IMAGE-BASED LITERAL NODE MATCHING FOR LINKED DATA INTEGRATION

Takahiro Kawamura[1,2] and Akihiko Ohsuga[2]

[1]Corporate Research & Development Center, Toshiba Corp.
[2]Graduate School of Information Systems, University of Electro-Communications, Japan

ABSTRACT

This paper proposes a method of identifying and aggregating literal nodes that have the same meaning in Linked Open Data (LOD) in order to facilitate cross-domain search. LOD has a graph structure in which most nodes are represented by Uniform Resource Identifiers (URIs), and thus LOD sets are connected and searched through different domains.However, 5% of the values are literal values (strings without URI) even in a de facto hub of LOD, DBpedia. In SPARQL Protocol and RDF Query Language (SPARQL) queries, we need to rely on regular expression to match and trace the literal nodes. Therefore, we propose a novel method, in which part of the LOD graph structure is regarded as a block image, and then the matching is calculated by image features of LOD. In experiments, we created about 30,000 literal pairs from a Japanese music category of DBpedia Japanese and Freebase, and confirmed that the proposed method determines literal identity with F-measure of 76.1-85.0%.

KEYWORDS
Linked Open Data, Literal Matching, Image feature

1. INTRODUCTION

DBpedia is currently a *de facto* hub of LOD which represents part of Wikipedia. However, approx. 5% of the values in DBpedia are literals (string values without URI) according to our research. Thus, we cannot trace the links in the LOD graphs of different domains without relying on regular expression. Although projects of LOD generation from the web and social media are collecting attentions, many literal values are created, at least in the initial stage of the projects. Thus, the literal node matching will become a major issue in the near future. The goal of this paper is to connect literals as much as possible in order to connect LOD sets of different domains and merge them to the LOD cloud in the world [1]. We thus propose a method for determining the identity of literal values and support data linkage.

The novelty of our method is that the target literal and surrounding information in LOD are regarded as a block image, and then the identity of the literals is determined through similarity discrimination of two images. The contribution of this paper is the introduction of a new feature in Linked Data integration. This method is inspired by recent computer shogi, in which records of games are regarded as figures, and a game is played to make a good figure in the record [2, 3]. Also, literal matching corresponds to name identification, which is a traditional but important problem in system integration (SI) projects. It is also similar to instance matching in ontology alignment, although the matching target is not an instance (resource in LOD) but a value.

The rest of this paper is organized as follows. Section 2 proposes image feature extraction from LOD. Then, Section 3 evaluates the matching results of the literals using a machine learning method, comparing with simple string matching. Finally, Section 4 presents related work, and Section 5 refers to future work and concludes this paper.

2. Image Feature Extraction from LOD

This section first defines literal matching as a binary classification problem and then proposes a method of extracting features for a classifier.

2.1. Binary classification problem

In this section, we define literal matching as the following binary classification based on related work [11].

Definition 1. *If different literal values a and b refer to the same object in the real world, we denote $(a, b) \in \mathcal{R}$. Given two LOD graphs A and B as input, the goal of literal matching is to compute the set $\mathcal{M}$.*

$$\mathcal{M} = \{(a, b) | (a, b) \in A \times B, (a, b) \in \mathcal{R}\}$$

Also, the problem of finding matching literal pairs can be formalized as the following binary classification problem.

$$\mathcal{C} : (a, b) \to \{-1, 1\} \; for \; (a, b) \in A \times B$$

, where $\mathcal{C}$ is a classifier that maps the non-matching pairs to -1 and the matching pairs to 1.

2.2. Extraction process of image features

The workflow of literal matching is shown in Fig.1. First, feature vectors are constructed as input for the classifier. In order to construct the feature vector we adopted Scale-Invariant Feature Transform (SIFT) [18], which is a well-known feature extraction method in computer vision. SIFT extracts local features around a key point in an image. In detail, the area surrounding a key point is divided into 4x4 blocks, with each block providing illumination changes (gradient) of 8 orientations per 45 deg., and then a vector with 128 dimensions is created. As the feature extraction methods in computer vision, there are also Speeded Up Robust Features (SURF) that is a high speed version of SIFT, haar-like, and Histogram of oriented Gradient (HOG), although the methods have their own limitations in terms of recognition objects. The reason for adopting SIFT is that it extracts features for each key point in an image. By regarding the target literal as a key point, a graph structure can be mapped to the SIFT algorithm. In contrast, the haar-like feature is for each image, and the HOG is for a pixel.

