Automatic Weight Generation and Class Predicate Stability in RDF Summary Graphs

Mehmet Aydar, Serkan Ayvaz, and Austin Melton

Kent State University, Department of Computer Science,
241 Math and Computer Science Building. Kent, OH 44240, USA
{maydar,sayvaz1,amelton}@kent.edu
http://www.kent.edu/cs

Abstract. *In this current study, we use graph localities and neighborhood similarity to enhance the summary graph generation approach for building a summary graph structure for intelligent exploration of semantic data. The key improvements to what we have previously proposed include the addition of a string similarity measure for the literal neighbors, development of a stability measure to evaluate the accuracy of class relations, the addition of auto-generated property weights, and the detection of noise properties.*

Keywords: Semantic Web, RDF, Graph Summarization, Automatic Property Weight

1 Introduction

In the recent years, there has been significant progress in publishing semantic data in the Web as an ever-growing number of organizations adopt Semantic Web technologies. The Linked Open Data Initiative [4] along with several other Semantic Web projects has promoted publishing various open datasets in RDF model by using a standard methodology, with the links between data items from different data sources on the Web. While this has made available thousands of general purpose datasets including DBpedia [2], FreeBase [5] and GeoNames [1], and many domain-specific data sources in the Resource Description Framework (RDF) data model, there is still a long way to go as Linked Open Data is still a small portion of the information available on the Web.

An RDF graph consists of a set of RDF triples. In Semantic Web, the size of RDF graphs can be very large, and processing an entire graph for each query can be costly in terms of time and resources. A summary graph consists of the type classes, members of the type classes, and the relations between the type classes with each type class representing a collection of RDF resources having the same type. The summary graph structure demonstrating the inferred class types and class relations can be beneficial for intelligent explorations of semantic data as it helps understand underlying structure and provides an intermediate index structure for semantic searches to avoid unnecessary traversals of entire RDF graphs.

In our previous work, we proposed an efficient algorithm for auto-generating a summary graph structure from an RDF dataset; our main goal was faster computations. In this current study, we focus on key improvements in summary graph generation process for potentially more accurate results.

1.1 Contribution and Outline

The main contributions of present study include the following:

- We auto-generate the importance weight of each property and each string word for each of the reference IRIs, and we apply the weights in the pairwise similarity calculation.
- We add a string similarity measure when two graph vertices are literal type.
- We generate the summary graph along with the classes and class relations with a stability measure for each class relation. And we propose that the stability measures can also be utilized in semantic search algorithms to generate more accurate results.

The rest of the paper is organized as follows. We briefly review the graph summarization problem and discuss in detail the key improvements to our existing solution. Then, we present the results of the evaluations. Finally, we review the related work and follow this with our conclusion and future work.

2 Augmenting Graph Summary Computations

The Linked Data consist of a collection of RDF statements that intrinsically represents a labeled, directed multi-graph with which the resources are expressed unambiguously. RDF statements describe resources in the form of triples, consisting of subject-predicate-object expressions that describe a resource, the type of a resource (type triple), or a relationship between two resources [9]. The subject in an RDF triple is either an Internationalized Resource Identifier (IRI) or a blank node, the predicate is an IRI, and the object is either an IRI, a literal or a blank node. The subjects and objects of triples in the RDF graph form RDF nodes.

Each RDF node that corresponds to a unique RDF entity is represented with a unique IRI, and the values such as strings, numbers and dates are represented by literal nodes. A literal node can consist of two or three elements: a lexical form, a datatype IRI and a language tag. The language tag in a literal node is included if and only if the datatype IRI of the literal node corresponds to rdf:langString [6]. A predicate in an RDF triple is also called a property of the RDF subject node. A predicate can be one of two types: a DatatypeProperty where the subject of the triple is an IRI and the object of the triple is a literal or an ObjectProperty where both the subject and object of the triple are IRIs. Each object of a subject node is called a neighbor of that subject node.

