. Ontology Matching: State of the Art and Future Challenges. *IEEE Trans. Knowl. Data Eng.* **2013**, *25*, 158–176. [CrossRef]
20. Park, J.; Oh, S.; Ahn, J. Ontology selection ranking model for knowledge reuse. *Expert Syst. Appl.* **2011**, *38*, 5133–5144. [CrossRef]
21. Ruy, F.B.; Guizzardi, G.; Falbo, R.A.; Reginato, C.C.; Santos, V.A. From Reference Ontologies to Ontology Patterns and Back. *Data Knowl. Eng.* **2017**, *109*, 41–69. [CrossRef]
22. Nikolaos, T.; Franjo, C. Ontology evaluation for reuse in the domain of Process Systems Engineering. *Comput. Chem. Eng.* **2016**, *85*, 177–187. [CrossRef]
23. Zhao, M.; Du, Y.; Du, H.; Zhang, J.; Chen, Y. Research on Ontology Non-taxonomic Relations Extraction in Plant Domain Knowledge Graph Construction. *Trans. Chin. Soc. Agric. Mach.* **2016**, *47*. [CrossRef]
24. Ren, B.; Bu, F.; Hou, Z.; Fu, Y.; Liu, X. Analysis on the Construction of Knowledge Graph of Mass Events Based on Ontology. *J. Phys. Conf. Ser.* **2021**, *1802*, 042056. [CrossRef]
25. He, L.; Jiang, P. Manufacturing Knowledge Graph: A Connectivism to Answer Production Problems Query with Knowledge Reuse. *IEEE Access* **2019**, *7*, 101231–101244. [CrossRef]
26. Wu, T.; Wang, H.; Li, C.; Qi, G.; Shi, C. Knowledge graph construction from multiple online encyclopedias. *World Wide Web* **2019**, *23*, 1–28. [CrossRef]
27. Shen, Y.; Yuan, K.; Dai, J.; Tang, B.; Yang, M.; Lei, K. KGDDS: A System for Drug-Drug Similarity Measure in Therapeutic Substitution based on Knowledge Graph Curation. *J. Med. Syst.* **2019**, *43*, 92. [CrossRef] [PubMed]

28. Fan, L.Y.; Wang, A.M.; Xiao, T.Y. Evaluation criteria of ontology integration method & its application. *Comput. Integr. Manuf. Syst.* **2007**, *13*, 911.
29. Oh, S.; Yeom, H.Y. Evaluation Criteria Ontology Modularization Tools. In Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence & Intelligent Agent Technology, Washington, DC, USA, 22 August 2011.
30. Dastgerdi, A.F. Ontology Evaluation:Consideration of Criteria, Approaches and Layers. *Iran. J. Inf. Process. Manag.* **2012**, *27*, 533–559.
31. Hooi, Y.K.; Hassan, M.F.; Shariff, A.M. Ontology evaluation—A criteria selection framework. In Proceedings of the 2015 International Symposium on Mathematical Sciences and Computing Research (iSMSC), Perak, Malaysia, 19–20 May 2015; pp. 298–303.
32. Brack, A.; Hoppe, A.; Stocker, M.; Auer, S.; Ewerth, R. *Requirements Analysis for an Open Research Knowledge Graph*; Springer International Publishing: Cham, Switzerland, 2020; pp. 3–18. [CrossRef]
33. Guan, S.P.; Jin, X.L.; Jia, Y.T.; Wang, Y.Z.; Cheng, X.Q. Knowledge Reasoning Over Knowledge Graph: A Survey. *J. Softw.* **2018**, *29*, 2966–2994. [CrossRef]
34. Giunchiglia, F.; Fumagalli, M. Entity Type Recognition—Dealing with the Diversity of Knowledge. In Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, Rhodes, Greece, 12–18 September 2020.
35. Park, N.; Kan, A.; Dong, X.L.; Zhao, T.; Faloutsos, C. MultiImport: Inferring Node Importance in a Knowledge Graph from Multiple Input Signals. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, CA, USA, 6–10 July 2020. [CrossRef]
36. Sequeda, J.; Priyatna, F.; Villazón-Terrazas, B. Relational database to RDF mapping patterns. In Proceedings of the 3rd International Conference on Ontology Patterns-Volume 929, Boston, MA, USA, 12 November 2012; pp. 97–108.
37. Winiewski, D.; Potoniec, J.; Awrynowicz, A.; Keet, C.M. Analysis of Ontology Competency Questions and their formalisations in SPARQL-OWL. *J. Web Semant.* **2019**, *59*, 100534. [CrossRef]
38. Chatterjee, U.; Giunchiglia, F.; Madalli, D.P.; Maltese, V. Modeling Recipes for Online Search. In Proceedings of the ODBASE 2016, Rhodes, Greece, 25–26 October 2016.
39. Budanitsky, G.A. Semantic distance in WordNet: An experimental, application-oriented evaluation of five measures. In Proceedings of the Workshop on Wordnet & Other Lexical Resources, Pittsburgh, PA, USA, 3–4 June 2001.