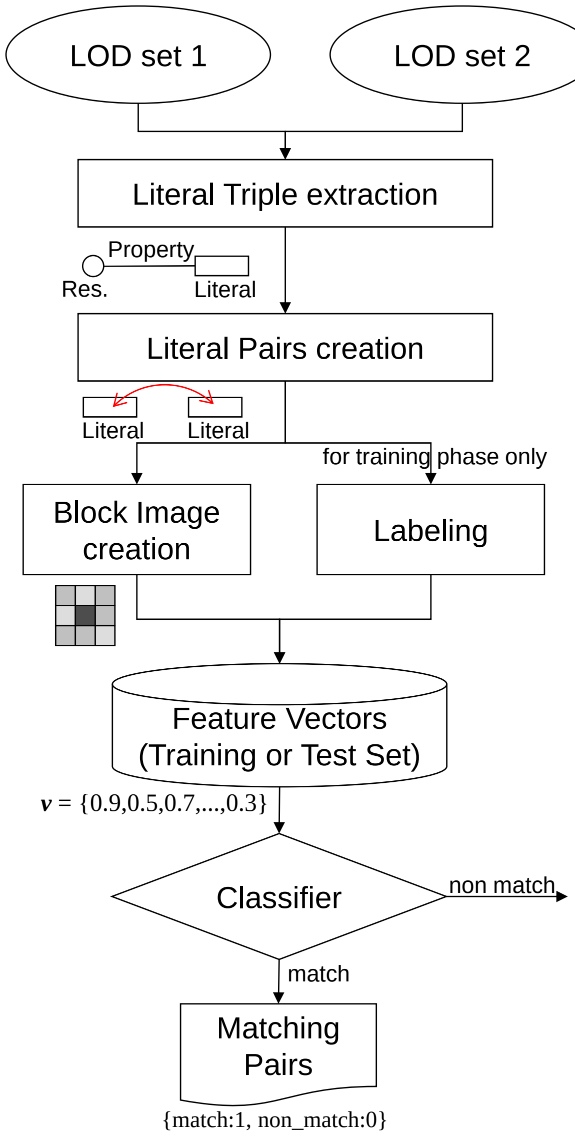

Figure 1. Workflow of literal matching

The method of generating a block image from two LOD graphs is shown in Fig. 2. Properties and resources connected from two target literals are selected for comparison, and another two properties and values connected from the above resources are also selected. Then, a grayscale image of 3x3 blocks is created from each similarity Sim_l, Sim_p, Sim_r. The similarities Sim_l, Sim_p, Sim_r are composed of string similarity and semantic similarity and defined as follows. Each block has a value [0,1], where 1 represents black. We attempt to generate an image with specified regularity. If we find that the above two resources have common properties, we alphabetically select two properties and their values as the second and the third row. If they don't have common properties, we select the two most similar properties and their values.

