An Entropy-Based Class Assignment Detection Approach for RDF Data

Molood Barati[1], Quan Bai[1] and Qing Liu[2]

[1] Auckland University of Technology, New Zealand,
[2] Data61, CSIRO, Australia
mbarati@aut.ac.nz; quan.bai@aut.ac.nz; q.liu@csiro.au

Abstract. The RDF-style Knowledge Bases usually contain a certain level of noises known as Semantic Web data quality issues. This paper has introduced a new Semantic Web data quality issue called Incorrect Class Assignment problem that shows the incorrect assignment between instances in the instance-level and corresponding classes in an ontology. We have proposed an approach called CAD (Class Assignment Detector) to find the correctness and incorrectness of relationships between instances and classes by analyzing features of classes in an ontology. Initial experiments conducted on a dataset demonstrate the effectiveness of CAD.

Keywords: Semantic Web data quality issue, Ontology refinement, Incorrect assignment, Knowledge Discovery.

1 Introduction

Recently, researchers are tackling with SW data quality issues for refining and re-engineering RDF-style Knowledge Bases (KBs). In this paper, we have identified a new SW data quality issue called Incorrect Class Assignment (ICA) problem that shows incorrect assignment between instances in the instance-level and corresponding classes in ontology. The DBpedia ontology defines a Royal class with two subclasses of BritishRoyalty and PolishKing for all royalties. There exist some instances that are incorrectly assigned to unrelated classes in ontology. For example, John I Albert, king of Poland, has been assigned to BritishRoyalty class instead of PolishKing class defined in the DBpedia ontology. This example can be described as an incorrect assignment issue between instance-level data and corresponding classes in ontology. The research problem used in this paper has been modelled in Fig. 1. We name all instances which have been correctly assigned to corresponding classes in ontology as CA. The data quality issue can be defined as the Incorrect Class Assignment problem (ICA) where at least one instance has been incorrectly assigned to class A instead of class B. Under this motivation, we proposed an approach called Class Assignment Detector (CAD) to deal with ICA problem. The main contribution of this paper is threefold including (I) identifying and defining a new SW data quality issue called Incorrect Class Assignment problem (ICA), (II) proposing CAD approach to detect the correctness and incorrectness of relationships between instances and classes in ontology by analyzing features of classes, and (III) conducting initial experiments over DBpedia dataset 3.8 to show the effectiveness of CAD.

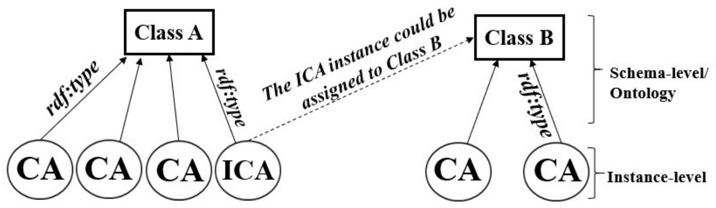

Fig. 1. Modeling the ICA problem.

This paper is structured as follows. Section 2 reviews related work on SW data quality issues. Sections 3 describes the CAD approach. Section 4 reveals the experimental results. Section 5 explains the conclusion and future work.

2 Related work

There exist various forms of SW data quality issues such as missing type prediction, incorrect or incomplete statements, invalid links to external resources, missing link prediction, etc. Studies based on first issue aim to predict missing *rdf:type* relation in the KBs [1][2]. Second issue refers to the incorrect or incomplete statements of SW data [3][4]. Consider an RDF triple (Rodrigo_Salinas, birthPlace, Puebla_F.C.) shared by DBpedia. The DBpedia provides an invalid object value that is the name of a stadium for Rodrigo Salinas instead of sharing a city or a country name. Third common issue refers to the faulty and invalid links to external RDF-style KBs [5]. Predicting Missing links (i.e. predicates) is another frequent issue in the KBs. Consider Barack Obama as a subject and Honolulu as an object of an RDF triple. Here the question is that how to learn birthPlace relation by mining existing RDF data. A Path ranking Algorithm proposed by [6][7] focused on this issue. To the best of our knowledge, the ICA problem has not been explicitly addressed by the most of existing work.

3 The architecture of Class Assignment Detector

The CAD architecture contains two main modules including (1) Class features extraction, and (2) Instance-Class relationship analysis. In Module 1, the CAD extracts features of classes in ontology. The output of Module 1 has been used in Module 2 to assess the correctness and incorrectness of relationships between instances and classes in ontology.

