Yuan-Fang Li · Wei Hu
Jin Song Dong · Grigoris Antoniou
Zhe Wang · Jun Sun
Yang Liu (Eds.)

LNCS 10055

Semantic Technology

6th Joint International Conference, JIST 2016
Singapore, Singapore, November 2–4, 2016
Revised Selected Papers

Springer

Yuan-Fang Li · Wei Hu
Jin Song Dong · Grigoris Antoniou
Zhe Wang · Jun Sun
Yang Liu (Eds.)

Semantic Technology

6th Joint International Conference, JIST 2016
Singapore, Singapore, November 2–4, 2016
Revised Selected Papers

Springer

Editors
Yuan-Fang Li
Information Technology
Monash University
Melbourne, VIC
Australia

Wei Hu
Computer Science and Technology
Nanjing University
Nanjing
China

Jin Song Dong
Computer Science
National University of Singapore
Singapore
Singapore

Grigoris Antoniou
University of Huddersfield
Huddersfield
UK

Zhe Wang
Information and Communication Technology
Griffith University
Brisbane, QLD
Australia

Jun Sun
ISTD
Singapore University of Technology and Design
Singapore
Singapore

Yang Liu
Computer Science and Engineering
Nanyang Technological University
Singapore
Singapore

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-50111-6 ISBN 978-3-319-50112-3 (eBook)
DOI 10.1007/978-3-319-50112-3

Library of Congress Control Number: 2016959173

LNCS Sublibrary: SL3 – Information Systems and Applications, incl. Internet/Web, and HCI

Preface

This volume contains the papers presented at JIST 2016: the 6th Joint International Semantic Technology Conference held during November 2–4, 2016, in Singapore. JIST 2016 was co-hosted by National University of Singapore, Nanyang Technological University (Singapore), and Monash University (Australia). JIST is a regional federation of semantic technology-related conferences. It attracts many participants from mainly the Asia Pacific region and often Europe and the USA. The mission of JIST is to bring together researchers in semantic technology research and other areas of semantic-related technologies to present their innovative research results and novel applications.

The main topics of JIST 2016 include ontology and reasoning, linked data, and knowledge graph, among others. JIST 2016 consisted of two keynotes, a main technical track, including (full and short) papers from the research and the in-use tracks, a poster and demo session, a workshop, and two tutorials. There were a total of 34 submissions for the main technical tracks from 17 countries. All papers were reviewed by at least three reviewers and the results were rigorously discussed by the program co-chairs. In all, 16 full papers (47%) and eight short papers were accepted in the technical tracks.

The paper topics are divided into six categories: Ontology and Data Management, Linked Data, Information Retrieval and Knowledge Discovery, RDF and Query, Knowledge Graph, and Applications of Semantic Technologies.

We would like to thank the JIST Steering Committee, Organizing Committee, and Program Committee for their significant contributions. We would also like to especially thank the co-hosts for their support in making JIST 2016 a successful and memorable event. Finally, we would like to express our appreciation to all speakers and participants of JIST 2016. This book is an outcome of their contributions.

November 2016

Yuan-Fang Li
Wei Hu
Jin Song Dong
Grigoris Antoniou
Zhe Wang
Jun Sun
Yang Liu

Organization

Program Committee

Paolo Bouquet	University of Trento, Italy
Nopphadol Chalortham	Silpakorn University, Thailand
C. Chantrapornchai	Kasetsart University, Thailand
Gong Cheng	Nanjing University, China
Paola Di Maio	ISTCS.org/IIT Mandi, India
Stefan Dietze	L3S Research Center, Germany
Dejing Dou	University of Oregon, USA
Jae-Hong Eom	Seoul National University, South Korea
Naoki Fukuta	Shizuoka University, Japan
Volker Haarslev	Concordia University, Canada
Armin Haller	Australian National University, Australia
Masahiro Hamasaki	National Institute of Advanced Industrial Science and Technology (AIST), Japan
Sungkook Han	Wonkwang University, South Korea
Koiti Hasida	AIST, Japan
Wei Hu	Nanjing University, China
Eero Hyvönen	Aalto University, Finland
Ryutaro Ichise	National Institute of Informatics, Japan
Vahid Jalali	Indiana University, USA
Jason Jung	Chung-Ang University, South Korea
Yong-Bin Kang	Monash University, Australia
Takahiro Kawamura	Japan Science and Technology Agency, Japan
Pyung Kim	Jeonju National University of Education, South Korea
Seiji Koide	Ontolonomy, LLC, Japan
Kouji Kozaki	Osaka University, Japan
Seungwoo Lee	KISTII, South Korea
Tony Lee	Saltlux, Inc., South Korea
Yuan-Fang Li	Monash University, Australia
Riichiro Mizoguchi	Japan Advanced Institute of Science and Technology, Japan
Takeshi Morita	Keio University, Japan
Ralf Möller	Universität zu Lübeck, Germany
Shinichi Nagano	Toshiba Corporation, Japan
Ikki Ohmukai	National Institute of Informatics, Japan
Artemis Parvizi	Oxford University Press, UK
Yuzhong Qu	Nanjing University, China
Ulrich Reimer	University of Applied Sciences St. Gallen, Switzerland

Giorgos Stoilos	National Technical University of Athens (NTUA), Greece
Umberto Straccia	ISTI-CNR, Italy
Boontawee Suntisrivaraporn	DTAC, Thailand
Hideaki Takeda	National Institute of Informatics, Japan
Holger Wache	University of Applied Science Northweastern Switzerland, Switzerland
Haofen Wang	East China University of Science and Technology, China
Peng Wang	Southeast University, China
Xin Wang	Tianjin University, China
Zhe Wang	Griffith University, Australia
Krzysztof Wecel	Poznan University of Economics, Poland
Gang Wu	College of Information Science and Engineering, Northeastern University, China
Guohui Xiao	KRDB Research Centre, Free University of Bozen-Bolzano, Italy
Bin Xu	DCST, Tsinghua University, China
Yasunori Yamamoto	Database Center for Life Science, Japan
Xiang Zhang	Southeast University, China
Yuting Zhao	IBM Italy, Italy
Amal Zouaq	Royal Military College of Canada, Canada

Organizing Committee

General Chairs

Jin Song Dong	National University of Singapore, Singapore
Grigoris Antoniou	University of Huddersfield, UK

Program Co-chairs

Yuan-Fang Li	Monash University, Australia
Wei Hu	Nanjing University, China

Publicity Chair

Zhe Wang	Griffith University, Australia

Local Chair

Jun Sun	Singapore University of Technology and Design, Singapore

Workshop Co-chairs

Xin Wang	Tianjing University, China
Hanmin Jung	Korea Institute of Science and Technology Information, Korea

Tutorial Co-chairs

Armin Haller	Australian National University, Australia
Gong Cheng	Nanjing University, China

Poster and Demo Co-chairs

Zhichun Wang	Beijing Normal University, China
Kouji Kozaki	Osaka University, Japan

Finance Chair

Yang Liu	Nanyang Technological University, Singapore

Publicity Chair

Haofen Wang	East China University of Science and Technology, China

Keynotes

Managing Dynamic Ontologies: Belief Revision and Forgetting

Kewen Wang

Griffith University, Brisbane, Australia
k.wang@griffith.edu.au

Ontologies have recently been used in a wide range of practical domains such as e-Science, e-Commerce, medical informatics, bio-informatics, and the Semantic Web. An *ontology* is a formal model of some domain knowledge of the world. It specifies the *formalization* of the domain knowledge as well as the *meaning* (semantics) of the formalization. The Web Ontology Language (OWL), with its latest version, OWL 2, is based on description logics (DLs). Thus, an ontology is often expressed as a knowledge base (KB) in DLs, which consists of both terminological knowledge (or schema information) in the TBox and assertional knowledge (or data information) in the ABox. As with all formal knowledge structures, ontologies are not static, but may evolve over time. Indeed, ontology engineering is described as a life-cycle, which is based on evolving prototypes and specific techniques peculiar to each ontology engineering activity. An important and challenging problem is thus how to effectively and efficiently modify ontologies.

In this talk, we discuss some recent developments and challenges for two paradigms of ontology changes. We focus on model-based approaches.

Knowledge Update: Outdated and incorrect axioms in an ontology have to be eliminated from the ontology and newly formed axioms have to be incorporated into the ontology. In the field of belief change, extensive work has been done on formalising various kinds of changes over logical knowledge bases. In particular, elimination of old knowledge is called contraction and incorporation of new knowledge is called revision. The dominant approach in belief change is the so called AGM framework. Regardless of its wide acceptance, the AGM framework is incompatible with DLs due to its assumption on an underlying logic that includes propositional logic. The incompatibility is the major difficulty in defining DL contraction and revision. Additionally, DL revision is more involved than AGM revision. AGM revision aims to resolve any inconsistency caused while incorporating a new formula. Since a meaningful DL ontology has to be both consistent and coherent (i.e, absence of unsatisfiable concepts), DL revision has to resolve not only inconsistency but also incoherence. Finally, DL contraction and revision should lead to tractable instantiations and at the same time respecting the mathematical properties of AGM contraction and revision.

Forgetting: To support the reuse and combination of ontologies in Semantic Web applications, it is often necessary to obtain smaller ontologies from existing larger ontologies. In particular, applications may require the omission of many terms, e.g., concept names and role names, from an ontology. However, the task of omitting terms from an ontology is challenging because the omission of some terms may affect the relationships between the remaining terms in complex ways. The technique of forgetting provides an effective way for extracting modules from a large ontology.

The Rise of Approximate Ontology Reasoning: Is It Mainstream Yet?

Jeff Z. Pan

University of Aberdeen, Aberdeen, UK

The last five years have seen a growing volume and complexity of ontologies and large-scale linked data available,[1] which present a pressing need for efficient and scalable ontology reasoning services. Major technology vendors are starting to embrace semantic technologies by supporting new standards and integrating with state of the art semantic tools. For example, in their new release 12.1, Oracle Spatial and Graph supports both RDF and OWL2-EL natively,[2] and integrates with an OWL2-DL reasoner (TrOWL) via OWL-DBC.[3]

The second version of the ontology standard OWL (Web Ontology Language) offers a family of ontology languages, including OWL2-DL, the most expressive decidable language in the family, and three tractable sub-languages of OWL2-DL, i.e. OWL2-EL, OWL2-QL and OWL2-RL. Such a two-layered language architecture allows approximate reasoning for OWL2-DL, by approximating OWL2-DL ontologies to those in its tractable sub-languages, so as to exploit efficient and scalable reasoners of the sublanguages. This is motivated by the fact that real-world knowledge and data are hardly perfect or completely digitalised. State of the art approximate reasoners, such as the TrOWL reasoner, can out-perform sound and complete reasoners in time constrained sound-and-complete reasoner competitions, such as the ORE competitions.

In this talk, we will look into how and why approximate reasoners work. Indeed, approximation approaches bring a new dimension – quality, in terms of completeness and soundness of reasoning, into the trade-off between expressiveness and performance, attempting to strike a balance among the three. Once we start to consider such a third dimension, many interesting questions follows: What are the typical approximate reasoning approaches? Should we approximate the input ontology or the input query? Are approximations always finite and unique? Given an ontology and some target queries, are there any best approximations? Why do some approximate reasoning algorithms lose many reasoning results, while others can enjoy high recall? Are approximate reasoning algorithms relevant to optimisations for sound and complete reasoners? Can we extend approximate reasoning algorithm with some post-processing to ensure soundness and completeness? I will discuss many of these questions, in the context of the TrOWL reasoner and related work, and share some thoughts on what approximate reasoning might bring in the near future.

---

1 http://lod-cloud.net/state/.

2 http://download.oracle.com/otndocs/tech/semantic_web/pdf/semtech_datamining_v8.pdf.

3 http://download.oracle.com/otndocs/tech/semantic_web/pdf/trowl_integration_with_orasag.pdf.

Contents

Ontology and Data Management

How Can Reasoner Performance of ABox Intensive Ontologies Be Predicted?

Isa Guclu[1], Carlos Bobed[2], Jeff Z. Pan[1(✉)], Martin J. Kollingbaum[1], and Yuan-Fang Li[3]

[1] University of Aberdeen, Aberdeen, UK
jeff.z.pan@abdn.ac.uk
[2] University of Zaragoza, Zaragoza, Spain
[3] Monash University, Melbourne, Australia

Abstract. Reasoner performance prediction of ontologies in OWL 2 language has been studied so far from different dimensions. One key aspect of these studies has been the prediction of how much time a particular task for a given ontology will consume. Several approaches have adopted different machine learning techniques to predict time consumption of ontologies already. However, these studies focused on capturing general aspects of the ontologies (i.e., mainly the complexity of their TBoxes), while paying little attention to ABox intensive ontologies. To address this issue, in this paper, we propose to improve the representativeness of ontology metrics by developing new metrics which focus on the ABox features of ontologies. Our experiments show that the proposed metrics contribute to overall prediction accuracy for all ontologies in general without causing side-effects.

Keywords: Semantic web · Ontology reasoning · Prediction · Random forests · Knowledge graph · Practical reasoning

1 Introduction

Semantic technologies have been utilized in various application domains for assisting knowledge management thus far, e.g., data management [13] and software engineering [17]. The worst case complexity 2NEXPTIME-complete [6] of OWL 2 DL, the most expressive profile of OWL 2, constitutes a bottleneck for performance critical environments. Empirical studies show that even the EL profile, with PTIME-complete complexity and less expressiveness, can become too time-consuming [4,11]. To have a scalable environment for implementing semantic technologies, an accurate prediction of ontology time consumption which will guide us about the feasibility of ontology reasoning is needed.

There have been several studies regarding the performance prediction of ontologies. Kang et al. [10] investigated the *hardness category* (categories according to reasoning time) for reasoner-ontology pairs and used machine learning techniques to make a prediction. Using FaCT++ [25], HermiT [5], Pellet [23], and TrOWL [16,18,20,24], they reached high accuracy in terms of hardness category, but not reasoning time.

Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 3–14, 2016.
DOI: 10.1007/978-3-319-50112-3_1

In another study, Kang et al. [12] investigated regression techniques to predict reasoning time. They made experiments using reasoners FaCT++, HermiT, JFact, MORe [21], Pellet and TrOWL with *their syntactic metrics* as features. These metrics are generally effective when there is a balance between TBox axioms and ABox axioms. Our experiments show that accuracy of these metrics decreases as ABox axiom sizes increase. As ABox constitutes the data in an ontology [1,8,27], where TBox constitutes the schema, an approach that can capture the changes in the ABox in a more detailed way is needed to make accurate overall predictions. As observed by Bobed et al. [2], there is an interest in using semantic technologies in mobile devices. In such scenarios, TBox axioms are expected to be more static and the ABox axioms (data) tend to be more frequently changing which necessitates high accuracy in ABox performance prediction. In this paper, we aim to investigate what metrics could help further improve reasoner predictions of ABox intensive ontologies.

Our main contributions can be summarized as follows.

1. We propose an initial set of metrics which estimate the complexity of the TBox concepts and propagates it into the estimated complexity of the ABox.
2. We show that our proposed new metrics for representing the structure of ontologies from the ABox perspective indicate a good research path to improve the accuracy of predicting time consumption of ontology reasoning.

The rest of the paper is as follows. In Sect. 2, we present some related works to place our proposal. In Sect. 3, we define the metrics that we propose in our ongoing work. In Sects. 4 and 5, we explain our experimental settings and the achieved results, respectively. Finally, in Sect. 6, we make some conclusions and draw some future work.

2 Related Work and Background

Ontology metrics, which are features of the ontology expressed numerically or categorically to represent the structure of an ontology, have been effectively utilised in analysing the complexity [28], energy consumption on mobile devices [7], cohesion [26], quality [3] and population task [15] of ontology reasoning.

Kang et al. [10] proposed a set of metrics in 2012 to classify raw reasoning times of ontologies into five large categories: [0 s.–100 ms.], (100 ms.–1 s.], (1 s.–10 s.], (10 s.–100 s.] and (100 s.–∞). Despite the high accuracy of prediction, over an 80%, this approach does not provide actual reasoning time but time categories, which may become obsolete or meaningless according to needs of implementation.

In 2014, Kang et al. [12] extended their work and proposed a new set of metrics to predict actual reasoning time by developing regression models. They extended the previous 27 metrics [10,28] and developed a set of 91 metrics that include 24 ontology-level (ONT) metrics, 15 class-level (CLS) metrics, 22 anonymous class expression (ACE) metrics, and 30 property definition and axiom (PRO) metrics.

While a high number of metrics are usually proposed by researchers, Sazonau et al. [22] proposed instead a local method which involved selecting a *suitable*, small subset of the ontology, and making extrapolation to predict total time consumption of ontology reasoning using the data coming from the processing of such small subset. To do so, they used *Principal Component Analysis* (PCA) [9]. In their experiments, Sazonau et al. [22] observed that 57 of the studied features can be replaced by just one or two features. Using a sample of size of a 10 % of the ontology for reasoning, they argue that they reached good predictions with simple extrapolations. They list advantages of their method as: (1) more accurate performance predictions, (2) not relying on an ontology corpus, (3) not being biased by this corpus, and (4) being able to obtain information about reasoner's behaviour of linear/nonlinear predictability on the corpus. A remarkable contribution of this approach is that it saves researchers from the difficulty/risk of selecting an unbiased corpus [14], which is very difficult while checking the validity of the prediction model and accuracy of the prediction. However, making reasoning with the 10 % of an ontology may not always be applicable especially when the ontology requires high reasoning times.

3 Our Approach

Our claim is that increasing the expressivity of ontology metrics directly helps increasing the accuracy of all the above studies, and enables new studies that target a more feasible implementation environment for semantic technologies.

Part of 91 metrics proposed by Kang et al. [12] are obtained by transforming an ontology into a graph which grasps the relationship between of ABox and TBox axioms. However, their approach calculates the effect of ABox axioms up to a certain extent. It is apparent that connected ABox axioms are more prone to cause more inferences than disconnected ABox axioms. These connections can trigger reasoning time enormously when they come along with a complex TBox. In our work, we have observed that the models trained with this set of 91 metrics begin to lose accuracy in predicting time consumption of ontologies as the ratio between the amount of ABox axioms and TBox axioms increases.

Thus, we propose to include the propagation of the complexity of the TBox into the ABox. To do so, we extend this set of metrics with our 15 *Class Complexity Assertions (CCA)* metrics, which can contribute to performance prediction of ontologies especially when we deal with ontologies which are ABox intensive (i.e., they exhibit a high ABox/TBox ratio). Experiment results and source codes are accessible[1].

3.1 Class Complexity Assertions Metrics

As above mentioned, to capture the interactions between the complexity of the different elements of the TBox and the individuals asserted in the ABox, we have

[1] http://sid.cps.unizar.es/projects/OWL2Predictions/JIST16/.

developed an initial set of features which aim at propagating the complexity of each of the concept expressions in the ontology to the ABox, as well as improving the richness of the TBox metrics.

Thus, let be $N_{CE} = \{CE_i \mid CE_i \in O\}$ with CE any concept expression appearing in any of the logical axioms of the ontology O. For each CE, we estimate its complexity as follows:

$$comp(CE_i) = \frac{height(CE_i) + sigSize(CE_i) + const(CE_i)}{3}$$

with $height(CE_i)$ being the height of the expression as a parsing tree, $sigSize(CE_i)$ being the number of different atomic class names that appear in the expression, and $const(CE_i)$ begin the number of class constructors participating in the class expression.

With this estimation for each CE_i, we calculate the following metrics:

- *TBoxSize:* The count of TBox axioms obtained from OWLAPI.
- *ABoxSize:* The count of ABox axioms obtained from OWLAPI.
- *ABoxTBoxRatio:* The ratio of ABox axioms to TBox axioms.
- *TCCA:* Total amount of estimated complexity of the ontology O (i.e., the class expressions in N_{CE}).

$$TCCA = \sum_{CE_i \in N_{CE}} comp(CE_i)$$

- *AVG_CCA:* Mean estimated complexity of the class expressions in N_{CE}.

$$AVG\_CCA = \frac{TCCA}{|N_{CE}|}$$

- *MAX_CCA:* Maximum estimated complexity of the class expressions in N_{CE}.
- *MIN_CCA:* Minimum estimated complexity of the class expressions in N_{CE}.
- *STD_CCA:* Standard deviation of complexity of the class expressions in N_{CE}.
- *ENT_CCA:* Entropy of the complexity distribution of N_{CE}.

To propagate the complexity of each concept expression to the ABox, we use each of the class assertions as a witness of the complexity of a class expression within the ontology. Then, we aggregate such values to capture what we name the witnessed complexity of the ABox. So, let $Ind_{N_{CE_i}} = \{a \mid a \in Ind(O) \wedge CE_i(a) \in O\}$ the individuals that are explicitly asserted to belong to CE_i. Thus, we define:

- *TWCCA:* Total witnessed complexity of the ABox, which is calculated summing all the products of the estimated complexities of the concept expressions with their *witness individuals.*

$$TWCCA = \sum_{CE_i \in N_{CE}} comp(CE_i) * |Ind_{N_{CE_i}}|$$

– *AVG_WCCA:* Mean witnessed complexity of the ABox of the concept expressions in O.
$$AVG\_WCCA = \frac{TWCCA}{|N_{CE}|}$$
– *MAX_WCCA:* Maximum witnessed complexity of a concept expression in O.
– *MIN_WCCA:* Minimum witnessed complexity of a concept expression in O.
– *STD_WCCA:* Standard deviation of witnessed complexity of the concept expressions in O.
– *ENT_WCCA:* Entropy of the witnessed complexity distribution of the concept expressions in O.

Note that we apply a Laplace smoothing[2] to include also into the metrics the concept expressions which appear in the ontology but do not have any explicit individual assertion.

4 Experimental Setup

4.1 Evaluation Metrics

R^2, $MAPE$ and $RMSE$ are referred to decide whether our regression model is valid for describing the relation between our metrics and the predictions made by the model. The coefficient of determination (R^2) is a crucial output of regression analysis, indicating to what extent the dependent variable is predictable. For example, a value 0.91 for R^2 means that 91% percent of the variance in Y is predictable from X. Let $y(t)$ be the observed value of y in second t, $\hat{y}(t)$ be the predicted value for y in second t, and $\bar{y}$ be the mean of the observed values, then:

$$R^2 = \frac{\sum_t (\hat{y}(t) - \bar{y})^2}{\sum_t (y(t) - \bar{y})^2} \tag{1}$$

The *Mean Absolute Percentage Error (MAPE)* is a measure of prediction accuracy of a prediction method in statistics that is used to expresses accuracy as a percentage. For calculating the *MAPE* of our prediction model, we will divide the difference of observed and predicted values, divide this by the observed values, and get the average of all observations in the scope. Related to this definition, we define the *Mean Absolute Accuracy Percentage (MAAP)* of our prediction model which is given by (1 - *MAPE*). In this paper, we will refer to *MAAP* to explain the accuracy of a model.

$$MAPE = 100. \frac{\sum\limits_{t=1}^{n} \frac{|\hat{y}(t) - y(t)|}{y(t)}}{n} \tag{2}$$

$$MAAP = 1 - MAPE \tag{3}$$

[2] Adapted from Natural Language Processing, basically, it consists in adding 1 to all the witnessed values of the concept expressions in the ontology.

Finally, the Root Mean Squared Error ($RMSE$) is the square root of the mean/average of the square of all of the error. $RMSE$ represents the sample standard deviation of the differences between observed and predicted values.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} (y(t) - \hat{y})^2}{n}} \tag{4}$$

4.2 Data Collection

Reasoner: We have used TrOWL 1.5 for testing EL ontologies as the reasoner to be tested. We deployed *ABox Materialization Task* with TrOWL as our experimental task. In our experiments, we implemented ABox materialization with one thread. We could benefit from parallelization in ABox materialization and it would improve the performance [19] to some extent. As RAM I/O becomes the bottleneck because of the limited bandwidth [19] of the RAM when many worker threads compete for RAM access and this would cause some side-effects in measuring the execution time, we preferred to analyse the performance prediction aspect parallel ABox materialization as future work.

Ontologies: We define an ontology as *ABox-intensive* if the count of ABox axioms in such an ontology is at least 10 times the count of TBox axioms. We made our experiments using ontologies in ORE2014 Reasoner Competition Dataset[3]. From 16,555 ontologies, we have filtered 74 ontologies in EL profile which have the ABox/TBox ratio of at least 10, and created artificial 2779 ABox-intensive ontologies[4] from these ABox-intensive ontologies *randomly* as follows: our method uses the TBox of the original ontology and creates a new ontology using different random subsets of the ABox axioms of the original ontology.

Prediction Model Construction: For predicting the time consumption of ontologies, a random forest based regression model is implemented, using the metrics (predictor variables). Standard 10-fold cross-validation is performed to ensure the generalizability of the model.

5 Results and Evaluation

In our study, we investigated the reasoning performance of a reasoner and ontology characteristics represented by available metrics and our new metrics (CCA). While developing our new metrics, we aimed at capturing the complexity of ontologies without losing accuracy when ABox sizes changed. Our claim is that developing high-quality metrics will increase the accuracy of the prediction

[3] https://zenodo.org/record/10791.

[4] You can find the code of the OntologyChopper at http://sid.cps.unizar.es/projects/OWL2Predictions/JIST16/.

model. Our goal is to make prediction models that can perform on any ontology with high stability using metrics that can represent the ontology with high expressivity.

To ensure the quality of the dataset, we created 2779 artificial ontologies from ORE2014 dataset. To avoid a biased corpus, which would result in misleading generalizations, we generated ontologies with random selection of ABox axioms. We did not put any threshold to cut the experiment, as we wanted to include every result of the dataset without missing any point that could be expressed by the dataset. We believe that wide range of ABox/TBox ratio will help increase the diversity in ontologies.

While working with the quality of the dataset, quality of the feature selection should also be taken into consideration. Inspired from the consistent high accuracy of the Random Forest based regression models in the study of Kang et al. [12], we adopted the same approach. Instead of categorising the time periods, we preferred metrics to give prediction results of time consumption in nanoseconds. We had specified R^2, $MAPE$, and $RMSE$ values as our performance criteria for prediction accuracy.

5.1 Combining 91 Metrics with CCA

In our first set of experiments, we combined 91 metrics with CCA metrics to train the model. We were expecting new CCA metrics would increase the accuracy of prediction, as it contained metrics that would better express the complexity of ABox axioms with TBox axioms. The results obtained in the cross validation procedure for the performance criteria can be seen in Table 1, and in Fig. 1, the $MAPE$ values obtained are visualized.

When we look at the R^2 values, which is indicative to which extent the dependent variable is predictable, we see that both available 91 metrics and combined metrics of 91 metrics and CCA metrics have the values between 97% and 98%. The difference of $RMSE$ values is ≈ 2.5 s. The values of $MAPE$ also show a difference of $\approx 2\%$.

Although this absolute value of a $\approx 2\%$ accuracy increase seems very small, it is a relative improvement of 11% with the first version of our transference metrics, which encourages us to continue to work in this direction improving and extending the definition of such kind of metrics.

Table 1. Contribution of CCA metrics to accuracy of prediction.

	91 Metrics	CCA + 91 Metrics
R^2	0.97607	0.97856
$MAPE$	23.58%	21.10%
$RMSE$	41.4 s	39.1 s

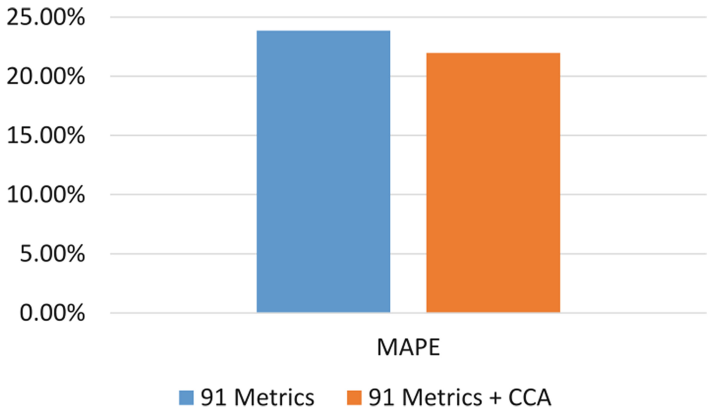


Fig. 1. Change in MAPE when new metrics are added to the prediction model.

5.2 Using CCA Metrics Instead of *some* ABox metrics in 91 Metrics v.1

We searched for the metrics in 91 metrics which are more sensitive to ABox axiom changes. By randomly adding ABox axioms to ontologies, we observed that the change in ABox axioms is highly correlated with some of 91 metrics, i.e., "SOV, CYC, RHLC, IHR, IIR, ITR, IND, aCID, mCID, tCID" [10,28].

In our second set of experiments, we removed the metrics "RHLC, IHR, IIR, ITR, IND, aCID, mCID, tCID" from 91 metrics and replaced with CCA metrics to train the model. The results obtained in the cross validation procedure for the performance criteria can be seen in Table 2, and in Fig. 2, the $MAPE$ values obtained are visualized.

When we look at the R^2 values, we see that both models have the values between 97% and 98%. The difference of $RMSE$ values is $\approx$ 1 s. The values of $MAPE$ show a difference of $\approx$ 4.5%.

The relative improvement in decreasing the average error rate of 91 metrics is about 20% by replacing some of ABox related metrics in 91 metrics with our CCA metrics.

Table 2. Contribution of CCA metrics when replaced with some ABox related metrics in 91 metrics (v.1).

	91 Metrics	CCA + 91 Metrics (v.1)
R^2	0.97607	0.97654
$MAPE$	23.58 %	19.03 %
$RMSE$	41.4 s	41.0 s

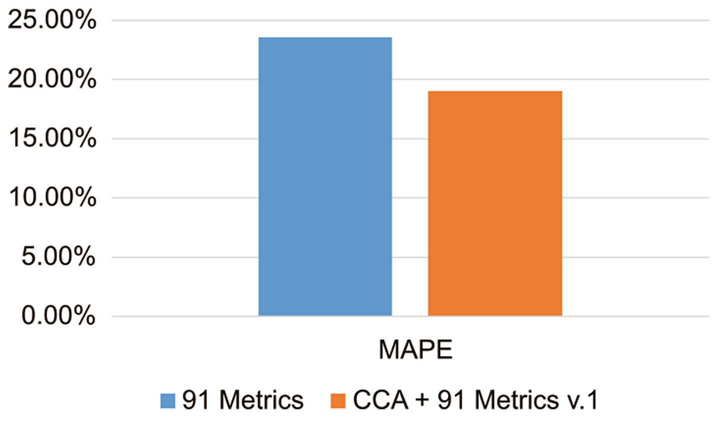


Fig. 2. Change in MAPE when some ABox related metrics in 91 metrics are replaced with CCA metrics (v.1) to the prediction model.

5.3 Using CCA Metrics Instead of *some* ABox metrics in 91 Metrics v.2

In our third case, we removed the metric "CYC" in addition to "RHLC, IHR, IIR, ITR, IND, aCID, mCID, tCID" from 91 metrics and replaced with CCA metrics to train the model. The results obtained in the cross validation procedure for the performance criteria can be seen in Table 3, and in Fig. 3, the $MAPE$ values obtained are visualized.

When we look at the R^2 values, we see that both models have the values between 97% and 98%. The value of $RMSE$ worsened here. The values of $MAPE$ show a difference of $\approx$ 4%, which is a general improvement but worst than the previous case.

The relative improvement in decreasing the average error rate of 91 metrics is about 18% by replacing some of ABox related metrics in 91 metrics with our CCA metrics.

Table 3. Contribution of CCA metrics when replaced with some ABox related metrics in 91 metrics (v.2).

	91 Metrics	CCA + 91 Metrics (v.2)
R^2	0.97607	0.97449
$MAPE$	23.58%	19.25%
$RMSE$	41.4 s	42.7 s

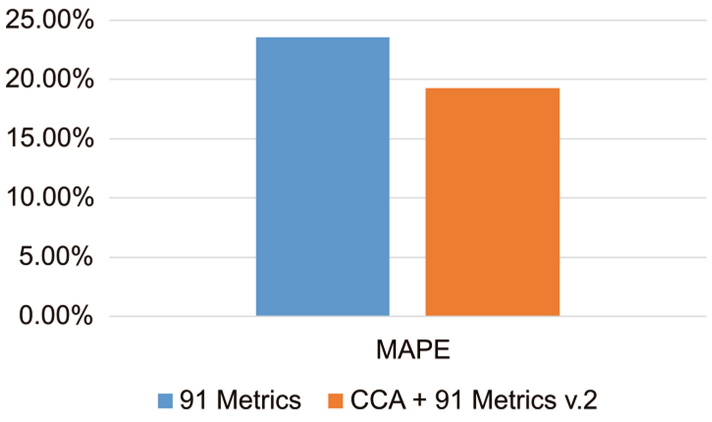


Fig. 3. Change in MAPE when some ABox related metrics in 91 metrics are replaced with CCA metrics (v.2) to the prediction model.

5.4 Evaluation

In our work, we have analysed available metrics and investigated how to bring expressivity of metrics further by developing new metrics to represent ABox axioms (and its interaction with TBox axioms) aspect of ontologies.

According to initial experiments, which compare 91 metrics with combination of 91 metrics and CCA metrics, we observe that adding CCA metrics increases the accuracy of prediction $\approx 2.5\%$ and relatively decreasing average error rate $\approx 11\%$.

When we replaced some of the metrics (RHLC, IHR, IIR, ITR, IND, aCID, mCID, tCID) in 91 metrics with CCA metrics, we observe the accuracy of prediction increase $\approx 4.5\%$ and relatively, average error rate decrease $\approx 20\%$.

In our third case, we also removed the metric (CYC) in 91 metrics and saw that there was again higher accuracy in prediction but it wasn't as good as the previous model.

Seeing the results above, we conclude that CCA metrics contributes to prediction of ABox-intensive ontologies in our preliminary work. Available metrics (91 metrics proposed by Kang et al. [12]) could grasp the complexity of ontologies to some extent. ABox materialization necessitates new metrics that will represent the interaction of ABox axioms with TBox axioms taking its complexity into account. The weight of ABox axioms in an ontology and their interactions can cause consuming more execution time than expected if their complexity is ignored. We propose our CCA metrics to measure the effect of ABox complexity in performance prediction of ontology reasoning and we want to improve these metrics to measure this aspect of ontologies more effectively. We believe that our study will lead to metrics that are generalizable regardless of the weight of TBox and ABox axioms.

6 Conclusion

Performance prediction of ontology reasoning is a very interesting and challenging topic. In this work, we have started to focus on the performance prediction of ABox-intensive ontologies. We proposed 15 new metrics by extending previous work of Kang et al. [12]. Preliminary results with adding these 15 metrics show slight increase ($\approx 4.5\%$) in the prediction accuracy. And, these results even at the early stages of our research encourage us to continue in this direction. We believe that awareness of the ABox axiom ratio in ontologies and bringing a solution to this change will increase the effectiveness and validity of a performance prediction model.

As future work, firstly, we plan to work on better representation of the interactions between ABox axioms and TBox axioms by developing new metrics. Secondly, we will make experiments with more reasoners on different ontologies that will help understanding the interaction of ABox axioms with TBox axioms in a broader sense. Thirdly, we will use different prediction mechanisms to leverage the contribution of these metrics.

Acknowledgments. This work was partially supported by the EC Marie Curie K-Drive project (286348), the CICYT project (TIN2013-46238-C4-4-R) and the DGA-FSE project.