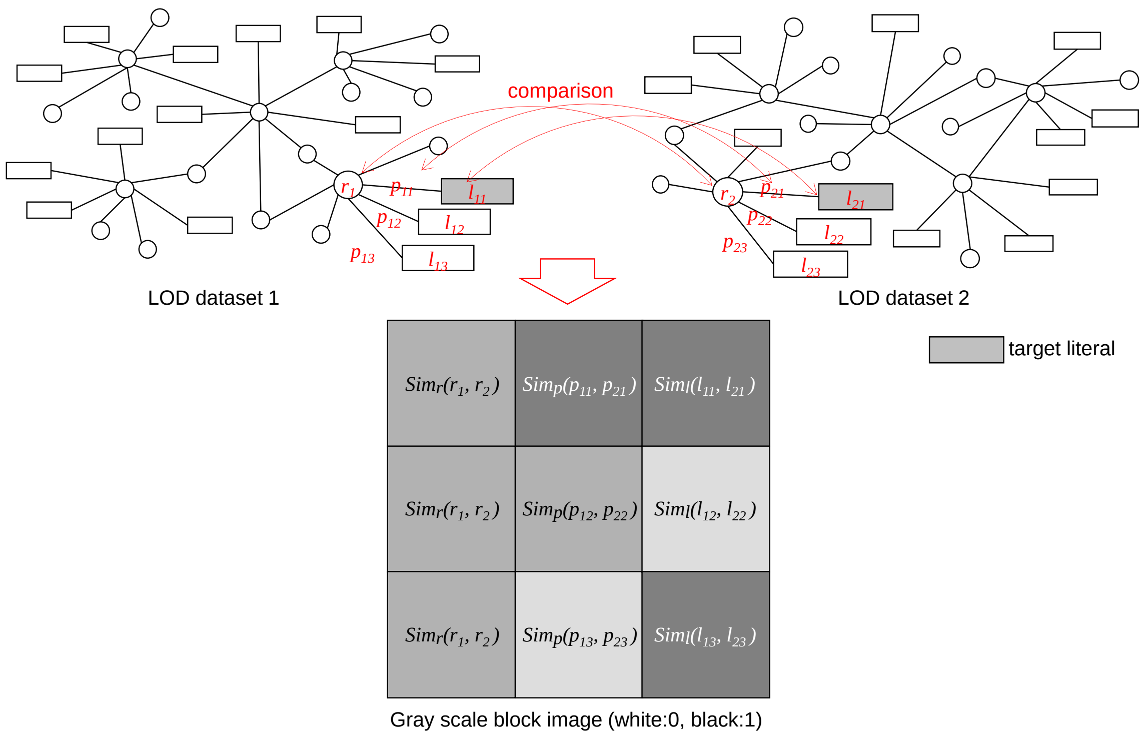

Figure 2. Image generation from LOD

$$\begin{aligned} Sim_l(l_{1i}, l_{2j}) =& \alpha\ StringSim(l_{1i}, l_{2j}) \\ & + \beta\ SemanticSim(l_{1i}, l_{2j}) \end{aligned} \tag{1}$$

$$\begin{aligned} Sim_p(p_{1i}, p_{2j}) =& \gamma\ StringSim(p_{1i}, p_{2j}) \\ & + \delta\ SemanticSim(p_{1i}, p_{2j}) \end{aligned} \tag{2}$$

$$\begin{aligned} Sim_r(r_1, r_2) =& \epsilon\ StringSim(r_1, r_2) \\ & + \zeta\ SemanticSim(r_1, r_2) \end{aligned} \tag{3}$$

$$StringSim(str1, str2) = \frac{|str1 \cap str2|}{|str1 \cup str2|} \tag{4}$$

$$\begin{aligned} & SemanticSim(str1, str2) = \\ & \begin{cases} 1.0 \quad \text{if} \qquad \text{str1} = \text{str2} \\ 0.75 \ \text{else if str1 is } \textit{Synonym} \text{ of str2} \\ 0.5 \quad \text{else if str1 is } \textit{Hypernym} \text{ or} \\ \qquad\qquad\qquad \textit{Hyponym} \text{ of str2} \\ 0.25 \ \text{else if str1 has same } \texttt{namespace} \text{ as str2} \\ 0.0 \quad \text{else} \end{cases} \end{aligned} \tag{5}$$

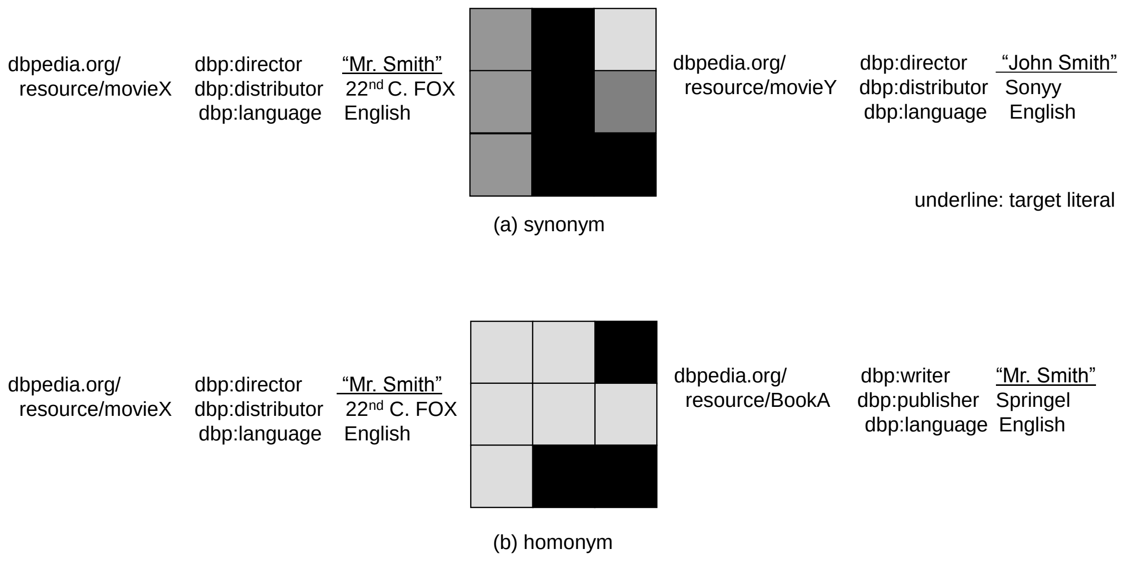
dbpedia.org/ resource/movieX
dbp:director "Mr. Smith"
dbp:distributor 22nd C. FOX
dbp:language English
dbpedia.org/ resource/movieY
dbp:director "John Smith"
dbp:distributor Sonyy
dbp:language English
underline: target literal
(a) synonym
dbpedia.org/ resource/movieX
dbp:director "Mr. Smith"
dbp:distributor 22nd C. FOX
dbp:language English
dbpedia.org/ resource/BookA
dbp:writer "Mr. Smith"
dbp:publisher Springel
dbp:language English
(b) homonym