2.1 Summary Graph Generation

A summary graph of a data graph is the directed graph such that each node in the summary graph is a subset of the original graph nodes of the same type. Let $G = (V, L, E)$ be a data graph such that V is the finite set of vertices; L denotes the finite set of edge labels; and E is the set of edges of the form $l(u, v)$, with $u \in V$, $v \in V$ and $l \in L$. Note that an edge $l(u, v)$ represents the RDF triple (u, l, v). We define a summary graph as $G' = (V', L', E')$, such that V' contains equivalence classes of V. E' and L' are, respectively, the sets of edges and labels in the graph G'. As we will see, $L' \subset L$, and the elements of E' are defined by the elements in the equivalence classes in V' and the edges in E.

There exists several methods to obtain a summary graph: (1) A summary graph can be obtained from the dataset ontology, if the dataset is already tied to an ontology. (2) Another way to obtain the summary graph is to locate the type triples in the dataset and to organize the type classes and relations accordingly, if the data set is published using a standard vocabulary [11]. (3) Or the summary graph can be built automatically by inferring the class types based on the similarity of the RDF nodes. Our graph summarization approach is based on method 3.

In our previous study, we proposed an algorithm for building a summary graph structure based on pairwise similarity matrices of the graph entities[3]. An efficient graph node pair similarity metric was introduced utilizing the graph localities and neighborhood similarity within the Jaccard measure context [13] in conjunction with RoleSim similarity [15], without relying on the existence of a common vocabulary such as rdf:type or owl:sameas. The intuition is that the nodes that have similar predicates connected to similar neighbors tend themselves to be similar nodes; thus, they should be in the same class. The properties of the entities were treated as the dimensions of the entities when measuring the entity similarity. The direct similarities of the entities were taken into account along with the similarity of the neighbors with which they interact. Therefore, our summary graph generation algorithm initially calculates the similarity of the IRI node pairs that share at least one common edge label. Consequently the algorithm generates the distinct classes based on a given threshold such that the nodes u and v get put into the same class if their dissimilarity is less than the defined threshold: ϵ. More details of our summary graph generation approach can be found in [3].

In this current study, we enhance our core summary graph generation approach [3] by incorporating literal node similarities, applying auto-generated importance weights of the IRI node descriptors and developing a measure describing the degree of confidence of the summary graph class relations. In the following sections, we describe these key improvement points in detail.

2.2 Literal Node Similarity

As stated in [3], our graph summarization approach is based on calculating the similarity of entities by utilizing the predicates of the IRI nodes. Our premise is that similar nodes tend to have similar properties and interact with similar neighbor nodes, which are either IRIs or literals. It is challenging to infer the semantics of literal nodes. An effective literal node similarity metric is needed for calculating the similarity of pairs of IRI nodes when some of the neighbors of the IRI nodes are literal nodes. We think that incorporating literals when computing the similarity of pairs can be beneficial when identifying similar entities, particularly in datasets where the entities are commonly described using literals. Thus, we are taking the similarity of literal neighbor nodes into account when doing similarity calculations in present study.

While incorporating literals in the computation of the similarity of IRI node pairs, we are assuming that all the literals are in the same language, as the same literals may have totally different meanings in different languages. Thus there is only one value for the rdf:langString component of the literal nodes if present. In this work, we disregard the third component of rdf:langString if present, which means we work only with the lexical form and the data type URI component of a literal node. Both the lexical form and the data type URI component clearly impact the similarity of a pair of literal nodes. Thus, they indirectly impact the similarity of IRI nodes in the calculation of neighborhood similarity, and their impacts need to be weighted. Since calculating the similarity of literal nodes when data types are different is meaningless, we only give weight to the data type factor when the two data types are equal. For the lexical form components of the literal nodes, we use a string similarity technique based on common words within the two lexical forms along with their auto-generated importance weights. More details about the importance weights will be given in the following section.

2.3 Descriptor Importance and Automatic Detection of Noise Labels

An IRI node is described through its predicates and the collection of literal neighboring nodes in the lexical form. We call these the descriptors of the IRI nodes. The similarity of two IRI nodes is calculated from their descriptor similarities including the similarities of their neighbors. The accuracy of pairwise graph node similarity is often impacted by the weight of a property associated with the graph nodes when the nodes are object nodes or with the weight of a string literal word referenced by the graph nodes when the nodes are literal type. Each descriptor may have a different impact on an IRI node. Therefore, identifying appropriate metrics for generating weights for the IRI descriptors to be utilized in the pairwise graph nodes similarities is a formidable yet significant task.