References

1. Fokoue, A., Meneguzzi, F., Sensoy, M., Pan, J.Z.: Querying linked ontological data through distributed summarization. In: Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI2012) (2011)
2. Bobed, C., Yus, R., Bobillo, F., Mena, E.: Semantic reasoning on mobile devices: do androids dream of efficient reasoners? J. Web Semant. **35**(4), 167–183 (2015). ISSN 1570–8268, https://dx.doi.org/10.1016/j.websem.2015.09.002
3. Burton-Jones, A., Storey, V.C., Sugumaran, V., Ahluwalia, P.: A semiotic metrics suite for assessing the quality of ontologies. Data Knowl. Eng. **55**, 84–102 (2005)
4. Dentler, K., Cornet, R., ten Teije, A., de Keizer, N.: Comparison of reasoners for large ontologies in the OWL 2 EL profile. Semant. Web **2**, 71–87 (2011)
5. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: an OWL 2 reasoner. J. Autom. Reasoning **53**, 245–269 (2014)
6. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: the next step for OWL. J. Web Sem. **6**, 309–322 (2008)
7. Guclu, I., Li, Y.-F., Pan, J.Z., Kollingbaum, M.J.: Predicting energy consumption of ontology reasoning over mobile devices. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (eds.) ISWC 2016. LNCS, vol. 9981, pp. 289–304. Springer, Heidelberg (2016). doi:10.1007/978-3-319-46523-4_18
8. Hogan, A., Pan, J.Z., Polleres, A., Ren, Y.: Scalable OWL 2 reasoning for linked data. In: Polleres, A., d'Amato, C., Arenas, M., Handschuh, S., Kroner, P., Ossowski, S., Patel-Schneider, P. (eds.) Reasoning Web 2011. LNCS, vol. 6848, pp. 250–325. Springer, Heidelberg (2011). doi:10.1007/978-3-642-23032-5_5
9. Jolliffe, I.: Principal Component Analysis. Wiley StatsRef: Statistics Reference Online (2002)

10. Kang, Y.-B., Li, Y.-F., Krishnaswamy, S.: Predicting reasoning performance using ontology metrics. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012. LNCS, vol. 7649, pp. 198–214. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35176-1_13
11. Kang, Y.-B., Li, Y.-F., Krishnaswamy, S.: A rigorous characterization of classification performance - a tale of four reasoners. In: ORE (2012)
12. Kang, Y.-B., Pan, J.Z., Krishnaswamy, S., Sawangphol, W., Li, Y.-F.: How long will it take? Accurate prediction of ontology reasoning performance. In: AAAI (2014)
13. Li, Y.-F., Kennedy, G., Ngoran, F., Wu, P., Hunter, J.: An ontology-centric architecture for extensible scientific data management systems. Future Gener. Comput. Syst. **29**, 641–653 (2013)
14. Matentzoglu, N., Bail, S., Parsia, B.: A corpus of OWL DL ontologies. In: Proceedings DL13 (2013)
15. Maynard, D., Peters, W., Li, Y.: Metrics for evaluation of ontology-based information extraction (2006)
16. Pan, J.Z., Ren, Y., Zhao, Y.: Tractable approximate deduction for OWL. Artificial Intelligence **235**, 95–155
17. Pan, J.Z., Staab, S., Amann, U., Ebert, J., Zhao, Y.: Ontology-Driven Software Development. Springer Publishing Company, Incorporated, Ontology-Driven Software (2012)
18. Pan, J.Z., Thomas, E., Ren, Y., Taylor., S.: Tractable fuzzy and crisp reasoning in ontology applications. In: IEEE Computational Intelligence Magazine (2012)
19. Ren, Y., Pan, J.Z., Lee, K.: Optimising parallel ABox reasoning of EL ontologies. In: Description Logics (2012)
20. Ren, Y., Pan, J.Z., Zhao, Y.: Soundness preserving approximation for TBox reasoning. In: AAAI (2010)
21. Romero, A.A., Grau, B.C., Horrocks, I.: More: modular combination of OWL reasoners for ontology classification. In: SEMWEB (2012)
22. Sazonau, V., Sattler, U., Brown, G.: Predicting performance of OWL reasoners: locally or globally? In: Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20–24, 2014 (2014)
23. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a practical OWL-DL reasoner. J. Web Sem. **5**, 51–53 (2007)
24. Thomas, E., Pan, J.Z., Ren, Y.: TrOWL: tractable OWL 2 reasoning infrastructure. In: Aroyo, L., Antoniou, G., Hyvönen, E., Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6089, pp. 431–435. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13489-0_38
25. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: system description. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006). doi:10.1007/11814771_26
26. Yao, H., Orme, A.M., Etzkorn, L.: Cohesion metrics for ontology design and application. J. Comput. Sci. **1**, 107–113 (2005)
27. Yuan Ren, J.Z.P., Lee, K.: Optimising parallel ABox reasoning of el ontologies. In: Proceedings of the 25th International Workshop on Description Logics (DL2012) (2012)
28. Zhang, H., Li, Y.-F., Tan, H.B.K.: Measuring design complexity of semantic web ontologies. J. Syst. Softw. **83**, 803–814 (2010)

Inquiry into RDF and OWL Semantics

Seiji Koide[1,2(✉)] and Hideaki Takeda[1]

[1] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
koide@ontolonomy.co.jp

[2] Ontolonomy, LLC., 3-76-3-J901, Mutsukawa, Minami-ku, Yokohama, Japan

Abstract. The purpose of this paper is to present the higher order formalization of RDF and OWL with setting up ontological meta-modeling criteria through the discussion of Russell's *Ramified Type Theory*, which was developed in order to solve *Russell Paradox* appeared at the last stage in the history of set theory. This paper briefly summarize some of set theories, and reviews the RDF and OWL Semantics with higher order classes from the view of Russell's *Principia Mathematica*. Then, a set of criteria is proposed for ontological meta-modeling. Several examples of meta-modeling, including sound ones and unsound ones, are discussed and some of solutions are demonstrated according to the meta-modeling criteria proposed.

Keywords: RDF semantics · OWL semantics · Set theory · Principia mathematica · KIF · Membership loop · Higher order class · Meta-modeling

1 Introduction

The OWL specifications has been split into two parts, Direct Semantics and RDF-based Semantics. The reason of this unhappy partition originates in the different formalizations between OWL DL and RDF. In order to match DL-based semantics to RDF-based semantics, the term of "comprehension conditions (principles)" is introduced into "OWL Semantics and Abstract Syntax, Sect. 5" (Patel-Schneider et al. 2004a) and "OWL 2 Web Ontology Language RDF-Based Semantics (Second Edition)" (Schneider 2014). In addition, "RDF-Compatible Model-Theoretic Semantics" (Patel-Schneider et al. 2004a) states that the *only-if* semantic conditions are necessary to prevent semantic paradoxes with the fourteen comprehension conditions, and OWL2 Appendix[1] mentions "formal inconsistency" instead of "paradox". However, these statements are caused by misunderstanding paradoxes in set theories, and the word "comprehension conditions" is still left in "OWL2 (Second edition)".

Granted that no choice but to introduce some postulates in order to make OWL semantics compatible to RDF semantics, this misunderstanding is partly

[1] http://www.w3.org/TR/2012/REC-owl2-rdf-based-semantics-20121211/#Appendix:_Comprehension_Conditions_.28Informative.29.

attributed to RDF semantics itself; indeed, there are not enough explanations in RDF Semantics (Hayes 2004) on the setup of preventing "membership loops". Hayes only claimed that the semantic model distinguishes both properties and classes, regarded as objects, from their extensions, so that this distinction prevents "membership loops". He also stated that the violation of the *axiom of foundation*, which is one of the axioms of standard set theories like Zermelo-Fraenkel (ZF) that forbids infinitely descending chains of membership, does not happen in RDF. However, the mechanism of preventing membership loops is still obscure for readers of RDF Semantics.

In this paper, at Sect. 2, we describe Russell paradox that has roots in Cantor's naive set theory, and summarize the history of set theories for the resolution of the paradox. At Sect. 3, we review RDF Semantics and OWL Semantics with higher order classes. Then, at Sect. 4, we propose a set of criteria for higher order classes and ontological meta-modeling. Those criteria are actually derived from the axioms and principles introduced in *Ramified Type Theory* for the resolution of Russell paradox in Principia Mathematica (PM, Vol.1). At Sect. 5, several examples of meta-modeling, including sound ones and unsound ones, are discussed, and some of solutions are demonstrated according to the meta-modeling criteria presented here.

We call the OWL Full with these criteria *Restricted OWL Full*. Thus, ambiguous word "punning" by W3C is clearly fixed on the meta-modeling with higher order classes and it helps us to deeply understand the semantics and the inference mechanism of RDF/OWL systems that allow ontological meta-modeling.

2 History of Set Theory and Type Theory

The history of set theory is a history of coping with Russell paradox essentially contained by the mathematical concept of set. Russell paradox was resolved in two ways. One is *axiom of separation* by Zermelo and the other is *Ramified Type Theory* by Whitehead and Russell. In the current set theory, a set is discriminated from a class that cannot be a member of set[2]. It is simply stated by Bourbaki (Bourbaki 1966) that there are two sorts of relations, i.e., relations which can make sets and which cannot make sets. This section abstracts set theories by Cantor, Zermelo, type theories by Russell, and the resolution of Russell paradox in Knowledge Interchange Format (KIF 1994).

2.1 RDF Sematics and Sets of Objects

RDF semantics is built on the foundation of set theory as well as every other formal theory in mathematics and logics. In fact, Hayes invokes a set theory named Zermelo-Fraenkel in order to rationalize *membership loops* that do not cause any paradoxes.

[2] A class notion in set theories is different from one in ontology descriptions.

> When classes are introduced in RDFS, they may contain themselves. Such ‘membership loops’ might seem to violate the *axiom of foundation*, one of the axioms of standard (Zermelo-Fraenkel) set theory, which forbids infinitely descending chains of membership. However, the semantic model given here distinguishes properties and classes considered as objects from their extensions - the sets of object-value pairs which satisfy the property, or things that are ‘in’ the class - thereby allowing the extension of a property or class to contain the property or class itself without violating the axiom of foundation. (Hayes 2004)

Although the Recommendation claims that the semantic model in RDF semantics does not violate the axiom of foundation on membership loops, yet there are not enough discussions on why and how the membership loops do not violate the axiom of foundation. It only states that $x \neq \mathbb{C}\mathrm{EXT}(x)$ for an object x, where $\mathbb{C}\mathrm{EXT}(x)$ is the class extension of x, and $p \neq \mathbb{I}\mathrm{EXT}(p)$ for a property p, where $\mathbb{I}\mathrm{EXT}(p)$ is the extension of p.

This paper claims the rdfs:Resource and rdfs:Class are proper classes in sets that do not cause paradoxes on sets, because the extension of rdfs:Resource is a totality and the extension of rdfs:Class is also a totality. A totality is not regarded as a set in today’s theory. So, the rdfs:Resource and the rdfs:Class should be conceived to be a convention of referring the universal class concept and the universal metaclass concept respectively in the universe of discourse.

2.2 Cantor’s Paradox and His Final Legacy

The history of set theories started with Georg Cantor. It is obvious that Cantor assumed members of sets are countable objects and a set is a collection of objects (Cantor 1895). Yet, he actually did not mention about objects, and he clarified the concept of natural numbers based on a (naive) set theory. However, Cantor became to know a paradox (called Cantor’s paradox) contained by his own set theory in case of handling the totality of infinite sets including sets of sets[3], and he noticed it in his letter to Dedekind[4] (Cantor 1967). Aczel wrote,

> Cantor’s final legacy, beyond the discovery of the transfinite numbers and the continuum hypothesis, was his realization that there could be no set containing everything ... since, given any set, there is a larger set – its set of subsets, the power set. (Aczel 2000)

Today, we know the power set $\wp(a) = \{x \mid x \subseteq a\}$ is too powerful for making sets of infinite cardinality, whereby Cantor discovered the transfinite ordinal numbers, but he also opened a door to lead paradoxes involved by set theories.

[3] The universe of natural numbers is factually defined as sets of sets that include the empty set as number zero and powersets of sets as number successors.

[4] In the letter, it is stated that “The system Ω of all numbers is an inconsistent, absolutely infinite multiplicity.”

2.3 Comprehension Principle and Russell Paradox

The followings are some of axioms in naive set theory (Boolos 1971).

$$\forall x \forall y \forall z[(z \in x \Leftrightarrow z \in y) \Rightarrow x = y] \tag{1}$$
$$\exists y \forall x[x \in y \Leftrightarrow x \neq x] \tag{2}$$
$$\forall z \forall w \exists y \forall x[x \in y \Leftrightarrow (x = z \vee x = w)] \tag{3}$$
$$\exists y \forall x \exists w[x \in y \Leftrightarrow (x \in w \wedge w \in z)] \tag{4}$$
$$\exists y \forall x[x \in y \Leftrightarrow x = x] \tag{5}$$

The first one is called *axiom of extensionality* (1), and in order respectively, *empty set* (2), *pairing* (3), *union* (4), and *universal* (5). Especially, the empty set $\emptyset$ is defined by the second formula, such that $\emptyset \equiv \{x \mid x \neq x\}$.

The intensional definition of sets such that $\{x \mid \varphi(x)\}$, where $\varphi(x)$ is any formula, is called *comprehension principle* (Kamareddine et al. 2004).

Definition 1 *(Comprehension Principle).*

$$\exists y \forall x[x \in y \Leftrightarrow \varphi(x)] \qquad \text{where } y \text{ is not free in } \varphi(x) \tag{6}$$

It was once conceived that the *comprehension principle* was very natural for intensionally defining a set because any (unary) predicate could be applied to a given object in order to determine the membership of an object to a set that holds a property featured by the formula under the law of *excluded middle*; namely, we can determine for any objects whether an object belongs to the set or not. Meanwhile, Russell keenly pointed a paradox in sets. If we take $\varphi(x)$ to $x \notin x$, which intends to denote a set that does not include any membership loops, then it follows that $y = \{x \mid x \notin x\}$ or $\exists y \forall x[x \in y \Leftrightarrow x \notin x]$. Then, in case of instantiating an arbitrary x to y, we obtain a contradiction as follows.

Definition 2 *(Russell Paradox).*

$$\exists y[y \in y \Leftrightarrow y \notin y].$$

Russell revealed that the *comprehension principle* inevitably involves a paradox[5]. J. van Heijenoort stated,

> Bertrand Russell discovered what became known as the Russell paradox in June 1901 [...]. In the letter [to Frege], written more than a year later and hitherto unpublished, he communicates the paradox to Frege. The paradox shook the logicians' world, and the rumbles are still felt today.(From Frege to Gödel in van Heijenoort 1967)

[5] Exactly, Russell pointed the paradox in the expression of functions rather than sets, in the letter to Frege (van Heijenoort 1967).

2.4 Zermelo's Axiom of Separation

To avoid Russell paradox, Ernst Zermelo replaced the *comprehension principle* with the *axiom of separation (aussonderung)* (Kamareddine et al. 2004).

Definition 3 *(Axiom of Separation).*

$$\exists y \forall x [x \in y \Leftrightarrow x \in z \land \varphi(x)] \quad \text{where } y \text{ does not occur in } \varphi(x) \tag{7}$$

In this case, y must be a proper subset of a set z, of which members satisfy the formula. It may be phrased that the paradox is work-arounded by introducing z distinctively existing. Today, a set theory based on a series of axioms by Zermelo is called Zermelo-Fraenkel (ZF) set theory together with Fraenkel's *axiom of replacement*.

It is worthy to note that in some of modern computer languages a functionality to make a list whose elements are selected from another list so that they satisfy a specified condition is accidentally misnamed *list comprehension*[6]. We claim that this functionality in computer languages should be properly named "list-separation" or "list-selection" in order to avoid misunderstanding between *comprehension principle* in set theories and *list comprehension* in computer languages. As well, the term of "comprehension conditions (principles)", which is strongly associated with paradoxes in sets, in "OWL Semantics and Abstract Syntax, Sect. 5", should be renamed to "list conditions by selection" or something else, in order to avoid unfounded fear of paradoxes associated with *comprehension principle.*

2.5 NBG and Set Theory of KIF 3.0

The Knowledge Interchange Format (KIF) 3.0 theory proposed a special set theory that is based on von Neumann-Bernays-Gödel (NBG) set theory. The Chap. 7 of the reference manual (KIF 1994), which is titled "Sets", starts with the statement below.

> In many applications, it is helpful to talk about sets of objects as objects in their own right, e.g. to specify their cardinality, to talk about subset relationships, and so forth. The formalization of sets of simple objects is a simple matter; but, when we begin to talk about sets of sets, the job becomes difficult due to the threat of paradoxes (like Russell's hypothesized set of all sets that do not contain themselves).
> (Knowledge Interchange Format version 3.0 Reference Manual (KIF 1994))

It is obvious that the KIF Group were worried about involving the paradoxes on sets in the KIF specification. NBG allows us to make sets that include individuals and sets or classes of individuals. The axioms in NBG is roughly separated into two parts; one is for sets, in which a set is expressed using variables with lower letters, and the other is for classes, in which a class is expressed using variables

[6] For example, in Haskell, $[x \uparrow 2 \mid x \leftarrow [1..5]]$ produces $[1, 4, 9, 16, 25]$. (Hutton 2007).

with upper letters. Although both parts of axioms are composed of almost same forms, they are separated owing to the idea that classes are a different sort from sets of individuals in NBG.

KIF distinguishes bounded set (set) from unbounded set (proper class). A bounded set can be a member of a set, but an unbounded set cannot be a member of set. A bounded set is finite. A finite set is bounded, but an infinite set is unbounded. It is consistent to the standard set theories, ZF and NBG. KIF contains one more notion, that is, individuals.

> In KIF, a fundamental distinction is drawn between individuals and sets. A set is a collection of objects. An individual is any object that is not a set. A distinction is also drawn between objects that are bounded and those that are unbounded. This distinction is orthogonal to the distinction between individuals and sets. There are bounded individuals and unbounded individuals. There are bounded sets and unbounded sets. The fundamental relationship among these various types of entities is that of membership. Sets can have members, but individuals cannot. Bounded objects can be members of sets, but unbounded objects cannot. (It is this condition that allows us to avoid the traditional paradoxes of set theory.) (KIF 1994, Sect. 7.1 Basic Concepts)

Although we may have a curiosity what *unbounded individuals* are, there is no explanation in KIF what they are, and the study is beyond the scope of this paper. Russell paradox is described in KIF as follows.

> The paradoxes appear only when we try to define set primitives that are too powerful. We have defined the sentence '(member τ σ)' to be true in exactly those cases when the object denoted by τ is a member of the set denoted by σ, and we might consider defining the term '(setofall τ ϕ)' to mean simply the set of all objects denoted by τ for any assignment of the free variables of τ that satisfies ϕ. Unfortunately, these two definitions quickly lead to paradoxes.
> Let $\phi_{\nu/\tau}$ be the result of substituting term τ for all free occurrences of ν in sentence ϕ. Provided that τ is a term not containing any free variables captured in $\phi_{\nu/\tau}$, then the following schema follows from our informal definition. This schema is called the *principle of unrestricted set abstraction.*
>
> $$(\Leftrightarrow \text{(member } \tau \text{ (setofall } \nu\ \phi))\ \phi_{\nu/\tau})$$
>
> (KIF 1994, Sect. 7.4 Paradoxes)

Note that this form is equivalent to the unrestricted *comprehension principle* (6) and it causes Russell paradox.

Instead of this principle, Russell paradox is avoided in KIF by *restricted set abstraction*, in which a set is restricted to bounded sets.

> In the von-Neuman-Gödel-Bernays version of set theory, these paradoxes are avoided by replacing the *principle of unrestricted set abstraction* with the *principle of restricted set abstraction* given above.
>
> $$(\Leftrightarrow \text{(member } \tau \text{ (setofall } \nu\ \phi))\ \text{(and (bounded } \tau)\ \phi_{\nu/\tau}))$$ (*ibid.*)

Note that this form resembles the *axiom of separation* (7) in ZF. Here a bounded object '(bounded τ)' is used instead of the restriction '$x \in z$'.

KIF succeeded to eliminate paradoxes by the concept of bounded set of objects, whereas Russell's *Ramified Type Theory* is much suitable to explain how RDFS can avoid Russell paradox for the correct comprehension of the framework for meta-modeling[7].

2.6 Russell's Ramified Type Theory

Alfred N. Whitehead and Bertrand Russell developed the first type theory, *Ramified Type Theory*, in the epoch-making three-volume books, 'Principia Mathematica' (hereafter *PM* for short). They attempted to solve the Russel paradox together with other paradoxes by capturing them as variations of *vicious circle.*

> An analysis of the paradoxes to be avoided shows that they all result from a certain kind of vicious circle. (*PM, Vol.1*, Introduction, Chap. 2)

They emphasized that statements about "all propositions" are meaningless owing to the totality contained in the statements.

> [...] if we suppose the set to have a total, it will contain members which presuppose this total, then such a set cannot be a total. (*ibid.*)

Therefore, they, first of all, postulate the *vicious-circle principle* in order to avoid paradoxes caused by self-reference.

> The principle which enables us to avoid illegitimate totalities may be stated as follows: "Whatever involves *all* of a collection must not be one of the collection"; or, conversely: "If, provided a certain collection had a total, it would have members only definable in terms of that total, then the said collection has no total." We shall call this the "vicious-circle principle", [...] (*ibid.*)

Whitehead and Russell introduced the idea of propositional function, in which sentences may include variables for not only objects but also functions[8], and value assignments for all variables of objects and functions turn open sentences unambiguous propositions. Thus, functions are also applied to as logical expression. For example, Leibniz equality of $x = y$ is defined as $\forall f[f(x) \Leftrightarrow f(y)]$.

Ramified Types. Types in *PM* have a double hierarchy, that is, (simple) types and orders. The second hierarchy is introduced by regarding also the types of the variables that are bound by a quantifier. Kamareddine, et al. explained the reason using a propositional function $z(\,) \vee \neg z(\,)$, which can involve an arbitrary

[7] The set theory in NBG for individuals and sets can be regarded as a sort of first order logic, and then classes can be regarded as first order. However, RDFS can be regarded as much higher order logic as shown at Sect. 3.

[8] Predicates are functions that return truth value.

proposition for z, then $\forall z \uparrow (\,)[z(\,) \vee \neg z(\,)]$[9] quantifies over all propositions for z in the universe. We must distinguish a simple proposition $C(a)$ and quantified $\forall z \uparrow (\,)[z(\,) \vee \neg z(\,)]$. The former does not involve any self-reference but the latter may involve the self reference for z. This problem is solved by dividing types into *orders*. An order is simply a natural number that starts with 0, and in $\forall z \uparrow (\,)[z(\,) \vee \neg z(\,)]$ we must restrict the form by mentioning the order of propositions. Thus, propositional function of the form $\forall z \uparrow (\,)^n[z(\,) \vee \neg z(\,)]$ quantifies over only all propositions of order n, and this form has its own order $n + 1$.

Definition 4 *(Ramified Types). PM explained ramified types for only unary and binary functions. Kamareddine, et al. extended the definition to n-ary.* (Kamareddine et al. 2004).

1. 0^0 is a ramified type;
2. If $t_1^{a_1}, \ldots, t_n^{a_n}$ are ramified types and $a \in \mathbb{N} > max(a_1, \ldots, a_n)$, then $(t_1^{a_1}, \ldots, t_n^{a_n})^a$ is a ramified type (if $n = 0$ then take $a \geq 0$) ;
3. All ramified types can be constructed using rules 1 and 2.

Note that in $(t_1^{a_1}, \ldots, t_n^{a_n})^a$ we demand that $a > a_i$ for all i. Furthermore, Whitehead and Russell defined *predicative* condition on ramified types.

Definition 5 *(Predicative Types).*

$$(t_1^{a_1}, \ldots, t_n^{a_n})^a \qquad \text{where} \;\; a = 0 \;\; \text{if} \;\; n = 0, \; \text{else} \; a = 1 + max(a_1, \ldots, a_n)$$

The followings are some examples of predicative types.

- 0^0;
- $\left(0^0\right)^1$;
- $\left(\left(0^0\right)^1, \left(0^0\right)^1\right)^2$;
- $\left(\left(0^0\right)^1, \left(\left(0^0\right)^1, \left(0^0\right)^1\right)^2\right)^3$.

The above expressions of ramified types are also expressed as follows by Stevens (Stevens 2003) using function symbol F, G, H, and I, and their arguments.

- ${}^0x^0$;
- ${}^1F^{(0/0)}\left({}^0x^0\right)$;
- ${}^2H^{(1/(0/0),1/(0/0))}\left({}^1F^{(0/0)}\left({}^0x^0\right), {}^1G^{(0/0)}\left({}^0y^0\right)\right)$;
- ${}^3I^{(1/(0/0),2/(1/(0/0),1/(0/0)))}\left({}^1F^{(0/0)}\left({}^0x^0\right), {}^2H^{(1/(0/0),1/(0/0))}\right)$.

[9] In this paper '$\uparrow$' is used to indicate the type of variable instead of colon that is usually used in type theory, as a colon is confusing with the notation for namespace in the syntax of Semantic Web.

Here the first 0 in (0/0) stands for the order as argument and the last 0 stands for the type as argument at the individual level. 1 in 1/(0/0) stands for the order as first order as argument. The prefix number of function stands for orders of the form.

Suggested by ramified types, we can put orders to classes on RDF semantics with interpreting a class name as unary predicates in predicate calculus or propositional functions in *PM*. In the next section, we attempt to formalize RDF semantics taking into account of orders in Ramified Type Theory for classes, and claims that RDFS may avoid Russell paradox.

3 Formalization of RDF/OWL Semantics Based on Higher Order Types

3.1 Preliminary Explanations of Notations, Denotations, and Universe of Discourse

Notation. In this formulation, $\mathbb{R}$ stands for the universe of discourse, $\mathcal{P}$ stands for a finite set of logical predicate symbols, $\mathcal{F}$ stands for a finite set of functional symbols, and $\mathcal{V}$ stands for a countable set of vocabularies. Every sentence in this formulation is a triple like $\langle s, p, o \rangle$ composed of words in a vocabulary in $\mathcal{V}$.

Interpretation $\mathcal{I}$ is a mapping from a set of triples and vocabularies $\mathcal{V}$ into the universe of discourse $\mathbb{R}$. Logical symbols, i.e., $\in$ (relation between elements and a set), $\subseteq$ (inclusiveness among sets), $\sqsubseteq$ (sub/super concept of class relation), are used in addition to common logical connectives $\wedge$ (conjunction) and $\vee$ (disjunction). In the domain of RDF and OWL, $\mathcal{F}$ contains only $\mathbb{E}$XT(.) (extension of property) and $\mathbb{C}$EXT(.) (extension of class).

There are no variables except for blank nodes in RDF and OWL. So, note that every symbol in RDF and OWL standing for an object is a constant term in logics. However, discussions on sets require variables. Thus, variables for sets are expressed x, y, ... or x_1, x_2, ..., and x_i, y_i. When we indicate variables in logical forms, they may be explicitly expressed with quantifiers $\forall$ or $\exists$.

Tarskian Denotational Semantics. We discriminate sentences and words in sentences from their denotations (Tarski 1946). For example, a word "Tokyo" as logical term *Tokyo* interpreted as a city named Tokyo in Japan, and the denotation is expressed as $\mathcal{I}[\![Tokyo]\!]$ or $Tokyo^{\mathcal{I}}$. In this case, we say "*Tokyo* denotes $Tokyo^{\mathcal{I}}$" or "$Tokyo^{\mathcal{I}}$ is the denotation of *Tokyo*" for $\mathcal{I}[\![Tokyo]\!] = Tokyo^{\mathcal{I}}$. In this representation, $x = x$ is interpreted as $x^{\mathcal{I}} = x^{\mathcal{I}}$ (as tautology), and $a = b$ is interpreted to $a^{\mathcal{I}} = b^{\mathcal{I}}$ (when both denotations are identical). A sentence denotes truth value, of which truth or falsity is decided depending on rules of interpretation and models constructed by ontologists. For example, $New\_York^{\mathcal{I}} = Apple\_City^{\mathcal{I}}$ is true in some case or false in another case.

The interpretation by denotational semantics do not require us to make *unique name assumption*. We represent $a \simeq b$ for owl:sameAs and $a \not\simeq b$ for owl:differentFrom in OWL sentences. Then, the followings hold on different nodes of RDF graph $a^{\mathcal{I}}$ and $b^{\mathcal{I}}$ in OWL[10].

$$\mathcal{I}[\![a \simeq b]\!] \Rightarrow a^{\mathcal{I}} = b^{\mathcal{I}}$$
$$\mathcal{I}[\![a \not\simeq b]\!] \Rightarrow a^{\mathcal{I}} \neq b^{\mathcal{I}}$$

Universe of Discourse by Set Theory. In set theories, a set is extensionally defined by enumerating all members of the set, or intensionally defined by using logical conditions that all members in the set satisfy, which is like *comprehension principle*. However, the expression $x^{\mathcal{I}} = x^{\mathcal{I}}$ is always true in any case, thus the universe of discourse that stands for the totality can be defined as follows,

$$\mathbb{R} \equiv \{x^{\mathcal{I}} \mid x^{\mathcal{I}} = x^{\mathcal{I}}\}.$$

Note that $\mathbb{R}$ as the universe of discourse can contain all denotations (objects in models), and every entity in the universe always belongs to $\mathbb{R}$ because $x^{\mathcal{I}} = x^{\mathcal{I}}$ is always true for any $x^{\mathcal{I}}$. Also note that this form looks like a set but $\mathbb{R}$ is actually not a set but a proper class that contains everything in the universe and cannot be a member of a set.

Property Extension. In this paper, the interpretation of a triple $\langle s, p, o\rangle$ or $\langle s, o\rangle \in \mathrm{EXT}(p)$ is represented as $\langle s^{\mathcal{I}}, o^{\mathcal{I}}\rangle \in \mathbb{E}\mathrm{XT}(p^{\mathcal{I}})$,

$$\mathcal{I}[\![\langle s, o\rangle \in \mathrm{EXT}(p)]\!] \Rightarrow \langle s^{\mathcal{I}}, o^{\mathcal{I}}\rangle \in \mathbb{E}\mathrm{XT}(p^{\mathcal{I}}).$$

$\mathbb{E}\mathrm{XT}(p^{\mathcal{I}})$ is called the (semantic) extension of property $p^{\mathcal{I}}$. $\mathbb{E}\mathrm{XT}(p^{\mathcal{I}})$ is a mapping into the powerset of direct product $\mathbb{R} \times \mathbb{R}$, thereby the arguments $x^{\mathcal{I}}$ and $y^{\mathcal{I}}$ of an ordered pair $\langle x^{\mathcal{I}}, y^{\mathcal{I}}\rangle$ are in $\mathbb{R}$, namely, $(x^{\mathcal{I}} \in \mathbb{R} \wedge y^{\mathcal{I}} \in \mathbb{R})$[11].

Class Extension. We express a triple $\langle s, \mathit{rdf}:\mathit{type}, o\rangle$ as $s \upharpoonleft o$ in this paper, then the class extension can be captured as a set of which members can be interpreted as instances of classes. Namely, for an instance $x^{\mathcal{I}}$ of class $y^{\mathcal{I}}$,

$$\mathcal{I}[\![x \upharpoonleft y]\!] \Rightarrow x^{\mathcal{I}} \Uparrow y^{\mathcal{I}} \equiv \langle x^{\mathcal{I}}, y^{\mathcal{I}}\rangle \in \mathbb{E}\mathrm{XT}(\mathit{rdf}:\mathit{type}^{\mathcal{I}}) \equiv x^{\mathcal{I}} \in \mathbb{C}\mathrm{EXT}(y^{\mathcal{I}}).$$

$\mathbb{C}\mathrm{EXT}(y^{\mathcal{I}})$ is called the (semantic) class extension of class $y^{\mathcal{I}}$.

3.2 Higher Order Classes

We introduce orders into the description of RDFS classes. Namely,

$${}^{n}x^{\mathcal{I}} \Uparrow {}^{m}y^{\mathcal{I}} \equiv \langle {}^{n}x^{\mathcal{I}}, {}^{m}y^{\mathcal{I}}\rangle \in \mathbb{E}\mathrm{XT}(\mathit{rdf}:\mathit{type}^{\mathcal{I}}) \equiv {}^{n}x^{\mathcal{I}} \in \mathbb{C}\mathrm{EXT}({}^{m}y^{\mathcal{I}}) \ \text{ where } m > n \geq 0.$$

[10] A question arises in the case of no statements of owl:sameAs and owl:differentFrom for atomic nodes in comparison of two different graphs. We proposed the algorithm named *UNA for atomic objects in the non-UNA condition*. See the motivation and the detail in Koide and Takeda 2011.

[11] See the simple interpretation 3 in RDF Semantics (Hayes 2004).

Here n and m is an order of class x and y, respectively. When exactly $m = n+1$, we call it *predicative* as well as *PM.*

Definition 6 *(Predicative Classes).*

$$ {}^{n}x^{\mathcal{I}} \Uparrow {}^{n+1}y^{\mathcal{I}} \equiv \langle {}^{n}x^{\mathcal{I}}, {}^{n+1}y^{\mathcal{I}} \rangle \in \mathbb{IEXT}(\mathit{rdf:type}^{\mathcal{I}}) \quad \text{where } n \geq 0, \tag{8}$$

$$ {}^{n}x^{\mathcal{I}} \Uparrow {}^{n+1}y^{\mathcal{I}} \equiv {}^{n}x^{\mathcal{I}} \in \mathbb{CEXT}({}^{n+1}y^{\mathcal{I}}) \quad \text{where } n \geq 0. \tag{9}$$

We distinguish ${}^{n}x^{\mathcal{I}}$ from ${}^{n+1}x^{\mathcal{I}}$ as different nodes in an RDF graph, while the x is the same lexical token as IRI or nodeID in $\mathcal{V}$. Therefore, we can obtain the following lemma without violating vicious circle principle,

Lemma 1.

$$ {}^{n}x^{\mathcal{I}} \Uparrow {}^{n+1}x^{\mathcal{I}} = \langle {}^{n}x^{\mathcal{I}}, {}^{n+1}x^{\mathcal{I}} \rangle \in \mathbb{IEXT}(\mathit{rdf:type}^{\mathcal{I}}) \quad \text{where } n \geq 0, \tag{10}$$

$$ {}^{n}x^{\mathcal{I}} \Uparrow {}^{n+1}x^{\mathcal{I}} = {}^{n}x^{\mathcal{I}} \in \mathbb{CEXT}({}^{n+1}x^{\mathcal{I}}) \quad \text{where } n \geq 0. \tag{11}$$

We assume that rdf:type has the same role in the universe from $n = 0$ to ∞. This is actually the same as *axiom of reducibility* (*PM, Vol.1*, Introduction, VI), which is required to enable that there exists a (higher order) function as formally the same as the predicative function that takes individuals as arguments. We extend this principle to every RDF and OWL properties later on.

3.3 Subsumption in Higher Order Classes

The RDFS semantic condition on rdfs:subClassOf contains the following condition that is obtained with extending RDFS-original classes to higher order classes,

$$ {}^{n}x^{\mathcal{I}} \sqsubseteq {}^{m}y^{\mathcal{I}} \equiv {}^{n}x^{\mathcal{I}} \in \mathbb{C} \land {}^{m}y^{\mathcal{I}} \in \mathbb{C} \land \mathbb{CEXT}({}^{n}x^{\mathcal{I}}) \subseteq \mathbb{CEXT}({}^{m}y^{\mathcal{I}}) \quad \text{where } m, n \geq 1.$$

Here $\sqsubseteq$ represents subclass-superclass relation that is designated by rdfs: subClassOf, and $\mathbb{C}$ may be called the *universal domain of classes*, to which all classes in the universe of discourse belong. The above condition is called *subsumption.*

Then, we introduce a new notion onto the subsumption by higher order classes. If $n = m$ on the above condition, let us call it *interpretable.*

Definition 7 *(Interpretable Class Condition).*

$$ {}^{n}x^{\mathcal{I}} \sqsubseteq {}^{n}y^{\mathcal{I}} \equiv {}^{n}x^{\mathcal{I}} \in \mathbb{C} \land {}^{n}y^{\mathcal{I}} \in \mathbb{C} \land \mathbb{CEXT}({}^{n}x^{\mathcal{I}}) \subseteq \mathbb{CEXT}({}^{n}y^{\mathcal{I}})) \quad \text{where } n \geq 1. \tag{12}$$

Namely, both classes related by rdfs:subClassOf must be the same order.

The order n of classes should be greater than zero ($n > 0$), since order 0 is assigned only individuals and an individual cannot have its own extension. Note that this *interpretable class condition* is constructively obtained in accordance with the definition of "being of the same type" *PM* *9.131, in which it is stated u and v "are the same type," if "(1) both are individuals, (2) both are elementary functions taking arguments of the same type". Individuals are interpretable. So, the first order classes are also interpretable. Then, we consider rdfs:subClassOf relation among the first order classes also interpretable. Thus, this procedure may be repeated again and again from order 0 to order n.

3.4 Universal Class in Higher Order Classes

As shown in the entailment rule **rdfs4a** and **rdfs4b** (Hayes 2004), every entity in the universe of discourse is an instance of $\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}$,

$$\vdash \forall u^{\mathcal{I}}[u^{\mathcal{I}} \in \mathbb{C}\mathrm{EXT}(\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}})].$$

RDFS entailment lemma in (Hayes 2004) states that every entity as class is a subclass of the class rdfs:Resource,

$$\vdash \forall c^{\mathcal{I}}[c^{\mathcal{I}} \sqsubseteq \mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}].$$

We extend these forms for entities and classes to higher order classes in the universe as well as described above.

$$\vdash \forall\, {}^{n}u^{\mathcal{I}}[{}^{n}u^{\mathcal{I}} \in \mathbb{C}\mathrm{EXT}({}^{n+1}\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}) \equiv {}^{n}\mathbb{R}] \qquad \text{where } n \geq 0, \tag{13}$$

$$\vdash \forall\, {}^{n}c^{\mathcal{I}}[{}^{n}c^{\mathcal{I}} \sqsubseteq {}^{n}\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}] \qquad \text{where } n \geq 1. \tag{14}$$

We see that all individuals, which is expressed as ${}^{0}u^{\mathcal{I}}$ belong to ${}^{0}\mathbb{R}$, and all first classes, which is expressed as ${}^{1}u^{\mathcal{I}}$ belong to ${}^{1}\mathbb{R}$, and so forth. Using this extended rule (13), we see that the universe of discourse $\mathbb{R}$ stratifies by orders. Every entity in n-th order universe ${}^{n}\mathbb{R}$ is an instance of $(n+1)$-th order ${}^{n+1}\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}$. Therefore, all extensions of ${}^{n}\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}$ $(n \geq 1)$ covers all entities in the universe and the union of ${}^{n}\mathbb{R}$ coincides with the universe of discourse $\mathbb{R}$.

Definition 8 *(Universal Class and Stratified Universe).*

$$\bigcup_{i=1\to\infty} \mathbb{C}\mathrm{EXT}({}^{i}\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}) = \bigcup_{i=0\to\infty} {}^{i}\mathbb{R} = \mathbb{R} \tag{15}$$

We abbreviate this form to the following that is described in RDF Semantics.

$$\mathbb{C}\mathrm{EXT}(\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}) = \mathbb{R}$$

Thus, $\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}$ is appropriate to name *universal class* due to the extension being the universe of discourse $\mathbb{R}$.

3.5 Universal Metaclass in Higher Order Classes

As well as the universe of discourse $\mathbb{R}$, we set up the universal domain of classes in discourse, $\mathbb{C}$, in which all classes in the universe exist. Then, we can define for higher order classes,

$$^{n}c^{\mathcal{I}} \in \mathbb{C}\mathrm{EXT}(^{n+1}\mathit{rdfs}:\mathit{Class}^{\mathcal{I}}) \equiv {}^{n}\mathbb{C} \qquad \text{where } n \geq 1. \tag{16}$$

Definition 9 *(Universal Metaclass and Stratified Universe of Classes).*

$$\bigcup_{i=2\to\infty} \mathbb{C}\mathrm{EXT}(^{i}\mathit{rdfs}:\mathit{Class}^{\mathcal{I}}) = \bigcup_{i=1\to\infty} {}^{i}\mathbb{C} \equiv \mathbb{C} = \mathbb{R}\backslash{}^{0}\mathbb{R} \tag{17}$$

The RDF semantics shows the following condition. It is deemed to be an abbreviation of the universal metaclass (17),

$$\mathbb{C}\mathrm{EXT}(\mathit{rdfs}:\mathit{Class}^{\mathcal{I}}) = \mathbb{C}.$$

While $\mathit{rdfs}:\mathit{Class}^{\mathcal{I}}$ is appropriate to be called *universal metaclass* as a representative class for the universal domain of classes in discourse, we need to make clear the relation between $\mathbb{R}$ and $\mathbb{C}$ or $\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}$ and $\mathit{rdfs}:\mathit{Class}^{\mathcal{I}}$.