Figure 3. Example features of synonym and homonym

l_{ij}, p_{ij}, r_{ij} mean identifiers of a literal node, a property and a resource, respectively. Also, parameters *α, β, γ, δ, ε,* and *ζ* satisfy $0 \leq \alpha, \beta, \gamma, \delta, \varepsilon, \zeta$, and $\alpha+\beta$, $\gamma+\delta$, $\varepsilon+\zeta \leq 1$. These parameters depend on problems to be solved, but we set 0.5 for each in the next section. *StringSim* is a Jaccard coefficient, and *SemanticSim* corresponds to a simplified calculation of the similarity indicator based on path lengths between two classes in ontology matching. *SemanticSim* is based on the semantic similarity measurement between two words according to path lengths between two corresponding classes in ontology [19]. We then simplified the calculation to reduce the processing time. Also, *str1, str2* are strings, and are regarded as sets of characters in *StringSim*. To determine *Synonym, Hypernym,* and *Hyponym*, we refer to the external ontologies, and we use WordNet [20] and Japanese WordNet [21] in the next section.

This image represents a type of contextual information around the target literal. For example, in the case that the target literals are synonyms with different notations, but their semantic similarity cannot be determined by ontology, the image will have high similarities in surrounding blocks, except for a block of the target literals like Fig. 3(a). In the case of homonyms with the same notation, the image will have low similarities in surrounding blocks, except for a block of the target literals like Fig. 3(b). In this manner, the image characterizes the connection of meaning around the target literal in the graph.

Finally, a target literal value is regarded as a key point in SIFT, and the feature vector is constructed from the similarity changes of the adjacent 9 blocks. In SIFT, the gradient is taken in order to absorb the illuminance difference by lighting etc. In our method, in the case that both LOD sets are described in the same schema (property definitions), the similarities of the links and nodes surrounding the target literals are expected to be rather high due to the same terminology and convention. However, in the case of LOD sets with different schemas, they are expected to be lower than the similarities of the same sets. Therefore, in order to absorb the difference in the relative level of the similarity, the proposed method does not use the absolute values of *Sim*, but the differences from adjacent blocks (similarity gradient) as the training data. But no orientation is determined, and an absolute value of the similarity at the key point is included as a basis unlike SIFT. The feature vector ***v*** is represented as follows. Also, each of the parameters *η, θ,* and *λ* is set to 1.0 in the next section.

$$\begin{aligned} v =& Sim_l(l_{11}, l_{21}) + \Delta_{p1} + \Delta_{r1} \\ & \Delta_{l2} + \Delta_{p2} + \Delta_{r2} \\ & \Delta_{l3} + \Delta_{p3} + \Delta_{r3} \\ \Delta_{li} =& \eta \left(Sim_l(l_{1i}, l_{2i}) - Sim_l(l_{1(i-1)}, l_{2(i-1)}) \right) \\ \Delta_{pi} =& \theta \left(Sim_p(p_{1i}, p_{2i}) - Sim_l(l_{1i}, l_{2i}) \right) \\ \Delta_{ri} =& \lambda \left(Sim_r(r_1, r_2) - Sim_p(p_{1i}, p_{2i}) \right) \quad (i = 1, 2, 3) \end{aligned} \tag{6}$$

3. Literal Matching Experiment

This section shows the experimental results for evaluating the effectiveness of the proposed feature vectors. Unlike the instance matching, we could not find any public research tool or algorithm to perform a comparison for literal matching on LOD. We thus compared the proposed method with a baseline method using a string matching technique.

As LOD sets with the same schema, we extracted triples (N = 8,504) that have literal values, from the resources in the “JPOP” (Japanese pop music) category of DBpedia Japanese [22], and created literal pairs ((N^2-N)/2 = 36,154,756). We then selected 17,391 pairs, excluding the pairs obviously not matching such as time and location, and manually marked the literal matching to them TRUE or FALSE.

As LOD sets with different schemas, we extracted triples (M = 2,879) that have literal values, from the resources in the “J-POP” category of Freebase LOD [23], and created literal pairs (N*M = 24,483,016) combining the above triples (N = 8,504) extracted from DBpedia. We then selected 13,702 pairs, excluding the pairs obviously not matching as mentioned above, and manually marked the literal matching to them TRUE or FALSE.

The literal pairs regarded as matching include: inconsistent spellings of values for properties such as label and alias, inconsistent spellings and different notations of album titles and artists' names that have not become resources and award record such as “Oscar nomination”. Some descriptions of addresses and dates have also not become resources.