In this paper, we investigate the factors that can impact the weight of a descriptor. We propose an approach for generating the importance weights of the IRI node descriptors automatically. Our approach is based on two premises: (1) the weight of a descriptor may differ for each IRI for which it is a descriptor and (2) the weight increases proportionally by the number of times a descriptor appears in the reference IRI, but it is offset by the frequency of the descriptor in the entire RDF dataset. It is a similar notion to the term frequency-inverse document frequency (tf-idf) [16, 20], a commonly used technique in information retrieval, indicating that some words may be important in some documents but not as important in other documents. More exactly, the importance of a word in a document increases by its frequency in the document but its importance decreases by its frequency in the corpus [18]. We apply the tf-idf concept to the properties and nodes in RDF graphs to compute the weight of properties. tf-idf is calculated as follows:

$$tf - idf(p, u, G) = tf(p, u) \times idf(p, G). \tag{1}$$

where the term frequency (tf) [16] represents the frequency of a proposition p with respect to a graph subject node u. More exactly, when $u \in V$ and $p \in L$, then

$$f(p, u) = |\{v \in V : p(u, v) \in E\}|. \tag{2}$$

Equivalently, $f(p, u)$ is the number of RDF triples with subject u and property p.

To define $tf(p, u)$, it is helpful to have a notation for the set of all properties with subject u. Thus, for $u \in V$, $L(u) = \{q \in L : \exists v \in V \text{ with } q(u, v) \in E\}$. Then

$$tf(p, u) = \frac{f(p, u)}{\sum_{q \in L(u)} f(q, u)}. \tag{3}$$

The inverse document frequency (idf) [20] represents the frequency of a property usage across all graph nodes, and it is defined as

$$idf(p, G) = \ln \frac{|V|}{|\{u \in V : p \in L(u)\}|}. \tag{4}$$

We apply a similar approach to calculate the weight of word importance in literal nodes, which can consist of a set of words. A string literal is a range for a DatatypeProperty. We assert that the weight of word importance depends upon the source subject node, the frequency of the word within the triple collection for each subject node, and the frequency of the word within the entire data set.

We calculate the property importance and assign weights depending on the degree of distinctiveness of a property describing an entity. With property distinctiveness, we mean the uniqueness of a property in describing the key characteristics of an entity type. For instance, if a property is specific to an entity type, it is a distinguishing character of the type from other types. When a property exists in all entity types, its quality of being distinctive is low. The noise labels tend to be common for a majority of entities if not for all entities. By increasing importance weights of properties with a higher degree of distinctiveness, we reduce the importance of noise labels automatically. As a result, the noise labels have significantly less impact on the overall similarity measures.

2.4 Class Relation Stability Metric

The summary graph from an RDF dataset is built automatically, and the constructed summary graph is also represented in RDF in our approach. The IRI nodes that have similarity higher than a defined threshold are considered to be of the same type, and they are categorized in the same class in the summary graph. A class relation between a class $c1$ and a class $c2$ is generated as a predicate and represented as $l(c1, c2)$ when there is at least one relation $l(u, v)$ such that u and v are IRIs in the dataset, $G = (V, E, L)$, and $u \in c1$ and $v \in c2$ with both $c1$ and $c2$ being type classes in the summary graph, $G' = (V', E', L')$. Then we have $l \in L'$ and $l(c1, c2) \in E'$. However, automatically generated summary graphs can be error prone. Therefore, a metric to measure the degree of confidence of a relation between classes in the summary graph would be beneficial. We call this metric Class Predicate Stability (CPS). The CPS is similar to the stability concept introduced by Paige and Tarjan [17].