From stratified universe (13) and stratified universal domains of classes (16), we obtain the followings,

$$\begin{aligned} ^{n}\mathit{rdfs}:\mathit{Class}^{\mathcal{I}} &\in \mathbb{C}\mathrm{EXT}(^{n+1}\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}) \equiv {}^{n}\mathbb{R} && \text{where } n \geq 2,\\ ^{n}\mathit{rdfs}:\mathit{Class}^{\mathcal{I}} &\sqsubseteq {}^{n}\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}} && \text{where } n \geq 2. \end{aligned}$$

We distinguish $^{n}\mathit{rdfs}:\mathit{Class}^{\mathcal{I}}$ and $^{n+1}\mathit{rdfs}:\mathit{Class}^{\mathcal{I}}$ as well as we distinguish $^{n}\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}$ and $^{n+1}\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}$. Thus, we obtain the followings from (13) and (16),

$$\begin{aligned} ^{n}\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}} &\in \mathbb{C}\mathrm{EXT}(^{n+1}\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}) \equiv {}^{n}\mathbb{R} && \text{where } n \geq 1,\\ ^{n}\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}} &\in \mathbb{C}\mathrm{EXT}(^{n+1}\mathit{rdfs}:\mathit{Class}^{\mathcal{I}}) \equiv {}^{n}\mathbb{C} && \text{where } n \geq 1,\\ ^{n}\mathit{rdfs}:\mathit{Class}^{\mathcal{I}} &\in \mathbb{C}\mathrm{EXT}(^{n+1}\mathit{rdfs}:\mathit{Class}^{\mathcal{I}}) \equiv {}^{n}\mathbb{C} && \text{where } n \geq 2. \end{aligned}$$

Let us call these complex relations between rdfs:Resource and rdfs:Class *hemi-cross subsumption*, as these equations draw a picture like cross fire but $^{n}\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}} \not\sqsubseteq {}^{n}\mathit{rdfs}:\mathit{Class}^{\mathcal{I}}$.

Thus, if we neglect the orders of classes, the membership loops appear on rdfs:Resource and rdfs:Class, but by seeing the orders, no membership loops exist in the universe.

$$\begin{aligned} \mathit{rdfs}:\mathit{Class}^{\mathcal{I}} &\in \mathbb{C}\mathrm{EXT}(\mathit{rdfs}:\mathit{Class}^{\mathcal{I}})\\ \mathit{rdfs}:\mathit{Resource}^{\mathcal{I}} &\in \mathbb{C}\mathrm{EXT}(\mathit{rdfs}:\mathit{Resource}^{\mathcal{I}}) \end{aligned}$$

4 Meta-Modeling Criteria in RDFS and OWL

As shown above, several principles about the order number of classes are addressed to avoid infinite membership loops. Here we set up them as criteria for meta-modeling.

1. (reducible) Every property that is applicable to individuals and the first order classes is applicable to much higher order classes.
2. (predicative) In respect of properties that make the relation of instance and class, i.e., rdf:type, owl:oneOf, rdfs:domain, rdfs:range, etc., the order of class must be plus one to the order of the instances.
3. (interpretable) In respect of properties that make the relation among non-literal objects, the order of arguments must be the same. Note that this principle is not applied to instance objects of Datatype such as strings, numbers, or URLs as datatype.
4. (constructive) Even if we adopt the predicative and the interpretable principles, ambiguous and undecidable entities on orders may still remain. Such a case, the orders must be decidable by ascendingly computing orders from individuals ($n = 0$), and first classes ($n = 1$), or descendingly computable starting at higher orders to lower orders so that the computation terminates at individuals level ($n = 0$).

In RDF-based OWL semantics, the class extension of OWL is defined as follows[12],

$$\mathbb{C}\mathrm{EXT}(c^{\mathcal{I}}) = \{x^{\mathcal{I}} \in \mathbb{R} \mid \langle x^{\mathcal{I}}, c^{\mathcal{I}} \rangle \in \mathbb{E}\mathrm{XT}(\mathit{rdf\!:\!type}^{\mathcal{I}})\}.$$

We extend this definition to higher order classes as

$$\mathbb{C}\mathrm{EXT}({}^{n+1}c^{\mathcal{I}}) = \{{}^{n}x^{\mathcal{I}} \in {}^{n}\mathbb{R} \mid \langle {}^{n}x^{\mathcal{I}}, {}^{n+1}c^{\mathcal{I}} \rangle \in \mathbb{E}\mathrm{XT}(\mathit{rdf\!:\!type}^{\mathcal{I}})\}. \tag{18}$$

Namely, all individuals ${}^{0}x^{\mathcal{I}}$ in OWL belong to ${}^{0}\mathbb{R}$ of the RDF universe, and all first classes ${}^{1}c^{\mathcal{I}}$ belong to ${}^{1}\mathbb{R}$ of the RDF universe, and so forth.

In the document of RDF-based OWL semantics, a special syntax form is used for sequence of entities, i.e., $\mathbb{S}\mathrm{EQ} \equiv \mathbb{E}\mathrm{XT}(\mathit{rdf\!:\!List})$. In this paper, we express the sequence of entities simply $(x, y, \dots)$. So, owl:intersectionOf and owl:unionOf[13] are extended to higher order classes as follows,

$$\langle\ {}^{n}z^{\mathcal{I}}, ({}^{n}c_1^{\mathcal{I}}, \dots, {}^{n}c_m^{\mathcal{I}})\ \rangle \in \mathbb{E}\mathrm{XT}(\mathit{owl\!:\!intersectionOf}) \Leftrightarrow$$
$${}^{n}z^{\mathcal{I}},\ {}^{n}c_1^{\mathcal{I}}, \dots, {}^{n}c_m^{\mathcal{I}} \in {}^{n}\mathbb{C}\ \wedge\ \mathbb{C}\mathrm{EXT}({}^{n}z^{\mathcal{I}}) = \bigcap_{i=1 \to m} \mathbb{C}\mathrm{EXT}({}^{n}c_i^{\mathcal{I}}), \tag{19}$$

$$\langle\ {}^{n}z^{\mathcal{I}}, ({}^{n}c_1^{\mathcal{I}}, \dots, {}^{n}c_m^{\mathcal{I}})\ \rangle \in \mathbb{E}\mathrm{XT}(\mathit{owl\!:\!unionOf}) \Leftrightarrow$$
$${}^{n}z^{\mathcal{I}},\ {}^{n}c_1^{\mathcal{I}}, \dots, {}^{n}c_m^{\mathcal{I}} \in {}^{n}\mathbb{C}\ \wedge\ \mathbb{C}\mathrm{EXT}({}^{n}z^{\mathcal{I}}) = \bigcup_{i=1 \to m} \mathbb{C}\mathrm{EXT}({}^{n}c_i^{\mathcal{I}}). \tag{20}$$

[12] http://www.w3.org/TR/owl-rdf-based-semantics/#Class_Extensions.

[13] http://www.w3.org/TR/owl2-rdf-based-semantics/#Semantic_Conditions_for_Boolean_Connectives.

As well, the semantic conditions of enumeration is extended as

$$\langle\ {}^{n+1}z^{\mathcal{I}}, ({}^{n}a_1^{\mathcal{I}}, \ldots, {}^{n}a_m^{\mathcal{I}})\ \rangle \in \mathbb{E}\mathrm{XT}(\mathit{owl}\!:\!\mathit{oneOf}) \Leftrightarrow \\ {}^{n+1}z^{\mathcal{I}} \in {}^{n+1}\mathbb{C}\ \wedge\ \mathbb{C}\mathrm{EXT}({}^{n+1}z^{\mathcal{I}}) = \{{}^{n}a_1^{\mathcal{I}}, \ldots, {}^{n}a_m^{\mathcal{I}}\}. \quad (21)$$

5 Related Work and Discussion

On Punning. W3C posed six use cases on "punning"[14], but only the first and the second cases in these use cases deserve to discuss in ontological view.

The first case is solved by making $a\!:\!Service$ a meta-class.

$$\boldsymbol{{}^{2}a\!:\!Service\ rdf\!:\!subClassOf\ {}^{2}owl\!:\!Class}\ .$$
$${}^{2}a\!:\!Service\ rdf\!:\!type\ {}^{3}owl\!:\!Class\ .$$
$${}^{1}a\!:\!Person\ rdf\!:\!type\ {}^{2}owl\!:\!Class\ .$$
$${}^{1}s1\ rdf\!:\!type\ {}^{2}a\!:\!Service\ .$$
$${}^{1}s1\ a\!:\!input\ {}^{1}a\!:\!Person\ .$$

The first triple shown above is newly added to the original set of the triples, so that the system becomes decidable and $s1$ becomes to be interpreted as the first order class rather than an individual because $n > 1$ for ${}^{n}owl\!:\!Class$. The last triple must be modified to the form for domain $s1$ and range $a : Person$ *constraints*.

The second use case is a typical quiz for meta-classing. Harry as individual is a eagle, and the eagle as species is in the Red List as endangered species. In the following triples, the first and second triples are newly added to the others, so that they set up $a\!:\!Species$ and $a\!:\!EndangeredSpecies$ as meta-classes. Then, the case becomes decidable.

$$\boldsymbol{{}^{2}a\!:\!Species\ rdfs\!:\!subClassOf\ {}^{2}owl\!:\!Class}.$$
$$\boldsymbol{{}^{2}a\!:\!EndangeredSpecies\ rdfs\!:\!subClassOf\ {}^{2}a\!:\!Species}.$$
$${}^{1}a\!:\!Eagle\ rdf\!:\!type\ {}^{2}owl\!:\!Class.$$
$${}^{0}a\!:\!Harry\ rdf\!:\!type\ {}^{1}a\!:\!Eagle.$$
$${}^{1}a\!:\!Eagle\ rdf\!:\!type\ {}^{2}a\!:\!Species.$$
$${}^{1}a\!:\!Eagle\ rdf\!:\!type\ {}^{2}a\!:\!EndangeredSpecies.$$

Domino-Tilting Puzzle. Motik posed Domino-tilting Puzzle to exemplify undecidable OWL Full (Motik 2007). In this example, GRID is an OWL class and does not interpreted as property. However, this model involves the infinite ascending higher order computation by the resulted stratified form such as ${}^{n}GRID \sqsubseteq \exists rdf\!:\!type.{}^{n+1}GRID$, starting from an individual GRID $a_{0,0}$ at the coordinate $(0, 0)$, and going to $a_{\infty,\infty}$. Then we have no way to terminate the computation.

[14] http://www.w3.org/2007/OWL/wiki/Punning#Treating_classes_as_instances_of_metaclasses_.28Class_.E2/86/94_Individual.29.

6 Conclusion

We focused on set theories involved in RDF and RDF-based OWL Semantics, and clarified that stratified proper classes such as ${}^n rdfs\!:\!Resource$, ${}^n rdfs\!:\!Class$ (${}^n owl\!:\!Thing$ and ${}^n owl\!:\!Class$ as well) do not include membership loops. Then we proposed a set of criteria for meta-modeling that is derived from *Ramified Type Theory* in Principia Mathematica. While it is obvious that unrestricted OWL Full may be undecidable, the proposed meta-modeling criteria is not enough to make meta-modeling computation decidable, even if we fulfill these criteria in meta-modeling as shown in Domino-tilting Puzzle. Let us call such ones *unsound meta-modeling setup*. We need further ways in well-mannered OWL Full meta-modeling so that the systems would be decidable with the computation of higher order classes.

References

Aczel, A.D.: The Mystery of the Aleph, Four Walls Eight Windows (2000)

Boolos, G.S.: The Iterative Conception of Set. J. Philosophy **68–8**, 215–231 (1971)

Bourbaki, N.: Éléments de Mathématique, Chapitres 1 et 2. Hermann (1966)

Cantor, G.: Beiträge zur Begründung der transfiniten Mengenlehre. Mathematische Annalen, Bd.46, S.481-512 (1895). Contributions to the Founding of the Theory of Transfinite Numbers, Dover (1955)

Cantor, G.: Letter to Dedekind (1899) in "From Frege to Gödel A Source Book in Mathematical Logic, 1879–1931". In: van Heijenoort, J. (ed.). Harvard (1967)

Doets, H.C.: Zermelo-Fraenkel Set Theory (2002). http://staff.science.uva.nl/vervoort/AST/ast.pdf

Hutton, G.: Programming in Haskell. Cambridge University Press, New York (2007)

Hayes, P.: RDF Semantics. W3C Recommendation (2004). http://www.w3.org/TR/2004/REC-rdf-mt-20040210/

Kamareddine, F., Laan, T., Nederpelt, R.: A Modern Perspective on Type Theory. Kluwer, New York (2004)

Genesereth, M.R., Fikes, R.E.: Knowledge Interchange Format version 3.0 Reference Manual (1994). http://logic.stanford.edu/kif/Hypertext/kif-manual.html

Koide, S., Takeda, H.: Common Languages for Web Semantics, Evaluation of Novel Approaches to Software Engineering. In: Communications in Computer and Information Science, vol. 230, pp. 148–162. Springer (2011)

Motik, B.: On the properties of metamodeling in OWL. J. Logic Comput. **17**(4), 617–637 (2007). doi:10.1093/logcom/exm027

Patel-Schneider, P.F., Hayes, P., Horrocks, I.: OWL web ontology language semantics and abstract syntax Sect. 5. RDF-Compatible Model-Theoretic Semantics (2004a). http://www.w3.org/TR/owl-semantics/rdfs.html

Patel-Schneider, P.F., Horrocks, I.: OWL web ontology language semantics and abstract syntax Sect. 3. Direct Model-Theoretic Semantics (2004b). http://www.w3.org/TR/owl-semantics/direct.html

Schneider, M.: OWL 2 web ontology language RDF-based semantics, 2nd edn. (2014). http://www.w3.org/TR/owl-rdf-based-semantics/

van Heijenoort, J. (ed.): Russell, B.: Letter to Frege (1902) in "From Frege to Gödel A Source Book in Mathematical Logic, 1879–1931". Harvard (1967)

Whitehead, A.N., Russell, B.: Principia Mathematica, vol. 1. Merchant Books (1910)
Graham, S.: Re-examining Russell's paralysis: ramified type-theory and Wittgenstein's objection to Russell's theory of judgment. J. Bertrand Russell Stud. **23**, 5–26 (2003)
Tarski, A.: Introduction to Logic. Dover (1946/1995). This book is an extended edition of the book title "On Mathematical Logic and Deductive Method," appeared at 1936 in Polish and 1937 in German

Designing of Ontology for Domain Vocabulary on Agriculture Activity Ontology (AAO) and a Lesson Learned

Sungmin Joo[1(✉)], Seiji Koide[2], Hideaki Takeda[1], Daisuke Horyu[3], Akane Takezaki[3], and Tomokazu Yoshida[3]

[1] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
{joo,takeda}@nii.ac.jp

[2] Ontolonomy, LLC., 3-76-3-J901, Mutsukawa, Minami-ku, Yokohama, Japan
koide@ontolonomy.co.jp

[3] National Agriculture and Food Research Organization, 1-31-1, Kannondai, Tsukuba, Ibaraki, Japan
{horyu,akane,jones}@affrc.go.jp

Abstract. This paper proposes Agriculture Activity Ontology (AAO) as a basis of the core vocabulary of agricultural activity. Since concepts of agriculture activities are formed by the various context such as purpose, means, crop, and field, we organize the agriculture activity ontology as a hierarchy of concepts discriminated by various properties such as purpose, means, crop and field. The vocabulary of agricultural activity is then defined as the subset of the ontology. Since the ontology is consistent, extendable, and capable of some inferences thanks to Description Logics, so the vocabulary inherits these features. The vocabulary is also linked to existing vocabularies such as AGROVOC. It is expected to use in the data format in the agricultural IT system. The vocabulary is adopted as the part of "the guideline for agriculture activity names for agriculture IT systems" issued by Ministry of Agriculture, Forestry and Fisheries (MAFF), Japan. Also we investigated the usefulness of the ontology as the method for defining the domain vocabulary.

Keywords: Ontology · Agriculture · Agronomic sciences · Knowledge representation · Core vocabulary · Vocabulary management

1 Introduction

The various IT systems have been introduced in farm management to realize better management, i.e., more efficient resource management, finer production control and better product quality. Now data management is indispensable in farm management. Data in farm management is also used in own purpose but the aggregated data is used for statistics, analysis and prediction for area agriculture.

Data in agricultural IT systems is nonetheless not easy to federate and integrate since the languages to describe data are not unified. Terminology in agriculture such as names of activity, equipment, and crop has not been well standardized mainly because agriculture has been *local*. Some of locality comes from diversity of culture and environment and others from the way of business, i.e., farms are small and run independently. But introduction of IT systems changed the situation; farms can be connected beyond the barrier of individual farms, regions, and even culture. But un-unified terminology exists as the problem. Without unified terminology, smooth data exchange cannot be enabled. So standardization of terminology is the key to enhance agriculture with IT systems. We focus on agriculture activity in this paper. Agriculture activity is the most basic element of farm management and also the most difficult to standardize since it is more abstract than other types of terminology like equipment and crop.

In this paper, we investigate the existing vocabulary system for agricultural activities. Then we propose the agriculture activity ontology by paying attention to the linguistic feature of agricultural activities. We also explore reasoning functions with the ontology and web services to utilize the ontology. Finally, we discuss the future directions of the improvement and extension of the ontology.

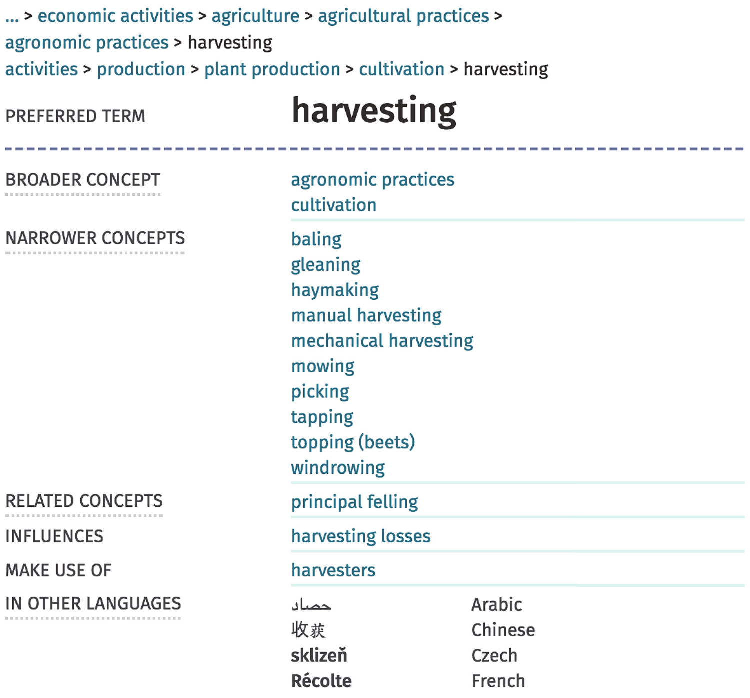
2 An Existing Resource: AGROVOC

In this section, we survey the features of AGROVOC (a portmanteau of agriculture and vocabulary) [1] as an existing agricultural vocabulary system. AGROVOC is the most well-known vocabulary in agriculture supervised by Food and Agriculture Organization (FAO) of the United Nations. AGROVOC is the thesaurus containing more than 32,000 terms of agriculture, fisheries, food, environment and other related fields. It has international interoperability as it is provided in 21 languages. Each term can have the hierarchical structure with narrower concept and broader concept, and there are 25 top-most concepts such as activities, organisms, location, products and so on. 1,434 narrower concepts are provided in activities which contains the concepts about agriculture activity.

AGROVOC is the well-known vocabulary system which has international interoperability and it contains many terms. However, There are some insufficient features in order to use as the core vocabulary. First of all, the relationship between concepts is not clear. Most of narrower/broader relationship is attached only by considering the pair-wise relationship. Thus hierarchy by these relationships are not so consistent. For instance, *Vegetative propagation* has *Rooting* as the narrower concept. Figure 1 shows the broader concept and the narrower concept of *harvesting. mowing* is located as the narrower concept of *harvesting*, but it is not an appropriate classification considering the general meaning of *mowing*. This kind of problem occurs because AGROVOC is established vocabulary system as the thesaurus. This vague relationship between concepts makes the problem when adding a new term; it is difficult to define the relation with concepts in AGROVOC, i.e., to find the best position to the new term.

In addition, the number of activity names about rice farming, which is important in Asia including Japan, are insufficient. For example, in rice farming, especially *pulling seedlings* and *midseason drainage* which are important activity in a rice paddy, are not contained in AGROVOC.

The ambiguousness of relationship among concepts in the existing vocabulary system of agricultural activities is thus problematic. It is required to clearly


... > economic activities > agriculture > agricultural practices > agronomic practices > harvesting
activities > production > plant production > cultivation > harvesting

PREFERRED TERM	harvesting
BROADER CONCEPT	agronomic practices cultivation
NARROWER CONCEPTS	baling gleaning haymaking manual harvesting mechanical harvesting mowing picking tapping topping (beets) windrowing
RELATED CONCEPTS	principal felling
INFLUENCES	harvesting losses
MAKE USE OF	harvesters
IN OTHER LANGUAGES	حصاد Arabic 收获 Chinese sklizeň Czech Récolte French

Fig. 1. The broader concept and the narrower concept of *harvesting* in AGROVOC (http://oek1.fao.org/skosmos/agrovoc/en/page/c_3500).

define the relationships among concepts and specify them. In order to solve these problems, this paper suggests the establishment of the ontology for agriculture activity. The ontology can define the clear concepts by separating concepts and representation. Also it can reflect the characteristics of the domain more clearly by structuralizing the relationships.

3 Designing of Agricultural Activity Ontology

This paper describes the Agriculture Activity Ontology (AAO) as the basis of the core vocabulary of agriculture activity, and it provides semantics for agricultural activity names. Also, AAO is formalized by Description Logics in order to define and classify the agricultural activities clearly. Formalization by Description Logics makes it possible to judge the inconsistency and subsumption among concepts, and to enable more logical inferences. The ontology designed by Description Logics can be converted to OWL so that it can processed by computers.

3.1 The Structuralization of the Agricultural Activities

Our strategy to structuralize agricultural activities is the top-down, i.e., starting from the most general activity and expanding it to more specific activities.

The important criteria for the top-down approach is how we can classify more specific concepts consistently. We define more specific concepts by specifying attributes and their values. We furthermore define the general rule for specifying attributes.

We start with the top concept *Agriculture Activity* which denotes all kind of activities related on crop and/or fields. Then we break down the concept into more concrete concepts. When farmers plan or do a certain agricultural activity, the first decision is what for they would take the action, i.e., *purpose* is the first attribute to distinguish agriculture activities. After the purpose is well specified, we use other attributes, i.e., *act* (type of action), *target*, *place*, *means*, *equipment*, and *season* in this order. *Crop* is also introduced so as to define the activity for a specific crop. These eight attributes are used to define the concept and to form the hierarchy of the agricultural activity.

The basic idea of formalization of Agriculture Activity Ontology is that concepts correspond to concepts in Descriptions Logics (DLs) while attributes such as purpose correspond to roles in DLs. By adding the role and the role value, the concepts is defined as the narrower concept of the original concept. If the added role is what is already used in the original concept and the value of the role of the new concept is narrower than that in the original concept, the new concept is also narrower concept of the original concept. It should be noted; not all concepts correspond to terms for farm management since some abstract concepts are introduced just to classify. So we distinguish the abstract concepts not corresponding to terms and concrete concepts corresponding to terms. We call former *category* and the latter *term*. For example, *thinning* and *cutting root* are terms, and their broader concept *activity for uniformity* is the category. The category and the term are succeeding the values of the attributes, and they have the relationship of inclusion.

Now let's look at the ontology in detail. At first, we classify the agriculture activity into two; *crop production activity* which is related to the crop production directly, and *administrative activity* which is related to the farm management. *crop production activity* is classified into the following four activities: *crop growth activity* which is for the purpose of crop growth, *activity for environmental control* which is for the purpose of the environment control, *activity for post production* which is for the purpose of the post production, and *activity for support for crop production* which is for the purpose of the indirect support in the crop production. So as to define narrower concepts, *purpose*, *act*, *target*, *place*, *means*, *equipment*, *season*, *crop* were used as attributes. Classification is conducted by using values of these attributes.

The activity *Seeding* can be defined as follows; First of all, *Seeding* is one of the activities of *Activity for seed propagation*, and *Activity for control of propagation* is the broader concept of *Activity for seed propagation*, *Crop growth activity* is the broader concept of *activity for control of propagation. Crop production activity* is the broader concept of *Crop growth activity.* Lastly, the broader concept of *Crop production activity* is *agriculture activity* which is the broadest concept. All these concepts are classified by *purpose* so that *purpose* attribute is used.

Since the values of purpose attributes are hierarchical, activity concepts are hierarchical. *Seed propagation* is the narrower concept of *Control of propagation*, and it then the narrower of *Crop growth. Crop growth* is the narrower concept of *Crop production.*

$$
\begin{aligned}
Crop\_production\_activity &\equiv Agriculture\_activity \\
&\sqcap \forall purpose.crop\_production \qquad (1) \\
Crop\_growth\_activity &\equiv Crop\_production\_activity \\
&\sqcap \forall purpose.crop\_growth \qquad (2) \\
Activity\_for\_control\_of\_propagation &\equiv Crop\_growth\_activity \\
&\sqcap \forall purpose.control\_of\_propagation \qquad (3) \\
Activity\_for\_seed\_propagation &\equiv Activity\_for\_control\_of\_propagation \\
&\sqcap \forall purpose.seed\_propagation \qquad (4)
\end{aligned}
$$

The formula (1), (2), (3), (4) can be represented as the formula (5).

$$
\begin{aligned}
Activity\_for\_seed\_propagation \equiv\ & Agriculture\_activity \\
& \sqcap \forall purpose.seed\_propagation \\
& \sqcap \forall purpose.control\_of\_propagation \\
& \sqcap \forall purpose.crop\_growth \\
& \sqcap \forall purpose.crop\_production \qquad (5)
\end{aligned}
$$

Here the purposes of *seed propagation*, *control of propagation*, *crop growth*, *crop production* have the following relation of inclusion by definition.

$$seed\_propagation \sqsubseteq control\_of\_propagation \sqsubseteq crop\_growth \sqsubseteq crop\_production \qquad (6)$$

Thus the formula (5) is represented as below.

$$
\begin{aligned}
Activity\_for\_seed\_propagation \equiv\ & Agriculture\_activity \\
& \sqcap \forall purpose.seed\_propagation \qquad (7)
\end{aligned}
$$

On the other hand, *Seeding* is the activity whose *purpose* is *seed propagation*, *place* is *field*, *target* is *seed*, and *act* is *sow*. So it is defined as below.

$$
\begin{aligned}
Seeding \equiv\ & Activity\_for\_seed\_propagation \\
& \sqcap \forall act.sow \\
& \sqcap \forall target.seed \\
& \sqcap \forall place.field \qquad (8)
\end{aligned}
$$

Therefore, *Seeding* in the agricultural activities is defined like below from the formula (7) and (8).

$$\begin{aligned} Seeding \equiv\ & Agriculture\_activity \\ & \sqcap \forall purpose.seed\_propagation \\ & \sqcap \forall act.sow \\ & \sqcap \forall target.seed \\ & \sqcap \forall place.field \end{aligned} \tag{9}$$

Among *Seeding*, when the *crop* is *rice* and the *place* is *nursery box*, it is classified as *seeding on nursery box*, when the *place* is *paddy field*, it is classified as *direct seeding in flooded paddy field* and when the *place* is *well drained paddy field*, it is classified as *direct seeding in well drained paddy field*. As a result, *Seeding on nursery box*, *Direct seeding in flooded paddy field* and *Direct seeding in well drained paddy field* are in the relationship of siblings, and they are the narrower concept of *seeding*. Here these activities can be represented by the description logic as below.

$$\begin{aligned} Seeding\_on\_nursery\_box \equiv\ & Seeding \\ & \sqcap \forall crop.rice \\ & \sqcap \forall place.nursery\_box \end{aligned} \tag{10}$$

$$\begin{aligned} Direct\_seeding\_in\_flooded\_paddy\_field \equiv\ & Seeding \\ & \sqcap \forall crop.rice \\ & \sqcap \forall place.paddy\_field \end{aligned} \tag{11}$$

$$\begin{aligned} Direct\_seeding\_in\_well\_drained\_paddy\_field \equiv\ & Seeding \\ & \sqcap \forall crop.rice \\ & \sqcap \forall place.well\_drainded\_paddy\_field \end{aligned} \tag{12}$$

The *place* value of *nursery box*, *paddy field* and *well drained paddy field* are defined as a part of *field*.

$$\begin{aligned} nursery\_box &\sqsubseteq field, \\ paddy\_field &\sqsubseteq field, \\ well\_drained\_paddy\_field &\sqsubseteq field \end{aligned} \tag{13}$$

Thus, from the formula (11), (12) and (13), we define the activity *Seeding on nursery box*, *Direct seeding in flooded paddy field* and *Direct seeding in well drained paddy field* as below.

$$\begin{aligned} Seeding\_on\_nursery\_box \equiv\ & Agriculture\_activity \\ & \sqcap \forall purpose.seed\_propagation \\ & \sqcap \forall act.sow \\ & \sqcap \forall target.seed \\ & \sqcap \forall crop.rice \\ & \sqcap \forall place.nursery\_box \end{aligned} \tag{14}$$

$$\begin{aligned} Direct\_seeding\_in\_flooded\_paddy\_field \equiv\ & Agriculture\_activity \\ & \sqcap \forall purpose.seed\_propagation \\ & \sqcap \forall act.sow \\ & \sqcap \forall target.seed \\ & \sqcap \forall crop.rice \\ & \sqcap \forall place.paddy\_field \end{aligned} \tag{15}$$

$$\begin{aligned} Direct\_seeding\_in\_well\_drained\_paddy\_field \equiv\ & Agriculture\_activity \\ & \sqcap \forall purpose.seed\_propagation \\ & \sqcap \forall act.sow \\ & \sqcap \forall target.seed \\ & \sqcap \forall crop.rice \\ & \sqcap \forall place.well\_drainded\_paddy\_field \end{aligned} \tag{16}$$

By combining adding more attributes and subdividing the attribute values, we can flexibly form the hierarchical structure to represent terminology used in agriculture without loosing logical consistency.

3.2 Polysemic Concepts

There are many activities conducted for the multiple purposes in the agricultural activities. The typical case is *Activity for mulching.* One of its purposes is spreading organic matter and other things on the surface of the soil, but there are other purposes; to keep the temperature optimal and controls the temperature, and to refrain weeds. The other example is *Puddling.* It is to plow the bottom of a rice field, but it is also intended to conduct for the purpose of water retention, i.e., preventing from the water leak, and for the purpose of land leveling, i.e., flattening the soil. In our formalization, these concepts are interpreted as polysemic concept and modelled as disjunction of multiple concepts since none of multiple concepts are mandatory rather optional. Here puddling can be expressed with DL as follows;

$$\begin{aligned} Puddling \equiv\ & Pulverization \\ & \sqcup Land\_leveling \\ & \sqcup Activity\_for\_water\_retention \end{aligned} \tag{17}$$

Now *Puddling* is expanded as follows;

$$\begin{aligned}Puddling \equiv\ & (Agriculture\_activity \\ & \sqcap \forall purpose.land\_preparation \\ & \sqcap \forall act.crush \\ & \sqcap \forall place.paddy\_field) \\ \sqcup\ & (Agriculture\_activity \\ & \sqcap \forall purpose.land\_preparation \\ & \sqcap \forall act.level \\ & \sqcap \forall target.field) \\ \sqcup\ & (Agriculture\_activity \sqcap \forall purpose.water\_retention)\end{aligned} \tag{18}$$

By converting the disjunction form into the formula (18) to the conjunction form, we can infer the formula (19).

$$\begin{aligned}Puddling \sqsupseteq\ & Agriculture\_activity \\ & \sqcap \forall purpose.(land\_preparation \sqcup water\_retention) \\ & \sqcap \forall act.(crush \sqcup level) \\ & \sqcap \forall place.paddy\_field\end{aligned} \tag{19}$$

The polysemic concepts defined with multiple concepts can properly express features for the activities conducted for multiple attributions in the Agriculture Activity Ontology.

3.3 Synonym

There are many synonyms in the vocabulary for agricultural activities. It is easily treated in DL as follows;

$$Seeding \equiv Sowing \tag{20}$$

In addition, expressions in multiple languages are also represented as synonyms. It is important especially for non-English speaking countries[1].

$$\begin{aligned}Seeding &\equiv Sowing \\ &\equiv \text{は種}\end{aligned} \tag{21}$$

So as to correspond to the variety of the vocabulary, the Agriculture Activity Ontology enables to separate the concepts themselves and expressions of the concepts properly.

[1] Indeed, AAO is basically written in Japanese and expressions of concepts and roles in English are optional. But we here provide the English version of AAO for simplicity of explanation.

4 Reasoning by Agriculture Activity Ontology

Generally speaking, the more abstract concepts are, the more difficult it is to define them. The more specific concepts are, the easier it is to take the specific attributes into account. On a specific agriculture activity, it is easy to define the activity with the specific attributes, such as the purpose, the target, the means, etc. Since the purpose of AAO is to keep the agriculture activity terms consistent and well-organized, placing new terms at the appropriate location in the ontology is mandatory. For instance, suppose that we want to add a new term *Making scarecrow*. It is composed of attribute the *purpose* of *pest animal suppression* and attribute the *means* of *physical means*, then the abstract activity *Activity for pest animal suppression by physical means* may become the abstraction of *Making scarecrow*, even if it is not specified explicitly. Furthermore, more abstract *Activity for pest animal suppression* must be the abstraction of *Activity for pest animal suppression by physical means* without attribute the means.

$$\begin{aligned} making\_scarecrow \equiv\ & \forall purpose.pest\_animal\_suppression \\ & \sqcap \forall act.make \\ & \sqcap \forall target.scarecrow \\ & \sqcap \forall means.physical\_means \end{aligned} \tag{22}$$

The question is what is the broader concept of *Making scarecrow* in AAO, and how we can find it. We set up the ontology of attributes with the relationship of inclusion as follows.

$$pest\_animal\_suppression \sqsubseteq biotic\_suppression \sqsubseteq biotic\_control \tag{23}$$

We also set up the hierarchical structure of the agriculture activity as follows.

$$\begin{aligned} & Activity\_for\_pest\_animal\_suppression\_by\_physical\_means \equiv \\ & Activity\_for\_pest\_animal\_suppression \sqcap \forall means.physical\_means \end{aligned} \tag{24}$$

$$\begin{aligned} & Activity\_for\_pest\_animal\_suppression \equiv \\ & Activity\_for\_biotic\_control \sqcap \forall purpose.pest\_animal\_suppression \end{aligned} \tag{25}$$

$$\begin{aligned} Activity\_for\_biotic\_control \equiv\ & activity\_for\_environmental\_control \\ & \sqcap \forall purpose.biotic\_control \end{aligned} \tag{26}$$

Activity for pest animal suppression by physical means is a conjunction of *pest animal suppression* (for purpose) and *physical means* (for means). Thus, there is no contradiction by making *Activity for pest animal suppression by physical means* a broader concept of *Making scarecrow*.

The main task of Description Logics is to compute truth value in subsumption checking [2]. However, it cannot discover subsumers or subsumees for a given subsumee or subsumer in a given ontological hierarchies. Therefore, we introduced

Schank's algorithm for Case-Based Reasoning (CBR) [3] into our OWL [4] reasoning engine named SWCLOS [5,6][2], whereby an appropriate position of a given collection of pairs of attributes and values can be automatically discovered in coherent hierarchies of concepts and their attribute values, starting from a given domain top concept and descending subsuming chains to specific ones. In the systematization of AAO reasoning, the knowledge expressed in DLs is described in OWL. The following shows an example of the formula (12) described in Turtle [7].

```
cavoc.aao:Direct_seeding_in_well_drained_paddy_field a rdfs:Class ;
    rdfs:subClassOf cavoc.aao:Seeding ,
    [ a owl:Restriction ;
      owl:onProperty cavoc:crop ;
      owl:allValuesFrom cavoc:rice ] ,
    [ a owl:Restriction ;
      owl:onProperty cavoc:place ;
      owl:allValuesFrom cavoc:well_drained_paddy_field] .
```

In SWCLOS, we can see the form of any OWL entity in lisp-like expression. The following demonstrates the expression of `cavoc.aao:making_scarecrow` in SWCLOS. Note that it has no subsumer concept defined here. Command `refine-abstraction-from` performs Schank's algorithm with parameters, a domain top concept and an entity for the discovery of position.

```
gx(8): (get-form cavoc.aao:Making_scarecrow)
(owl:Class cavoc.aao:Making_scarecrow
  (rdfs:subClassOf
    (owl:Restriction _:g1937
      (owl:onProperty cavoc:purpose)
      (owl:allValuesFrom cavoc:pest_animal_suppression))
    (owl:Restriction _:g1938
      (owl:onProperty cavoc:act)
      (owl:allValuesFrom cavoc:make))
    (owl:Restriction _:g1939
      (owl:onProperty cavoc:target)
      (owl:allValuesFrom cavoc:scarecrow))
    (owl:Restriction _:g1940
      (owl:onProperty cavoc:means)
      (owl:allValuesFrom cavoc:physical_means)))
  (rdfs:label"\"Making scarecrow\"@en"))
gx(9): (refine-abstraction-from
        cavoc.aao:Crop_production_activity cavoc.aao:making_scarecrow)
#<node cavoc.aao:Activity_for_pest_animal_suppression_by_physical_means>
gx(10): (get-form cavoc.aao:making_scarecrow)
(owl:Class cavoc.aao:Making_scarecrow
  (rdfs:subClassOf cavoc.aao:Activity_for_pest_animal_suppression_by_physical_means
    (owl:Restriction _:g1937
      (owl:onProperty cavoc:purpose)
      (owl:allValuesFrom cavoc:pest_animal_suppression))
    (owl:Restriction _:g1938
      (owl:onProperty cavoc:act)
```

[2] SWCLOS is a lisp-based OWL Full processor on top of Common Lisp Object System (CLOS). It is downloadable from https://github.com/SeijiKoide/SWCLOS.

```
      (owl:allValuesFrom cavoc:make))
    (owl:Restriction _:g1939
      (owl:onProperty cavoc:target)
      (owl:allValuesFrom cavoc:scarecrow))
    (owl:Restriction _:g1940
      (owl:onProperty cavoc:means)
      (owl:allValuesFrom cavoc:physical_means)))
  (rdfs:label "\"Making scarecrow\"@en"))
```

Here, in `gx(9)` the appropriate position in the hierarchy was decided, and `gx(10)` demonstrated the `cavoc.aao:Making_scarecrow` should be a subclass of `cavoc.aao:Activity_for_pest_animal_suppression_by_physical_means`. Namely, following results were inferred.

$$\begin{aligned} making\_scarecrow \sqsubseteq \{ & (\forall purpose.activity\_for\_pest\_animal\_suppression \\ & \sqcap \forall act.make \\ & \sqcap \forall target.scarecrow \\ & \sqcap \forall means.physical\_means) \\ & \sqcap Activity\_for\_pest\_animal\_suppression\_by\_physical\_means\} \end{aligned} \tag{27}$$

5 Web Services Based on Agricultural Activity Ontology

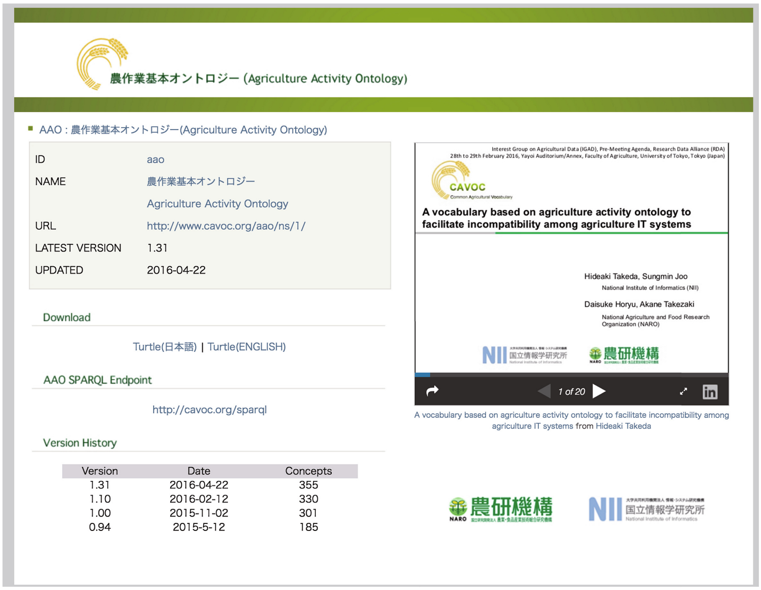
AAO is hosted on CAVOC (Common Agricultural VOCabulary, www.cavoc.org). In this section, we explain the web services of CAVOC based on the agricultural activity ontology.