3.1 Module 1: Class features extraction

Generally, a class is a category of things having some common features that make those things distinct from others. To detect the features of classes in our scenario, the initial step is to analyze instance-level data to extract common features among RDF triples. In the following, we first explain the idea behind mining common features from RDF triples. Then, we describe how mining common features of instance-level data leads CAD to extract features of classes.

Identifying common features from RDF triples. The assertion of an RDF triple (i.e., subject, predicate, object) shows a meaningful relationship between a subject and an object provided by a predicate. Considers RDF triples (John I Albert, deathPlace, Poland) and (Casimir III, deathplace, Poland). The subjects of these RDF triples, i.e., John I Albert and Casimir III, have a common feature, i.e., (deathPlace Poland). To extract this behavior from RDF triples, we have defined the concepts of Group Feature and Common Feature as follows.

Definition 1 (*Group Feature*). Given RDF triples, a Group Feature gf_i is a 2-tuple, i.e., gf_i=(g_i, f_i). g_i is a Group of subjects or objects, i.e., $\{s_1, s_2, ..., s_n\}$ or $\{o_1, o_2, ..., o_n\}$. f_i is a Feature shared by g_i. Corresponding with the content in g_i, f_i contains a combination of predicate-object or predicate-subject, i.e., (p, o) or (p, s).

Definition 2 (*Common Feature*). Given a Group Feature gf_i=(g_i, f_i), Feature f_i is a Common Feature cf_i for Group g_i, if the number of instances in the g_i is greater than or equal to the Minimum Instance Number (*MinIN*).

Detecting features of classes. RDF-style KBs suffer from ICA problem since publishing SW data is manually maintained by contributors. To this end, CAD takes advantage from information theory [8] to analyze the level of uncertainty from this situation. As explained, a Common Feature shows a common behavior of instances (subjects or objects) in a Group. The information gain allows us to measure which common features are more certain to be used as features of classes in ontology. The following first introduces a measure based on entropy that calculates the uncertainty associated with the whole random space. In the SW, we have defined the entropy of a random space as follows.

Definition 3 (*Random Space Entropy*). Given an ontology, a Random Space S is a space built up from instances of different classes in ontology. The entropy of S can be calculated by Equation 1:

$$Entropy(S) = -\sum_{i=1}^{N} p(c_i) \log_2 p(c_i) \tag{1}$$

where N is the total number of classes in the ontology and $p(c_i) = \frac{|Ins_{c_i}|}{|INS|}$ is the probability of Class c_i in the Random Space S. $|Ins_{c_i}|$ is the total number of instances of Class c_i. $|INS|$ is the total number of instances in the Random Space S.

In the information theory, more information can be obtained by a random space with lower entropy, and vice versa. In our scenario, a Common Feature can be shared by instances of different classes in ontology. Therefore, the information gained from a Common Feature depends on the types of instances in its Group. Few types can cause lower entropy and consequently more information gained from the Common Feature. By relying on the fact, we have defined the concepts of Common Feature Information and Common Feature Space as follows.

Definition 4 (*Common Feature Information*). Given a Random Space S and a Common Feature cf_i, the information gained from cf_i is a normalized value measured by Equation 2:

$$NormGain(S, cf_i) = \frac{Entropy(S) - \frac{|Ins_{cf_i}|}{|INS|} Entropy(S_{cf_i})}{Entropy(S)} \tag{2}$$

where *Entropy(S)*$\neq 0$ and $0 \leq NormGain(S, cf_i) \leq 1$.

Definition 5 (*Common Feature Space*). Given a Common Feature cf_i, RDF triples and its corresponding ontology, a Common Feature Space S_{cf_i} is a space built up from instances that share the Common Feature cf_i. The Entropy of Common Feature Space S_{cf_i} can be computed by Equation 3:

$$Entropy(S_{cf_i}) = -\sum_{i=1}^{n} p(c_{i.cf_i}) \log_2 p(c_{i.cf_i}) \tag{3}$$

where n is the total number of classes in the Common Feature Space S_{cf_i} and $p(c_{i.cf_i}) = \frac{|Ins_{c_i.cf_i}|}{|Ins_{cf_i}|}$ is the probability of class c_i in the Common Feature Space S_{cf_i}. $|Ins_{c_i.cf_i}|$ is the total number of instances of class c_i that share cf_i. $|Ins_{cf_i}|$ is the total number of instances that share cf_i.