For a triple $(c1, p, c2)$ in the summary graph G' with $c1$ and $c2$ being type class IRI nodes and p being a predicate between them, the CPS metric is calculated as the number of the IRI nodes u in class $c1$ having a triple of the form (u, p, v) with $u \in c1$ and $v \in c2$ divided by the total number of the IRI nodes in $c1$ in the summary graph. $CPS(c1, p, c2)$ is formulated as

$$CPS(c1, p, c2) = \frac{|(u, p, v) : u \in c1, v \in c2\}|}{|c1|} \tag{5}$$

where $|c1|$ is the number of IRI nodes in the class $c1$. Note that $|c1| > 0$. We define full CPS as follows: for two classes in the summary graph either all the IRI nodes from $c1$ are connected with a predicate p to at least one IRI node in $c2$ or none of the IRI nodes in $c1$ are connected with the predicate p to an IRI node in $c2$.

The CPS value for a triple $(c1, p, c2)$ in the summary graph indicates how strongly connected and how coarsely partitioned the type classes $c1$ and $c2$ are with the predicate p. Thus, the average of all the CPS values in the summary graph is a measure of accuracy for the generated summary graph. $CPS(G')$ is formulated as

$$CPS(G') = \frac{\sum_{i=1}^{|E'|} CPS(c1^i, p^i, c2^i)}{|E'|} \tag{6}$$

where $G' = (V', E', L')$ is the summary graph and $p^i(c1^i, c2^i) \in E'$, and thus $|E'| > 0$.

Another advantage of calculating the CPS metric is that it can further be utilized in semantic search algorithms. In traditional semantic search algorithms, the relations between two different type classes are assumed to be tightly coupled [21]. In real situations this assumption may not always be true, especially if the summary graph is auto-generated as in our study. We propose that the CPS metric can be used as an impact factor between two type classes and utilized in the semantic search graph traversal for more accurate results.

3 Evaluations

In the evaluations, we assessed the effectiveness of the proposed improvements on three datasets: a subset of DBpedia [2]; a subset of SemanticDB [10], a Semantic Web content repository for Clinical Research and Quality Reporting; and a subset of Lehigh University Benchmark (LUBM) [12], a benchmark for OWL knowledge base systems.

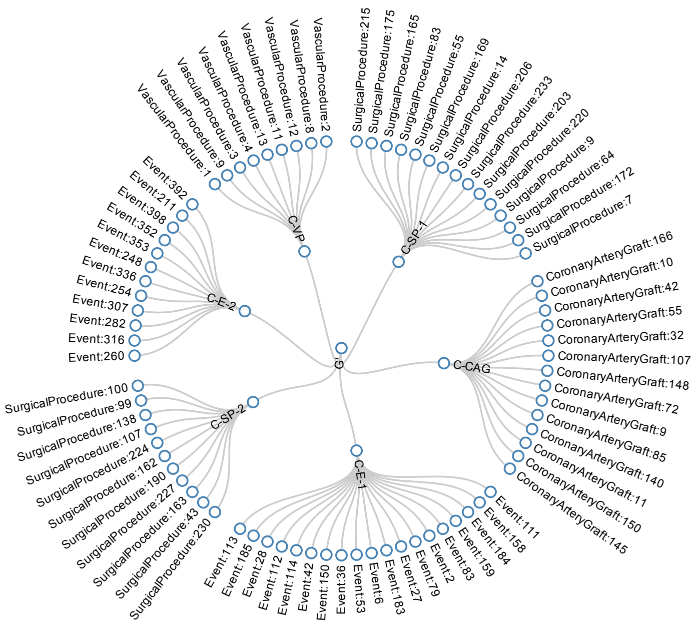

Fig. 1. A figure consisting of different types of entities and elements belonging to the class types.

We ran the datasets for summary graph generation with the core summary graph generation algorithm that was proposed in our previous work [3] and with the improvements suggested in this work. The improvements include taking the literal neighbor similarity and the dynamic property weight assignment into account in type generation. The goal of our evaluations was to investigate the impact of the improvements in real world datasets.