Then, the image features described in the previous section are calculated for each of the above-mentioned pairs. As the training and classification method of the feature vectors, Random Forest [24] was used. Random Forest is composed of multiple weak learners (decision trees) to obtain better predictive performance, and is advantageous in that it gives estimates of what features are important in the classification, and has an effective method for estimating missing data and maintains accuracy when a proportion of the data is missing. It also runs efficiently on large datasets and is faster than, for example, Support Vector Machine (SVM). Other research on instance matching [11] indicated that Random Forest surpasses other methods owing to the nature of the problem. We also confirmed that the precision of SVM can be easily raised, but the recall tends to fall in the same experimental setting, since SVM is largely dependent on training dataset. In detail, 10 decision trees were generated, each of which randomly takes 4 features and unlimited depth of the tree, and the algorithm of each tree is an implementation (modified REPTree) based on C4.5.

Finally, 10-fold cross-validation was conducted for each evaluation. Figure 4 indicates our evaluation procedure.

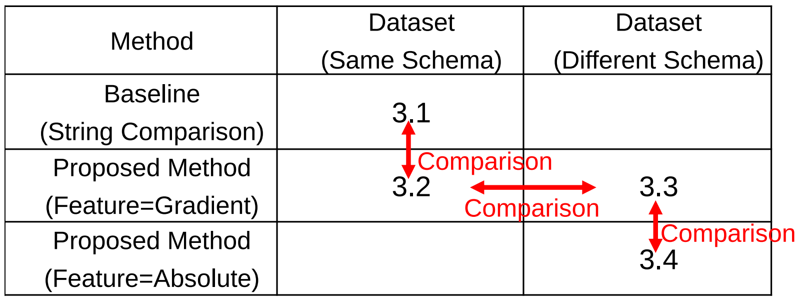
Method
Dataset (Same Schema)
Dataset (Different Schema)
Baseline (String Comparison)
Proposed Method (Feature=Gradient)
Proposed Method (Feature=Absolute)
3.1
Comparison
3.2
Comparison
3.3
Comparison
3.4

Figure 4. Evaluation Procedure

3.1. Accuracy of the baseline method for the same schema set

As a baseline method, we used string comparison of literal values with Jaccard coefficient *jcc*. The matching result is shown in Table 1, in which TRUE and FALSE mean matching pairs and non-matching pairs, respectively.

Table 1. Classification result of string comparison (%).

	$jcc > 0.7$		$jcc > 0.9$	
	Precision	Recall	Precision	Recall
TRUE	61.4	58.0	82.8	54.7
FALSE	98.8	99.0	98.7	99.7

In the table, both precision and recall of FALSE have high scores. Thus, we can easily raise the total accuracy by strictly determining the identities of the pairs, and raising the accuracy of FALSE, since the matching pairs (TRUE) are only about 3% of all the pairs. Also, we can easily raise the precision of TRUE by strictly determining the string matching, as shown in comparison of *jcc* > 0.7 and *jcc* > 0.9. However, it results in a decrease of the recall of TRUE and loss of the matching pairs. Therefore, we can confirm that solving the problem of literal matching involves raising the recall of TRUE without decreasing the precision by separately looking at the results of TRUE and FALSE.

3.2. Accuracy of the proposed method for the same schema set

The matching result using the proposed method is shown in Table 2.

Table 2. Classification Result of image features for the same schema sets (%).

	Precision	Recall	F-Measure
TRUE	87.5	82.6	85.0
FALSE	99.5	99.7	99.6
Weighted Avg.	99.2	99.2	99.2

In the table, the proposed method improved the recall of TRUE 25 points compared with the case of *jcc* > 0.7, and also improved the precision 5 points compared with the case of *jcc* > 0.9. Moreover, the weighted average according to the number of pairs including FALSE resulted in over 99% precision, recall and F-measure, although it highly depends on a large number of FALSE pairs. The out-of-bag (OOB) error, which is an unbiased estimate of the classification error with unused (out of sample) data, was about 1%. Thus, the effectiveness of the proposed method was confirmed.

3.3. Accuracy of the proposed method for the different schema set

The matching result of the proposed method for the different schema set is shown in Table 3.

Table 3. Classification Result of image features for different schema sets (%).

	Precision	Recall	F-Measure
TRUE	83.6	69.8	76.1
FALSE	99.5	99.8	99.7
Weighted Avg.	99.3	99.3	99.3

In the table, the recall of TRUE fell 13 points compared with the result of the same schema sets, although the precision remained 4 points fall. The reason for less recall than for the same schema sets is considered to be the difference of naming convention and terminology of the properties. The common property between DBpedia Japanese and Freebase was only http://www.w3.org/2000/01/rdf-schema\#label. Even in the case of the same meaning property, Freebase defines the original property name. Also, DBpedia has many kinds of properties, whereas Freebase does not necessarily have properties corresponding to them. Therefore, discrete patterns of the matching properties could not be learned between the different schema sets, and thus the increase of the data size will improve the recall.