5.1 Namespace of Agriculture Activity

CAVOC allows browsing and searching concepts of AAO (Fig. 2, www.cavoc.org/aao). The key feature of CAVOC is that it provides URIs for names of agriculture activities. The agriculture activity ontology has unique namespace, and each agriculture activity has URI. Each URI is structured using the http://cavoc.org/aao/ns/1/ namespace, therefore all of the terms and categories are preceded with this URL. Figure 3 is an example of the URI for *Seeding*. In the page, the hierarchical structure is represented in order to indicate the narrower concept, the broader concept, and the relationship between concepts. In addition, it provides the brief natural language explanation of the concept by using values of attributes. The simple interface allows users to browse concepts of AAO through a tree interface, and to search for specific terms.

5.2 Version History

We have developed the agriculture activity ontology with some versions. Table 1 shows the overview of the versions of the agriculture activity ontology. In version 0.94, the first version to open publicly, the concepts were classified by two



農作業基本オントロジー (Agriculture Activity Ontology)

■ AAO : 農作業基本オントロジー(Agriculture Activity Ontology)

ID	aao
NAME	農作業基本オントロジー
	Agriculture Activity Ontology
URL	http://www.cavoc.org/aao/ns/1/
LATEST VERSION	1.31
UPDATED	2016-04-22

Download

Turtle(日本語) | Turtle(ENGLISH)

AAO SPARQL Endpoint

http://cavoc.org/sparql

Version History

Version	Date	Concepts
1.31	2016-04-22	355
1.10	2016-02-12	330
1.00	2015-11-02	301
0.94	2015-5-12	185

Interest Group on Agricultural Data (IGAD), Pre-Meeting Agenda, Research Data Alliance (RDA)
28th to 29th February 2016, Yayoi Auditorium/Annex, Faculty of Agriculture, University of Tokyo, Tokyo (Japan)

CAVOC
Common Agricultural Vocabulary

A vocabulary based on agriculture activity ontology to facilitate incompatibility among agriculture IT systems

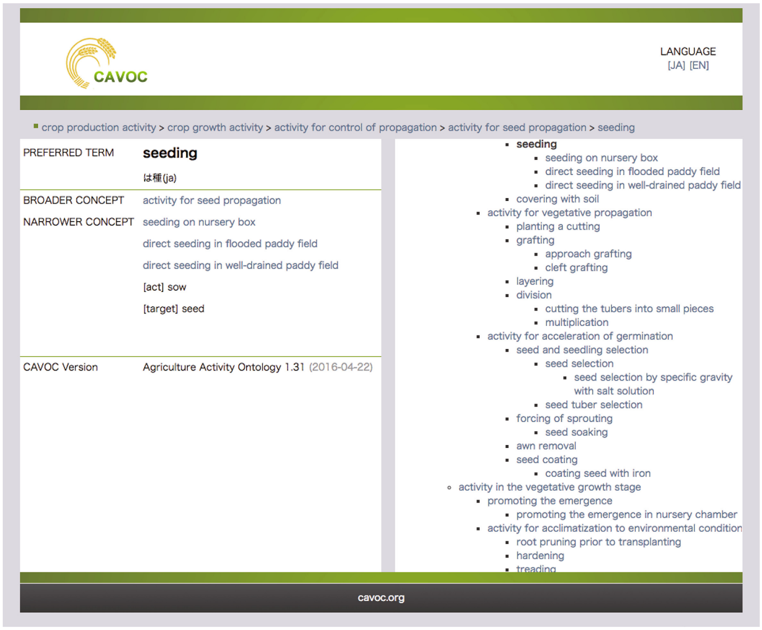
Hideaki Takeda, Sungmin Joo
National Institute of Informatics (NII)

Daisuke Horyu, Akane Takezaki
National Agriculture and Food Research Organization (NARO)

1 of 20

A vocabulary based on agriculture activity ontology to facilitate incompatibility among agriculture IT systems from Hideaki Takeda

Fig. 2. Main page of AAO (http://cavoc.org/aao).



CAVOC

LANGUAGE
[JA] [EN]

■ crop production activity > crop growth activity > activity for control of propagation > activity for seed propagation > seeding

PREFERRED TERM	**seeding**
	は種(ja)
BROADER CONCEPT	activity for seed propagation
NARROWER CONCEPT	seeding on nursery box
	direct seeding in flooded paddy field
	direct seeding in well-drained paddy field
	[act] sow
	[target] seed
CAVOC Version	Agriculture Activity Ontology 1.31 (2016-04-22)

- **seeding**
  - seeding on nursery box
  - direct seeding in flooded paddy field
  - direct seeding in well-drained paddy field
- covering with soil

- activity for vegetative propagation
  - planting a cutting
  - grafting
    - approach grafting
    - cleft grafting
  - layering
  - division
    - cutting the tubers into small pieces
    - multiplication
- activity for acceleration of germination
  - seed and seedling selection
    - seed selection
      - seed selection by specific gravity with salt solution
    - seed tuber selection
  - forcing of sprouting
    - seed soaking
  - awn removal
  - seed coating
    - coating seed with iron

- activity in the vegetative growth stage
  - promoting the emergence
    - promoting the emergence in nursery chamber
  - activity for acclimatization to environmental condition
    - root pruning prior to transplanting
    - hardening
    - treading

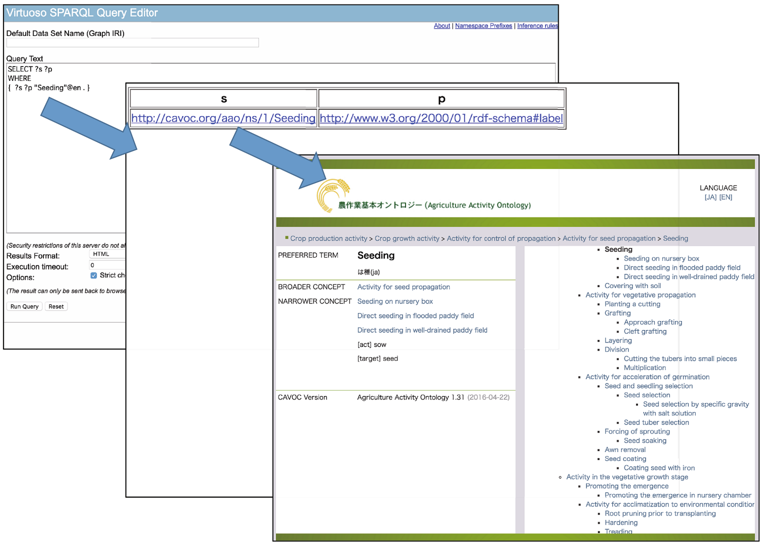
cavoc.org

Fig. 3. The namespace of seeding (http://cavoc.org/aao/ns/1/seeding).

Table 1. Listing of the history of AAO version changes

Version	Date initiated	Categories	Terms	Concepts	Attributes	Maximum layer
0.94	12/05/2015	59	126	185	2	6
1.00	02/11/2015	67	234	301	7	6
1.10	12/02/2016	73	257	330	7	7
1.31	22/04/2016	86	269	355	8	7

attributes of purpose and method. In version 1.00, the attributes were used to specify definition of activity concepts and they also have hierarchical structure. Now the description of the hierarchical relationship was logically defined based on the description logic. In the version 1.10, More concepts were introduced by consulting experts in agricultural fields and farm management systems. In the version 1.31, the newest version, the concepts were classified with eight attributes of purpose, act, target, place, means, equipment, season and crop.


Virtuoso SPARQL Query Editor
About | Namespace Prefixes | Inference rules
Default Data Set Name (Graph IRI)
Query Text
(Security restrictions of this server do not a
Results Format:
HTML
Execution timeout:
0
Options:
☑ Strict ch
(The result can only be sent back to browse
Run Query
Reset
農作業基本オントロジー (Agriculture Activity Ontology)
LANGUAGE
[JA] [EN]

```
SELECT ?s ?p
WHERE
{  ?s ?p "Seeding"@en . }
```

s	p
http://cavoc.org/aao/ns/1/Seeding	http://www.w3.org/2000/01/rdf-schema#label

Crop production activity > Crop growth activity > Activity for control of propagation > Activity for seed propagation > Seeding

PREFERRED TERM	Seeding
	は種(ja)
BROADER CONCEPT	Activity for seed propagation
NARROWER CONCEPT	Seeding on nursery box
	Direct seeding in flooded paddy field
	Direct seeding in well-drained paddy field
	[act] sow
	[target] seed
CAVOC Version	Agriculture Activity Ontology 1.31 (2016-04-22)

- Seeding
  - Seeding on nursery box
  - Direct seeding in flooded paddy field
  - Direct seeding in well-drained paddy field
- Covering with soil
- Activity for vegetative propagation
  - Planting a cutting
  - Grafting
    - Approach grafting
    - Cleft grafting
  - Layering
  - Division
    - Cutting the tubers into small pieces
    - Multiplication
- Activity for acceleration of germination
  - Seed and seedling selection
    - Seed selection
      - Seed selection by specific gravity with salt solution
    - Seed tuber selection
  - Forcing of sprouting
    - Seed soaking
  - Awn removal
  - Seed coating
    - Coating seed with iron
- Activity in the vegetative growth stage
  - Promoting the emergence
    - Promoting the emergence in nursery chamber
  - Activity for acclimatization to environmental conditior
    - Root pruning prior to transplanting
    - Hardening
    - Treading

Fig. 4. The SPARQL endpoint of AAO.

5.3 Data Sharing

The data of AAO can be downloaded in the RDF/Turtle formats from http://cavoc.org/aao/. This format is well supported by many semantic tools, and it is possible to convert it into other RDF formats if needed. Also, we provide a SPARQL endpoint for users to explore AAO data using SPARQL queries (Fig. 4).

6 Discussion and Future Work

The agriculture activity ontology ver 1.31, the latest version of the agriculture activity ontology, has 355 concepts which are either categories and terms. It covers most of terms in the national agricultural statistics in Japan. Structuralization with attributes are our own idea so that we need discussion and communication with experts in agriculture more extensively to verify the value of the ontology.

The extension to crop-specific ontologies is one of the important directions of AAO. The scope of the agriculture activity ontology is not just the general terms of agriculture but also covers agriculture activities specialized by crop. The activity specialized in the crop currently contains 10 types of crop such as rice, melon, and other things. By using the attribute of the crop, it is possible to extend to the crop-specific ontologies. We are now developing the crop-based ontologies that can define crop-specific concepts by using crop independent concepts.

There are still issues in Structuralization of the ontology. One of them is *composite* concept. In the agriculture IT systems, there are cases in which the multiple works are managed as a single activity by combining multiple activities. For example, *Raising seedling* is *composite* including*Seeding*, *Fertilization*, *Watering* and other things. However, when the agriculture is planned or implemented, it is managed as *Raising seedling*. We express this activity by combining existing activities. The concept can be expressed with `part-of` relationship, but the simple solution is not always suitable since all of the concepts are not sometimes necessary. We are now considering more appropriate formalization for combination of the relevant activities.

International interoperability is next to do. We have already connections with other activities for agricultural ontologies (for example Crop Ontology Group[3]). Our research has begun from the purpose of establishing the core vocabulary for the field of agriculture of Japan, although it can be independent regardless of language culture so that it can be applied to various languages and cultures. We will improve the ontology in order to develop international core vocabulary.

We designed a domain vocabulary by using ontology based on Description Logics. The design used by the Description Logics can make a concept which has ambiguous meaning classified clearly and deal with the situation when new vocabularies have to be added. The meaning of the concept was defined as suitable attributes for the domain. The structure was constructed to make sense the meaning of the concept by using the value of attributes and it could make the effective processing when the vocabulary lists have to be generated automatically or when the related applications have to be realized. Also it can be used as a tool like dictionary in a namespace for each vocabulary.

[3] https://sites.google.com/a/cgxchange.org/cropontologycommunity/.

7 Conclusion

We provide the Agriculture Activity Ontology (AAO) so as to standardize the vocabulary for agricultural activities. By using the ontology, it is possible to define concepts of agriculture activities beyond the linguistic diversity of the vocabulary for agricultural activities. The agriculture activity ontology was adopted as the part of "the guideline for agriculture activity names for agriculture IT systems" issued by Ministry of Agriculture, Forestry and Fisheries (MAFF), Japan in 2016, which is one of the achievements of this study [8]. We are now working to extend our idea to other agriculture domains, i.e., the standardization of vocabulary for agriculture such as the crop, distribution, and agricultural pesticide.

Acknowledgement. This work was supported by Council for Science, Technology and Innovation (CSTI), Cross-ministerial Strategic Innovation Promotion Program (SIP), "Technologies for creating next-generation agriculture, forestry and fisheries" (funding agency: Bio-oriented Technology Research Advancement Institution, NARO).

References

1. AGROVOC Multilingual agricultural thesaurus. Subsequences. J. Mol. Biol. **147**, 195–197 (1981). http://aims.fao.org/vest-registry/vocabularies/agrovoc-multilingual-agricultural-thesaurus
2. Baader, F., et al. (eds.): The Description Logic Handbook. Cambridge University Press, Cambridge (2003)
3. Riesbeck, Christopher K., Roger C. Schank, Inside Case-Based Reasoning, ISBN 0-89856-767-6, LEA (1989)
4. OWL Web Ontology Language Guide, W3C Recommendation 10, February 2004 https://www.w3.org/TR/owl-guide/
5. Koide, S., Takeda, H.: OWL-full reasoning from an object oriented perspective. In: Mizoguchi, R., Shi, Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 263–277. Springer, Heidelberg (2006). doi:10.1007/11836025_27
6. Koide, S.: Theory and implementation of object oriented semantic web language Dr.thesis, Department Informatics School of Multidisciplinary Sciences, The Graduate University for Advanced Studies (SOKENDAI) (2010)
7. RDF 1.1 Turtle, Terse RDF Triple Language, W3C Recommendation 25, February 2014. https://www.w3.org/TR/turtle/
8. Ministry of Agriculture, Fisheries and Forestry (MAFF), Japan: the guideline for agriculture activity names for agriculture IT systems (2016). http://www.kantei.go.jp/jp/singi/it2/senmon_bunka/shiryo/shiryo04.pdf (in japanese)

SQuaRE: A Visual Approach for Ontology-Based Data Access

Michał Blinkiewicz(✉) and Jarosław Bąk(✉)

Institute of Control and Information Engineering, Poznan University of Technology, Piotrowo 3a, 60-965 Poznan, Poland
{michal.blinkiewicz,jaroslaw.bak}@put.poznan.pl

Abstract. We present the SPARQL Query and R2RML mappings Environment (SQuaRE) which provides a visual interface for creating mappings expressed in R2RML. SQuaRE is a web-based tool with easy to use interface that can be applied in the ontology-based data access applications. We describe SQuaRE's main features, its architecture as well as implementation details. We compare SQuaRE with other similar tools and describe our future development plans.

1 Introduction

Ontologies, as a way of expressing knowledge, are becoming more and more popular in various research and practical fields. They allow to define a knowledge base using abstract concepts, properties and relations between them. Ontologies can be expressed in the Web Ontology Language 2 (OWL 2). This is a well-known format of ontologies and most widely used. Ontologies require data to be in a format of RDF[1] triples. Then, using an appropriate reasoner we can obtain new data in the same format. Moreover, we can query such RDF data using SPARQL[2] queries. Nevertheless, ontologies and data need to follow the RDF-based representation. Due to a fact that most of data are stored in different formats, any application of an OWL/OWL2 ontology rises the integration problem between an ontology and stored data. In this case we can transfer our current data format into RDF-based representation and change our software as well as architecture environment or we can create mappings between ontology and our data and then use an appropriate tool that handles such a solution. The first option is very cost-expensive and needs a lot of changes in the current software architecture. The second approach is easier and cheaper. We need to create mappings and then query non-RDF data with SPARQL using ontology, mappings and a tool that enables on-the-fly transfer from non-RDF into RDF data. In this method the most important part is to create appropriate mappings. Currently, a very popular standard for expressing mappings from relational databases to RDF data is W3C's R2RML[3]. The standard allows to use existing relational data in the RDF data model, and then use SPARQL to query such data.

1 https://www.w3.org/RDF/.
2 https://www.w3.org/TR/sparql11-overview/.
3 https://www.w3.org/TR/r2rml/.

Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 47–55, 2016.
DOI: 10.1007/978-3-319-50112-3_4

In this paper we describe SQuaRE, the SPARQL Queries and R2RML mappings Environment, which provides a graphical editor for creating and managing R2RML mappings and for creating and executing SPARQL queries. SQuaRE is a web-based application that simplifies the creation of mappings between a relational database and an ontology. It also enables to test created mappings by defining and executing SPARQL queries. The remainder of this paper is organized as follows. Firstly, we provide preliminary information, then we describe main features of SQuaRE. Next, we present its architecture as well as implementation details. Then, we compare SQuaRE with other similar tools. Finally, we provide conclusions along with future development plans.

2 Preliminaries

SQuaRE is an OBDA-oriented tool which helps inexperienced user to create mappings between a relational database and ontology, and then test those mappings by creating SPARQL queries. Moreover, the tool can be used to write and execute SPARQL queries. In this section we present the main overview of OBDA and R2RML.

Ontology-based Data Access (OBDA) is an approach [8] to separate a user from data sources by means of an ontology which can be perceived as a conceptual view of data. Moreover, by using concepts and relations from the ontology one can define a query in a convenient way. In this case the user operates on a different abstract level than data source. As a result the user defines queries using concepts and relations from the domain of interest and creates complex semantic conditions instead of expressing queries in terms of relational data model. Nevertheless, in order to use OBDA approach with relational data one needs to develop mappings between a relational database and an ontology.

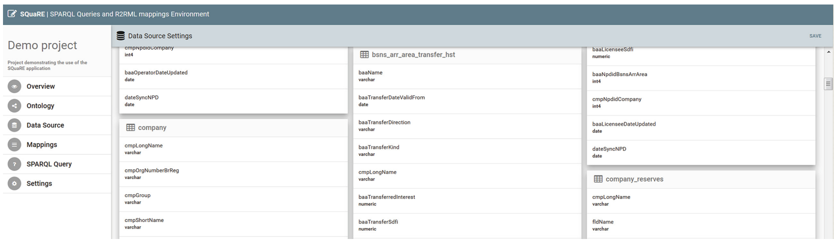
The R2RML recommendation provides a language for expressing mappings from a relational database to RDF datasets. Those mappings allow to view the relational database as a virtual RDF graph. Then, the relational database can be queried using the SPARQL language. Each R2RML mapping is a triples map (an RDF graph) that contains: a logic table (which can be a base table, a view or a valid SQL query); a subject map which defines the subject of all RDF triples that will be generated for a particular logical table row; and a set of predicate-object maps that define the predicates and objects of the generated RDF triples. In order to create R2RML mappings manually, one needs to know about ontologies, RDF, R2RML and SQL at the same time.

SQuaRE tries to overcome the aforementioned issues. The main goal of the tool is to support creation of R2RML mappings and SPARQL queries in a graphical manner. However, at the current state of development SQuaRE supports a graphical editor for R2RML mappings and a text-based interface for creating and executing SPARQL queries.

3 SPARQL Queries and R2RML Mappings Environment

Features. The SQuaRE environment is aimed at providing easy-to-use functions that will support creation and execution of SPARQL queries as well as creation of R2RML mappings. Moreover, SQuaRE allows for management of queries and mappings. A user can save both: mappings and queries for future reference, execution and management. Currently, the tool supports a graphical interface for creating mappings and a text-based interface for creating and executing SPARQL queries. Nevertheless, SQuaRE provides the following useful features:

1. Browsing a relational database – a user can choose a data source and browse its schema. In this view the user sees table names, column names as well as data types stored in each column. An example view of a relational database is shown in Fig. 1.


SQuaRE | SPARQL Queries and R2RML mappings Environment
Demo project
Project demonstrating the use of the SQuaRE application
Overview
Ontology
Data Source
Mappings
SPARQL Query
Settings
Data Source Settings
SAVE

cmpNpdidCompany
int4
baaOperatorDateUpdated
date
dateSyncNPD
date

company
cmpLongName
varchar
cmpOrgNumberBrReg
varchar
cmpGroup
varchar
cmpShortName
varchar

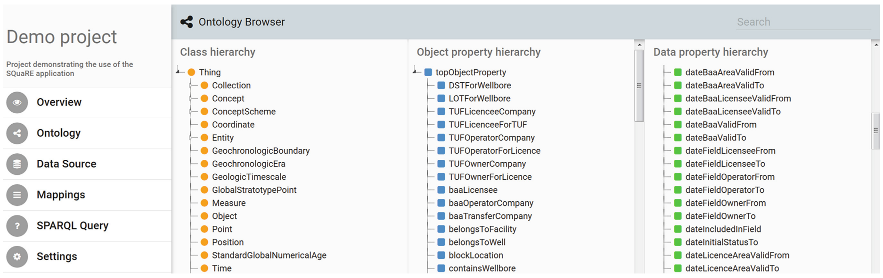
bsns_arr_area_transfer_hst
baaName
varchar
baaTransferDateValidFrom
date
baaTransferDirection
varchar
baaTransferKind
varchar
cmpLongName
varchar
baaTransferredInterest
numeric
baaTransferSdfi
numeric

baaLicenseeSdfi
numeric
baaNpdidBsnsArrArea
int4
cmpNpdidCompany
int4
baaLicenseeDateUpdated
date
dateSyncNPD
date

company_reserves
cmpLongName
varchar
fldName
varchar

Fig. 1. View of a database schema.

2. Browsing an OWL ontology – a user sees hierarchies of classes, object properties and datatype properties. The user can browse an ontology and search for its elements (Fig. 2).



Class hierarchy
Thing
Collection
Concept
ConceptScheme
Coordinate
Entity
GeochronologicBoundary
GeochronologicEra
GeologicTimescale
GlobalStratotypePoint
Measure
Object
Point
Position
StandardGlobalNumericalAge
Time

Object property hierarchy
topObjectProperty
DSTForWellbore
LOTForWellbore
TUFLicenceeCompany
TUFLicenceeForTUF
TUFOperatorCompany
TUFOperatorForLicence
TUFOwnerCompany
TUFOwnerForLicence
baaLicensee
baaOperatorCompany
baaTransferCompany
belongsToFacility
belongsToWell
blockLocation
containsWellbore

Data property hierarchy
dateBaaAreaValidFrom
dateBaaAreaValidTo
dateBaaLicenseeValidFrom
dateBaaLicenseeValidTo
dateBaaValidFrom
dateBaaValidTo
dateFieldLicenseeFrom
dateFieldLicenseeTo
dateFieldOperatorFrom
dateFieldOperatorTo
dateFieldOwnerFrom
dateFieldOwnerTo
dateIncludedInField
dateInitialStatusTo
dateLicenceAreaValidFrom
dateLicenceAreaValidTo

Fig. 2. View of an OWL ontology.

3. Browsing mappings – a list of all created mappings is shown to a user. The user can choose a mapping and then edit it in the mapping creation view (shown in Fig. 3).
4. Graphical creation of R2RML mappings – in a mapping creation view a user can create R2RML mappings. The user needs to choose tables that are going to be mapped. Then, she/he needs to search for an appropriate classes and properties to create mappings using a graphical interface. An example mapping of a table that contains data about companies to the NPD-benchmark ontology[4] is shown in Fig. 3. Classes are represented with an orange background, datatype properties with a green background and object properties with a blue background.

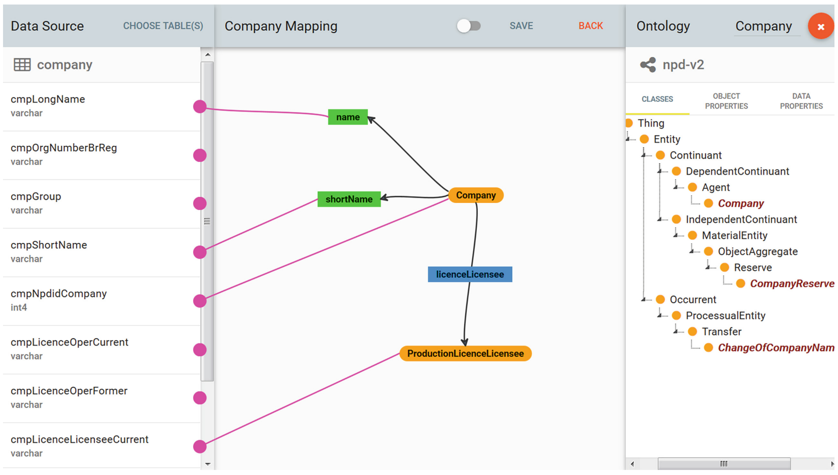
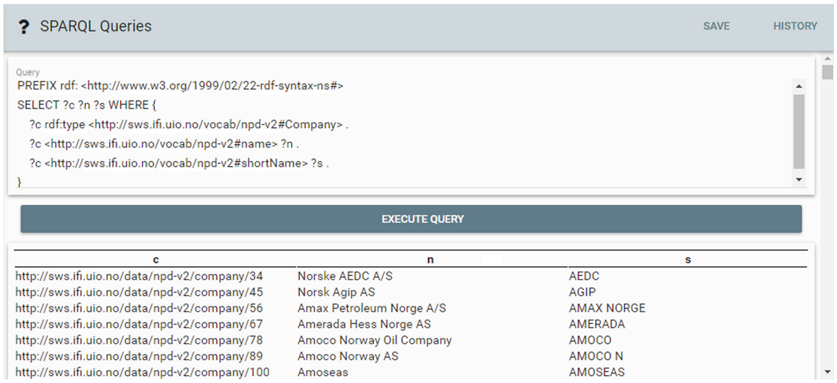


Fig. 3. Creating a mapping.

5. Management of R2RML mappings – each created mapping can be saved for future reference. A user can delete mappings, correct them or generate an R2RML file that contains all or selected mappings.
6. Textual creation of SPARQL queries – current version of SQuaRE provides an option to create SPARQL queries using a text-based interface. A user can write and execute a query. The view of a user interface for creating queries is shown in Fig. 4. Graphical editor for creating queries is one of our development plans.

[4] https://github.com/ontop/npd-benchmark.

7. Management of SPARQL queries – each constructed SPARQL query can be saved and used in future. A user can execute, delete or export a SPARQL query. Moreover, the user can select few queries (or all of them) and generate a separate .txt file that contains their definitions.
8. Execution of SPARQL queries – created SPARQL queries can be executed and results are shown as a table.



```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?c ?n ?s WHERE {
  ?c rdf:type <http://sws.ifi.uio.no/vocab/npd-v2#Company> .
  ?c <http://sws.ifi.uio.no/vocab/npd-v2#name> ?n .
  ?c <http://sws.ifi.uio.no/vocab/npd-v2#shortName> ?s .
}
```

c	n	s
http://sws.ifi.uio.no/data/npd-v2/company/34	Norske AEDC A/S	AEDC
http://sws.ifi.uio.no/data/npd-v2/company/45	Norsk Agip AS	AGIP
http://sws.ifi.uio.no/data/npd-v2/company/56	Amax Petroleum Norge A/S	AMAX NORGE
http://sws.ifi.uio.no/data/npd-v2/company/67	Amerada Hess Norge AS	AMERADA
http://sws.ifi.uio.no/data/npd-v2/company/78	Amoco Norway Oil Company	AMOCO
http://sws.ifi.uio.no/data/npd-v2/company/89	Amoco Norway AS	AMOCO N
http://sws.ifi.uio.no/data/npd-v2/company/100	Amoseas	AMOSEAS

Fig. 4. Defining and executing a SPARQL query.

The aforementioned main list of features provides an intuitive ontology-based access to relational data. Moreover, by exporting functionality (importing features are still in development) a user can use SQuaRE to create mappings and test them by creating SPARQL queries, and then save everything into external files. This allows to import queries and mappings into another tool that supports SPARQL and R2RML.

Architecture and Applied Tools. SQuaRE is developed in Java as a web application. The architecture of SQuaRE is presented in Fig. 5.

The main module, from the user's point of view, is Visual R2RML Mapper which provides tools for visual (graph-based) mappings of relational database metadata, such as table columns, and user provided ontology entities.

Moreover, there are modules responsible for data source configuration and ontology management. The former allows the user to configure a data source by providing DBMS, host location and port, username and password. The latter provides an interface to import an ontology and browse hierarchies of classes and properties.

The server-side modules consist of Data Source and DBMS Manager which manages the user defined data sources and provides JDBC-based access; Ontology Handler for OWL ontology processing; and SPARQL Query Executor which utilizes already defined mappings and allows to execute SPARQL queries in the context of a relational database.

Another key server-side module is Graph to R2RML Converter which is responsible for converting user defined mappings, supplied with client-side visual mapper, in the form of a serialized graph. This is a graph of connections between relational database columns and ontology entities. The graph is then translated into a valid R2RML representation which may be used by the SPARQL Query Executor module.

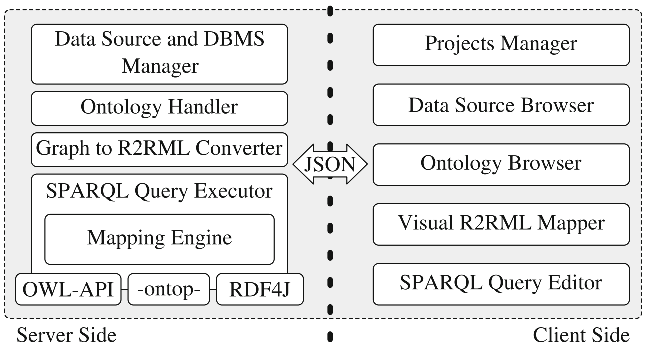


Fig. 5. The architecture of SQuaRE.

SQuaRE applies well-known tools to handle OWL ontologies, relational data and SPARQL queries. The main tools that SQuaRE uses are the following:

– OWL-API [4] – this is the most often used Java library to handle OWL/OWL2 ontologies. It contains a lot of features that are useful when manipulating ontology elements, using reasoner or serialising ontologies.
– The Spring Framework[5] – it is an application framework for the Java platform. Among others, it allows for easy creation of RESTful web services and building backend API.
– -ontop- [2] – it is a platform to query relational databases as virtual RDF graphs using SPARQL. The tool accepts mappings in R2RML and its own OBDA mapping language. SQuaRE uses -ontop- to query relational database using mappings and SPARQL.
– RDF4J[6] – it is a framework for processing RDF data. The tool supports SPARQL in version 1.1 and is used in many third party storage applications. SQuaRE uses it to save all data connected with created mappings and queries.
– Javascript libraries – we use a set of popular Javascript tools like: AngularJS, jQuery, Cytoscape.js, jsPlumb and jsTree.

[5] http://projects.spring.io/spring-framework/.
[6] http://rdf4j.org/.

4 Related Tools

Several tools have been implemented to support a user in defining mappings between data sources and ontologies. We provide the comparison table of the most similar tools to SQuaRE (Table 1).

Table 1. Comparison of main features of SQuaRE and related tools

Features	SQuaRE	OntopPro	Map-On	ODEMapster	Karma	RBA
Visual mappings editor	✓	–	✓	✓	✓	–
Visual SPARQL queries creator	–	–	–	–	–	–
SPARQL queries executor	✓	✓	–	–	–	–
Relational database support	✓	✓	✓	✓	✓	✓
Other data formats support	–	–	–	–	✓	–
Ontology browser	✓	✓	✓	✓	✓	✓
Database schema browser	✓	–	✓	✓	✓	✓
Web-based	✓	–	✓	–	✓	–

SQuaRE, Map-On [9], ODEMapster[7] and Karma [6] are equipped with a visual mapping editor. Map-On provides a graph layout for creating mappings as well as viewing ontologies and databases. ODEMapster supports a tree graphical layout for database schema and ontology. Karma provides a table-like interface for representing data sources and tree layout for visualising an ontology. RBA (R2RML By Assertion) [7] supports a tree layout for displaying databases and ontologies but a user is not able to see a graphical form of mappings. None of compared tools likewise SQuaRE do not have a visual SPARQL query creator. However, only SQuaRE and OntopPro[8] enable SPARQL queries execution against created mappings. All of selected tools are capable to map relational databases but only Karma also supports other data formats (like JSON, CSV etc.). Ontology and database schema browsers are built in all aforementioned tools but OntopPro which is a Protégé plugin does not allow to browse database schema. Furthermore, only SQuaRE, Map-On and Karma are accessible via a Web-based interface.

The aforementioned tools provide different features that overlap in some cases. However, none of them provide the comprehensive functionality for

[7] http://neon-toolkit.org/wiki/ODEMapster.
[8] http://ontop.inf.unibz.it/components/sample-page/.

OBDA-based scenario. SQuaRE provides features for creating and managing of both: R2RML mappings and SPARQL queries. Moreover, it supports users in the execution of queries and presents results in a table-like way. Moreover, we are going to implement support for a graphical creation of SWRL rules [5], which will be another difference to the mentioned tools.

SQuaRE is aimed at providing a simple user interface and easy to use methodology. Nevertheless, it should be perceived as a tool that tries to acquire the best features of other applications and provide them in a graphical way with an easy-to-use interface. The most similar tool at this stage of development is Karma, but without handling SPARQL queries and results in a graphical manner (but Karma provides more mapping methods than SQuaRE and more features regarding data integration). It is worth to notice that SQuaRE is still at the early stage of development whereas most of the tools from the list are being developed in the last few years. Some of them are even discontinued, like ODEMapster or RBA.

5 Summary and Future Work

In this paper we presented the SQuaRE tool which is a web-based environment that provides: (i) creation of R2RML mappings between relational databases and OWL ontologies, and (ii) creation and execution of SPARQL queries. The tool provides a lot of useful features that can be applied in an OBDA-based scenario.

Currently, we are implementing a graph-based method for creating SPARQL queries. In this case we will fully support a graphical environment for handling R2RML and SPARQL. We also plan to include RuQAR [1] to extend reasoning capabilities and provide support for SWRL rules [5]. Moreover, the long term plans are to support other mapping languages, like D2RQ[9] and RML [3]. As a result we will be able to map different data sources like CSV or JSON.

Acknowledgments. This research has been supported by Polish Ministry of Science and Higher Education under grant 04/45/DSPB/0149.

References

1. Bak, J.: Ruqar: reasoning with OWL 2 RL using forward chaining engines. In: ORE (2015)
2. Calvanese, D., Cogrel, B., Komla-Ebri, S., Kontchakov, R., Lanti, D., Rezk, M., Rodriguez-Muro, M., Xiao, G.: Ontop: Answering SPARQL queries over relational databases. Semantic Web, (Preprint), pp. 1–17
3. Dimou, A., Sande, M.V., Colpaert, P., Verborgh, R., Mannens, E., Van de Walle, R.: RML: a generic language for integrated RDF mappings of heterogeneous data. In: Proceedings of the 7th Workshop on Linked Data on the Web, April 2014
4. Horridge, M., Bechhofer, S.: The OWL API: a Java API for OWL ontologies. Semant. Web **2**(1), 11–21 (2011)

[9] http://d2rq.org/.