The reason for selecting these three datasets was that they represent different aspects of real world semantic data. Thus, we tested the applicability of our approach in different types of datasets. SemanticDB is a domain specific semantic data repository in Healthcare. It provides structured type information for the entities that we utilized as the ground truth for automatic verification of the accuracy in the evaluations. Lehigh University Benchmark (LUBM) is a struc-

Table 1. A Sample of RDF Triples from Each Dataset

Dataset	Subject	Predicate	Object
SemanticDB	SurgeryProcedure:236	hasCardiacValveAnatomyPathologyData	CardiacValveAnatomyPathologyData:70
SemanticDB	SurgeryProcedure:236	hasCardiacValveRepairProcedureData	CardiacValveRepairProcedureData:16
SemanticDB	SurgeryProcedure:236	SurgeryProcedureClass	”cardiac valve”
SemanticDB	SurgeryProcedure:236	CardiacValveEtiology	”other”
SemanticDB	SurgeryProcedure:236	CardiacValveEtiology	Event:184
SemanticDB	SurgeryProcedure:236	belongsToEvent	Event:184
SemanticDB	SurgeryProcedure:236	SurgeryProcedureDescription	”pulmonary valve repair”
SemanticDB	SurgeryProcedure:236	CardiacValveStatusologyData	”native”
SemanticDB	SurgeryProcedure:104	hasCardiacValveAnatomyPathologyData	CardiacValveAnatomyPathologyData:35
SemanticDB	SurgeryProcedure:104	SurgeryProcedureClass	”cardiac valve”
SemanticDB	SurgeryProcedure:104	CardiacValveEtiology	”rheumatic”
SemanticDB	SurgeryProcedure:104	belongsToEvent	Event:81
SemanticDB	SurgeryProcedure:104	SurgeryProcedureDescription	”mitral valve repair”
SemanticDB	SurgeryProcedure:104	CardiacValveStatus	”native”
LUBM	Student49	telephone	”xxx-xxx-xxxx”
LUBM	Student49	memberOf	http://www.Department3.University0.edu
LUBM	Student49	takesCourse	Course32
LUBM	Student49	name	”UndergraduateStudent49”
LUBM	Student49	emailAddress	”Student49@Department3.University0.edu”
LUBM	Student49	type	UndergraduateStudent
LUBM	Student10	telephone	”xxx-xxx-xxxx”
LUBM	Student10	memberOf	http://www.Department3.University0.edu
LUBM	Student10	takesCourse	Course20
LUBM	Student10	name	”UndergraduateStudent10”
LUBM	Student10	emailAddress	”Student10@Department3.University0.edu”
LUBM	Student10	type	UndergraduateStudent
DBPedia	Allen_Ginsberg	wikiPageUsesTemplate	Template:Infobox_writer
DBPedia	Allen_Ginsberg	influenced	John_Lennon
DBPedia	Allen_Ginsberg	occupation	”Writer, poet”@en
DBPedia	Allen_Ginsberg	influences	Fyodor_Dostoyevsky
DBPedia	Allen_Ginsberg	deathPlace	”New York City, United States”@en
DBPedia	Allen_Ginsberg	deathDate	”1997-04-05”
DBPedia	Allen_Ginsberg	birthPlace	”Newark, New Jersey, United States”@en
DBPedia	Allen_Ginsberg	birthDate	”1926-06-03”
DBPedia	Allen_Ginsberg	deathPlace	”New York City, United States”@en
DBPedia	Albert_Camus	wikiPageUsesTemplate	Template:Infobox_philosopher
DBPedia	Albert_Camus	influenced	Orhan_Pamuk
DBPedia	Albert_Camus	influences	Friedrich_Nietzsche
DBPedia	Albert_Camus	schoolTradition	Absurdism
DBPedia	Albert_Camus	deathPlace	”Villeblevin, Yonne, Burgundy, France”@en
DBPedia	Albert_Camus	deathDate	”1960-01-04”
DBPedia	Albert_Camus	birthPlace	”Drean, El Taref, Algeria”@en
DBPedia	Albert_Camus	birthDate	”1913-11-07”

tured and well-known benchmark dataset, which has type information available. However, the entities can have multiple types. Unlike SemanticDB, LUBM data has hierarchical types. For instance, an entity can have both types: Student type and Graduate Student type. Therefore, we performed a manual verification process for the ground truth to ensure the accuracy of evaluations. Lastly, DBPedia is a commonly used general purpose dataset, which is a central source in the Linked Open Data Cloud [4]. Type information is not always present for entities in DBPedia. Moreover, some entities have several types, including hierarchical types, which makes it problematic for automatic verification of accuracy results. Therefore, we manually verified the accuracy of the ground truth in the evaluations. Table 1 demonstrates a sample of RDF triples from each dataset in the evaluations.