3.4. Accuracy of the proposed method with absolute values

Finally, we used the absolute values of *Sim*, not the differences from adjacent blocks (similarity gradient), as the feature vector. The evaluation on about 30 thousand pairs of the same schema and the different schema set, in which the relative level of the similarity is different as mentioned in section 2.2, was conducted to confirm that similarity gradient can be more generic features than the absolute values. As a result, however, the gradient slightly improved the precision of TRUE, by 1 point (84.7% → 85.5%), and there was no difference in the recall of TRUE (approx. 77%) and the accuracy of FALSE. The reason for this result is that the *StringSim* and *SemanticSim* are normalized to [0,1], and the absolute values did not differ markedly in terms of the relative level of similarity. The average of similarities between the same schema sets (DBpedia) was 0.40, and the average of similarities between different schema sets (DBpedia and Freebase) was 0.27.

We also calculated predictor importance (Mean Decrease Accuracy in Random Forests) in both kinds of feature vectors. Fig. 5 shows an absolute value of similarity between target literals has the highest importance in the both features. However, we can confirm that the importance decreased and importance of surrounding nodes and links increased in the features of the gradient, comparing with the features of absolutes values. Thus, the features of the gradient can be regarded as less dependence on a particular predictor, and effective when the training data is missing and changing.

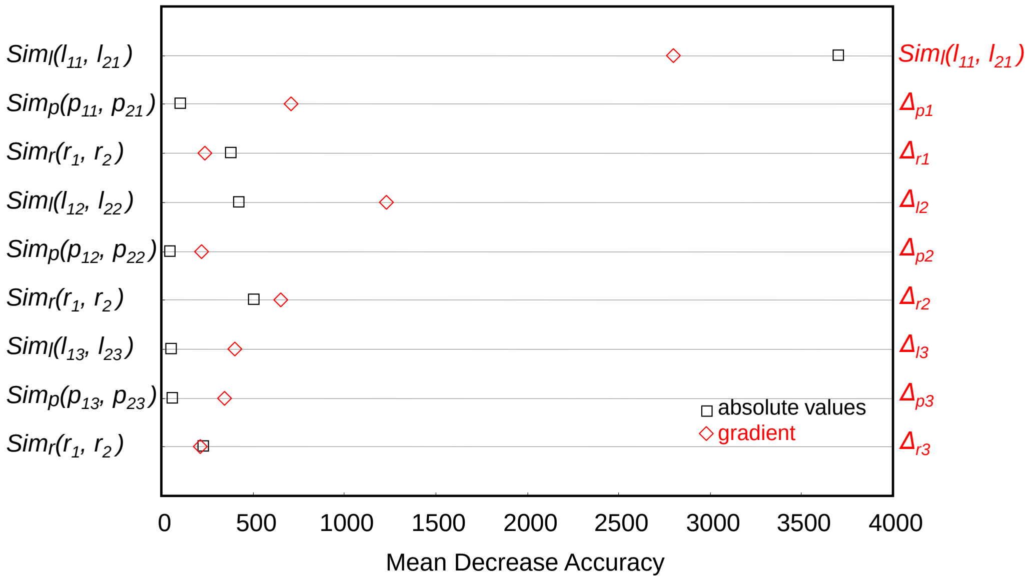

Figure 5. Comparison of predictor importance

4. Related Work

Since we could not find research similar to literal matching of LOD, we first introduce Apache Solr [13] that is used for name identification in SI projects. Name identification means confirming the consistency of customers' information for financial institutions and administrative organizations, whereas it usually refers to data cleansing. Frameworks such as Solr calculate word similarity using TF/IDF etc., and then suggest words that might have the same meaning, and then developers finally confirm the words and register them in a dictionary. Thus, if users search for "dentist" on the web, "dental clinic" is also included in the results owing to such frameworks. However, synonyms which have totally different notations are not suggested by using word similarity, and homonyms which have the same notations cannot be distinguished by dictionary registration in the frameworks.

Next, we first introduce related research on instance matching. With a well-known tool for instance matching, SILK [4, 5], a user selects features to identify the similarity and defines the link specifications for each dataset. Maali et al. [9] also implemented four kinds of extensions for instance matching in Google Refine [8], that is a data cleansing tool provided by Google:

1. SPARQL query
2. SPARQL with full-text search
3. SILK Server API
4. Keyword-based Sindice search API [7] with 2., since Sindice result is a list of document URIs containing matching RDF data, and not a list of the actual matching resources.