5. Horrocks, I., Patel-schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: a semantic web rule language combining OWL and RuleML (2004). Accessed 04 Apr 2013
6. Knoblock, C.A., et al.: Semi-automatically mapping structured sources into the semantic web. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 375–390. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30284-8_32
7. Neto, L.E.T., Vidal, V.M.P., Casanova, M.A., Monteiro, J.M.: *R2RML by Assertion*: a semi-automatic tool for generating customised R2RML mappings. In: Cimiano, P., Fernández, M., Lopez, V., Schlobach, S., Völker, J. (eds.) ESWC 2013. LNCS, vol. 7955, pp. 248–252. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41242-4_33
8. Poggi, A., Lembo, D., Calvanese, D., Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. In: Spaccapietra, S. (ed.) Journal on Data Semantics X. LNCS, vol. 4900, pp. 133–173. Springer, Heidelberg (2008). doi:10.1007/978-3-540-77688-8_5
9. Siciliaa, Á., Nemirovskib, G., Nolleb, A.: Map-on: A web-based editor for visual ontology mapping. Semantic Web Journal (Preprint), pp. 1–12

Compression Algorithms for Log-Based Recovery in Main-Memory Data Management

Gang Wu(✉), Xianyu Wang, Zeyuan Jiang, Jiawen Cui, and Botao Wang

College of Computer Science and Engineering, Northeastern University, Shenyang, People's Republic of China
wugang@mail.neu.edu.cn, wangxianyu04@gmail.com

Abstract. With the dramatic increases in performance requirement of computer hardware and decreases in its cost in recent years, the relevant research in main-memory database is becoming more and more popular and has a prosperous future. Log-based recovery, which is one of its most important research directions, is a set of problems accompanied by volatile memory. Its problem of stagnation in memory/CPU resulted from the slow I/O speed of non-volatile storage now needs to be addressed urgently. However, there is no specific platform for log-based recovery research. So the study aims to address this issue.

For the specific platform issue, we design and implement a simulation platform called RecoS. RecoS aims at an implementation of recovery sub-system of the main-memory database. It uses cluster substrate to simulate more real data storage and developed interfaces for a variety of recovery strategies. We propose three log compression methods in this paper: (1) the dictionary encoding, (2) the indirectly encoding with no threshold limit and (3) the indirect encoding with a threshold limit. We also adapt ARIES and command logging on the platform, which represents physical and logical logging respectively, focusing on their recovery process and some important details. Regard the recovery platform as the core to investigate the performance of the recovery platform with different load by using different log sets.

Keywords: Main-memory database · Logging · Checkpointing · Failure recovery

1 Background

The development of information technology has a great influence on all walks of life. As a representative one of various techniques, main-memory database has its outstanding advantages in access time and made a significant impact on many applications. It has been one of the important research files in the main- memory database management in recent years. Relying on the background of the main-memory database, this paper mainly studied on[1]:

[1] This paper is partially supported by the National Natural Science Foundation of China No. 61370154 and No. 61332006, and the Fundamental Research Funds for the Central Universities No. N140404009.

1. We design and implement a simulation platform called RecoS. RecoS aims at an implementation of recovery sub-system of main-memory database. It uses cluster substrate to simulate realer data storage, and developed interfaces for variety of recovery strategy.
2. We implement the important steps of physical logging and logical logging in RecoS, stressing the detailed difference caused by volatile memory in log, failure recovery and checkpointing between main-database and disk database [1].

2 Overview

RecoS is mainly used to implement the recovery sub-system in MMDB, means three key step, observing log, checkpointing and recovery. The goal of this paper is to implement the platform based on available means independently.

The superstratum of RecoS refers to the recovery part of H-Store, and using Redis as storage in the substratum. [2, 3] RecoS is supposed to simulate and implement some properties and functions such as relational, row store and cluster environment.

2.1 Architecture

RecoS can be a recovery program which is used to compare different strategies with each other. RecoS consists of master nodes which controls the program and Redis instances. Redis includes the cluster which is regarded as storage nodes(the cluster has its own distributed protocols) and the singletons which are regarded as logging record nodes. Using Redis as storage node and logging record node, and control through the master nodes, the Redis instances can be abstracted as a kind of distributed system, it can also abstracted as a kind of distributed system.

Master node: The master node keeps control of the Redis instance and get status through the network connection. It is mainly used to be responsible for all functions except storing data and log, including sending a read/write command, simulated transactions, control timing of logging reading and writing, etc. The master node is aim to implement three functions: mapping table, recovery simulator, transaction simulator. Redis cluster: The main roles Redis cluster is of are data storage and checkpointing. There are some operations for each separate node in the cluster. The connection between master nodes and sub-nodes is the only problem that the super stratum need to pay attention to. Redis logging node: Logging nodes are composed of multiple separate Redis instance. Logging nodes accept the logging of master node and finish the persistence of logging independently. Under the condition that Redis cluster nodes are used to store data, the logging nodes are used to access the log.

2.2 The Implements of Recovery Strategy

The simulate platform stores the logging dispersedly in the cluster. However, there are some differences in the recovery strategy between merging the logging and recovering the logging singly [4].

Physical logging.

(1) Logging format

The platform only record LSN, TxnID, TupleID, and OldValue(NewValue).

(2) Logging record

The logging record are stored in the logging database, then flush the redo logging into the disk of that node. The undo logging will be cleared after committing the transaction.

(3) Logging strategy

The logging can be divided into private logging and group logging. The private logging means a logging chain of a transaction maintenance. The private one will be combined with other logging into a group logging after the commit of transaction [4].

Logical logging: Compared with physical logging, logical logging has differences in format, strategy and recovery. In logging format, the logical logging record LSN, TxnID, SPP, and Params. SPP is the pointer of the store procedure which has been stored. [6] We need to add a kind of string to the logging as SPP. When the string is loaded into the master node to recover the logging, it is supposed to point out the location of stored procedure in the main-memory.

In the logging strategy, we come up with a method which only need to be located in one node. Firstly, we need to number the working nodes in the cluster. If there were two nodes in one stored procedure, we would choose the small one to record the logging.

CheckPointing: The platform uses conforming checkpointing. All the transactions in the checkpointing are in a same status. In order words, all the transactions should be ended with committing, so that we just need to REDO once. This can simplify and accelerate the recovery process [7, 8].

(1) Checkpoints in physical logging
Checkpoints in RecoS use the RDB snapshot of Redis. The system use the SAVE or BGSAVE command on the all nodes in the cluster at a given checkpoint timing. It represents that there are some persisting RDB operation in the background.
(2) Checkpoints in logical logging
Command logging need a transaction conforming checkpointing to cooperate. [9] Transaction conforming checkpointing is similar to delaying the new transactions, waiting for the ongoing transaction has been completed, then making checkpointing.

3 Algorithm

3.1 Recovery

(1) Physical logging recovery
Recovery algorithm for physical logging in RecoS

Algorithm3.1 Recovery algorithm for physical logging

```
Input: the node number N which to be recovered.
Output: null
for(each node in N):
    If (checkPointing) import the node ,and implement the recovery of checkpointing in chapter 3.3.3
        Reload the redo_log into main-memory.
Take the redo_log out from master node.
For(all nodes)
        Scan all the logging chains excpet redo_log, means scan the txn_redo logging which has not been committed.
        Destroy these private txn_redo log.
After mater node have taken all logging of broken nodes, rank the logging according to LSN.
According to the ranked redo logging, do the transaction fragments again.
```

(2) Logical logging recovery
Recovery algorithm for Logical logging in RecoS.

Algorithm3.2 Recovery algorithm for logical logging

```
Input :the number N of nodes in trouble
Output :null
If(node N is in trouble)
    Stop all nodes in the system, if there were a checkpointing, then load the checkpointing and implement the recovery in 3.3.3
Take the redo_log of all nodes, merge it with master node.
Master nodes rank the logging by LSN.
Execute unified recovery.
For(all nodes)
Scan all the logging chains excpet redo_log, and destroy private txn_redo log if possible.
```

3.2 Logging Compression

Compression is aimed to optimize the space of logging and commit ways. We put forward the coding compression method based on group commit strategy in the commit of logging process [10, 11].

(1) Dictionary encoding for private logging.
Directory encoding for private log

```
Algorithm3.4 Directory encoding for private log
Input: the time T of commit, the length L of temporary storage region
Output: a set of directory encoding and relative directory
while(in time T || TS is not full)
   TS receive the logging produced by transactions
   Query all dictionary according to txnID.
   If(the txnID is not in the dictionary)
      Record the txnID in the dictionary and then establish a new private logging
of this txnID.
      Return dictID, the direction number.
      Replace the txnID to dictID
      Put this logging into the private logging, and add 1 to the temporary storage
region.
Submit all private logging chains with the order of the dictionary
Submit the corresponding direction of the group.
Clear the temporary storage region
```

(2) Indirect encoding with threshold limit.
In indirect encoding, suppose that we would like to divide the data into blocks in a same size, if there were less different values in a block, we could compress this block in dictionary encoding, the opposite is not.

(1) Finish a group submission in size L and time T, no matter the temporary storage region is full or not. The logging must be committed when time T out. Set two pointers, p1 and p2, points to the head and tail of the temporary storage region. P1 moves from head to tail, pointing to the position where the logging will be stored. P2 moves from tail to head, pointing to the logging beyond threshold value.
(2) When the logging is to fill the i block, we establish the local dictionary di and a table ti which can only accommodate t value for the range block and the initial value of these table is empty.
(3) When writing logging items into main-memory, we record the TxnID in the table and follow the rules below in block i.
If ti is not full, then write the value into the table, establish an item in the dictionary and write the logging into the block, p1 move to the next position.

If there is such value in ti, write the logging into the block, p1 move to the next position.
If ti is not full and there is not such a value in ti, write the logging into the position that p2 points to, then p2 move to the former position.

(4) If pi point to the next block, then rebuild di + 1and ti + 1, and now ti is useless, return to step2. If pointers meet or time out, end the algorithm, commit the partly logging, the information of pointers in the temporary storage region.

4 Experiments

We use the method of group submission in the experiment. [12] We no longer to consider whether the transaction are rolled back in order to pay attention to the compression of transaction logging encoding. In addition, in a simple submission and submission with the dictionary encoding group, the time occupancy we use for the encoding is not obvious high. Considering the loggings submission, transaction concurrency taking up too much time, the advantages of no-encoding have already been basically eliminated.

4.1 Experimental Environment

The Redis cluster in the experiment contains three nodes. Each node matches to a logging node, logging nodes are directly connected to the master node.

The master node use an 8 GB memory, a disk of 512 GB and Windows8.1 as operating system. Cluster nodes and logging nodes use a memory of 10 GB, a disk of 20 GB, and CentOS6.5 operating system. Nodes in the cluster use Redis3.0 version, other configuration is the same as the default cluster configuration. Logging nodes are separate Redis3.0 instance with the default configuration. Jedis version is 2.7.0.

4.2 Group Commit of Dictionary Encoding

Figure 1 shows the size of L and the relationship of the space taken up by logging and dictionary after dictionary encoding. In this experiment, the number of transaction is fixed to 50, each transaction logging has an average number of 20 and each logging is about 60 B. We will discuss compression effect on these 1000 loggings in this paper.

In Fig. 1, the “uncompressed” part of the logging remains at 69686 bytes, all the stores is loggings. The “compressed” part of the storage is the corresponding loggings and dictionary, the results is about less than 68000 bytes. If the number of logging is overmuch in each group, compression effect not rises linearly, such as compression rate rises steadily from 20 to 500 in abscissa, but fall from 500 to 1000. This illustrates that if the global dictionary contains too many entries, it may affect the final result. At the L = 500, compression rate is about 95.5%.

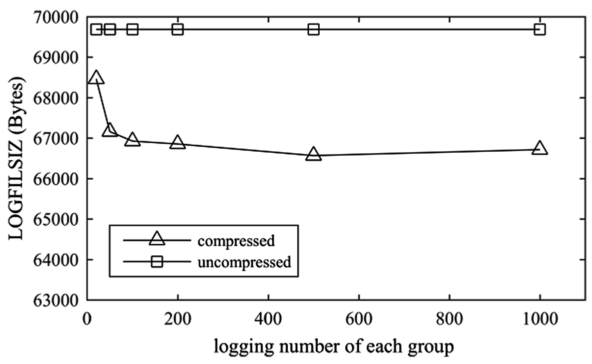


Fig. 1. Group commit with dictionary encoding

4.3 Group Commit of Indirect Encoding

Indirect encoding without threshold limit.
In Fig. 2, the experiment is based on the combination of different transaction number and the average logging each transaction produced. The diagram shows that indirect encoding has advantage on the compression when transactions are more and the average loggings are less.

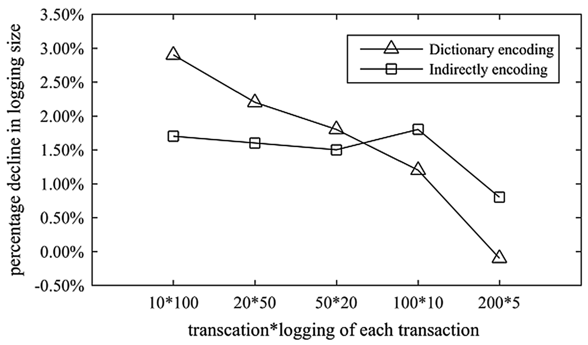


Fig. 2. A comparison between indirect and dictionary encoding on the combination of different transaction number and log per transaction

Indirect encoding with threshold limit.
Encoding with a threshold limit have obvious advantages when the part of to be compressed is lower. If the logging is 60 bytes. The concurrent transactions increase but the compressible part become less. Using the dictionary encoding and indirect encoding are unable to achieve the effect of compression, and indirect encoding with threshold limit doesn't produce dictionary for block, then avoid the overhead of the dictionary, and this makes some effect of compression.

4.4 Comparing of Recovery

We compare the recovering efficiency from no-compression, dictionary encoding and indirect encoding. Regarding the platform as the core, we inspect the performance through different loggings scale in different size of the recovery platform load. As shown in the Fig. 3, there is a list of the recovery time in a scale of 1000, 3000, 6000, 10000, 20000, and 50000.

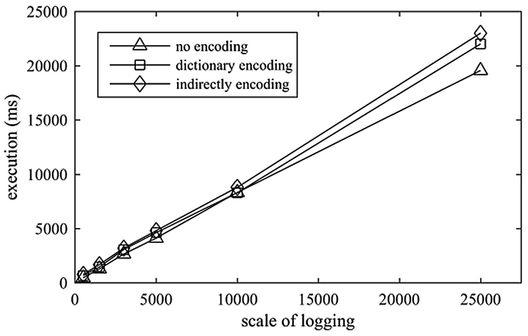


Fig. 3. Comparison with several methods the recovery of submission

5 Conclusions

This article has studied study the traditional and active logging recovery from the availability of the main-memory database. The paper mainly stated the two aspects:

1. In order to study the recovery strategy of main-memory database, especially in the cluster, we design and implement a simulate platform—RecoS. The platform is aimed to separate user's focus from the multifarious database system to the deep study of recovery sub-system. We use Redis to store data and logging, and the client logic program to control the nodes.
2. As to logging recovery, the strategies that are not based on logging will be a main direction of main-memory data recovery development.

References

1. Woo, S., Ho Kim, M., Joon Lee, Y.: Accommodating logical logging under fuzzy checkpointing in main memory databases. In: Proceedings of the International Database Engineering and Applications Symposium. IDEAS 1997. IEEE, pp. 53–62 (1997)
2. Kallman, R., Kimura, H., Natkins, J., et al.: H-store: a high-performance, distributed main memory transaction processing system. Proc. VLDB Endowment **1**(2), 1496–1499 (2008)

3. Stonebraker, M., Madden, S., Abadi, D.J., et al.: The end of an architectural era: (it's time for a complete rewrite). In: Proceedings of the 33rd International Conference on Very Large Data Bases. VLDB Endowment 2007, pp. 1150–1160 (2007)
4. Yao, C., Agrawal, D., Chen, G., et al.: Adaptive logging: optimizing logging and recovery costs in distributed In-memory databases. In: International Conference (2016)
5. Mohan, C., Haderle, D., Lindsay, B., et al.: ARIES: a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging. ACM Trans. Database Syst. (TODS) **17**(1), 94–162 (1992)
6. Stonebraker, M., Weisberg, A.: The VoltDB main memory DBMS. IEEE Data Eng. Bull. **36** (2), 21–27 (2013)
7. Salem, K., Garcia-Molina, H.: Checkpointing memory-resident databases. In: Proceedings of the Fifth International Conference on Data Engineering, pp. 452–462. IEEE (1989)
8. Elnozahy, E.N., Johnson, D.B., Zwaenepoel, W.: The performance of consistent checkpointing. In: Proceedings of the 11th Symposium on Reliable Distributed Systems, pp. 39–47. IEEE(1992)
9. Malviya, N., Weisberg, A., Madden, S., et al.: Rethinking main memory OLTP recovery. In: 2014 IEEE 30th International Conference on Data Engineering (ICDE), pp. 604–615. IEEE (2014)
10. Stonebraker, M., Abadi, D.J., Batkin, A., et al.: C-store: a column-oriented DBMS. In: Proceedings of the 31st International Conference on Very Large Data Bases. VLDB Endowment, pp. 553–564 (2005)
11. Abadi, D.J., Boncz, P.A., Harizopoulos, S.: Column-oriented database systems. Proc. VLDB Endowment **2**(2), 1664–1665 (2009)
12. Garcia-Molina, H., Salem, K.: Main memory database systems: an overview. IEEE Trans. Knowl. Data Eng. **4**(6), 509–516 (1992)

Linked Data

An Empirical Study on Property Clustering in Linked Data

Saisai Gong, Haoxuan Li, Wei Hu(✉), and Yuzhong Qu

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
ssgong.nju@gmail.com, hxli.nju@gmail.com, {whu,yzqu}@nju.edu.cn

Abstract. Properties are used to describe entities, a part of which are likely to be clustered together to constitute an aspect. For example, first name, middle name and last name are usually gathered to describe a person's name. However, existing automated approaches to property clustering remain far from satisfactory for an open domain like Linked Data. In this paper, we firstly investigated the relatedness between properties using five different measures. Then, we employed three clustering algorithms and two combination methods for property clustering. Based on a moderate-sized sample of Linked Data, we empirically studied the property clustering in Linked Data and found that a proper combination of different measures gave rise to the best result. Additionally, we showed how the property clustering can improve user experience in our entity browsing system.

Keywords: Property clustering · Property relatedness · Entity browsing · Empirical study · Linked data

1 Introduction

With the development of Linked Data, billions of RDF triples have been published to describe numerous entities. An entity usually involves multiple aspects and its property-values may focus on different aspects. For instance, graduate from and work at reveal the career information of a person, while parent, spouse and child deliver her family information. Therefore, it is natural to cluster properties into meaningful groups based on the aspects that they intend to describe. Property clustering is useful for many applications such as entity browsing, ontology editing, query completion, etc. It makes the presented information more formatted and understandable and significantly enhances the capability of users to consume the large-scale Linked Data [8].

Take, for example, the case of entity browsing. Many state of the art systems support users to manually cluster properties [13]. But due to the limited energy and knowledge of the users, this type of manual operations is only effective at a small scale. In consideration of an open domain like Linked Data, automated property clustering is needed to solve the scalability issue, but its performance

© Springer International Publishing AG 2016
Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 67–82, 2016.
DOI: 10.1007/978-3-319-50112-3_6

is still far from satisfactory. One reason is the fact that, when browsing entities, the data is multi-sourced and the vocabularies involved are barely predictable, i.e. probably use any vocabularies, thus it is difficult to find out useful patterns among properties in advance and make use of them to guide clustering. Another reason is that the properties used by the entities are largely heterogeneous, which makes identifying similar aspects more difficult and less reliable.

In this paper, we empirically studied the property clustering in Linked Data, which is defined as to automatically assign a subset of property set $\mathbf{P}$ (collected from the descriptions of entities) to a set of disjoint clusters $\{g_1, g_2, \ldots, g_m\}$, such that the properties in a cluster g_k focus on an aspect or a dimension of the content. But, $g_1 \cup g_2 \cup \ldots \cup g_m$ is unnecessarily equal to $\mathbf{P}$.

In order to achieve property clustering, we firstly measured the relatedness of properties from five perspectives: lexical similarity between property names, semantic relatedness between property names, distributional relatedness between properties, range relatedness between properties and overlap of property values. We then employed three widely-used algorithms, namely density-based clustering, hierarchical clustering and spectral clustering. Furthermore, to combine various relatedness measures and clustering results, we developed two combination methods based on linear combination and consensus clustering.

We sampled a moderate-sized dataset of Linked Data for our empirical study and manually built gold standards to assess the relatedness measures, clustering algorithms and combination methods. Moreover, we integrated the property clustering in our entity browsing system called SView[1], in order to observe the use of property clustering in practice. Overall, we tried our best in this study to provide answers to the following questions:

Q1. What is the most effective measure(s) for measuring the relatedness between properties?
Q2. What is the most effective algorithm(s) for clustering properties into meaningful groups?
Q3. Can the combination method(s) improve the property clustering and how largely?
Q4. Are there any general principles or guidelines for using the property clustering in practice?

In the rest of this paper, we first introduce the measures of property relatedness in Sect. 2. Then, we present the clustering algorithms in Sect. 3 and the combination methods in Sect. 4. Our experimental results are reported in Sect. 5. We show the application of property clustering to entity browsing in Sect. 6 and discuss related work in Sect. 7. Finally, we summarize our findings in Sect. 8 and conclude this paper in Sect. 9.

2 Property Relatedness Measures

In this section, we present five types of relatedness between properties and formalize them as numerical measures. All the measures are assumed non-negative.

[1] http://ws.nju.edu.cn/sview.

2.1 Lexical Similarity Between Property Names

A property is usually associated with several human-readable names, e.g. labels. When the names of two properties share many common characters, it indicates some kind of relatedness between their meanings. For example, both mouth position and mouth elevation describe the mouth information of a river. By detecting this aspect, we took advantage of I-Sub string measure [16] to characterize the lexical similarity. Let p_i, p_j be two properties and l_i, l_j be the names of p_i, p_j respectively, the *lexical similarity* between p_i and p_j, denoted by R_I, is measured by I-Sub as follows:

$$\begin{aligned} R_I(p_i, p_j) &= \textit{I-Sub}(l_i, l_j) \\ &= Comm(l_i, l_j) - Diff(l_i, l_j) + Winkler(l_i, l_j), \end{aligned} \tag{1}$$

where $Comm(l_i, l_j)$ represents the commonality of l_i, l_j, while $Diff(l_i, l_j)$ represents their difference. $Winkler(l_i, l_j)$ is a coefficient to adjust the result. The score of I-Sub was normalized from $[-1, 1]$ to $[0, 1]$; a larger value implies a higher similarity.

2.2 Semantic Relatedness Between Property Names

WordNet provides several semantic relations among concepts/words, e.g. hyperonymy, and it is widely considered as a knowledge base for measuring semantic relatedness between words [3], based on shortest paths, information theory, etc. To measure the WordNet-based relatedness of two properties p_i, p_j, we transformed their property names to the normalized forms by splitting names, removing stop words and stemming. Let l'_i, l'_j be the normalized forms of the names of p_i, p_j, respectively. $|.|$ counts the number of words in a normalized name. The *semantic relatedness* between p_i and p_j, denoted by R_W, is calculated based on WordNet 3.0 as follows:

$$R_W(p_i, p_j) = \min\left(\frac{\sum_{x \in l'_i} \max_{y \in l'_j} Lin(x, y)}{|l'_i|}, \frac{\sum_{y \in l'_j} \max_{x \in l'_i} Lin(x, y)}{|l'_j|}\right), \tag{2}$$

where $Lin(x, y)$ denotes the Lin's WordNet-based word relatedness between x and y [11], which is based on the shortest path and information theory.

2.3 Distributional Relatedness Between Properties

In the area of computational linguistics, distributional relatedness [5] is a measure of word relatedness in distributional semantics, through word co-occurrence in different contexts such as bigrams, sentences or documents. Based on the selected context, the strength of relatedness between co-occurrent words is quantified by using mutual information or other measures. Inspired by this research line, we studied property co-occurrence in use and conceived an entity's RDF description as the context from which co-occurrence is found, i.e. properties are used together to describe the entity. For example, founded by and key person co-occur to describe important people in a company. Specifically, we used the *symmetrical uncertainty coefficient*, denoted by R_U, which computes the strength of relatedness between variables in terms of normalized mutual information. Given two properties p_i, p_j, the *distributional relatedness* between p_i and p_j, denoted by R_U, is calculated using the symmetrical uncertainty coefficient as follows:

$$R_U(p_i, p_j) = 2 \cdot \frac{H(p_i) + H(p_j) - H(p_i, p_j)}{H(p_i) + H(p_j)}, \tag{3}$$

where $H(p_i) = -\sum_{x \in \{p_i, \overline{p_i}\}} P(x) \log P(x)$ obtains the entropy of p_i, $H(p_i, p_j) = -\sum_{x \in \{p_i, \overline{p_i}\}} \sum_{y \in \{p_j, \overline{p_j}\}} P(x, y) \log P(x, y)$ counts the joint entropy of p_i, p_j. The score of R_U is in $[0, 1]$; a higher value indicates a stronger relatedness.

To obtain R_U, we need the probabilities $P(p_i), P(p_i, p_j), P(p_i, \overline{p_j}), P(\overline{p_i}, \overline{p_j})$. Let $\mathbb{G}$ be the RDF dataset from which the probabilities would be estimated and p_i, p_j be two properties. $Res(\mathbb{G})$ denotes the entities appearing in the subject or object position of any RDF triples in $\mathbb{G}$, and $ResDesc(\mathbb{G}, p_i)$ be the entities in $\mathbb{G}$ that is particularly described by p_i (i.e. p_i appears at the predicate position of any RDF triples). The probabilities $P(p_i), P(p_i, p_j), P(p_i, \overline{p_j})$ and $P(\overline{p_i}, \overline{p_j})$ are estimated as follows:

$$P(p_i) = \frac{|ResDesc(\mathbb{G}, p_i)|}{|Res(\mathbb{G})|}, \tag{4}$$

$$P(p_i, p_j) = \frac{|ResDesc(\mathbb{G}, p_i) \cap ResDesc(\mathbb{G}, p_j)|}{|Res(\mathbb{G})|}, \tag{5}$$

$$P(p_i, \overline{p_j}) = \frac{|ResDesc(\mathbb{G}, p_i) \cap (Res(\mathbb{G}) - ResDesc(\mathbb{G}, p_j))|}{|Res(\mathbb{G})|}, \tag{6}$$

$$P(\overline{p_i}, \overline{p_j}) = \frac{|Res(\mathbb{G}) - (ResDesc(\mathbb{G}, p_i) \cup ResDesc(\mathbb{G}, p_j))|}{|Res(\mathbb{G})|}. \tag{7}$$

We leveraged the Billion Triples Challenge (BTC) 2011 dataset[2], a representative subset of the Linked Data to estimate the above probabilities. As different URIs may refer to the same entity, which are called *coreferent URIs*, we firstly found out the coreferent URIs and then merged their RDF descriptions by replacing coreferent URIs with a uniform ID. We used two kinds of ontology semantics owl:sameAs and inverse functional properties, and computed a transitive closure to identify coreferent URIs. More sophisticated coreference resolution algorithms can be found in [9]. This modified dataset would be treated as the RDF dataset $\mathbb{G}$ as described above.

2.4 Range Relatedness Between Properties

Property ranges may be URIs of related types (i.e. classes), which indicate certain kind of relatedness as well. For example, if two properties have the ranges

[2] http://km.aifb.kit.edu/projects/btc-2011/.

delicious food and handicraft respectively, both of them deliver the tourist information of a tourist city. We leveraged the ranges of properties to measure their relatedness, except rdfs:Resource and owl:Thing. If no axioms involving property ranges can be found, we used the conjunction of the types of the property values instead. Let p_i, p_j be two properties and $\mathbf{T}_i, \mathbf{T}_j$ be the sets of ranges for p_i, p_j respectively, the *range relatedness* between p_i, p_j, denoted by R_T, is calculated as follows:

$$R_T(p_i, p_j) = \max_{c_i \in \mathbf{T}_i, c_j \in \mathbf{T}_j} R_W(c_i, c_j), \tag{8}$$

where $R_W(c_i, c_j)$ reuses Eq. (2) to compute the WordNet-based relatedness of the names of c_i, c_j.

Many other relations exist between properties, such as sub-/super-properties or domain relatedness, which indicate the strength of property relatedness as well. For example, both medalist and champion are sub-properties of has participant, and they give the winner information of a sport event participants. A part of super-properties exist in ontology axioms, however, a larger amount of essential super-properties that can be used for property clustering by semantic relatedness are not defined formally in ontologies and thus only latent. For instance, in most ontologies, the three properties length, width and depth do not have a super-property like physical dimension. Besides, the work in [9] observed that property domains are not as useful as ranges. Therefore, we do not consider those relatedness using other relations presently.

2.5 Overlap of Property Values

There may also exist synonymous properties to describe the same entity. For example, both has book and write describe a book written by an author. In this case, common values should be frequently shared by these properties. We used the vector space model (specifically, the TF/IDF model) to represent the values of a property. The text of each property value is collected, e.g. local names of URIs and lexical forms of literals after normalization, and all the terms in the text are used to construct a term frequency vector, where each component corresponds to the number of occurrences of a particular term. Given two properties p_i, p_j, the *overlap of property values* between p_i, p_j, denoted by R_O, is computed by the cosine similarity of the corresponding vectors $\mathbf{v}_i, \mathbf{v}_j$:

$$R_O(p_i, p_j) = \frac{\sum_{k=1}^{n} v_{ik} \cdot v_{jk}}{\sqrt{\sum_{k=1}^{n} v_{ik}^2} \cdot \sqrt{\sum_{k=1}^{n} v_{jk}^2}}, \tag{9}$$

where n is the dimension of the vector space and v_{ik}, v_{jk} are the components of the vectors $\mathbf{v}_i, \mathbf{v}_j$.

3 Property Clustering Algorithms

To work with an arbitrary relatedness measure, we employed the following three well-known clustering algorithms in Weka 3 [15], which use the previously computed relatedness as input and generate a set of property clusters.

DBSCAN, denoted by C_D, finds clusters based on the density of properties in a region. Its key idea is for each property in a cluster, the neighborhood of a given radius (Eps) has to contain at least a minimum number of properties ($MinPts$). In other words, each non-trivial cluster in the result must own at least $MinPts$ properties.

Single linkage clustering, referred to as C_L, is an agglomerative hierarchical algorithm, which repeatedly merges two most related clusters in a bottom-up fashion until meeting some criteria. The single linkage relatedness of two clusters is derived from the two most related properties in the two clusters. The single linkage clustering is terminated when the maximum relatedness of any two clusters is no greater than a threshold θ.

Spectral clustering, denoted by C_S, leverages the spectrum of a relatedness matrix of properties to divide them into clusters. A threshold η for cluster number needs to be pre-defined.

Except the aforementioned parameters and thresholds, we kept the default settings in Weka for the three clustering algorithms.

4 Combination Methods

Combining various relatedness measures helps obtain a property clustering with better accuracy and coverage. There exist two typical methods to conduct combination. One is to combine the measures before clustering, for example, to use a *linear combination* of different relatedness measures for each property pair and carry out a clustering algorithm to produce clusters. Given the five relatedness measures, the combined relatedness, denoted by R_{all}, is defined as follows:

$$\begin{aligned} R_{all}(p_i, p_j) = {} & \omega_1 R_I(p_i, p_j) + \omega_2 R_W(p_i, p_j) + \omega_3 R_U(p_i, p_j) \\ & + \omega_4 R_T(p_i, p_j) + \omega_5 R_O(p_i, p_j), \end{aligned} \tag{10}$$

where $\omega_i \in [0, 1]$ denotes the weight coefficient value for a specific property relatedness measure, and $\sum_{i=1}^{5} \omega_i = 1$. In this study, we investigated various values for linearly combining our relatedness measures.

Another method is to first conduct clustering based on individual relatedness measures and then aggregate these individual results using *ensemble clustering*. In this study, we selected *consensus clustering* [1] to realize ensemble clustering. Given a set of individual clusterings corresponding to different relatedness measures, the goal of computing a consensus clustering is to achieve a clustering that minimizes the distance among individual clusterings. The problem of finding an optimal consensus clustering is NP-hard. We implemented CC-Pivot [1], a 3-approximation algorithm, to calculate the consensus clustering.

5 Empirical Study

In this section, we report our study of the relatedness measures, clustering algorithms and combination methods for the property clustering in Linked Data. The source code and sample data for this empirical study are all available at our website[3].

5.1 Dataset

We randomly sampled 20 entities of various types (classes) in Linked Data, each of which is integrated from a DBpedia URI with its coreferent URIs that refer to the same entity using owl:sameAs relations. The finally-selected URIs were required to be accessible via HTTP protocol (to eliminate outdated ones), and have sufficient properties and values, i.e. having more than 50 properties. Overall, the 20 entities involve 12 sources: DBpedia, DBTune, Freebase, GeoNames, LinkedGeoData, LinkedMDB, New York Times, OpenCyc, Project Gutenberg, RDF Book Mashup, The World Factbook and YAGO. We distinguished properties in the forward and backward directions, and considered that different directions of the same property represent different properties. The properties holding a sample entity at the subject position is referred to as the *forward* properties, while at the object position is referred to as the *backward* properties. Table 1 lists the names of the entities with their types and numbers of properties. Note that an entity can have multiple types and we just show an important one.

Table 1. 20 sample entities with their types and numbers of properties

Entity name	Type	#Prop.	Entity name	Type	#Prop.
Hong Kong Airport	Airport	95	Bob Jones University	Institution	130
Michael Nesmith	Artist	145	British Museum	Museum	110
Jeremy Shockey	Athlete	107	Load (album)	MusicalWork	91
Deep Purple	Band	108	Edmund Stoiber	Politician	82
A Clockwork Orange	Book	61	Amazon River	River	142
The Pentagon	Building	110	William H. Holmes	Scientist	65
Baltimore	City	351	Polymelus	Species	51
Adobe Systems	Company	127	Doom II	Software	79
Finland	Country	574	Burlington Township	Township	62
Eyes Wide Shut	Film	99	Barney & Friends	TVShow	79

5.2 Experiment Setup

To observe the prevalence of property clustering in Linked Data and assess the effectiveness of the relatedness measures, clustering algorithms and combination

[3] http://ws.nju.edu.cn/sview/propcluster.zip.

methods, we sought to build for each sample entity a reference clustering that is meaningful, aspect-coherent and compact, so as to compare the algorithmically-generated clustering with the reference ones. Due to the large number of properties (shown in Table 1), it is hard to ask users to manually build the reference clustering. Hence, we did not start from scratch but leveraged existing reasonably good clustering.

Freebase divides properties describing similar aspects into *types* and groups similar types into *domains*[4]. For example, /music/group_member describes the member information of a music group, where group_member is a type and music is a domain. Thus, we invited three PhD candidates in the field of Linked Data to assign each property of a sample entity to the most relevant /domain/type, for example assign a property band member to /music/group_member, and created the reference clustering such that properties are clustered together if they are assigned to the same /domain/type. The average inter-rater agreement score of the 20 entities, measured by Fleiss' κ [6], is 0.895, and the minimum inter-rater agreement score for an entity is 0.814. From the high inter-rater agreement score, we saw that strong agreement exists among the three judges, which guarantees the statistical significance of our empirical study.

By using the reference clusterings as our golden standard, we evaluated the algorithmically-generated property clustering in terms of the following five metrics: *Precision*, *Recall*, *F-Score*, *Rand Index* and *Normalized Mutual Information* (NMI). These metrics are the well-known criteria assessing how well a clustering matches the golden standard. For a clustering π, $S(\pi)$ gives the total number of property pairs in the same clusters:

$$S(\pi) = \{(p_i, p_j) \mid \exists g_k \in \pi,\, p_i, p_j \in g_k,\, i < j\}. \tag{11}$$

Let π_{gs} be a golden standard clustering. The Precision, Recall and F-Score for a computed clustering π w.r.t. π_{gs} are calculated as follows:

$$\text{Precision} = \frac{S(\pi) \cap S(\pi_{gs})}{S(\pi)}, \tag{12}$$

$$\text{Recall} = \frac{S(\pi) \cap S(\pi_{gs})}{S(\pi_{gs})}, \tag{13}$$

$$\text{F-Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{14}$$

Rand Index penalizes both false positive and false negative decisions in clustering, while NMI can be information-theoretically interpreted. Their values are both rational numbers in $[0, 1]$ range, and a higher value indicates a better clustering. We refer the reader to [17] for the detailed calculation.

5.3 Results of Relatedness Measures and Clustering Algorithms

We clustered the properties of the 20 sample entities by using each relatedness measure and clustering algorithm, and computed the harmonic mean (h-mean)

[4] https://developers.google.com/freebase/guide/basic_concepts.

of Precision, Recall, F-Score, Rand Index and NMI, respectively. To obtain the optimal parameters of a clustering algorithm, we enumerated the values of Eps and $MinPts$ of DBSCAN in $\{0.6, 0.7, 0.8, 0.9\}$ and $\{2, 3, 4, 5\}$ respectively, θ of single linkage in $\{0.6, 0.7, 0.8, 0.9, 1.0\}$, η of spectral clustering from 5 to 20 with 1 interval, and selected the parameters that achieved the highest h-mean of F-Score for the 20 entities. As a result, we set $MinPts = 2, Eps = 0.9, \theta = 0.1$ and $\eta = 5$.

Table 2. Average performance w.r.t. relatedness measures and clustering algorithms

(a) Precision

	C_D	C_L	C_S
R_I	.235	.235	.184
R_W	.215	.215	.198
R_U	.242	.242	.177
R_T	.170	.170	.215
R_O	.247	.247	.188

(b) Recall

	C_D	C_L	C_S
R_I	.273	.273	.449
R_W	.266	.266	.337
R_U	.433	.433	.410
R_T	.381	.381	.329
R_O	.137	.138	.427

(c) F-Score

	C_D	C_L	C_S
R_I	.253	.253	.261
R_W	.238	.238	.250
R_U	.310	.310	.248
R_T	.235	.235	.260
R_O	.176	.177	.261

(d) Rand Index

	C_D	C_L	C_S
R_I	.549	.549	.500
R_W	.672	.672	.584
R_U	.644	.644	.503
R_T	.547	.547	.628
R_O	.709	.708	.516

(e) NMI

	C_D	C_L	C_S
R_I	.387	.387	.229
R_W	.441	.441	.231
R_U	.507	.507	.224
R_T	.364	.364	.255
R_O	.520	.520	.216

Table 2 depicts the result of this experiment, where each row and column represent the relatedness measure and clustering algorithm used, respectively. From the first two columns of each table, we found that the results of DBSCAN (C_D) and single linkage (C_L) are very similar. In fact, when the value of $MinPts$ of DBSCAN is set to 2, DBSCAN is nearly identical to single linkage. From Table 2(a), R_I, R_U and R_O achieved the highest values in Precision by using C_D or C_L. Among them, R_U and R_I also achieved high Recall using either clustering algorithm, as shown in Table 2(b), and thus both of them achieved the highest values in F-Score as shown in Table 2(c). R_T achieved low Precision, but had a relatively good Recall. Although the F-Score of R_T seems good in Table 2(c), the values of R_T in Rand Index and NMI are almost the lowest, as listed in Table 2(d) and (e). To summarize, R_I, R_U and R_O may be the most effective measures for clustering in terms of Precision in general. R_U, R_T and R_I may be the most effective measures in terms of Recall in general. R_U and R_I may be the most effective measures in terms of F-Score. These best measures may vary w.r.t the nature of clustering algorithms used.