Table 2. An Excerpt from Dynamically Assigned Weights of Descriptors

Dataset	Node_Pair	Descriptor_Type	Descriptor	Weight
LUBM	(Student49,Student10)	Property	memberOf	14.7%
LUBM	(Student49,Student10)	Property	takesCourse	44.1%
LUBM	(Student49,Student10)	Property	emailAddress	14.0%
LUBM	(Student49,Student10)	Property	type	5.7%
LUBM	(Student49,Student10)	Property	name	7.5%
LUBM	(Student49,Student10)	Property	telephone	14.0%
SemanticDB	(Procedure:236,Procedure:104)	Literal	"cardiac"	13.6%
SemanticDB	(Procedure:236,Procedure:104)	Literal	"native"	15.2%
SemanticDB	(Procedure:236,Procedure:104)	Literal	"other"	14.3%
SemanticDB	(Procedure:236,Procedure:104)	Literal	"pulmonary"	22.8%
SemanticDB	(Procedure:236,Procedure:104)	Literal	"repair"	17.2%
SemanticDB	(Procedure:236,Procedure:104)	Literal	"valve"	16.9%
DBPedia	(Allen_Ginsberg,Albert_Camus)	Property	wikiPageUsesTemplate	2.2%
DBPedia	(Allen_Ginsberg,Albert_Camus)	Property	influences	58.3%
DBPedia	(Allen_Ginsberg,Albert_Camus)	Property	deathDate	2.2%
DBPedia	(Allen_Ginsberg,Albert_Camus)	Property	birthDate	2.4%
DBPedia	(Allen_Ginsberg,Albert_Camus)	Property	birthPlace	2.1%
DBPedia	(Allen_Ginsberg,Albert_Camus)	Property	deathPlace	2.1%
DBPedia	(Allen_Ginsberg,Albert_Camus)	Property	influenced	30.7%

We also evaluated the performance of dynamic assignment of descriptor weights. Table 2 shows a sample of dynamically assigned descriptor weights from each dataset. As expected, the algorithm assigned higher weights to the properties with a higher degree of distinctiveness describing the resource type. For instance in LUBM dataset, takesCourse property is more descriptive of the Student type than the name property, which is a common property for all class types in the dataset. Thus, takesCourse was assigned a weight of 44.1% as compared to the weight of 7.5% for name.

We observed that higher class dissimilarity threshold results in more coarse classes, whereas the classes become more granular when the threshold is chosen smaller. The beta factor and the class dissimilarity threshold can be tuned differently in various datasets. Their optimum values depend on the characteristics of the datasets. For each dataset, we kept the beta factor and the class dissimilarity threshold the same in both evaluations; core algorithm and algorithm with the improvements. We found that the class dissimilarity threshold ranging between 0.3 to 0.6 in combination of the beta factor of 0.15 appeared to work well in our evaluations.

It is clear that the evaluation with the suggested improvements generates a summary graph with better accuracy and stability, as demonstrated in Table 3. We noticed that the literal similarity improves the class generation accuracy in datasets that have frequently used terminology as in the case of SemanticDB and LUBM. On the other hand, it may have an adverse effect in datasets with lengthy and diverse vocabulary of literals as in the example of DBPedia.