Then, Maali et al. measured accuracy and computational performance for each extension. Although the approaches can be applied for literal matching, all of them need to know structures of two datasets to be compared in advance for constructing search queries and translation rules. Research on automatic rule generation based on Genetic Programming [6] is also conducted. Also, [10] acquires matching rules for each dataset based on semi-supervised learning and refines repeatedly. In addition, [12] is a useful reference for surveying instance matching. Literal matching is a problem similar to instance matching, but a different problem. The problem remains even after performing instance matching and property matching [14]. Figure 6 indicates the relationship of three kinds of matching. For example, there is a case that 'Director' property of a movie resource and 'Author' property of a book resource indicate the same person. In regard to technical aspects, instance matching also utilizes semantic and structural similarities of properties and nodes connected to the instance as feature vectors. However, literal matching is for a terminal node connected only to a property, and then needs to define a new feature.

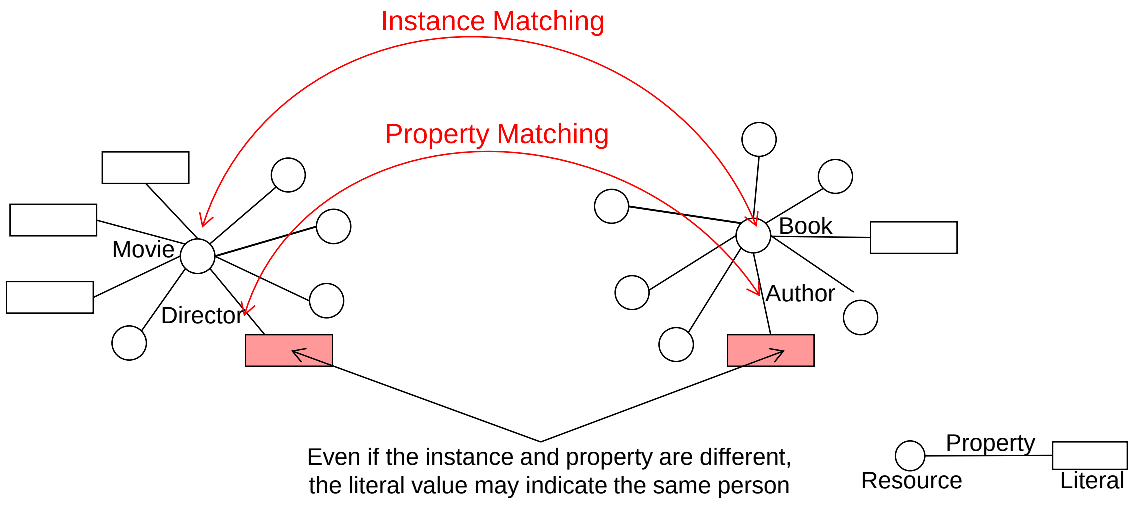
Instance Matching
Property Matching
Movie
Director
Book
Author
Even if the instance and property are different, the literal value may indicate the same person
Property
Resource
Literal

Figure 6. Instance, Property and Literal Matching

LOD matching techniques have been implemented in commercial applications. Fujitsu [15] released a data matching solution that is included in its open data platform. They match terms in a microblog with corresponding entities in their knowledge base, that contains a vast amount of name variations of entities such as acronyms, confusable names, spelling variations, and nick names etc. for the purpose of the classification of microblogs, advertisement and product recommendation. However, to the best of our knowledge from [16, 17], the dataset is not in a linked data format.

5. Conclusion and Future work

Literal matching to determine identities of literal values in LOD was addressed in this paper, in order to give a URI to the literals and allow search through different LOD sets. For this problem, we proposed a novel method to extract image features from LOD. In experiments, we extracted literal pairs from Japanese music category of DBpedia Japanese and Freebase, and determined the identities of the pairs based on the features. The experiments showed that the matching pairs were successfully classified with F-measure of 76.1-85.0%.

Future works include performance evaluations. However, literal matching between datasets will not be conducted at online, but at the dataset creation and modification. Therefore, the processing time is not a critical issue in practice. In the future, however, the relationship between data size and the processing time needs to be investigated.

The proposed method does not depend on the music domain in principle, and we intend to apply it to other domains. In addition, we intend to make a module of the classifier and publish it as a web service that dynamically matches the literals. These initiatives will facilitate cross-domain search through links in LOD.

REFERENCES

[1] C. Bizer, T. Heath, and T. Berners-Lee: “Linked Data - The Story So Far”, International Journal on Semantic Web and Information Systems (IJSWIS), Vol. 5, No. 3, pp. 1-22, 2009.