From the third column of each table, we saw that the performance of spectral clustering (C_S) is very different from C_D (or C_L). R_T may be the most effective measure for clustering in terms of Precision in general. R_U, R_I and R_O may be the most effective measures in terms of Recall. R_I, R_T and R_O may be the most effective measures in terms of F-Score. Also, we can see that the Rand Index and NMI of C_S are generally much lower than the other two algorithms. These indicate that DBSCAN and single linkage performed better than spectral clustering in our evaluation.

5.4 Results of Combination Methods

The second experiment is to evaluate whether a certain combination method can improve the performance of clustering. In this experiment, we used DBSCAN (C_D) with $MinPts = 2$ and $Eps = 0.9$, and spectral clustering (C_S) with $\eta = 5$ as the clustering algorithms since these parameter settings generally achieved good F-Score according to the results of the previous experiment.

The total number of possible linear combinations of five measures is 26 ($= 2^5 - 1 - 5$), which is large. So we did not investigate all these possible combinations (Recall that we intend to investigate how a proper linear combination improves the performance compared to single measures, not for the best combination on specific datasets). Instead, for C_D we tried to find whether the linear combination of R_I, R_U and R_O (each of which has a relatively high Precision) can improve the performance, and whether R_T or R_W can improve the performance when combined with R_I, R_U and R_O. As a result, seven linear combinations were investigated in Table 3 (C_D), where $\omega_i \in [0, 1]$ is a weight coefficient. For C_S, similar to the C_D, we tried to find seven linear combinations as well. Because the values of the measures in Precision are similar, we chose to try the linear combination of R_I, R_T and R_O, each of which has a relatively high F-Score instead of Precision. Finally, seven linear combinations were investigated in Table 3 (C_S), where $\omega_j \in [0, 1]$ is also a weight coefficient like ω_i as aforementioned. We enumerated each weight coefficient value ω_i or ω_j from 0 to 1 with 0.05 interval and finally selected the combinations that achieved the highest F-Score in average. Additionally, we investigated the performance of the consensus clustering induced by using the corresponding measures. The average runtime of a clustering algorithm on the 20 entities is in 40 s.

Table 3 shows the harmonic means of Precision, Recall, F-Score, Rand Index and NMI achieved by using single measures for C_D and C_S, linear combinations of various measures and ensemble clustering. For each table, the 6th to 12th rows represent the linear combination with their weight coefficient values, and the 13th to 19th rows represent the performance of consensus clustering induced by using various measures.

For C_D, we observed that the linear combination of different measures may greatly improve the Recall values (about 50% from 0.433 to 0.899 in some cases). However, the linear combination achieved lower Precision as compared with individual measures. Furthermore, it also has a substantial decrease in the values of Rand Index and NMI (about 50% from 0.709 to 0.364 in some cases on Rand Index and about 50% from 0.520 to 0.268 in some cases on NMI). These results may be due to that different measures complemented each other to cover more

Table 3. Comparison on single relatedness measures and two combination methods

Clustering algorithm: C_D	Precision	Recall	F-Score	Rand Index	NMI
R_I	.235	.273	.253	.549	.387
R_W	.215	.266	.238	.672	.441
R_U	.242	.433	.310	.644	.507
R_T	.170	.381	.235	.547	.364
R_O	.247	.137	.176	.709	.520
$.3R_I + .7R_U$	.218	.757	.339	.471	.379
$.5R_I + .5R_O$	.209	.619	.313	.411	.265
$.6R_U + .4R_O$	.214	.716	.330	.477	.375
$.3R_I + .5R_U + .2R_O$	.211	.883	.341	.398	.318
$.3R_I + .5R_U + .1R_T + .1R_O$	.205	.878	.333	.372	.277
$.2R_I + .1R_W + .2R_U + .5R_O$	.216	.790	.339	.438	.344
$.2R_I + .1R_W + .15R_U + .1R_T + .45R_O$	.207	.899	.337	.364	.268
R_I, R_U	.287	.148	.196	.732	.563
R_I, R_O	.331	.051	.089	.744	.566
R_U, R_O	.290	.066	.108	.755	.575
R_I, R_U, R_O	.273	.210	.237	.706	.513
R_I, R_U, R_T, R_O	.292	.102	.151	.744	.560
R_I, R_W, R_U, R_O	.290	.115	.165	.726	.548
R_I, R_W, R_U, R_T, R_O	.256	.213	.232	.677	.493
Clustering algorithm: C_S	Precision	Recall	F-Score	Rand Index	NMI
R_I	.184	.449	.261	.500	.229
R_W	.198	.337	.250	.584	.231
R_U	.177	.410	.248	.503	.224
R_T	.215	.329	.260	.628	.255
R_O	.188	.427	.261	.516	.216
$.75R_I + .25R_T$	.209	.410	.277	.563	.269
$.8R_T + .2R_O$	.215	.356	.268	.613	.255
$.7R_I + .3R_O$	.194	.449	.271	.520	.242
$.7R_I + .25R_T + .05R_O$	.206	.418	.276	.558	.278
$.2R_I + .5R_U + .2R_T + .1R_O$	.209	.392	.272	.585	.285
$.7R_I + .1R_W + .1R_T + .1R_O$	.191	.464	.271	.497	.249
$.2R_I + .1R_W + .5R_U + .1R_T + .1R_O$	.198	.455	.276	.499	.234
R_I, R_T	.181	.211	.195	.620	.331
R_T, R_O	.237	.167	.196	.705	.376
R_I, R_O	.218	.160	.184	.691	.361
R_I, R_T, R_O	.184	.298	.228	.552	.268
R_I, R_U, R_T, R_O	.200	.151	.172	.681	.361
R_I, R_W, R_T, R_O	.202	.129	.158	.691	.362
R_I, R_W, R_U, R_T, R_O	.207	.213	.210	.662	.320

types of properties describing similar aspects while bringing noises. The ensemble clustering based on various measures has an increase in Precision (0.247 to 0.331 in some cases) without loss of Rand Index and NMI, due to the fact that two properties were assigned to the same cluster by ensemble clustering only if there is sufficient number of individual clustering results. This indicates that, when seeking for a clustering with a high Precision for C_D, the ensemble clustering may be better than the linear combination, while the linear combination tends to find a clustering with a high Recall.

For C_S, we found that the linear combination of different measures cannot improve the performance for C_S. The ensemble clustering based on various measures has an great decrease in Recall (about 60% from 0.449 to 0.184 in some cases). This indicates that the combination methods may not be helpful for improving the performance for C_S. The bad results may be related to the similarity of these measure for C_S.

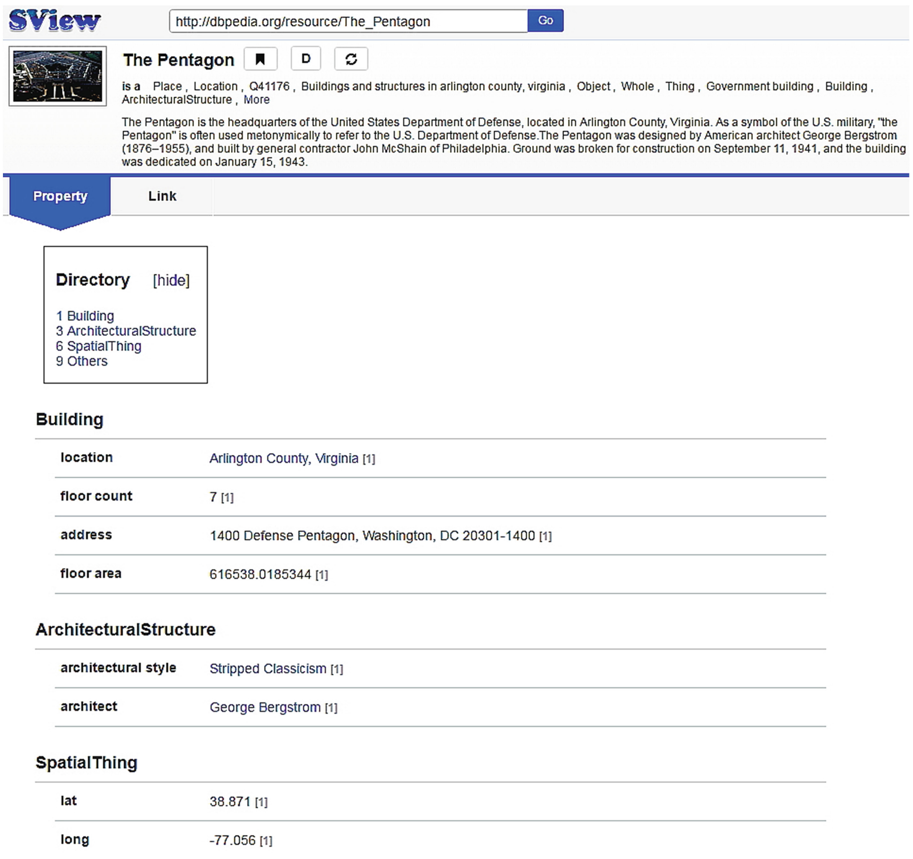
6 Application to Entity Browsing

For years, it has been a great challenge to provide general users with smart views for browsing interlinked RDF descriptions of entities. As a use scenario of property clustering, we developed an online system called SView for browsing linked entities. It groups and orders property-values of entities by lenses for a neat presentation, and offers various mechanisms for discovering related entities such as exploration based on link patterns and similarity-based entity recommendation. Moreover, users can personalize their browsing experience and they never work alone. They can edit lenses and consolidate entities following their own opinions, and their efforts are alleviated due to crowdsourced contributions from all users. Additionally, SView leverages users' contributions to generate smart views, e.g. global lenses and global viewpoints on entity consolidation. The smart views are, in turn, shared among all users when browsing linked entities.

Figure 1 shows the screenshot for viewing an entity "The Pentagon"[5] in SView. Since this entity contains hundreds of property-values, it is not very readable if there is no appropriate organizing method. To address this issue, SView groups and orders property-values with lenses (e.g. "Building"), which reuse property clustering to describe closely related aspects of an entity and thus to help users capture related information quickly. A weighted set cover problem is formulated and solved to automatically pick up a small number of the lenses that can cover as many relevant properties as possible.

We invited 24 master students in computer science to compare the presentation of 10 DBpedia entities with and without property clustering in SView. The SUS (System Usability Scale) scores in average indicated that nearly 16% improvement can be achieved by using the property clustering (72.85 versus 62.86), and this result is statistically significant ($p < 0.05$).

[5] http://dbpedia.org/resource/The_Pentagon.


SView

http://dbpedia.org/resource/The_Pentagon Go

The Pentagon

is a Place , Location , Q41176 , Buildings and structures in arlington county, virginia , Object , Whole , Thing , Government building , Building , ArchitecturalStructure , More

The Pentagon is the headquarters of the United States Department of Defense, located in Arlington County, Virginia. As a symbol of the U.S. military, "the Pentagon" is often used metonymically to refer to the U.S. Department of Defense.The Pentagon was designed by American architect George Bergstrom (1876–1955), and built by general contractor John McShain of Philadelphia. Ground was broken for construction on September 11, 1941, and the building was dedicated on January 15, 1943.

Property Link

Directory [hide]

1 Building
3 ArchitecturalStructure
6 SpatialThing
9 Others

Building

location	Arlington County, Virginia [1]
floor count	7 [1]
address	1400 Defense Pentagon, Washington, DC 20301-1400 [1]
floor area	616538.0185344 [1]

ArchitecturalStructure

architectural style	Stripped Classicism [1]
architect	George Bergstrom [1]

SpatialThing

lat	38.871 [1]
long	-77.056 [1]

Fig. 1. Screenshot for browsing entities in SView with property clustering

It is worth noting that Wikidata[6] and Freebase[7] also organize related properties in adjacent positions. See Reasonator[8] for an item-type-optimized manner of Wikidata entity browsing.

7 Related Work

Property similarity is a special kind of property relatedness. Existing work that dedicates to property similarity finds synonymous or equivalent properties for applications like ontology mapping [14], entity linkage [10] and query expansion [2, 18]. Specifically, the work in [2] leveraged association rule mining to exploit synonymous properties. The work in [18] defined statistical knowledge patterns,

[6] http://www.wikidata.org.
[7] http://www.freebase.com.
[8] http://tools.wmflabs.org/reasonator.

which identified synonymous properties in and across datasets in terms of triple overlap, cardinality ratio and clustering. But synonymous or equivalent properties are inadequate to cover the properties describing similar aspects.

There are also works focusing on a more general notion of relatedness. The work in [7] used the Web as its knowledge source and utilized the use frequency provided by search engines to define semantic relatedness measure between ontology terms. The work in [4] characterized the relatedness between vocabularies from four angles: well-defined semantic relatedness, lexical similarity in content, closeness in expressivity and distributional relatedness. The work in [9] refined association rule mining to discover frequent property combinations in use. Many of these works focus on specified vocabularies or ontologies. However, for open domain entity browsing, the vocabularies are multi-sourced, heterogeneous and unpredictable. More importantly, none of them further considered property clustering or combination.

Faceted categorization and clustering organize items into meaningful groups to make sense of the items and help users decide what to do next during Linked Data exploration [8]. Automated facet construction attracts attentions in many studies [12], but its accuracy is often limited. Moreover, faceted categorization is generally used to group entities while our work focuses on clustering properties. Several browsing systems enable users to manually divide properties and values [13], but user contributions are usually sparse, especially at a large scale.

8 Discussion of Findings

The experimental results that we have shown allow us to answer our questions in Sect. 1.

- We empirically evaluated five kinds of relatedness measures between properties: lexical similarity between property names (R_I), semantic relatedness between property names (R_W), distributional relatedness between properties (R_U), range relatedness between properties (R_T) and overlap of property values (R_O). The result of our empirical study is uneven for every measure, which can be explained in two aspects. On one hand, most sample entities have considerable variance due to difference sources and property numbers; on the other hand, there is no measure that can achieve a high value for every clustering algorithm on either of Precision, Recall, Rand Index and NMI. In terms of F-Score, R_I and R_U generally generate the clusterings that are closer to the reference ones in our study than the other measures.
- We empirically evaluated three clustering algorithms: DBSCAN (C_D), single linkage (C_L) and spectral clustering (C_S). From the overall results, C_D is similar to C_L under our parameter settings, and C_S is greatly different from them. The results of C_D, C_L and C_S are uneven for each measure, and that of C_S is relatively stable. However, C_D and C_L usually generate better clustering results.

- We empirically evaluated the linear combination of measures and ensemble clustering using C_D and C_S. For C_D, our empirical study shows that the linear combination of relatedness measures tends to generate a clustering that features a high Recall, while ensemble clustering (consensus clustering) is recommended to use if a high Precision is preferred. For C_S, both of them are of little avail.

However, there are some issues that have not been fully covered during our evaluation:

- There are a diversity of methods to calculate lexical similarity between property names. In our study, we only tried I-Sub based on our previous experience in ontology matching. But it is possible that there is a great difference among the performances of different similarity measures. Additionally, distributional relatedness between properties highly depends on the underlying dataset used for estimation. Leveraging a more appropriate dataset can improve the performance.
- There are not a few well-known clustering algorithms that we have not considered in our evaluation, i.e. non-negative matrix factorization [15], which may achieve better performance.
- The two combination methods can improve C_D significantly, but can work on C_S barely. It is probable that both of the two combination methods have special favorites on clustering algorithms. However, due to the limited sample entities, we have not observed such correlation between the combination methods and the clustering algorithms.
- In our evaluation, properties used for clustering came from multiple sources and they were largely heterogeneous. At present, the clustering algorithms and combination methods have not achieved satisfiable F-Score values. This implies that more sophisticated solutions need to be developed for property clustering in Linked Data.

9 Conclusion

In this paper, we studied the property clustering in Linked Data and evaluated five property relatedness measures, three property clustering algorithms and two combination methods. Our experimental results demonstrated the feasibility of the automated property clustering. We also showed that property clustering can enhance entity browsing in practice.

In future work, we will improve the quality of property clustering by leveraging user feedback and active learning. We will also explore more use scenarios for property clustering in Linked Data.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (No. 61370019). We appreciate our students' participation in the experiments.

References

1. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: ranking and clustering. J. ACM **55**(5), 23 (2008)
2. Abedjan, Z., Naumann, F.: Synonym analysis for predicate expansion. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 140–154. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38288-8_10
3. Budanitsky, A., Hirst, G.: Evaluating WordNet-based measures of lexical semantic relatedness. Comput. Linguist. **32**(1), 13–47 (2006)
4. Cheng, G., Gong, S., Qu, Y.: An empirical study of vocabulary relatedness and its application to recommender systems. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 98–113. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25073-6_7
5. Evert, S.: Corpora and collocations. In: Lüdeling, L., Kytö, M. (eds.) Corpus Linguistics: An International Handbook, pp. 1212–1248. Mouton de Gruyter, Berlin (2008)
6. Fleiss, J.: Measuring nominal scale agreement among many raters. Psychol. Bull. **76**(5), 378–382 (1971)
7. Gracia, J., Mena, E.: Web-based measure of semantic relatedness. In: Bailey, J., Maier, D., Schewe, K.-D., Thalheim, B., Wang, X.S. (eds.) WISE 2008. LNCS, vol. 5175, pp. 136–150. Springer, Heidelberg (2008). doi:10.1007/978-3-540-85481-4_12
8. Hearst, M.: Clustering versus faceted categories for information exploration. Commun. ACM **49**(4), 59–61 (2006)
9. Hu, W., Jia, C.: A bootstrapping approach to entity linkage on the semantic web. J. Web Semant. **34**, 1–12 (2015)
10. Isele, R., Bizer, C.: Active learning of expressive linkage rules using genetic programming. J. Web Semant. **23**, 2–15 (2013)
11. Lin, D.: An information-theoretic definition of similarity. In: ICML 1998, pp. 296–304. Morgan Kaufmann, San Francisco (1998)
12. Oren, E., Delbru, R., Decker, S.: Extending faceted navigation for RDF data. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 559–572. Springer, Heidelberg (2006). doi:10.1007/11926078_40
13. Quan, D., Karger, D.: How to make a semantic web browser. In: WWW 2004, pp. 255–265. ACM, New York (2004)
14. Shvaiko, P., Euzenat, J.: Ontology matching: state of the art and future challenges. IEEE Trans. Knowl. Data Eng. **25**(1), 158–176 (2013)
15. Smith, T., Frank, E.: Introducing machine learning concepts with WEKA. In: Mathé, E., Davis, S. (eds.) Statistical Genomics, pp. 353–378. Springer, Heidelberg (2016)
16. Stoilos, G., Stamou, G., Kollias, S.: A string metric for ontology alignment. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 624–637. Springer, Heidelberg (2005). doi:10.1007/11574620_45
17. Wagner, S., Wagner, D.: Comparing clusterings: an overview. Universität Karlsruhe, Fakultät für Informatik (2007)
18. Zhang, Z., Gentile, A.L., Blomqvist, E., Augenstein, I., Ciravegna, F.: Statistical knowledge patterns: identifying synonymous relations in large linked datasets. In: Alani, H., et al. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 703–719. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41335-3_44

A MapReduce-Based Approach for Prefix-Based Labeling of Large XML Data

Jinhyun Ahn[1], Dong-Hyuk Im[2], and Hong-Gee Kim[1,3](✉)

[1] Biomedical Knowledge Engineering Laboratory and Dental Research Institute, Seoul National University, Seoul, South Korea
{jhahncs,hgkim}@snu.ac.kr

[2] Department of Computer and Information Engineering, Hoseo University, Cheonan, South Korea
dhim@hoseo.edu

[3] Institute of Human-Environment Interface Biology, Seoul National University, Seoul, South Korea

Abstract. A massive amount of XML (Extensible Markup Language) data is available on the web, which can be viewed as tree data. One of the fundamental building blocks of information retrieval from tree data is answering structural queries. Various labeling schemes have been suggested for rapid structural query processing. We focus on the prefix-based labeling scheme that labels each node with a concatenation of its parent's label and its child order. This scheme has been adapted in RDF (Resource Description Framework) data management systems that index RDF data in tree by grouping subjects. Recently, a MapReduce-based algorithm for the prefix-based labeling scheme was suggested. We observe that this algorithm fails to keep label size minimized, which makes the prefix-based labeling scheme difficult for massive real-world XML datasets. To address this issue, we propose a MapReduce-based algorithm for prefix-based labeling of XML data that reduces label size by adjusting the order of label assignments based on the structural information of the XML data. Experiments with real-world XML datasets show that the proposed approach is more effective than previous works.

1 Introduction

A large volume of XML (Extensible Markup Language) data from various areas are publicly available on the web, with some examples including DBLP (computer science bibliography), UniprotKB (protein information), SwissProt (protein sequence database), and Treebank (tagged sentences). XML data can be viewed as a tree data model that represents parent-child relationships between elements. XPath [1] is widely used to represent structural queries against XML data. One of the fundamental structural queries requires the determination of whether an ancestor/descendant relationship exists between two given elements. The simplest method to answer such queries is to traverse the tree to determine if a path exists between the two given elements. However, if the elements are far apart, one has to visit many elements. To overcome this disadvantage, labeling

Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 83–98, 2016.
DOI: 10.1007/978-3-319-50112-3_7

schemes have been proposed, in which each node is labeled so that its ancestor/descendant relationship can be determined by considering only those two labels. Interval-based, prefix-based, and prime number-based labeling schemes exist for tree data models. Of these schemes, we focus on the prefix-based labeling scheme that has been extensively utilized in many practical systems [2], including Microsoft SQL Server [3] and SiREN [4]. Note that SiREN is an information retrieval engine for RDF (Resource Description Framework) data that is a graph data model. In SiREN, RDF triples are converted into tree data structures by grouping subjects. The prefix-based labeling scheme is adopted to index the tree. With the popularity of the prefix-based labeling scheme, [5] proposed a MapReduce-based algorithm for prefix-based labeling of XML data to make it more applicable for massive real-world XML datasets (or Linked Open Data[1] if processed like SiREN).

Although previous approaches can label large XML data in parallel, they are not efficient at producing smaller label sizes. This makes it difficult for the prefix-based labeling scheme to handle massive XML data. Previous approaches assign labels to each node in the order presented in the XML file. We note that if we change the label assignment order appropriately, the resultant prefix-based label size is reduced while conforming to the prefix-based labeling scheme. Reducing label size is important because the query processing performance depends on how large the labels are. Obviously, the smaller the labels are, the faster ancestor/descendant relationships can be determined. Therefore, devising a way of generating the smallest label size is very important.

In this paper, we propose a dynamic labeling technique for the prefix-based labeling scheme, designed to produce a smaller label size than previous works. Specifically, the proposed approach extends [6]'s MapReduce-based repetitive prime labeling algorithm, which is similar to prefix-based labeling. The proposed approach adjusts the label assignment order during the labeling process, enabling it to reduce label size. The adjustment of label assignment is based on structural properties of the tree that can be obtained during the labeling process. The proposed approach is implemented on MapReduce, which allows multiple machines to perform labeling in parallel.

The contribution of this paper is summarized as follows:

– We extend the existing MapReduce-based repetitive prime labeling algorithm [6] to perform prefix-based labeling more efficiently than the state-of-the-art prefix-based labeling systems.
– We devise a novel technique that improves the above extension.
– Experiments with real-world XML datasets are conducted to show the effectiveness of the proposed approaches compared with state-of-the-art works.

This paper is organized as follows. Problem definitions and notations are stated in Sect. 2. Section 3 briefly overviews related work. In Sect. 4, we motivate our work by discussing the drawbacks of the state-of-the-art approach. Our approach is demonstrated in Sect. 5. Experimental results are discussed in Sect. 6. The paper is concluded in Sect. 7.

[1] http://linkeddata.org.

2 Preliminarily

We briefly introduce the MapReduce framework, XML data and prefix-based labeling scheme in this section, as these are closely related in our problem setting.

MapReduce is a programming model in a distributed computing environment. A MapReduce-based program consists of a map and reduce phase [7]. First, input data is split into several parts by an InputFormat, each of which is denoted as InputSplit. Each mapper reads each InputSplit and sends a part of InputSplit that is grouped by a map key to reducers. The reducer then processes the received data. All mappers and reducers run on each machine independently in a parallel fashion.

XML data can be viewed as a sequence of *start/end-tags*. The *empty-element-tags* (encoded in `<.../>`) and text contents between *start-tag* and *end-tag* are ignored in this paper to simplify notation. These can be represented in an expanded form consisting explicitly of dummy *start-tags* and *end-tags*. An *element* is a logical component that is a pair of a *start-tag* and a matching *end-tag*. Specifically, XML InputSplit is taken into account in order to model XML data in the context of the MapReduce framework. The formal definition of XML InputSplit is stated in Definition 1.

Definition 1. ***XML InputSplit***: Given an XML file D and a split size b in bytes. An ith InputSplit $I_i(D, b)$ is the sequence of *start/end-tags* split by b bytes, such that

$$I_i(D, b) = (t_j, t_{j+1}, ..., t_{j+n})$$

where t_j is the jth tag in D such that

$$t_j = (name, offset, type)$$

The $name$ denotes the string between two brackets. The position of its first bracket in D is $offset$, which is provided automatically by the MapReduce framework. The $type$ is either $start$ or end. For readability, we sometimes denote $t^{start} = (*, *, start)$ and $t^{end} = (*, *, end)$. If the context is clear, we simply state I_i to indicate $I_i(D, b)$.

Example 1. The start-tag of a root element is represented by (dblp, 0, *start*) if the name of the root element is "dblp". The $offset$ of the first child of the root element is set as 6 since `<dblp>` has six characters.

We implemented an XMLInputFormat to split an XML file into a sequence of XML InputSplits, as TextInputFormat provided by Apache Hadoop cannot be used here because it splits by *characters*. A tag may be split into different XML InputSplit (e.g., `<d` in I_i and `blp/>` in I_{i+1}). Thus, the XMLInputFormat is designed to be split by *brackets* to split into tags correctly.

Definition 2. ***XML Data Labeling Problem***: Given an XML file D, output an injective map L from a set of elements E of D to a set LS of particular labels.

$$L : E \longrightarrow LS$$

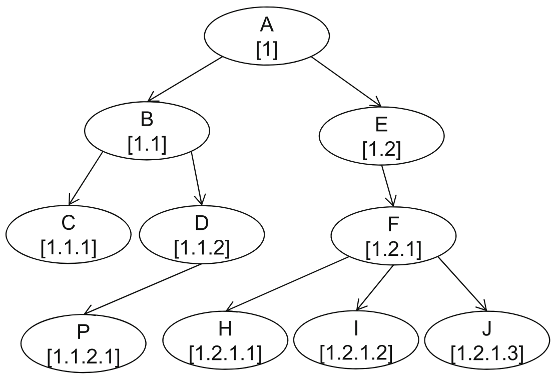


Fig. 1. A tree labeled by the prefix-based labeling scheme

L should be injective because every element $e \in E$ must be labeled and no two elements have the same label. LS is based on the choice of labeling scheme.

In particular, we focus on the conventional prefix-based labeling scheme [8–11], defined in Definition 3.

Definition 3. ***Prefix-based Labeling Scheme***: The prefix-based label $L(e)$ of an element e is defined as follows:

$$L(e) = L(pa(e)).order(e)$$

where $pa(e)$ is the parent of e, period(.) is the delimiter, and $order(e)$ is the child order of e under $pa(e)$. The label size $size(L(e))$ of $L(e)$ is the number of digits in $L(e)$ except for delimiters. To indicate the suffix label $order(e)$, we denote $suffix(L(e))$.

Example 2. See Fig. 1. The root element A has 1 as its prefix-based label. $L(\mathrm{E})$ is 1.2 because we have $L(\mathrm{A}) = 1$ and E is the second child corresponding to $order(\mathrm{E}) = 2$. The label size of $L(\mathrm{H})$ is 4, while the label size of $L(\mathrm{E})$ is 2.

3 Related Work

Answering ancestor/descendant relationships between two given elements in XML data can be achieved by tree traversal. However, this approach is inefficient when the two given elements are far away from each other because it requires visiting all of the intermediate elements in a path connecting them. To avoid visiting numerous nodes, we may maintain additional data for each element. The additional data is called an *index* or *label*. For two given elements a and b, we can determine the ancestor/descendant relationships by applying operations to the labels of a and b. The simplest labeling scheme is to have the label to retain a list of ancestor/descendant elements. However, this approach is inefficient because the space requirement is too large. To overcome the space disadvantage, diverse tree labeling schemes have been proposed [12].

Previous works are classified in two categories: tree labeling schemes and tree labeling algorithms. Tree labeling schemes define *what* the label is, while tree labeling algorithms define *how* to label each element. Note that the focus of this paper is a tree labeling algorithm based on the prefix-based labeling scheme.

Tree Labeling Schemes: The interval-based labeling scheme labels each element with (*start*, *end*, *level*) [13–17]. The *start* and *end* of an element represents the interval that includes all of its child element's intervals recursively. Therefore, the parent/child relationship can be determined from whether or not an interval is included. Additionally, *level* is used for determining whether the child is a direct child. The prime number labeling scheme calculates labels on the basis of the product of prime numbers [18,19]. In [18], after a unique prime number is assigned to every element, the product of the parent element's label and its own prime number (self-label) becomes the element's label. The disadvantage is that a unique prime number must be assigned to every element. If the number of elements increases, the number of prime numbers used will increase accordingly. However, variations have been proposed to overcome this disadvantage. In [19], the prime number recycling method is proposed, in which the self-label becomes the prime number following the prime number of its parent. This has two advantages over [18]. First, the label size is reduced because this scheme reuses prime numbers. Second, the keyword-based search query processing is efficient because self-labels of descendant elements of an element are larger than or equal to the self-label of the element. The prefix-based labeling scheme is a labeling method in which the parent element label is the prefix, and the child's order number is the suffix [8–11]. A delimiter is needed to distinguish between the prefix and the suffix. For example, if the root element's label is 1, its first child element's label is 1.1 and its second child element's label is 1.2, where the period(.) is the delimiter.

Tree Labeling Algorithm: In-memory tree labeling algorithms can be straightforwardly implemented. However, a massive XML file cannot be processed if there is insufficient memory. To solve this, the MapReduce framework has been adapted.

There exists a few studies on XML data labeling algorithms based on MapReduce [5,6]. [5] proposed two MapReduce-based tree labeling algorithms for interval-based labeling and prefix-based labeling schemes. Let's first discuss the interval-based labeling algorithm. During the map phase, *start* and *end* values are assigned to each element in InputSplit. In InputSplit, for a start/end-tag, if there is no corresponding end/start-tag, then *end* (or *start*) value is left empty. These incomplete labels are sent to reducers. At each map step, the label of the last element and the number of elements are recorded on an HDFS (Hadoop Distributed File System) file. When all map steps are completed, all of the information collected in the HDFS file is combined to generate one offset table, which

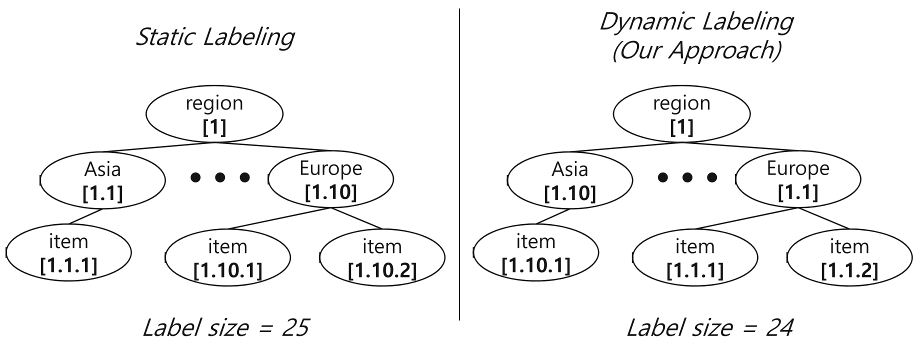


Fig. 2. Label size differs depending on the order of labels for unordered XML data.

can be accessed by all machines. In the reduce step, the offset table is used for completing the incomplete label of each element.

To explain the motivation for our work, the state-of-the-art MapReduce-based prefix-based labeling algorithm and its drawbacks are discussed in details in Sect. 4.

4 Motivation

Regarding the order of child elements, XML data are classified as unordered and ordered. The order of child elements is significant in ordered XML data, whereas this order is not significant in unordered XML data. In most real-world XML data, the order among child elements is unimportant [20,21]. In this regard, this paper focuses on unordered XML data. Specifically, we note that for unordered XML data, the order of label assignments affects the size of the resultant labels, as discussed in [6].

In Fig. 2, two different labeling results are depicted for the same XML data. The same prefix-based labeling scheme is used here. The figure is drawn on the assumption that the Asia element precedes the Europe element in the XML data. We also assume that there are eight elements between Asia and Europe, which are omitted in the figure. Label assignment on the left is by the order of the elements serialized in the XML data. We call it *static* labeling because the order of labels is determined by the order in the XML data. Asia is assigned 1.1 because it is the first child. Europe is assigned 1.10 because it is the 10th child. On the other hand, label assignment on the right is *dynamic* labeling because label order is not determined by the order in the XML data. Here, we have $L(\text{Europe}) = 1.1$ and $L(\text{Asia}) = 1.10$. Asia is assigned 10 because it has fewer descendants than Europe. In *dynamic* labeling, 10 is inherited to one node, whereas it is inherited to two elements in *static* labeling. By changing the label assignment order, we can reduce the label size (e.g., from 25 to 24 in Fig. 2).

The state-of-the-art prefix-based labeling algorithm [5] contains *static* labeling and does not support *dynamic* labeling. InputSplits are created from an input

XML file by a given split size, each of which is then assigned to each mapper. In the map phase, each element with an incomplete prefix-based label is sent to reducers using the name of the element as the map key. In addition, the following information is output from each mapper: the number of not paired end-tags (called `basement`), the sibling order of the last element, and the incomplete label of the last element. The information is stored in an HDFS file to create an offset table that has n rows, where n is the number of InputSplits. All previous rows from 1 to $n-1$ are combined to obtain the nth row in the offset table. In the reduce phase, the label in the offset table is appended to the prefix part, and s suffixes in the incomplete label are removed when s is the `basement` value. Note that a row in the offset table is made of all previous rows. Therefore, the order of the label assignment is the same as the order of InputSplit, which is the same as the order of rows. In other words, the order of label assignment is fixed to the order present in the XML file, which corresponds to *static* labeling.

5 The Proposed Approach

The proposed approach is extended from RepMR [6], which proposes a MapReduce-based XML data labeling algorithm for the repetitive prime labeling scheme [19]. It appears that a simple modification of RepMR can achieve our goal as these two labeling schemes are closely related. We call the simple modification PrxMR. However, we observe that label size can be further reduced using our novel *DCL(dynamic compressed element labeling)* technique. The approach with *DCL* is called PrxMR+. Section 5.1 briefly reviews PrxMR. Section 5.2 discusses the limitations of PrxMR and then explains our alternative PrxMR+.

5.1 PrxMR

PrxMR is a modification of RepMR [6] so that prefix-based labels are generated instead of repetitive prime labels. The overall algorithm is the same; however, the difference between them is the labeling scheme employed. This is possible because prefix-based labels and repetitive prime labels are similar in that labels are based on parent's label and child order. These two schemes deal with child order differently though. An illustrative example of PrxMR is shown in Fig. 3. Upper Tree Labeling is the process of labeling a portion of elements extracted from the root (called an upper tree). The label assignment order is determined based on the number of descendants. The upper tree is stored in an HDFS file for reducers to access. Label population is performed by reducers independently by referring to the shared upper tree.

Upper Tree Labeling. Figure 4 illustrates the Upper Tree Labeling step. For a given split size, we assume that three XML InputSplits are created. One sub-upper tree (denoted as SUP) is extracted for each InputSplit, in which nested elements are not included (e.g., type in SUP_1). Since these nested elements can be labeled locally in the Label Population step. Start/end-tags are depicted in

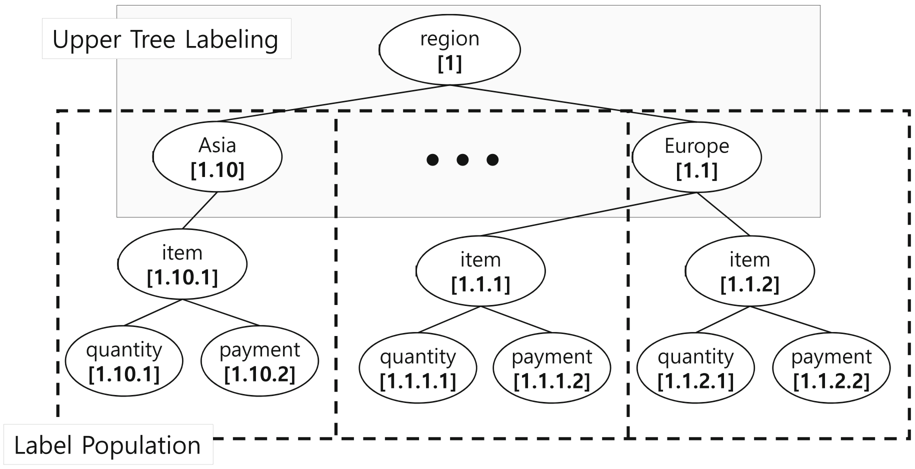


Fig. 3. Conceptual diagram of PrxMR. Prefix-based labels are represented in the bracket below the element name.

dotted rectangles while elements are in solid rectangles. For sub-upper trees (e.g. SUP_2) that have two or more root elements, a $DUMMY$ tag becomes the root. These sub-upper trees are merged by paring root elements to create the upper tree (UP) depicted in the right-most part. The number of all descendants including itself is dsc, while $sibs$ is the number of siblings including itself. Note that the order of book, device, and food elements is changed by dsc. The bold rectangle in UP represents a compressed element that corresponds to a sequence of elements, defined in Definition 4.