Table 3. Evaluation Results

Dataset	Algorithm	#Triples	Class_Threshold	#Iterations	Stability	Accuracy
SemanticDB	Core	6,450	0.5	4	61.0%	87.3%
SemanticDB	With Improvements	6,450	0.5	4	68.2%	94.1%
LUBM	Core	6,484	0.3	3	67.8%	90.7%
LUBM	With Improvements	6,484	0.3	3	78.4%	98.6%
DBPedia	Core	10,000	0.6	3	82.4%	92.8%
DBPedia	With Improvements	10,000	0.6	3	89.1%	92.2%

Figure 1 illustrates a small sample set of entities in the RDF graph from SemanticDB and their corresponding class types in the summary graph. As demonstrated in Figure 1, the classes C-E1 and C-E2 represent the entities that are patient event types. They are classified in two different classes because when compared with the original dataset we observed that the entities in C-E1 are more specifically patient surgery-related event types while the entities in C-E2 are patient-encounter related event types. Also, the classes E-SP1 and E-SP2 are surgical procedure types. More specifically, the entities in E-SP1 are coronary artery and vascular procedure-related procedures while the entities in E-SP2 are cardiac valve related-procedures. The classes C-VP and C-CAG represent the entities that are related to vascular procedures and coronary artery grafts, respectively. We implemented a basic algorithm to name the classes based on the class member IRIs. The classes C-E1, C-E2, C-SP1, C-SP2, C-VP and C-CAG are named as C-Event-1, C-Event-2, C-SurgicalProcedure-1, C-SurgicalProcedure-2, C-VascularProcedure and C-CoronaryArteryGraft, respectively.

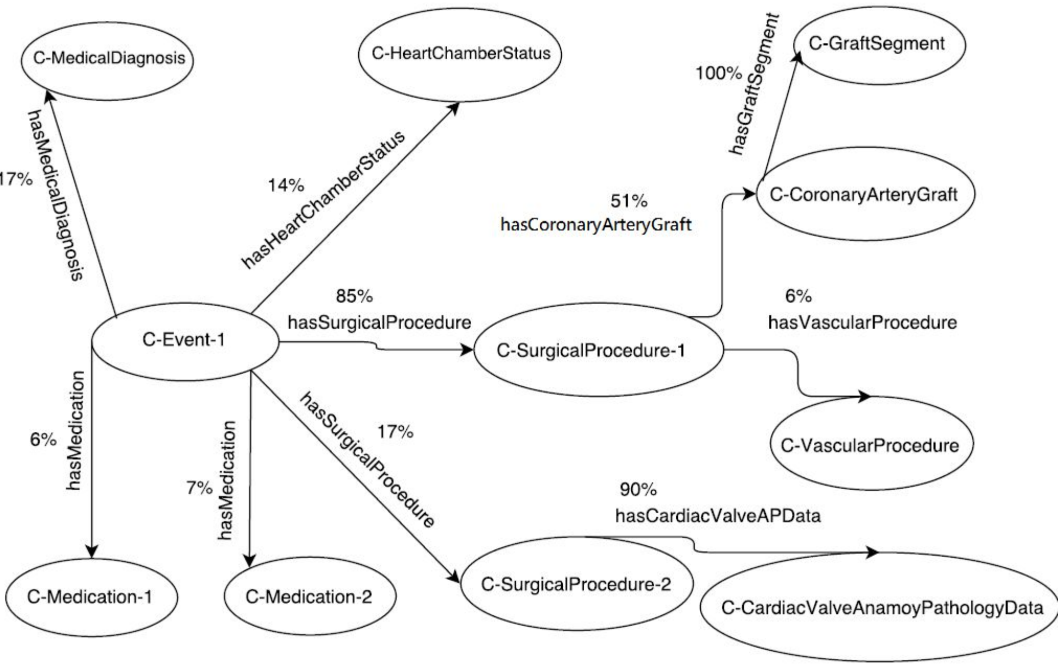

Fig. 2. An excerpt from the generated summary graph.

The summary graph is generated along with the classes and the class relations with a stability measure for each relation. Figure 2 shows an excerpt from the summary graph representing the class relations from SemanticDB dataset. The percentage values beside the predicates are the stability (CPS) measure.

4 Related Work

Many methods have been proposed for calculating the graph node similarities in an RDF data set, including our previous study [3]. While most of the similarity calculations do not take the property weights into account, for example, [3] and [14], there are some studies that try to calculate the property weights and apply them in similarity calculations.

H-Match[7] tried to detect the property weights using the distinct value based weight generation, assigning higher weight to a property that references more distinct values. However, a training set of instances may not always be available.