According to Equation 2, the more information gained by a Common Feature indicates fewer types in the random space generated by the Common Feature.

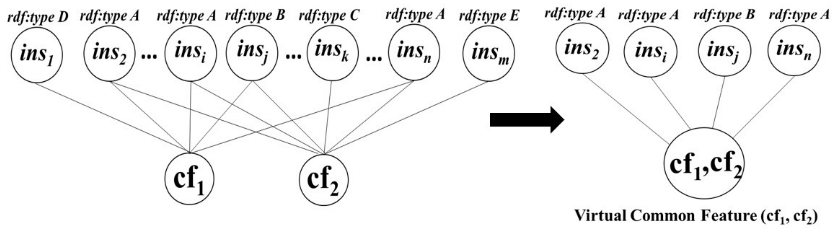

Fig. 2. An example of a Virtual Common Feature.

On the one side, a Common Feature can be shared by instances of different classes. On the other side, instances of a class might share more than one Common Feature. In this paper, we have evaluated the information gained from combinations of common features. Consider Fig. 2 to explain the idea. Common

Feature cf_1 is shared by instance ins_1 with *rdf:type D*, and ins_2, ins_i, ins_n with *rdf:type A*, and ins_j with *rdf:type B*. Common Feature cf_2 is also shared by ins_2, ins_i, ins_n with *rdf:type A*, and ins_k with *rdf:type C*, and ins_m with *rdf:type E*. The combination of (cf_1,cf_2) gains more information to compare with each cf_1 and cf_2. Because the Random Space of (cf_1,cf_2) contains lower entropy and fewer types (i.e., types A and B) to compare with cf_1 and cf_2 that contain instances with {*rdf:type A*, *rdf:type B*, *rdf:type D*} and {*rdf:type A*, *rdf:type C*, *rdf:type E*}, respectively. Based on this motivation, the concept of Virtual Common Feature is defined as follows.

Definition 6 (*Virtual Common Feature*). A Virtual Common Feature vcf is a combination of n ($n \geq 2$) common features where the number of instances that share vcf is greater than or equal to $MinIN$.

A Virtual Common Feature vcf shows there is a number of instances that share more than one Common Feature. In this regard, the information gained from a Virtual Common Feature can be computed by Equations 2 and 3. We just need to replace $|Ins_{cf_i}|$ and Entropy (S_{cf_i}) with $|Ins_{vcf_i}|$ and Entropy (VS_{vcf_i}). The $|Ins_{vcf_i}|$ is the total number of instances that share vcf_i and Entropy (VS_{vcf_i}) is the Entropy of Virtual Common Feature Space.

Given a Common Feature cf_i and a Virtual Common Feature vcf_i, the more information indicates lower entropy (i.e., lower uncertainty) in the random spaces generated by cf_i and vcf_i. This fact reveals that most of instances that have shared cf_i and vcf_i have the same type. Based on this motivation, the concept of Class Feature is defined as follows.

Definition 7 (*Class Feature*). A Common Feature cf_i or a Virtual Common Feature vcf_i is a Class Feature for Class c_i, if the information gained from cf_i or vcf_i is greater than or equal to NormGain Thresholds ($NGTh$).

Note that cf_i (or vcf_i) is a Class Feature for Class c_i where most instances that share cf_i have been assigned to Class c_i. It is important to mention that a class can have multiple class features including common features and virtual common features with information gained greater than or equal to *NGTh*.

3.2 Module 2: Instance-Class relationship analysis

If we take an instance with type and features, the goal of Module 2 is to analyse the correctness (i.e., CA) and incorrectness (i.e., ICA) of relationships between the instance and classes. To this end, Algorithm 1 has been implemented to assess the above targets for a given instance in four different statuses including (I) the CA status of an instance with one Feature, (II) the ICA status of an instance with one Feature, (III) the CA status of an instance with multiple features, and (IV) the ICA status of an instance with multiple features. Algorithm 1 receives Minimum Instance Number (*MinIN*), NormGain Thresholds (*NGTh*), classes (C), features of classes (*C.features*), instances (*Iset*), and features of instances (*I.features*) as inputs. Algorithm 1 returns *CA and ICA* statutes for given instances as an output. Note that the status of ins_i is *Undecidable* if ins_i is neither CA nor ICA.