Definition 4. ***Compressed Element***: A compressed element C in UP is constructed based on a sequence $W \subset I_i$ of tags in an XML InputSplit I_i.

$$W = \{\texttt{<}e_i^{start}\texttt{>}, ..., \texttt{</}e_i^{end}\texttt{>}, \texttt{<}e_j^{start}\texttt{>}, ..., \texttt{</}e_k^{end}\texttt{>}\}$$

such that there exists a start-tag `<e>` in W if and only if there exists a matching end-tag `</e>` in W, which means that all tags are paired. The first element (e_i^{start},e_i^{end}) is called $leader$. Consider a sub-sequence $S \subset W$ of $leader$'s sibling elements as follows:

$$S = \{(s^{start}, s^{end}) | level(s^{start}) = level(leader^{start}) \text{ and } s^{start}, s^{end} \in W\}$$

The $level$ values are calculated by iterating over W. The $level(leader^{start})$ is initially set to 0. The value increases by 1 when a start-tag is encountered and decreases by 1 for an end-tag. Then C is encoded in a 3-tuple:

$$(leader, sibs, dsc),$$

where $sibs = |S|$ and $dsc = \sum_{s \in S} s.dsc$.

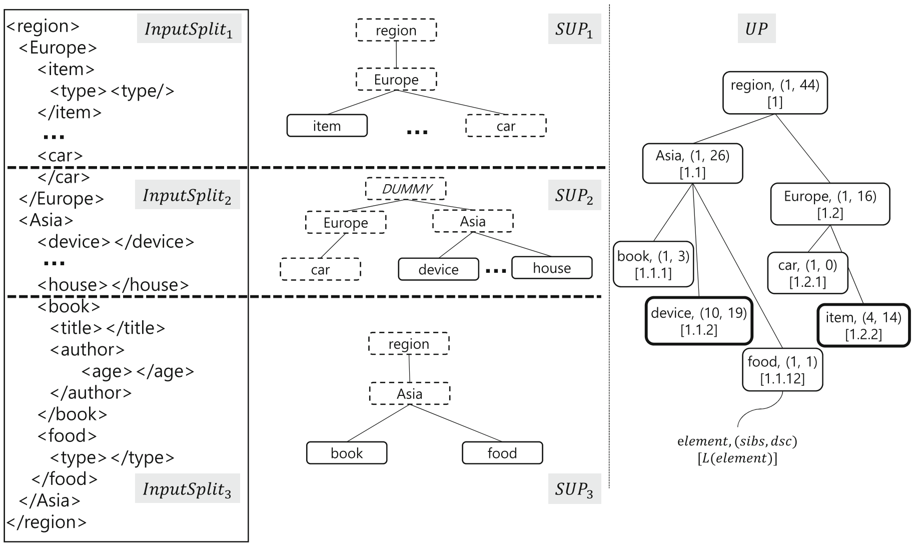


Fig. 4. Example dataflow of Upper Tree Labeling. Sub-upper trees (SUP) are extracted from each InputSplits and then merged into an upper tree (UP). Note that elements that can be labeled locally are not included in UP.

Example 3. (device,10,19) is a compressed element. There exist ten sibling elements in the same level such that devise is the first one and house is the last one. Therefore, food element is assigned 1.1.12 because device.$sibs$ = 10.

The order between two (compressed or regular) elements A and B in UP is determined by the average number of descendants of each direct children defined in Eq. 1.

$$order(A) < order(B) \text{ iff } \frac{A.dsc}{A.sibs} \geq \frac{B.dsc}{B.sibs} \tag{1}$$

Example 4. In Fig. 4, we have $order(\textbf{book}) < order(\textbf{device})$ because $\frac{book.dsc}{book.sibs} = \frac{3}{1} = 3$ and $\frac{device.dsc}{device.sibs} = \frac{19}{10} = 1.9$.

Label Population. Figure 5a is an illustrative example of PrxMR that assigns prefix-based labels to each element by referring to UP in Fig. 4. Three reducers are executed because InputSplit id is chosen as the map key, which means that elements in an InputSplit are sent exclusively to a reducer. There are four cases where each element e is labeled by matching to UP. The $offset$ value is utilized to match between elements.: ① If e is found in UP, then take the label in UP. ②: If e is not found in UP, but its parent $pa(e)$ is found in UP, then we automatically obtain $L(e)$ since $pa(e)$ has already been encountered in the same reducer and therefore $L(pa(e))$ is available. For example, in Reducer #3, the title is not in UP, but its parent book is. Therefore, book must have been

(a) PrxMR

Reducer #1

	①	②	③	④
region	1			
Europe	1.2			
Item			**1.2.2**	
type				**1.2.2.1**
police			**1.2.3**	
...				
car	1.2.1			

Reducer #2

	①	②	③	④
Asia	1.1			
device			**1.1.2**	
school			**1.1.3**	
office			**1.1.4**	
...				
house			**1.1.11**	

Reducer #3

	①	②	③	④
book	1.1.1			
title		1.1.1.1		
author		1.1.1.2		
age				1.1.1.2.1
food	1.1.12			
type		1.1.12.1		

(b) PrxMR+

Reducer #1

	dsc.	①	②	③	④
region		1			
Europe		1.2			
Item	**1**			**1.2.3**	
type					**1.2.3.1**
police	**2**			**1.2.2**	
...					
car	1	1.2.1			

Reducer #2

	dsc.	①	②	③	④
Asia		1.1			
device	**0**			**1.1.11**	
school	**3**			**1.1.8**	
office	**1**			**1.1.10**	
...					
house	**2**			**1.1.9**	

Reducer #3

	dsc.	①	②	③	④
book		1.1.1			
title			1.1.1.1		
author			1.1.1.2		
age					1.1.1.2.1
food		1.1.12			
type			1.1.12.1		

Fig. 5. Example dataflow of the Label Population step of PrxMR and PrxMR+. Each element e is assigned $L(e)$ differently in four cases (① ∼ ④) by referring to UP in Fig. 4. The number of descendants is represented by dsc, which is only exploited by PrxMR+. Elements in compressed nodes are labeled differently, highlighted in boldface.

encountered before title according to the sequence serialized in the XML data. ③: e is the compressed element in UP. For example, device is the compressed element with $L(\text{devise}) = 1.1.2$ from UP. The sub-sequent sibling elements are labeled in the presented order. ④: The other case. For example, in Reducer #3, the age element is automatically labeled because it is the child of the author element that has been labeled by case ②.

5.2 PrxMR+

Although PrxMR successfully reduces the label size, it is still inefficient in the way elements in a compressed element are labeled, corresponding to case ③ in the Label Population step.

See elements in bold in Fig. 5a. The suffix labels increase sequentially in the presented order. For example, in Reducer #2, device, school, office, ... , house elements are assigned suffix labels from 2 to 11. This corresponds to *static* labeling; its inefficiency in terms of the label size was discussed in Sect. 4. We observed that this inefficiency can be overcome by utilizing the information about dsc of each sibling elements to adjust suffix labels. In this regard, we devise DCL in Definition 5.

Definition 5. ***DCL(Dynamic Compressed Element Labeling)***: Given a compressed element C, by Definition 4, we have a sequence W of tags and its

Table 1. Real-world XML Datasets

Name	Elements	Avg. fanout (max)	Avg. depth (max)
treebank	2,437,666	2.2 (51)	7.8 (36)
swissProt	2,977,030	6.6 (342)	3.5 (5)
dblp	3,332,130	9.0 (750)	2.9 (6)
psd7003	21,305,818	3.9 (151)	5.1 (7)
kegg	63,445,091	2.4 (188)	7.3 (10)

sub-sequence S of $C.leader$'s sibling elements. The dsc of $s \in S \subset W$ is obtained as follows:

$$s.dsc = \text{ the number of start-tags in } W \text{ between } s^{start} \text{ and } s^{end}$$

By ordering S in decreasing order of dsc, we obtain a sorted set $DSC(S)$. Then, the label of each element in S is defined as follows:

$$L(s) = L(pa(C.leader)).i$$

such that $i = suffix(L(C.leader)) + (d - 1)$ and s is the dth one in $DSC(S)$.

Example 5. See Fig. 5b. In Reducer #1, we have a compressed element $C = (\text{item}, 4, 14)$ and $S = \{\textbf{item}, ..., \textbf{police}\}$. We obtain $DSC(S) = \{\textbf{police}, ..., \textbf{item}\}$, because $\text{item}.dsc < \text{police}.dsc$ assuming that they have the minimum and maximum dsc among elements in S. We also have $L(pa(\text{item})) = 1.2$ and $suffix(L(\text{item})) = 2$. Therefore, we have $L(\text{police}) = 1.2.2$ because police is the first element in $DSC(S)$.

6 Performance Study

Experiments with real-world XML datasets were conducted. Our cluster consists of four machines (2.6 GHz CPU, 32 GB RAM). Hadoop 2.7.1 and JDK 1.8 were used for the implementations. The proposed approaches (PrxMR and PrxMR+) were compared with the state-of-the-art approach [5], denoted as STATIC. In the experiments, the split size b was assumed to be 524,288 bytes.

Table 1 lists five datasets obtained from the XML Repository at University of Washington[2]. The number of children of an element is *fanout*. The number of elements from e to root is the *depth* of an element e.

6.1 Labeling Time

Figure 6 shows the labeling time for the five datasets. There is no significant difference between the three approaches for all datasets. Nevertheless, we observe

[2] http://www.cs.washington.edu/research/xmldatasets/www/repository.html.

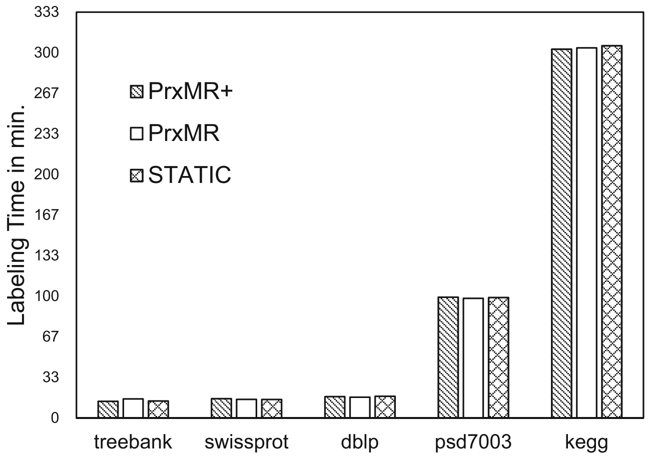


Fig. 6. Labeling time

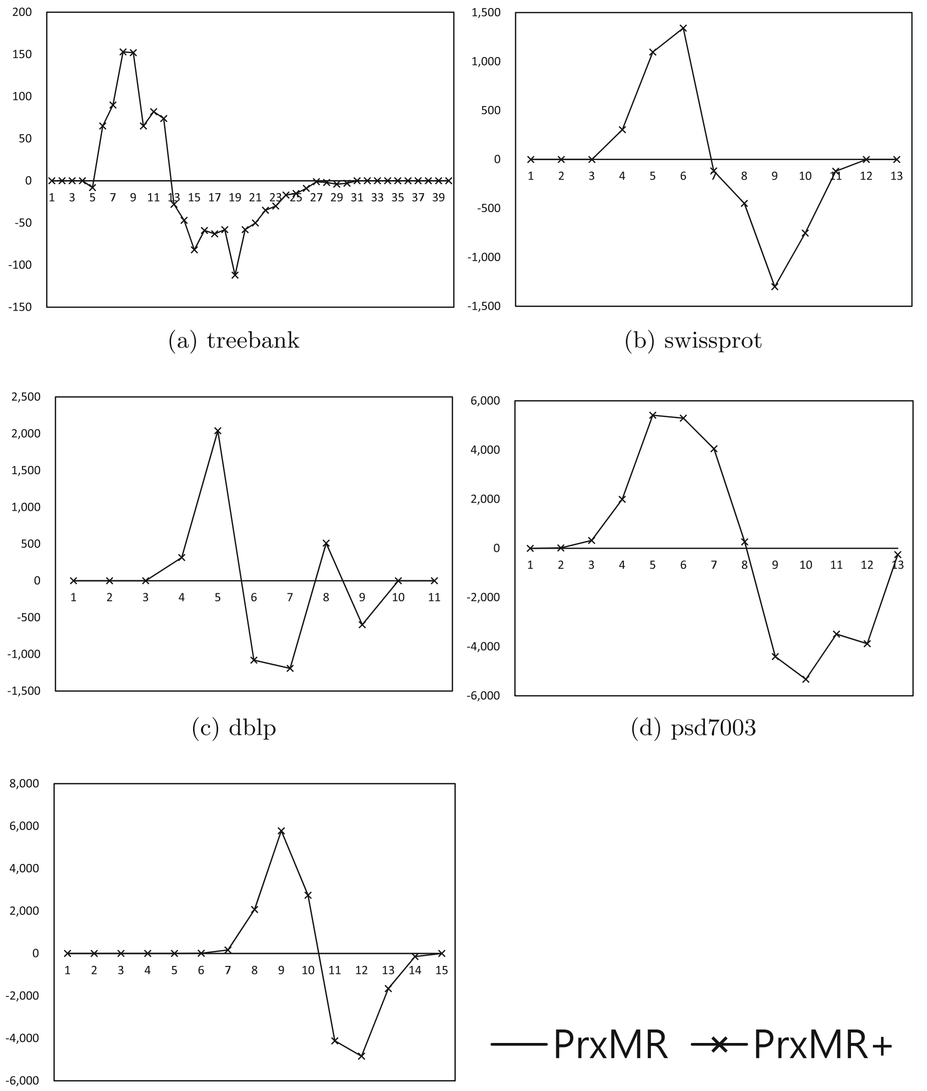
that our approaches are slightly better than STATIC for larger datasets. This can be understood by the fact that STATIC completes labels in the reduce phase using a slightly complex operation (i.e., calibration operation stated in [5]), whereas our approach employs a very simple operation (i.e., Label Population).

6.2 Label Size

The average label size is calculated in the experiments (Table 2). This measure is more important than the total label size, because prefix matching is performed between individual labels to answer structural queries. For all cases, our approaches generate smaller label size than STATIC. The Upper Tree Labeling technique helps avoid assignments of larger suffix numbers to elements with many descendants, which contributes to reducing label size. Note that label size is the number of digits in decimal representation, which means that even a very small reduction in label size is meaningful. The largest label sizes were seen in treebank even though it is the smallest dataset due to its large depth. According to the prefix-labeling scheme, suffix numbers are appended following descen-

Table 2. Average Label Size

	PrxMR+	PrxMR	STATIC
treebank	**11.656**	11.658	11.678
swissprot	**7.762**	7.765	8.291
dblp	**7.558**	7.559	7.672
psd7003	**9.796**	9.800	9.913
kegg	**10.466**	**10.466**	10.741



(a) treebank (b) swissprot

(c) dblp (d) psd7003

(e) kegg

Fig. 7. The relative number of labels for each label size, on the basis of PrxMR. X-axis indicates the label size. For example, individual label sizes of kegg range from 1 to 15. The y-axis indicates the relative number of elements for a label size x by calculating $|\{e|size(L_{\mathsf{PrxMR+}}(e)) = x\}| - |\{e|size(L_{\mathsf{PrxMR}}(e)) = x\}|$, where $L_{\mathsf{PrxMR+}}$ and L_{PrxMR} are the labeling map by PrxMR+ and PrxMR, respectively.

dants. In the case of kegg, PrxMR+ and PrxMR have the same label size. As seen in Table 1, kegg has a relatively small average fanout, which means that there are few chances to employ the *DCL* technique to help reduce the label size by adjusting suffix label assignments between elements in a compressed element.

More details on each dataset's label size are depicted in Fig. 7, which shows the relative label size of PrxMR+ on the basis of PrxMR. For example, in the case of psd7003, PrxMR+ assigns labels of size 5 to 5,000 more elements than PrxMR; on the other hand, PrxMR+ assigns labels of size 9 to 4,000 less elements than PrxMR. In other words, there are more elements that are assigned smaller labels by PrxMR+ than PrxMR. The same tendencies are observed in the other datasets. Among them, psd7003 shows a relatively larger positive area for smaller label sizes (i.e., label size from 3 to 7) and smaller negative area for larger label sizes (i.e., label size from 9 to 12). This can be understood by the fact that psd7003 has both larger average fanout and larger average depth overall. For larger average depth, there are more chances to reduce the label size by Upper Tree Labeling. For larger average fanout, there are more chances for *DCL* to have an effect on reducing the label size.

7 Conclusion

We proposed a MapReduce-based prefix-based labeling algorithm, which is extended from a MapReduce-based repetitive prime labeling algorithm [6]. This algorithm introduces a novel way of dealing with compressed elements to reduce label size. Experiments on massive XML data showed that the proposed technique generated smaller labels than the previous algorithms.

In the proposed approach, elements in a mapper are sent exclusively to a reducer by choosing InputSplit id as the map key. There is no other way because the sequence of elements should be preserved in the reduce phase to carry out the Label Population step appropriately. This method does not allow control of the amount of data sent to reducers via the network. This is not desired because it cannot fully utilize all workers efficiently in terms of data distribution across a cluster. A completely new algorithm design is needed to address this issue in future works.

Acknowledgement. This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. R0101-16-0054, WiseKB: Big data based self-evolving knowledge base and reasoning platform) and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014R1A1A1002236).

References

1. Clark, J., DeRose, S., et al.: XML path language (XPath) (1999)
2. Pal, S., Cseri, I., Seeliger, O., Rys, M., Schaller, G., Yu, W., Tomic, D., Baras, A., Berg, B., Churin, D., et al.: XQuery implementation in a relational database system. In: Proceedings of the 31st International Conference on Very Large Data Bases, VLDB Endowment, pp. 1175–1186 (2005)
3. O'Neil, P., O'Neil, E., Pal, S., Cseri, I., Schaller, G., Westbury, N.: ORDPATHs: insert-friendly XML node labels. In: Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, pp. 903–908. ACM (2004)
4. Delbru, R., Toupikov, N., Catasta, M., Tummarello, G.: A node indexing scheme for web entity retrieval. In: Aroyo, L., Antoniou, G., Hyvönen, E., Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010. LNCS, vol. 6089, pp. 240–256. Springer, Heidelberg (2010). doi:10.1007/978-3-642-13489-0_17
5. Choi, H., Lee, K.H., Lee, Y.J.: Parallel labeling of massive XML data with mapreduce. J. Supercomputing **67**(2), 408–437 (2014)
6. Ahn, J., Im, D.H., Lee, T., Kim, H.G.: A dynamic and parallel approach for repetitive prime number labeling of XML data with MapReduce. J. Supercomputing (To Appear)
7. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
8. Xu, L., Ling, T.W., Wu, H., Bao, Z.: DDE: from dewey to a fully dynamic XML labeling scheme. In: SIGMOD. ACM (2009)
9. Tatarinov, I., Viglas, S.D., Beyer, K., Shanmugasundaram, J., Shekita, E., Zhang, C.: Storing and querying ordered XML using a relational database system. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pp. 204–215. ACM (2002)
10. Lin, R.-R., Chang, Y.-H., Chao, K.-M.: A compact and efficient labeling scheme for XML documents. In: Meng, W., Feng, L., Bressan, S., Winiwarter, W., Song, W. (eds.) DASFAA 2013. LNCS, vol. 7825, pp. 269–283. Springer, Heidelberg (2013). doi:10.1007/978-3-642-37487-6_22
11. Lu, J., Meng, X., Ling, T.W.: Indexing and querying XML using extended dewey labeling scheme. Data Knowl. Eng. **70**(1), 35–59 (2011)
12. Klaib, A., Joan, L.: Investigation into indexing XML data techniques (2014)
13. Xu, L., Bao, Z., Ling, T.W.: A dynamic labeling scheme using vectors. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 130–140. Springer, Heidelberg (2007). doi:10.1007/978-3-540-74469-6_14
14. Li, C., Ling, T.W.: QED: a novel quaternary encoding to completely avoid re-labeling in XML updates. In: CIKM. ACM (2005)
15. Christophides, V., Karvounarakis, G., Plexousakis, D., Scholl, M., Tourtounis, S.: Optimizing taxonomic semantic web queries using labeling schemes. Web Semant. Sci. Serv. Agents World Wide Web **1**(2), 207–228 (2004)
16. Xu, L., Ling, T.W., Wu, H.: Labeling dynamic XML documents: an order-centric approach. IEEE Trans. Knowl. Data Eng. **24**(1), 100–113 (2012)
17. Subramaniam, S., Haw, S.C., Soon, L.K.: Relab: A subtree based labeling scheme for efficient XML query processing. In: 2014 IEEE 2nd International Symposium on Telecommunication Technologies (ISTT), pp. 121–125. IEEE (2014)
18. Wu, X., Lee, M.L., Hsu, W.: A prime number labeling scheme for dynamic ordered XML trees. In: ICDE (2004)

19. Sun, D.H., Hwang, S.C.: A labeling methods for keyword search over large XML documents. J. KIISE **41**(9), 699–706 (2014)
20. Wang, Y., DeWitt, D.J., Cai, J.Y.: X-Diff: An effective change detection algorithm for XML documents. In: 2003 Proceedings of the 19th International Conference on Data Engineering, pp. 519–530. IEEE (2003)
21. Leonardi, E., Bhowmick, S.S., Madria, S.: Xandy: Detecting changes on large unordered XML documents using relational databases. In: Zhou, L., Ooi, B.C., Meng, X. (eds.) DASFAA 2005. LNCS, vol. 3453, pp. 711–723. Springer, Heidelberg (2005). doi:10.1007/11408079_65

RIKEN MetaDatabase: A Database Platform as a Microcosm of Linked Open Data Cloud in the Life Sciences

Norio Kobayashi[1,2,3(✉)], Kai Lenz[1], and Hiroshi Masuya[2,1]

[1] Advanced Center for Computing and Communication (ACCC), RIKEN, 2-1 Hirosawa, Wako, Saitama 351-0198, Japan
{norio.kobayashi,kai.lenz}@riken.jp

[2] BioResource Center (BRC), RIKEN, 3-1-1, Koyadai, Tsukuba, Ibaraki 305-0074, Japan
hmasuya@brc.riken.jp

[3] RIKEN CLST-JEOL Collaboration Center, RIKEN, 6-7-3 Minatojima-minamimachi, Chuo-ku, Kobe 650-0047, Japan

Abstract. The amount and heterogeneity of life-science datasets published on the Web have considerably increased recently. However, biomedical scientists face numerous serious difficulties in finding, using and publishing useful databases. In order to solve these issues, we developed a Resource Description Framework-based database platform, called RIKEN MetaDatabase, which allows biologists to easily develop, publish and integrate databases. The platform manages metadata of both research data and individual data described with standardised vocabularies and ontologies, and has a simple browser-based graphical user interface for viewing data including tabular and graphical views. The platform was released in April 2015, and 110 databases including mammalian, plant, bioresource and image databases with 21 ontologies have been published through this platform as of July 2016. This paper describes the technical knowledge obtained through the development and operation of RIKEN MetaDatabase as a challenge for accelerating life-science data distribution promotion.

Keywords: Semantic web · Database cloud platform · Database integration · Life sciences

1 Introduction

The life sciences have been developed rapidly and subdivided into specialised study fields. Thus, the study of the life sciences has generated numerous heterogeneous datasets, making it difficult for researchers to find data from this flood of information, use them appropriately in their research and publishing them in a useful way for other researchers. Considering these difficulties, two major issues arise. The first issue is realising rich and useful data integration in a sustainable way: linking data, integrating data systematically using standardised vocabularies, representing semantics and publishing the data location. The second issue involves realising easy, flexible and low-cost operation that allows many data developers and biologists to participate in the process of data integration.

These difficulties also occur within a research institute. RIKEN is the largest Japanese comprehensive science institute, having both large-scale research centres and many small-scale laboratories, which generate large-scale life-science datasets in various fields. The institute is confronted with issues regarding the realisation of collaborative research promotion over different fields within it. Therefore, database infrastructure is required for the publication and promotion of RIKEN's research results. This situation can be presented as a microcosm of the linked open data cloud in the life sciences.

We consider RIKEN's problem described above as a case study of data utilisation in the life sciences. To solve this problem, we developed RIKEN MetaDatabase—which is a database platform based on the Resource Description Framework (RDF), which realises metadata management at low cost, systematic data integration and global publication on the Web. RIKEN MetaDatabase was published in April 2015 with RIKEN's original databases, as well as external databases associated with these databases and ontologies. Here, we discuss the advantages of RDF for solving life-science data distribution and future issues, focusing on RIKEN MetaDatabase implementation, data integration and comparison with other cases.

The rest of this paper is organised as follows. Section 2 presents previous work related to this study. Section 3 discusses the requirement specifications for the database platform. Sections 4, 5 and 6 discuss design issues including functions, workflow of database publication, detailed data view and implementation. Sections 7 and 8 introduce available databases and comprehensively review RIKEN MetaDatabase by introducing concrete database projects. Finally, Sect. 9 concludes this study.

2 Related Work

We here review the existing RDF-based database platforms for life-science data publication and integration related to RIKEN MetaDatabase.

The Harvard Catalyst (https://catalyst.harvard.edu) is an information resource-sharing platform for human health research that enables collaboration among researchers within a group of 31 institutes including Harvard University. RDF-based data integration and federated search among distributed servers are used as the network's mining tool [1]. However, the platform does not aim at hosting the researchers' databases for data integration purposes.

Bio2RDF (http://bio2rdf.org) provides major existing life-science datasets by converting them into the RDF format [2]. The generators of the original data and the informaticians for RDFising are different in this case. Instead, in our approach, the original data generators participate in the data integration by RDFising these data themselves.

BioPortal (http://bioportal.bioontology.org) in the National Center for Biomedical Ontology (NCBO) is a data federation platform based on RDF and OWL, as well as our platform [3]. However, the primary focus of BioPortal is data integration and coordination between ontologies, while our approach focuses on inter-linking data items in the researchers' databases directly.

RDF Portal (http://integbio.jp/rdf/) in the National Bioscience Database Center (NBDC) and our platform apply a common data integration concept, which collects RDF datasets from various study fields. RDF Portal allows researchers from different institutes and universities to combine their RDF datasets. In addition, it provides SPARQL query interfaces for each dataset and across all datasets. Instead, our platform supports both generating and collecting RDF data. Furthermore, RIKEN MetaDatabase provides a data browser that can also be used by non-RDF users.

EBI RDF Portal (https://www.ebi.ac.uk/rdf/) currently hosts six datasets produced by large-scale projects, which are indispensable for data analysis and integration to many bioinformatists. In contrast, our platform is aimed at hosting both large- and middle–small-scale projects and laboratories, all of which want to contribute to the life sciences through the publication of their research-based data.

To generate RDF data, we employ a spreadsheet to describe the raw data and convert them into RDF. A similar tool is OpenRefine (http://openrefine.org/), which can generate RDF data from various source files including a spreadsheet. In OpenRefine, the data that are to be converted into RDF are defined outside the source files. Instead, our spreadsheet includes all the data to be converted into RDF, and the RDF expert can easily recognise the RDF data structure from the spreadsheet.

RightField [4] is a tool for editing life-science data given in spreadsheets by embedding the ontology annotations. A RightField spreadsheet allows a user to select terms from a given ontology dataset which includes subclass relations, individuals, and combinations.

3 Requirement Specifications for the Life-Science Database Platform

3.1 Requirements for Cloud-Based Databases in the Life Sciences

As an in-house database platform for a comprehensive research institute, RIKEN MetaDatabase should support different types and sizes of datasets generated by research projects of all scales. At the same time, RIKEN MetaDatabase should form a uniform knowledge base by including not only RIKEN's internal datasets but also global datasets interlinked on the Web. In addition, it should provide a simple operational workflow, by which biologists can easily participate in global data integration without specialised data integration skills.

Ideally, both biologists and informaticians should cooperate closely for data integration and mining through the database platform. Therefore, we conclude that RIKEN MetaDatabase is required to be a database platform providing well-coordinated datasets, which can be easily integrated into global datasets and used by data scientists. Furthermore, the data publication workflow should be simple so that biologists can operate it easily.

To satisfy these requirements, we have designed the database platform as a cloud-based platform that allows many database developers to deploy their data without management hardware and to ensure significant and flexible computational resources. We also decided to adopt Semantic Web technologies. Easily operable interfaces for data generation and publication were also designed. At present, there is no other database platform that realises cloud-based data and database publication independently of the data types and sizes. In other words, this platform meets various needs of database developers in an organisation or community in a cost-effective way.

3.2 Data Integration

For developing RIKEN MetaDatabase, we aim to realise the following two types of data integrations:

1. Data integration in a specialised research field.
   Data integration to realise a comprehensive dataset across research projects (research organisations or international consortium) to form a unified database. In this case, we assume that research projects bring non-redundant datasets, which may belong to the same data class. Research projects should share the data structure defined using their class, rather than sharing data entities (or instances), to enable unified data handling and management.
2. Data integration among different research fields.
   Data integration among different research fields or for co-operable research to realise mutual data links across datasets. In this case, projects provide related datasets, which belong to different data classes. Here, some data entities may act as links between the different datasets.

Case 1 can be achieved by introducing common or standardised data schemata and ontologies, where each data entity is usually described as an instance of a class or an ontology term. Therefore, data integration is not achieved by providing a direct link between data entities but by sharing a common class and semantic links (RDF properties) defining the data structure. In case 2, data entities from different datasets are directly connected by semantic links, whereas each dataset is described by different specialised data schemata. Data entities themselves are the links that allow the expansive combination of different communities.

Semantic Web employing RDF is a technology that satisfies these two types of integrations simultaneously; case 1 can be realised using the RDF scheme and the web ontology language (OWL), whereas, in case 2, a data linking mechanism can be applied. However, as explained in Sect. 2, most platforms do not satisfy both cases 1 and 2. Therefore, we propose here a novel practical approach to solve this problem.

4 Grand Design of the RDF-based RIKEN MetaDatabase Platform

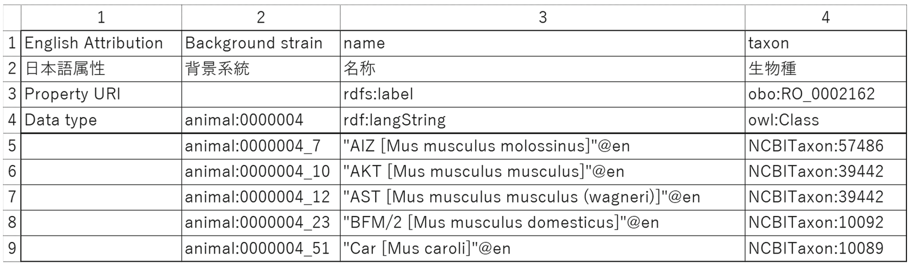
4.1 RDF Data Structure Suitable for Life-Science Data Integration

Prior to discussing the grand design, we will investigate the data structures in the RIKEN databases. These datasets are represented in tabular form or hosted by a relational database system (data not shown). Therefore, in order to realise a simple and user-friendly database infrastructure system, as described above, we restrict the RDF data handled by RIKEN MetaDatabase to tabular-type database data and tree-type ontology data.

Tabular form used to describe tabular-type database data represents the RIKEN MetaDatabase data that can be easily generated and browsed. Tree form used to describe ontology data represents the concepts and data classes with their conceptual hierarchy to refer to the databases. Using the two kinds of data forms, RIKEN MetaDatabase aims to build a single integrated RDF dataset by managing multiple tabular and tree ontology data individually.

4.2 Tabular Data Model

We introduce a tabular data model for describing RDF data in which all RDF resources are associated with an RDF class. A table is generated for each class of subject instances of RDF triplets. Figure 1 shows the data structure of the RDF data described in tabular form. The presented table is separated into an RDF scheme definition part and a data part.



	1	2	3	4
1	English Attribution	Background strain	name	taxon
2	日本語属性	背景系統	名称	生物種
3	Property URI		rdfs:label	obo:RO_0002162
4	Data type	animal:0000004	rdf:langString	owl:Class
5		animal:0000004_7	"AIZ [Mus musculus molossinus]"@en	NCBITaxon:57486
6		animal:0000004_10	"AKT [Mus musculus musculus]"@en	NCBITaxon:39442
7		animal:0000004_12	"AST [Mus musculus musculus (wagneri)]"@en	NCBITaxon:39442
8		animal:0000004_23	"BFM/2 [Mus musculus domesticus]"@en	NCBITaxon:10092
9		animal:0000004_51	"Car [Mus caroli]"@en	NCBITaxon:10089

Fig. 1. A spreadsheet describing RDF data of class Background strain (http://metadb.riken.jp/db/rikenbrc_mouse/animal_0000004) in tabular form. In this example, the second column includes a list of instances of class Background strain, the third column includes a list of literal values of rdf:langString and the fourth column includes a list of Taxon classes as instances of owl:Class.

Functions of Rows. The RDF scheme definition part is presented in the top four rows in the table. In the first and second rows, English and Japanese column names of the table are displayed on the GUI, respectively. The third and fourth rows describe the properties and classes of the objects of the triplets used to convert the tabular data into RDF, respectively. The fifth and subsequent rows include the data.

Functions of Columns. The first column is a comment column, which is not converted into RDF.

The second column includes a list of instances (resources) of the common class that is the subject of all triplets described in the table, namely a list of subject instances. Using the table coordinates (r, c) to locate the data points, where r is the row and c is the column, $(4, 2)$ contains the data class, $(3, 2)$ is empty, and $(m, 2)$ for $m \geq 5$ are the instances of class $(4, 2)$.

The third and subsequent columns describe the properties and objects for the subjects listed in the second column. $(3, n)$ is a property and $(4, n)$ is a class or a data type of the instances or the literals listed as (m, n), respectively, where $m \geq 5$ and $n \geq 3$. Here, the triplet $(m, 2), (3, n), (m, n)$ is equivalent to the following set of RDF triplets:

$$\begin{aligned}&(m, 2)\ \ (3, n)\ \ (m, n).\\&(m, 2)\ \ \texttt{rdf:type}\ \ (4, 2).\\&(m, n)\ \ \texttt{rdf:type}\ \ (4, n).\end{aligned}$$

where $(4, n)$ is a RDF class, or

$$\begin{aligned}&(m, 2)\ \ (3, n)\ \ (m, n).\\&(m, 2)\ \ \texttt{rdf:type}\ \ (4, 2).\end{aligned}$$

where $(4, n)$ is a data type and (m, n) is a literal denoted as the form of data type $(4, n)$.

Moreover, each pair of a property and an object class (or data type) in the third and subsequent columns can appear multiple times, in order to describe multiple triplets sharing a common subject, property and object class (or data type).

4.3 Correspondence with the RDF Scheme

In order to manage multiple RDF datasets as databases or ontologies in RIKEN MetaDatabse, we introduce a specialised data category corresponding to the existing RDF scheme elements, as shown in Table 1.

A database is an RDF dataset with tabular data which compose an individual database, and corresponds to an RDF named graph. An ontology is an OWL ontology managed as an RDF named graph. A property and a class are equivalent to an RDF property and an RDF class as an instance of rdf:Property and rdfs:Class, respectively. An instance is limited to an instance i of rdf:Resource explicitly described as triplet i `rdf:type` c, where c is an RDF class. The reason we introduce limited instances is to establish data re-usability; when a class is specified, the instances of that class can be accurately obtained without orphan instances, which are not associated with any class.

Table 1. Correspondence between RIKEN MetaDatabase and the RDF scheme

RIKEN MetaDatabase	RDF scheme	Description
database	named graph	an individual dataset with multiple classes
ontology	named graph	an individual ontology written in OWL
property	instance of rdf:Property	equivalent to rdf:Property
class	instance of rdfs:Class	a concept or a rdf:Resource set
instance	instance of class	an instance typed by a class

4.4 RDF Data Generation and Publication

We design a procedure through which users can generate and publish their RDF data. Tree ontology data—usually described in OWL, which can be downloaded from public repositories or generated by an existing ontology editor—can be uploaded directly to the RIKEN MetaDatabase platform and immediately published. On the other hand, for tabular data, we apply a spreadsheet-based workflow, which can be operated by biologists as follows:

Step 1. Generating a spreadsheet. In this step, the user (database developer) describes the spreadsheet as a Microsoft Excel file or Tab-Separated Values (TSV) files, which represents a tabular data model. Using multiple spreadsheets in Microsoft Excel or TSV files, the user can describe a complicated database in which multiple tables are linked in a relational database management system.

Step 2. Generating an RDF dataset. The spreadsheet generated in the previous step is converted into RDF by the user using our application program. The program generates not only the RDF data converted from the raw data but also a structure definition file that describes the order of the columns and the column names.

Step 3. Uploading the RDF dataset. Both the database and the structure definition files are uploaded by a service administrator. The uploaded data are immediately published.

4.5 User Interface for Data Input and Output

RIKEN MetaDatabase employs both a graphical user interface (GUI), working on the user's web browser, and an application programming interface (API), for data input and output.

For data input, a registration interface of RDFised tabular data and tree ontology data is implemented. This function is closed; only the service administrators can operate this function since they should be able to check the uploaded data before publication.

The data publishing function is implemented in both API and GUI. As an API, we use an interface that actuates as a SPARQL endpoint accessible via the HTTP protocol, which is a standardised RDF data access protocol. The GUI works on the user's web browser and displays RDF data in various formats such as tabular and tree formats. In addition, it offers a list of RDF data archives for download and access to the SPARQL endpoint described above with query editor and result display functions.

5 Data Display Functions

To demonstrate the RIKEN management of RDF data, several fixed display forms are prepared as views for each data category. The data are shown using only their multilingual labels rather than their Unified Resource Identifiers (URIs), as a default, but both labels and URIs can be shown to RDF experts. The implemented views are summarised as follows:

List view shows lists of databases and ontologies, and includes a keyword search function to filter those data.
Tree view shows OWL ontologies as trees based on the subclass relationships.
Tabular view shows a list of instances of a specified class.
Card view shows a selected instance.
Download view is used for downloading RDF data archives for each database.
SPARQL search view supports editing queries and result displaying.