In [8] the authors suggest that properties with a maximum or an exact cardinality of 1 have a higher impact in instance matching, thus having a higher property weight. This assumption does not work well in instance type discovery. For instance, in a university related dataset a more specific property hasPresident should have more impact in type discovery than a more general property hasName, even though both of the properties have the cardinality of one. In this case, the assumption would misleadingly assign the same weight to both of the properties.

On the other hand, [19] considers the ratio of the number of distinct values of a property to the number of instances in a dataset in addition to the number of distinct values referenced by the property. However, they primarily focus on instance matching, where property weights naturally yield precedence to properties that make the instances more unique. Unlike the instance matching approach, we emphasize the properties that would help describe the entity types more distinctively.

In [17] the authors defined the stability concept to be used in a coarsest partitioning problem. They utilized the stability concept on directed graphs. In our work, we leverage the stability concept to be used in a summary graph which is in RDF model.

5 Conclusion

In this paper, we described enhancements to our pairwise graph node similarity calculation with the addition of the property and string word importance weights. We introduced the Class Predicate Stability metric, which allows evaluation of the degree of confidence of each class predicate in the summary graph. We experimented with the enhanced method applied in our previous core summary graph generation technique. The results show that our enhanced method can yield more accurate results over the pure summary graph generation technique. Future work will focus on improvement of the scalability of the proposed method. Furthermore, our plan is to investigate obtaining the optimum value of the class dissimilarity threshold automatically and improving the class generation algorithm to discover the hierarchy of the class types.

Acknowledgments The authors would like to thank the Kent State University Semantic Web Research Group (SWRG) members for their helpful feedback.

References

1. GeoNames, June 2015. http://www.geonames.org/.
2. Sren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. *Dbpedia: A nucleus for a web of open data.* Springer, 2007.

3. Serkan Ayvaz, Mehmet Aydar, and Austin C Melton. Building summary graphs of rdf data in semantic web. In *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th International*. IEEE, 2015.
4. Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data-the story so far. *International journal on semantic web and information systems*, 5(3):1–22, 2009.
5. Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM, 2008.
6. Dan Brickley and R. V. Guha. RDF Schema 1.1. W3c Recommendation, February 2014.
7. Silvana Castano, Alfio Ferrara, Stefano Montanelli, and C Quix. H-match: an algorithm for dynamically matching ontologies in peer-based systems. In *SWDB*, pages 231–250. Citeseer, 2003.
8. Keith Cortis, Simon Scerri, Ismael Rivera, and Siegfried Handschuh. Discovering semantic equivalence of people behind online profiles. In *In Proceedings of the Resource Discovery (RED) Workshop, ser. ESWC*, 2012.
9. Richard Cyganiak, David Wood, and Markus Lanthaler. RDF 1.1 Concepts and Abstract Syntax. W3c Recommendation, February 2014.
10. Christopher D Pierce, David Booth, Chimezie Ogbuji, Chris Deaton, Eugene Blackstone, and Doug Lenat. Semanticdb: A semantic web infrastructure for clinical research and quality reporting. *Current Bioinformatics*, 7(3):267–277, 2012.
11. Songyun Duan, Anastasios Kementsietsidis, Kavitha Srinivas, and Octavian Udrea. Apples and oranges: a comparison of rdf benchmarks and real rdf datasets. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 145–156. ACM, 2011.
12. Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. Lubm: A benchmark for owl knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2):158–182, 2005.
13. Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.
14. Glen Jeh and Jennifer Widom. SimRank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.
15. Ruoming Jin, Victor E Lee, and Hui Hong. Axiomatic ranking of network role similarity. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 922–930. ACM, 2011.
16. Hans Peter Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317, 1957.
17. Robert Paige and Robert E Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.
18. Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.
19. Md Hanif Seddiqui, Rudra Pratap Deb Nath, and Masaki Aono. An efficient metric of automatic weight generation for properties in instance matching technique. *International Journal of Web & Semantic Technology*, 6(1):1, 2015.
20. Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

21. Thanh Tran, Haofen Wang, Sebastian Rudolph, and Philipp Cimiano. Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 405–416. IEEE, 2009.