[2] T. Obata, T. Sugiyama, K. Hoki, and T. Ito: “Consultation Algorithm for Computer Shogi: Move Decisions by Majority”, Computers and Games, Vol. 6515, pp. 156{165, 2011.

[3] K. Hoki, and T. Kaneko: “The Global Landscape of Objective Functions for the Optimization of Shogi Piece Values with a Game-Tree Search”, Advances in Computer Games, Vol. 7168, pp. 184{195, 2012.

[4] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov: “Silk - A link discovery framework for the web of data”, Proc. of 2nd Linked Data on the Web Workshop (LDOW), CEUR Workshop Proceedings, Vol. 538, 2009.

[5] J. Volz, C. Bizez, M. Gaedke, and G. Kobilarov: “Discovering and maintaining links on the web of data”, Proc. of 8th International Semantic Web Conference (ISWC), pp. 650-665, 2009.

[6] R. Isele and C. Bizer: “Learning linkage rules using genetic programming”, Proc. of 6th International Workshop on Ontology Matching (OM), CEUR Workshop Proceedings, Vol. 814, 2011.

[7] “Sindice - The Semantic Web Index”, http://sindice.com/, 2014.

[8] “Google Refine (in transition over to Open Refine)”, https://github.com/OpenRefine, 2014.

[9] F. Maali, R. Cyganiak, and V. Peristeras: “Re-using Cool URIs: Entity Reconciliation Against LOD Hubs”, Proc of 4th Linked Data on the Web Workshop (LDOW2011), CEUR Workshop Proceedings, Vol. 813, 2011.

[10] X. Niu, S. Rong, H. Wang, and Y. Yu: “An Effective Rule Miner for Instance Matching in a Web of Data”, Proc. of 21st ACM international conference on Information and knowledge management (CIKM), pp. 1085-1094, 2012.

[11] S. Rong, X. Niu, E. W. Xiang, H. Wang, Q. Yang, and Y. Yu: “A Machine Learning Approach for Instance Matching Based on Similarity Metrics”, Proc. of 11th International Semantic Web Conference (ISWC), pp. 460-475, 2012.

[12] S. Castano, A. Ferrara, S. Montanelli, G. Varese: “Ontology and Instance Matching”, Knowledge-Driven Multimedia Information Extraction and Ontology Evolution, pp. 167-195, 2011.

[13] “Apache Solr”, http://lucene.apache.org/solr/, 2014.

[14] K. Gunaratna, K. Thirunarayan, P. Jain, A. Sheth, and S. Wijeratne: “A Statistical and Schema Independent Approach to Identify Equivalent Properties on Linked Data”, Proc. of 9th International Conference on Semantic Systems (I-SEMANTICS), pp. 33-40, 2013.

[15] Fujitsu Laboratories of Europe Limited: “Fujitsu Laboratories Develops Technology for Automatically Linking with Open Data throughout the World”, http://www.fujitsu.com/global/news/pr/archives/month/2014/20140116-01.html, 2014.

[16] “TAC KBP 2013 Entity Linking Track”, http://www.nist.gov/tac/2013/KBP/EntityLinking/index.html, 2014.

[17] Q. Miao, H. Lu, S. Zhang, and Y. Meng: “Simple Yet Effective Method for Entity Linking in Microblog-Genre Text”, Proc. of Natural Language Processing and Chinese Computing (NLPCC), CCIS Vol.400, pp.440-447, 2013.

[18] D. G. Lowe: “Object recognition from local scale-invariant features”, Proc. of 7th International Conference on Computer Vision (ICCV), Vol. 2, pp. 1150-1157, 1999.

[19] Y. Li, Z. A. Bandar, and D. McLean: “An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources”, IEEE Transactions on Knowledge and Data Engineering, Vol. 15, No. 4, pp. 871-882, 2003.

[20] Princeton University: “WordNet: A lexical database for English”, http://wordnetweb.princeton.edu/perl/webwn, 2014.

[21] National Institute of Information and Communications Technology: “Japanese WordNet”, http://nlpwww.nict.go.jp/wn-ja, 2014.

[22] National Institute of Informatics: “DBpedia Japanese”, http://ja.dbpedia.org/sparql, 2014.

[23] OpenLink Software: “OpenSearch”, http://lod.openlinksw.com/sparql/, 2014.

[24] L. Breiman: “Random Forests”, Machine Learning, Vol. 45, No. 1, pp. 5-32, 2001.

Authors

Takahiro Kawamura is a Senior Research Scientist at Corporate Research and Development Center, Toshiba Corp., and also an Visiting Associate Professor at the Graduate School of Information Systems, the University of Electro-Communications, Japan.

Akihiko Ohsuga is a Professor at the Graduate School of Information Systems, the University of Electro-Communications, Japan.