Algorithm 1: Instance-Class relationship

```
input  : $MinIN, NGTh, C, C.features, Iset, I.features$
output : $CA\,and\,ICA$
1  $ICA \leftarrow \emptyset$, $CA \leftarrow \emptyset$, $Undecidable \leftarrow \emptyset$ ;
2  for each $ins_i \in Iset$ do
3      if $ins_i$ shares one Feature then
4          $IF$ the feature of $ins_i$ is a Common Feature, then $OneCommonFeature$ flag will be $true$ ; $ELSE\,ins_i$ is neither CA nor ICA and it will record in $Undecidable$ set, then the algorithm iterates for another instance;
5      else
6          1. The features of $ins_i$ will check and those which are common features will record in $CF_{ins_i}$;
7          2. $IF\,CF_{ins_i}$ has one Common Feature, then $OneCommonFeature$ flag will be true; $ELSEIF\,CF_{ins_i}$ has no Common Feature, so $ins_i$ is neither CA nor ICA and will record in $Undecidable$ set, then the algorithm iterates for another instance; $ELSE$ all common features of $CF_{ins_i}$ will store in $Feature_{ins_i}$ set;
8          3. $IF\,OneCommonFeature$ is not true, then the algorithm checks if $Feature_{ins_i} \geq MinIN$, if yes, then Virtual Common Feature will create with $vcf_{ins_i} = Feature_{ins_i}$ and $Multiplecommonfeatures$ flag will be true; $ELSE\,ins_i$ is neither CA nor ICA and will record in $Undecidable$ set, then the algorithm iterates for another instance;
9      if ($OneCommonFeature$) then
10         $IF$ Information gained by Common Feature of $ins_i \geq NGTh$, then do 4 and 5; $ELSE\,ins_i$ is neither CA nor ICA and it will record in $Undecidable$ set, then the algorithm iterates for another instance;
11         4. The common feature of $ins_i$ will check in the features of Class $c_j \in C$.;
12         5. $IF$ the feature of $ins_i$ is in the class features of $c_j$ and if $ins_i$ has the same type with class $c_j$, then $ins_i$ will record in $CA$,; $ELSE$ $ins_i$ has an $ICA$ status.;
13     if ($Multiplecommonfeatures$) then
14         $IF$ Information gained by Virtual Common Feature $vcf_{ins_i} \geq NGTh$, then do 6; $ELSE\,ins_i$ is neither CA nor ICA and it will record in $Undecidable$ set, then the algorithm iterates for another instance;
15         7. $IF$ the type of $ins_i$ is equal to the $classType$ of Virtual Common Feature $vcf_{ins_i}$, then $ins_i$ will record in $CA$ ; $ELSE$ $ins_i$ has an $ICA$ status.;
16 return $CA$ and $ICA$
```

4 Experiments and analysis

The following experiments have been conducted over DBpedia dataset 3.8 that is one of the most common errors encountered RDF-style KBs. By using DBpedia dataset, we considered two classes called Food and Hotel. Each class contains about 750 instances. The goal of following experiments is to check the accuracy of CAD in analyzing the correctness and incorrectness of relationships between instances and classes. Generally, the accuracy of a system is the degree of closeness between a measured value and the true value. In our scenario, a measured value refers to the number of correctly (i.e., CA) and incorrectly (i.e, ICA) assigned instances detected by CAD approach. While a true value indicates the predefined number of CA and ICA instances in a class. Thus, we correctly assigned 700 instances to the Hotel class that indicates a true value for CA instances. We also incorrectly assigned 50 instances of Hotel class to the Food class that shows a true value for ICA instances. To measure the accuracy of CAD approach, two measurements called $Accuracy_{CA}$ and $Accuracy_{ICA}$ are defined as follows.

Given a class, the accuracy of CAD in detecting correctly assigned instances can be computed by Equation 4:

$$Accuracy_{CA} = \frac{|ins_{CA} \cap INS_{CA}|}{|INS_{CA}|} \tag{4}$$

where ins_{CA} is the number of correctly assigned instances detected by CAD and INS_{CA} is a true value for the predefined number of CA instances in the class.

Given a class, the accuracy of CAD in detecting incorrectly assigned instances can be measured by Equation 5:

$$Accuracy_{ICA} = \frac{|ins_{ICA} \cap INS_{ICA}|}{|INS_{ICA}|} \tag{5}$$

where ins_{ICA} is the number of incorrectly assigned instances detected by CAD and INS_{ICA} is a true value for the predefined number of ICA instances in the class.

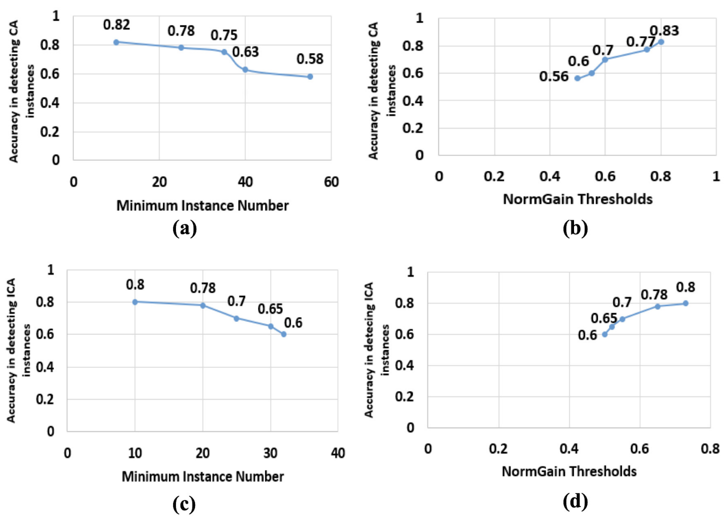

Fig. 3. (a) Accuracy in detecting CA instances in different *MinIN*, (b) Accuracy in detecting CA instances in different *NGTh*, (c) Accuracy in detecting ICA instances in different *MinIN*, (d) Accuracy in detecting ICA instances in different *NGTh*.

Fig. 3(a) shows that the accuracy of CAD approach in detecting CA instances has been gradually decreased by increasing *MinIN*. One reason behind such reduction is related to the strategy of selecting common features by using *MinIN*. Consider the process of analyzing common features in Algorithm 1. Given an instance, if the feature shared by ins_i is not a Common Feature, then the status of ins_i is undecidable. For example, an RDF triple (White House, location, Herm) detected indicates that White House is an instance of Hotel class with a particular feature i.e., (location, Herm). In the Hotel class, Herm is the only instance that has shared (location, Herm) as a particular feature. Algorithm 1 ignores some instances in case the features shared by them have not identified as common features. Fig. 3(b) shows that the accuracy of CAD approach in detecting CA instances has been grown by increasing *NGTh*. Fig. 3(c) represents that the accuracy of CAD in detecting ICA instances is reduced by increasing *MinI*. Consider again the RDF triple (White House, location, Herm). If White House has been incorrectly assigned to the Food class, Algorithm 1 ignores White House since (location, Herm) has not identified as a Common Feature.

5 Conclusions and future work

This paper has introduced a new SW data quality issue called Incorrect Class Assignment (ICA) problem that indicates incorrect assignment between instance-level data and corresponding classes in an ontology. So, we proposed an entropy-based approach called Correct Assignment Detector (CAD) to deal with ICA problem. A direction for future work is to apply Natural Language Processing (NLP) techniques on predicates of common features to find out more similar behaviors taken by instances in the groups.

References

1. A. Melo, J. Völker, and H. Paulheim, "Type prediction in noisy rdf knowledge bases using hierarchical multilabel classification with graph and latent features," *International Journal on Artificial Intelligence Tools*, vol. 26, no. 02, p. 1760011, 2017.
2. K. Gunaratna, K. Thirunarayan, A. Sheth, and G. Cheng, "Gleaning types for literals in rdf triples with application to entity summarization," in *International Semantic Web Conference*. Springer, 2016, pp. 85–100.
3. J. Lehmann, D. Gerber, M. Morsey, and A.-C. N. Ngomo, "Defacto-deep fact validation," in *International Semantic Web Conference*. Springer, 2012, pp. 312–327.
4. G. Töpper, M. Knuth, and H. Sack, "Dbpedia ontology enrichment for inconsistency detection," in *Proceedings of the 8th International Conference on Semantic Systems*. ACM, 2012, pp. 33–40.
5. J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov, "Discovering and maintaining links on the web of data," in *International Semantic Web Conference*. Springer, 2009, pp. 650–665.
6. N. Lao and W. W. Cohen, "Relational retrieval using a combination of path-constrained random walks," *Machine learning*, vol. 81, no. 1, pp. 53–67, 2010.
7. N. Lao, T. Mitchell, and W. W. Cohen, "Random walk inference and learning in a large scale knowledge base," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2011, pp. 529–539.
8. E. T. Jaynes, "Information theory and statistical mechanics," *Physical review*, vol. 106, no. 4, p. 620, 1957.