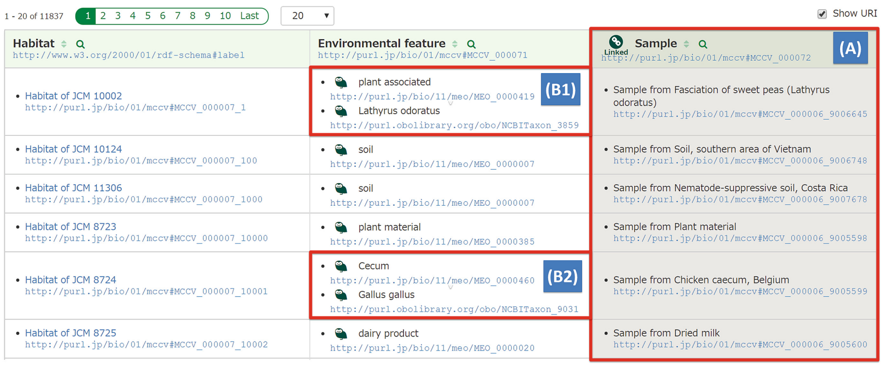
By default, the tabular and card views are devised to show RDF data to biologists. We describe these views in detail below.

Tabular View. Tabular view is a special feature of RIKEN MetaDatabase that shows an RDF graph data in tabular form. This view can be generated for each class and shows the name and description of the concerned class, all instances of the class and the triplets whose subject is one of the instances. Moreover, the triplets, whose object is an instance described not only in the database but also in other databases, are shown so that data integration via instances can be realised. A selected RDF class with its instances can be shown in this view. However, using a structure definition file generated from a spreadsheet of tabular data, the column names and column order can be customised.

An example of tabular view is shown in Fig. 2. The first column is a list of instances of the class. The second and subsequent columns form sets, each of which is associated with a property and describes a list of objects of triplets with the same property. Furthermore, the columns are reversely linked to the instances listed in the first column; thus, each column is associated with a property of triplets reversely linked to the instances of the first column.

By default, the name of the first column is the label of the class and those of the second and subsequent columns are the labels of the corresponding properties. However, a structure definition file can be uploaded and the column names can be overwritten by the column names in the structure definition.

The data in each row can be sorted in ascending or descending order as specified by the user. Furthermore, the data in the rows can be filtered by full-text search for humanly readable metadata of the data records using keywords specified by the user for each column.


1 - 20 of 11837 | 1 2 3 4 5 6 7 8 9 10 Last | 20 | ☑ Show URI

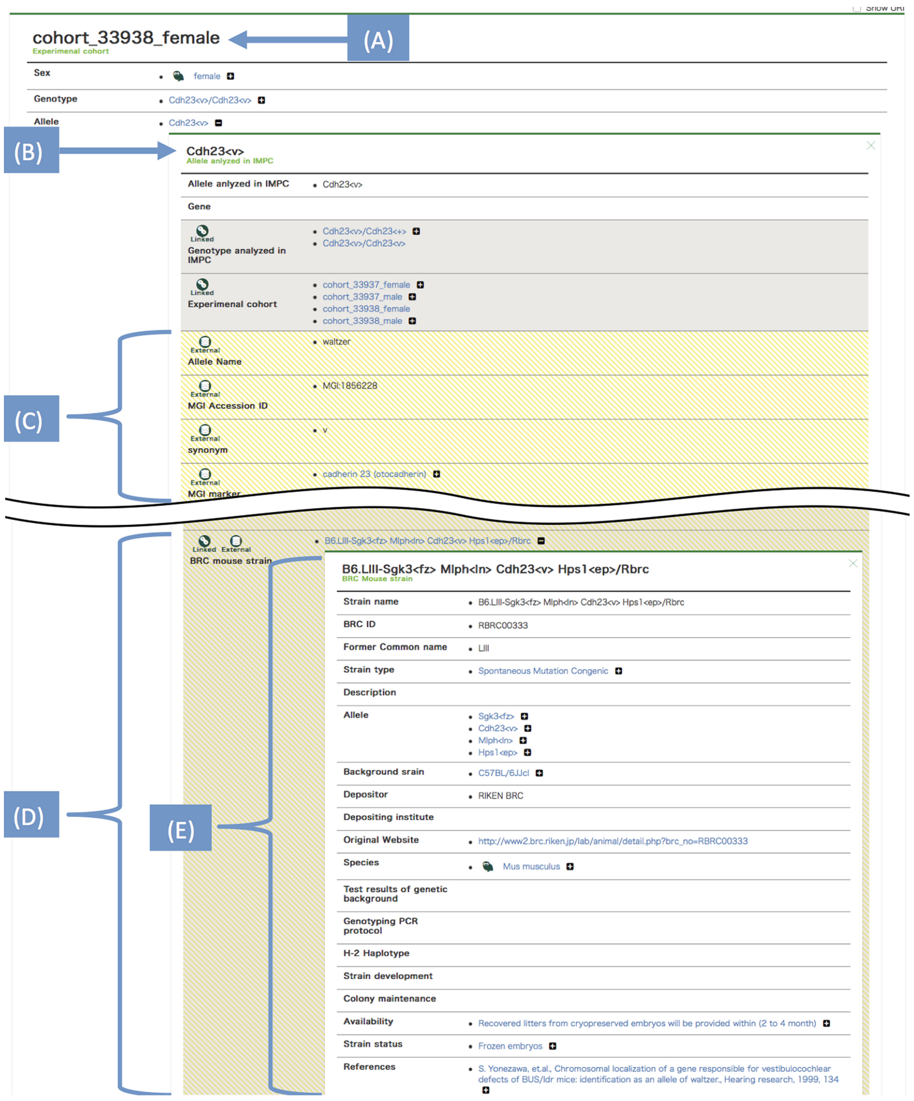
Habitat http://www.w3.org/2000/01/rdf-schema#label	Environmental feature http://purl.jp/bio/01/mccv#MCCV_000071	Linked Sample http://purl.jp/bio/01/mccv#MCCV_000072 (A)
• Habitat of JCM 10002 http://purl.jp/bio/01/mccv#MCCV_000007_1	• plant associated http://purl.jp/bio/11/meo/MEO_0000419 • Lathyrus odoratus http://purl.obolibrary.org/obo/NCBITaxon_3859 (B1)	• Sample from Fasciation of sweet peas (Lathyrus odoratus) http://purl.jp/bio/01/mccv#MCCV_000006_9006645
• Habitat of JCM 10124 http://purl.jp/bio/01/mccv#MCCV_000007_100	• soil http://purl.jp/bio/11/meo/MEO_0000007	• Sample from Soil, southern area of Vietnam http://purl.jp/bio/01/mccv#MCCV_000006_9006748
• Habitat of JCM 11306 http://purl.jp/bio/01/mccv#MCCV_000007_1000	• soil http://purl.jp/bio/11/meo/MEO_0000007	• Sample from Nematode-suppressive soil, Costa Rica http://purl.jp/bio/01/mccv#MCCV_000006_9007678
• Habitat of JCM 8723 http://purl.jp/bio/01/mccv#MCCV_000007_10000	• plant material http://purl.jp/bio/11/meo/MEO_0000385	• Sample from Plant material http://purl.jp/bio/01/mccv#MCCV_000006_9005598
• Habitat of JCM 8724 http://purl.jp/bio/01/mccv#MCCV_000007_10001	• Cecum http://purl.jp/bio/11/meo/MEO_0000460 • Gallus gallus http://purl.obolibrary.org/obo/NCBITaxon_9031 (B2)	• Sample from Chicken caecum, Belgium http://purl.jp/bio/01/mccv#MCCV_000006_9005599
• Habitat of JCM 8725 http://purl.jp/bio/01/mccv#MCCV_000007_10002	• dairy product http://purl.jp/bio/11/meo/MEO_0000020	• Sample from Dried milk http://purl.jp/bio/01/mccv#MCCV_000006_9005600

Fig. 2. A snapshot of the tabular view of a class Habitat of the Japan Collection of Microorganisms (JCM) resource database (http://metadb.riken.jp/metadb/db/rikenbrc_jcm_microbe). (A) The third column (class Sample) includes the instances to link to the subject instances of the first column, namely reversely linked instances of class Sample. (B1,B2) Multiple objects with same subject and predicate pairs can be displayed in the form of a list in the corresponding cell.

Card View. Card view, as shown in Fig. 3, is mainly used to show an instance and its triplets, which is linked to other instances or is reversely linked from other instances. In the card view, a user can view a long triplet path by traversing the connected triplets, in a sequence, from the corresponding instance. By default, only triplets including that particular instance are shown. A user can select an instance connected via a triplet to show further triplets having the selected instance, and the new triplets are shown as a new nested card in the original card view.

6 Implementation

Reducing of both development and operational costs is most important for realising persistent database services worldwide while ensuring service stability. In our implementation of RIKEN MetaDatabse, we adopt a simple architecture consisting of two components: (a) a web server providing GUI and (b) an RDF triplet store. The web server provides web pages having data display functions through a data display view, as described above. The RDF data displayed on a view are obtained from the RDF triplet store. In addition, the web server



cohort_33938_female
Experimenal cohort

Sex	• female
Genotype	• Cdh23<v>/Cdh23<v>
Allele	• Cdh23<v>

Cdh23<v>
Allele anlyzed in IMPC

Allele anlyzed in IMPC	• Cdh23<v>
Gene	
Genotype analyzed in IMPC	• Cdh23<v>/Cdh23<+> • Cdh23<v>/Cdh23<v>
Experimenal cohort	• cohort_33937_female • cohort_33937_male • cohort_33938_female • cohort_33938_male
Allele Name	• waltzer
MGI Accession ID	• MGI:1856228
synonym	• v
MGI marker	• cadherin 23 (otocadherin)
BRC mouse strain	• B6.LIII-Sgk3<fz> Mlph<ln> Cdh23<v> Hps1<ep>/Rbrc

B6.LIII-Sgk3<fz> Mlph<ln> Cdh23<v> Hps1<ep>/Rbrc
BRC Mouse strain

Strain name	• B6.LIII-Sgk3<fz> Mlph<ln> Cdh23<v> Hps1<ep>/Rbrc
BRC ID	• RBRC00333
Former Common name	• LIII
Strain type	• Spontaneous Mutation Congenic
Description	
Allele	• Sgk3<fz> • Cdh23<v> • Mlph<ln> • Hps1<ep>
Background srain	• C57BL/6JJcl
Depositor	• RIKEN BRC
Depositing institute	
Original Website	• http://www2.brc.riken.jp/lab/animal/detail.php?brc_no=RBRC00333
Species	• Mus musculus
Test results of genetic background	
Genotyping PCR protocol	
H-2 Haplotype	
Strain development	
Colony maintenance	
Availability	• Recovered litters from cryopreserved embryos will be provided within (2 to 4 month)
Strain status	• Frozen embryos
References	• S. Yonezawa, et.al., Chromosomal localization of a gene responsible for vestibulocochlear defects of BUS/Idr mice: identification as an allele of waltzer., Hearing research, 1999, 134

Fig. 3. A snapshot of the card view of an instance of an experimental cohort (http://metadb.riken.jp/db/IMPC_RDF/Cohort) used in the International Mouse Phenotype Consortium (IMPC) database (described in Sect. 8.2 in detail) that links to the instances of other databases. (A) is an instance of the Cohort class of KO mice, (B) represents an allele, *Cdh23-v* (*walther*), carried by the cohort in the IMPC database and described in the URI of the Mouse Genome Informatics (MGI) RDF database, (C) is a list of triplets in the MGI RDF database, (D) is a list of links from the BioResouce Center (BRC) Mouse Strain database and (E) is the detailed information of the mouse strain which has multiple alleles including *Cdh23-v* in the BRC Mouse Strain database.

functions as a SPARQL endpoint for submitting SPARQL queries generated by the web server. In our current platform, we employ the Openlink Virtuoso open-source version 7 as the RDF triplet store, and the web server is implemented as a Java servlet using Apache Tomcat version 8. To ensure stability, portability and continuity of the platform, we deploy these software components on RIKEN's private cloud, called RIKEN Cloud Service (http://cloudinfo.riken.jp), which provides multi-purpose Linux-based virtual machines. The web server (a) is located on a virtual machine connected to the global network. The RDF triplet store (b) is deployed on a specialised virtual machine to realise fast SPARQL operations, which connects a 1 TB flash memory storage via an InfiniBand network where the Virtuoso database directory is located.

7 Available Databases

As of July 2016, 21 public ontologies, including Gene Ontology (GO), Phenotypic Quality Ontology (PATO), NCBI Organismal Classification (NCBITaxon) and Semanticscience Integrated Ontology (SIO) have been selected and published as mirrors. These ontologies refer to 110 databases including 59 of RIKEN's original databases. The remaining 51 databases are external databases that are converted from originally non-RDF databases and linked from RIKEN's databases. In total, RIKEN MetaDatabase carries 148 million triplets, 797 classes, 2.94 million instances and 1,352 properties. The original databases are from various research fields, e.g., FANTOM (mammalian [5]), FOX Hunting (plant [6]), Heavy-atom Database System (protein [7]) and Metadata of BioResouce Center (BRC) resources (bioresources [8–10]).

7.1 Database Directory Service

RIKEN MetaDatabase provides a specialised database and RDF datasets that provide easy access to data. The specialised database is the RIKEN Database Directory, which is a catalogue of RIKEN's databases including the non-RDF databases. The catalogue data are designed to be compatible with the Integbio Database Catalog (http://integbio.jp/dbcatalog/?lang=en), which aims at inter-ministry integration of life-science databases in Japan. In addition, W3C's Health Care and Life Sciences (HCLS) Community Profile data (http://www.w3.org/TR/hcls-dataset/) including statistics data are generated for each database and for entire datasets, and are published as RDF archives and via the SPARQL endpoint. RIKEN MetaDatabase also provides the SPARQL Builder Metadata (http://sparqlbuilder.org/), which are generated and published for more intelligent SPARQL search. The SPARQL Builder Metadata is a profile of the SPARQL endpoint that describes the RDF graph structure. Thus, the SPARQL Builder tool [11] generates a SPARQL query that obtains triplet paths connecting two ontological concepts specified by the user.

8 Discussion

We developed RIKEN MetaDatabase as a cloud-based database platform, which realises Semantic-Web-based data integration with a simplified workflow, implemented through the cooperation of biologists and informaticians. In this section, we comprehensively review our methodology and the development process and operation of RIKEN MetaDatabase from various perspectives.

8.1 Contributions of RIKEN MetaDatabase for Different Types of Users

For Database Publishers. A data publisher is a biologist who has research results, converts the data into RDF and publishes the converted RDF data. The advantages of data generation using a spreadsheet are summarised as follows:

1. new columns can be easily added,
2. text format data such as TSV can be easily imported, and
3. the readability of tabular data is high for humans.

Adding new columns is required for including a triplet corresponding to a new RDF property. This feature is enabled because the RDF is open for adding new data (the open world assumption). Importing text format allows bioinformaticians to input data derived through the existing techniques where life-science data processing is often performed using script languages such as Perl and text data are often used for data exchange, rather than the RDF graph format. Finally, the tabular form is suitable for data typing and data confirmation before publication.

Especially for biologists, this methodology does not solve RDF-specific difficulties such as usage of URIs for data resource identification and selecting suitable vocabularies including ontologies, properties and classes. However, these difficulties are successfully reduced by generating spreadsheet templates in collaboration with informaticians.

Data integration based on RDF is also to the publisher's advantage. Though the integration can be realised by creating semantic links from one publisher's data to another's, as RDF triplets, a more attractive advantage is that data published later may be linked to existing data, which is already integrated by the original publishers. Furthermore, the appropriate data to link new data can be easily discovered through the tabular view without SPARQL.

For database users. Previously, database users had great difficulties in discovering the types of databases available, where these were published and how to use them. RIKEN MetaDatabase collects metadata of databases published by RIKEN and functions as a one-stop-shop of databases. Indeed, these metadata are published as a database catalogue in the RIKEN Database Directory and the HCLS Community Profile using standardised vocabularies, which help users to discover the data.

Furthermore, by employing standards for metadata publication such as RDF and SPARQL, RIKEN MetaDatabase provides standardised API to data access as a SPARQL endpoint. In addition, for users who are not familiar with RDF, it provides intuitive data views such as the tabular data view—which is a popular form for biologists, facilitating data view and operation.

For the RIKEN institute. Since RIKEN has researchers in various fields, including genome, plant, animal, brain, medical, bioresource and informatics, we can handle a wide range of metadata descriptions and bio-medical concepts. Development of a novel ontology is required for new types of research data and concepts. RIKEN easily realises this collaboration among various researchers for an internal collaborative research. Consequently, we are accomplishing the difficult task of the ontology development. We propose that this collaborative metadata integration model should be used in an open environment.

8.2 Contributions of RIKEN MetaDatabase to Inter-labs and Global Data Integration

In this section, we present the main contributions of RIKEN MetaDatabase.

Open data promotion. The development of RIKEN MetaDatabase is a step toward open access to research data. The platform provides easy and interactive access to previously untapped data stored in laboratory records. In addition, RIKEN MetaDatabase facilitates an easy, rapid and cost-effective publication of databases by small laboratories. For example, in the case of ENU-induced Mutations in RIKEN Mutant Mouse Library (http://metadbdev.riken.jp/sandbox/db/BRC-ENU-inducedMutationsInRIKENMutantMouseLibrary), the data developer did not have sufficient expertise and hardware to develop a public database. Using RIKEN MetaDatabase, they easily published their data on the Web. Furthermore, through collaboration with us and the RDF experts in DNA Data Bank of Japan (DDBJ), their data were integrated with data within RIKEN and DDBJ by applying a common data scheme as shown below.

Data coordination referring common data resources. The second contribution is the data coordination between different databases hosted in RIKEN MetaDatabase to share common URI of instances. In order to describe alleles and genes in mice, we applied common gene records (http://metadb.riken.jp/metadb/db/mgi_rdf) imported from the Mouse Genome Informatics (MGI) database (http://www.informatics.jax.org). Here, the MGI approved the publication of the RDF version of the mouse gene records. We have promoted the common use of MGI gene records in RIKEN MetaDatabase. As a result, MGI records are used in multiple databases, such as Metadata of BRC mouse resources and phenotypes (http://metadb.riken.jp/metadb/db/rikenbrc_celle), Metadata of Functional Glycomics with KO mice database (http://metadb.riken.jp/metadb/db/Glycomics_mouse) and the International Mouse Phenotype Consortium (IMPC: http://www.mousephenotype.org) RDF data (http://metadb.riken.jp/metadb/db/IMPC_RDF). In these databases, the data items related to genes are linked to MGI allele or gene records. Through this association, the integrated information, including what public experimental material (mouse strain in this case) are available correspond to the phenotype data published in IMPC, can be obtained, as shown in Fig. 3.

Scheme-level integration across databases. The third contribution is towards scheme-level integration. In the ENU-induced Mutations in RIKEN Mutant Mouse Library, next-generation sequencing (NGS) metadata are described in the common RDF scheme, which is developed by cooperation of DDBJ and RIKEN, based on the broadly used XML scheme for the NGS metadata. Using this scheme, RIKEN plans to develop a unified pipeline to publish NGS metadata on the Web and deposit NGS data as public archives operated by DDBJ. It is expected that this pipeline will promote worldwide sharing of NGS data from RIKEN. Moreover, metadata from the Japan Collection of Microorganisms (JCM) resources (http://metadb.riken.jp/metadb/db/rikenbrc_jcm_microbe) are described based on a common RDF scheme for strains of microorganisms, the Microbial Culture Collection Vocabulary (MCCV: http://bioportal.bioontology.org/ontologies/MCCV), which is used in MicrobeDB.jp (http://microbedb.jp/). The integrated database represents the encyclopaedia of microbes based on metagenome data. By applying the MCCV, basic information on the microbe strains released from JCM can be related to the metagenome data in MicorbeDB.jp. Phenotype data of experimental animals are also integrated by the J-phenome project (http://jphenome.info). J-phenome is a portal of phenotype databases hosted by RIKEN MetaDatabase in which the RDF scheme for the description of animals' phenotypes are unified using common phenotype ontologies such as the Mammalian Phenotype Ontology PATO. The unified scheme contributes to the development of a common application to determine the cross-species relationship between phenotypes using an inter-ontology relationship library produced by machine reasoning [12,13]. In summary, scheme-level integration in RIKEN MetaDatabase contributes to the common use of query, application and workflow pipelines to handle the same (or similar) data across databases.

International collaboration. The integration of multiple datasets in RIKEN MetaDatabase contributes to international collaboration. IMPC is an umbrella of comprehensive phenotyping of mouse mutants [14]. Through the cooperation of this consortium, multiple research centres released measurement data produced from the standardised phenotyping pipeline. As a member of IMPC, RIKEN BRC produced RDF version of IMPC phenotype data including more than 50 million triplets, which are now hosted by RIKEN MetaDatabase. Although the IMPC website provides a rich interface, visualising various phenotype data, the RDF version of the IMPC data presented in RIKEN MetaDatabase can be used by data scientists who want to integrate these phenotype data with other datasets related from different databases. For example, using the SPARQL endpoint of RIKEN MetaDatabase (http://metadb.riken.jp/sparql), a data user can perform a federated query between RIKEN and EBI RDF platform (https://www.ebi.ac.uk/rdf/) to retrieve what phenotype can be expressed when a specific biological pathway is inactivate, utilising the connection of IMPC dataset and Reactome (http://www.reactome.org) dataset.

In summary, using RIKEN MetaDatabase, seamless data integration can be performed from the inner-research institute to the worldwide level.

8.3 Open Issues

RIKEN MetaDatabase is a simple database system and platform built on our private cloud infrastructure. Data generation and publication costs for biologists are reduced, since they do not need to prepare and operate their own server. Since the system does not support data access control, it cannot handle private datasets or datasets under development. However, the system is lightweight and requires only two virtual machines. Therefore, we can build multiple instances of the system on the private cloud for each research project, with each project having its own access control using a firewall. However, the federated SPARQL search is not available yet. Our future work will include the development of such effective federation among these instances, as an ideal database federation model on the global Web.

Since the publication of RIKEN MetaDatabase in April 2015, our efforts for data dissemination are in progress. So far, we have participated in international database projects such as IMPC, for mouse phenotype databases, and W3C's HCLS group, for database profile, so that our published metadata can be easily linked to other published datasets.

To promote the reuse of common URIs, discussion-based cooperation among database developers is currently promoted by forming a working group of representatives from research centres in RIKEN. However, we have not yet implemented the automatic ontology annotation function on the RIKEN MetaDatabase. To address this issue, application of RightField in data construction workflow may prove useful in expansion of Excel spreadsheets, allowing semi-automatic ontology annotation.

9 Conclusions

We discussed the requirement specifications, design, development and operation of a database platform called RIKEN MetaDatabase handled by the comprehensive research institute RIKEN. One of the major difficulties is the practical co-localisation of open data framework RDF and the development of simple data processing methods for biologists. In order to solve these issues, we developed a template spreadsheet for data creation, which is a GUI that realises intuitive data views including tabular view. The database platform is deployed on our private cloud infrastructure and multiple system instances can be generated. Thus far, data integration from different research fields, such as IMPC, has been successfully realised on the platform.

Future work includes the realisation of practical federation among multiple system instances, so that an integrated database can be realised that supports our proposed data views. This will be accomplished by developing an individual database for each research project in a distributed environment and intelligent support for selecting suitable vocabularies for biologists.

References

1. Vasilevsky, N., Johnson, T., Corday, K., Torniai, C., Brush, M., Segerdell, E., Wilson, M., Shaffer, C., Robinson, D., Haendel, M.: Research resources: curating the new eagle-i discovery system. Database (Oxford), 20:2012 (2012)
2. Belleau, F., Nolin, M.A., Tourigny, N., Rigault, P., Morissette, J.: Bio2RDF: towards a mashup to build bioinformatics knowledge systems. J Biomed. Inform. **41**(5), 706–716 (2008)
3. 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. **39**(Web Server issue): W541–W545 (2011)
4. Wolstencroft, K., Owen, S., Horridge, M., Krebs, O., Mueller, W., Snoep, J.L., du Preez, F., Goble, C.: RightField: embedding ontology annotation in spreadsheets. Bioinformatics **27**(14), 2021–2012 (2011)
5. The FANTOM Consortium and the RIKEN PMI and CLST (DGT): A promoter-level mammalian expression atlas. Nature **507**, 462–470 (2014)
6. Ichikawa, T., Nakazawa, M., Kawashima, M., Iizumi, H., Kuroda, H., Kondou, Y., Tsuhara, Y., Suzuki, K., Ishikawa, A., Seki, M., Fujita, M., Motohashi, R., Nagata, N., Takagi, T., Shinozaki, K., Matsui, M.: The FOX hunting system: an alternative gain-of-function gene hunting technique. Plant J. **45**, 974–985 (2006)
7. Sugahara, M., Asada, Y., Shimada, H., Taka, H., Kunishima, N.: HATODAS II: heavy-atom database system with potentiality scoring. J. Appl. Crystallogr. **42**, 540–544 (2009)
8. Yoshiki, A., Ike, F., Mekada, K., Kitaura, Y., Nakata, H., Hiraiwa, N., Mochida, K., Ijuin, M., Kadota, M., Murakami, A., Ogura, A., Abe, K., Moriwaki, K., Obata, Y.: The mouse resources at the RIKEN BioResource center. Exp. Anim. **58**(2), 85–96 (2009)
9. Nakamura, Y.: Bio-resource of human and animal-derived cell materials. Exp. Anim. **59**(1), 1–7 (2010)
10. Yokoyama, K.K., Murata, T., Pan, J., Nakade, K., Kishikawa, S., Ugai, H., Kimura, M., Kujime, Y., Hirose, M., Masuzaki, S., Yamasaki, T., Kurihara, C., Okubo, M., Nakano, Y., Kusa, Y., Yoshikawa, A., Inabe, K., Ueno, K., Obata, Y.: Genetic materials at the gene engineering division. RIKEN BioResource Center. Exp. Anim. **59**(2), 115–124 (2010)
11. Yamaguchi, A., Kozaki, K., Lenz, K., Wu, H., Yamamoto, Y., Kobayashi, N.: Efficiently finding paths between classes to build a SPARQL query for life-science databases. In: Qi, G., Kozaki, K., Pan, J.Z., Yu, S. (eds.) JIST 2015. LNCS, vol. 9544, pp. 321–330. Springer, Heidelberg (2016). doi:10.1007/978-3-319-31676-5_24

12. Hoehndorf, R., Schofield, P.N., Gkoutos, G.V.: PhenomeNET: a whole-phenome approach to disease gene discovery. Nucleic Acids Res. **39**(18), 119 (2011)
13. Robinson, P.N., Khler, S., Oellrich, A.: Sanger Mouse Genetics Project, Wang, K., Mungall, C.J., Lewis, S.E., Washington, N., Bauer, S., Seelow, D., Krawitz, P., Gilissen, C., Haendel, M., Smedley, D.: Improved exome prioritization of disease genes through cross-species phenotype comparison. Genome Res. **24**(2), 340–348 (2014)
14. Dickinson, M.E., Flenniken, A.M., Ji, X., Teboul, L., Wong, M.D., White, J.K., Meehan, T.F., Weninger, W.J., Westerberg, H., Adissu, H., et al.: High-throughput discovery of novel developmental phenotypes. Nature **537**(7621), 508–514 (2016)

A Preliminary Investigation Towards Improving Linked Data Quality Using Distance-Based Outlier Detection

Jeremy Debattista(✉), Christoph Lange, and Sören Auer

University of Bonn and Fraunhofer IAIS, Bonn, Germany
{debattis,langec,auer}@cs.uni-bonn.de

Abstract. With more and more data being published on the Web as Linked Data, Web Data quality is becoming increasingly important. While quite some work has been done with regard to quality assessment of Linked Data, only few works have addressed quality improvement. In this article, we present a preliminary an approach for identifying potentially incorrect RDF statements using distance-based outlier detection. Our method follows a three stage approach, which automates the whole process of finding potentially incorrect statements for a certain property. Our preliminary evaluation shows that a high precision is maintained with different settings.

Keywords: Outlier detection · Data quality · Linked data

1 Introduction

A rationale of the Semantic Web is to provide real-world things, also called *resources*, with descriptions in common data formats that are meaningful to machines. Furthermore, Linked Data emphasises on the reuse and linking of these resources, thus assisting in the growth of the *Web of* (meaningful) *Data*. Schemas, some being lightweight and others being more complex, have been defined for various use cases and application scenarios in order provide structure to the descriptions of semantic resource based on a common understanding. Nevertheless, since linked datasets are usually originating from various structured (e.g. relational databases), semi-structured (e.g. Wikipedia) or unstructured sources (e.g. plain text), a complete and accurate *semantic lifting* process is difficult to attain. Such processes can often contribute to incomplete, misrepresented and noisy data, especially for semi-structured and unstructured sources. Issues caused by these processes can be attributed to the fact that either the knowledge worker is not aware of the various implications of a schema (e.g. incorrectly using inverse functional properties), or because the schema is not well defined (e.g. having an open domain and range for a property). In this article, we are concerned with the latter, aiming to identify potentially incorrect statements in order to improve the quality of a knowledge base.

Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 116–124, 2016.
DOI: 10.1007/978-3-319-50112-3_9

When analysing the schema of the DBpedia dataset we found out that from around 61,000 properties, approximately 59,000 had an undefined domain and range. This means that the type of resources attached to such properties as the subject or the object of an RDF triple can be very generic, i.e. `owl:Thing`. Whilst this is not forbidden, it makes a property ambiguous to use. For example, the property `dbp:author`, whose domain and range are undefined, has instances where the subject is of type `dbo:Book` and the object of type `dbo:Writer`, and other instances where the subject is of type `dbo:Software` and the object of type `dbo:ArtificialSatellite`.

The key research question in this paper is *can distance-based outlier techniques help in identifying quality problems in linked datasets?* In this article we investigate how triples can be clustered together based on their distance. This distance is identified by a semantic similarity measure that takes into consideration the subject type, object type, and the underlying schema. Furthermore, we evaluate complementary aspects of the proposed approach. More specifically, we were interested to see how different settings in our approach affect the precision and recall values.

This article is structured as follows. The state-of-the-art is described in Sect. 2. Our proposed approach is explained in Sect. 3. Experiments of our approach are documented in Sect. 4. Conclusions and an outlook to future work are discussed in Sect. 5.

2 Related Work

Various research efforts have tackled the problem of detecting incorrect RDF statements using different techniques. These include *statistical distribution* [8], *schema enrichment* [9,12] and *crowdsourcing* [1,10]. Outlier detection techniques such as [11] are used to validate the correctness of data literals in RDF statements, which is out of the scope of this research as our approach considers only statements where the subject and object are resources.

Statistical Distribution. Paulheim et al. [8] describe an algorithm based on the statistical distribution of types over properties in order to identify possibly faulty statements. Statistical distribution was used in order to predict the probability of the types used on a particular property, thus with some confidence verify the correctness of a triple statement. Their three step approach first computes the frequency of the predicate and object combination in order to identify those statements that have a low value. Cosine similarity is then used to calculate a confidence score based on the statement's subject type probability and the object type probability. Finally, a threshold value is applied to mark those statements that are potentially incorrect. Our approach uses semantic similarity to identify whether a statement could be a possibly incorrect statement or not, instead of statistical distribution probabilities. Therefore, our similarity approach takes into consideration the semantic topology of types and not their statistical usage.

Schema Enrichment. Schema enrichment is also a popular technique to detect incorrect statements. Töpper et al. [9] enrich a knowledge base schema with additional axioms before detecting incorrect RDF statements in the knowledge base itself. Such an approach requires external knowledge in order to enrich the ontology. Similarly, Zaveri et al. [12] apply a semi-automated schema enrichment technique before detecting incorrect triples.

Crowdsourcing WhoKnows? [10] is a crowdsourcing game where users contribute towards identifying inconsistent, incorrect and doubtful facts in DBpedia. Such crowdsourcing efforts ensure that the quality of a dataset can be improved with more accuracy, as a human assessor can identify such problems even from a subjective point of view. During the evaluation, the users identified 342 triples that were potentially inconsistent from a set of overall 4,051 triples, reporting a precision value of 46%. A similar crowdsourcing effort was undertaken by Acosta et al. in [1]. They used pay-per-hit micro tasks as a means of improving the outcome of crowdsourcing efforts. Their evaluation focuses on checking the correctness of the object values and their data types, and the correctness of interlinking with related external sources, thus making it incomparable to our approach. In contrast to crowdsourcing, our preliminary approach gives a good precision in identifying outliers without the need of any human intervention, in an acceptable time (± 3 min to compute outliers of a 10 K dump). Nonetheless, at some point, human expert intervention would still be required (in our approach) to validate the correctness of the detected outliers, but with any (semi-)automatic learning approaches, human intervention is reduced.

3 Improving Dataset Quality by Detecting Incorrect Statements

The detection and subsequent cleaning of potentially incorrect RDF statements aids in improving the quality of a linked dataset. There were a number of attempts to solve this problem in the best possible manner (cf. Sect. 2). We apply the distance-based outlier technique by Knorr et al. [6] in a Linked Data scenario. Exploiting reservoir sampling and semantic similarity measures, clusters of RDF statements based on the statement' subject and object types are created, thus identifying the potentially incorrect statements. We implemented[1] this approach as a metric for *Luzzu* [2].

3.1 Approach

Following [6], our proposed Linked Data adapted method has three stages: *initial*, *mapping*, and *colouring*. These three stages automate the whole process of finding potentially incorrect statements for a certain property. In the *initial* stage, k (the size of the reservoir) RDF statements are added to a reservoir sampler. Following the initialisations of the constants, the *mapping* stage groups data objects in various cells based on the mapping properties described in [6]. Finally, the *colouring* stage identifies the cells that contain outlier data objects.

[1] The Java code can be found in our GIT repository: https://goo.gl/bGRKxi.

Initial Stage. The initial steps are crucial for achieving a more accurate result, i.e. a better identification of potentially incorrect statements. We start by determining the approximate distance D that is used in the second stage to condition the mapping, and thus the final clustering of RDF statements. The approximate value D is valid for a particular property, i.e. the property whose triples are being assessed. Therefore, two properties (e.g. *dbp:author* and *dbp:saint*, i.e. the patron saint of, e.g., a town) will have different values of D according to the triples, their types, and ultimately the similarity measure chosen. Currently, in our approach we assume that a resource is typed with **only** one class, choosing the most specific type if a resource is multi-typed (e.g. *dbo:Writer* and not *dbo:Person*). Additionally, a threshold fraction p (between 0 and 1) is defined by the user during the initial phase, affecting the number of data objects in a cluster M. Therefore, p can be considered to be a sensitivity function that increases or decreases the amount of data objects in a cluster.

Determining the Approximate Distance. Our approach makes use of reservoir sampling as described in [3]. The rationale is that D is approximated by a sample of the data objects being assessed, to identify the acceptable maximum distance between objects mapped together in a cell, in a quick and automated way. To determine the approximate distance we applied two different implementations (cf. Sect. 4 for their evaluation), one based on a simple sampling of triples and another one based on a modified reservoir sampler, which we call the *type-selective*. From the sample set (for both implementations), a random data object is chosen to be the *host*, and is removed from the sampler. All remaining statements in the sampler are semantically compared with the host individually and their distance values are stored in a list. The median distance is than chosen from the list of distances. We chose the median value over the mean value as a central tendency since the latter can be influenced by outliers.

In the first implementation (simple sampling), the reservoir selects a sample of triples, irrelevantly of their subject and object types. The main limitation is that, irrelevantly of the size of the reservoir, the approximate distance D value can bias towards the more frequent pairs of the subject and object types. Therefore, the sampler might not represent the broad types attached to the particular property being assessed.

In order to attempt to solve the *sampler representation problem*, we propose the *type-selective* reservoir sampler. The proposed reservoir sampler modifies the simple sampler by adding a condition that only one statement with a certain subject type and object type can be added to the reservoir. In other words, when there are two distinct statements with matching subject types and object types, only one of these statements will be added to the reservoir.

Mapping Stage. The mapping stage attends to the clustering of data objects (i.e. RDF statements in our case) in cells. An RDF statement is chosen at random from the whole set of data objects and is placed in a random cell. This is called the *host* cell. Thereafter, every other RDF statement in the dataset is mapped to an appropriate cell by first comparing it to the data object in this host cell.

Semantic Similarity Measure. In order to check if an RDF statement fits in a cell with other similar RDF statements, a semantic similarity measure is used. More specifically, since we are mostly concerned about the distance between two statements, we use a normalised semantic similarity measure. The similarity between two statements S_1 and S_2 is defined as the average of the similarity between the statements' subjects, and the similarity between the statements' objects.

Colouring Stage. After mapping all data objects to the two-dimensional space, the *colouring* process colours cells to identify outlier data objects, based on the process identified in [6]. In [6], the minimum number of objects (M) required in a cell such that data objects are not considered as outliers is calculated as $M = N \cdot (1 - p)$ where N is the total number of data objects, and p is the threshold fraction value determined in the *initial* stage.

4 Experiments and Evaluations

The primary aim of this experiment is to compare if the automatic approach of setting approximate D value gives an advantage over the manual setting. All experiments in this part of the evaluation used the same similarity measure configuration, i.e. Zhou IC [13] with the Mazandu measure [7], as implemented in the Semantic Measures Library & Toolkit [4].

This experiment is split into two sub-experiments. In the first part, we evaluated triple statements in DBpedia with the predicate http://dbpedia.org/property/author using the proposed approach with the p and D parameters manually set to determine the precision and recall values. In the second part of this evaluation we repeat this experiment but the value of D is determined by the two automated approaches described in Sect. 3. For both experiments, p was set to: 0.99, 0.992, 0.994, 0.996, and 0.998.

Sub-experiment #1 – Setting Approximate D Manually. In this manual experiment, the D value for the evaluated property was obtained as an estimate from a manual calculation of the similarity values of the different types. From Fig. 1, we observe that on average our approach achieved around 76% precision. On the other hand, the recall values were low, with an average of 31%. We also observed that increasing the approximate value D does not result in an increasing precision. For example, in Fig. 1 we spot that the precision value for the D value of 0.3335 is greater than that of 0.3555 when p was set to 0.996. When D was set to 0.3555, 39 more outliers were detected, (true positives –7, false positives +42 data objects). This slight change in *true positives* and *false positives* was expected as the data objects cluster with similar data objects whose distance is the smallest. Therefore, the change in D might have moved some objects from one cell to another with the consequence that a previously non-outlier cell is now marked as an outlier, since a number of data objects might have moved to other

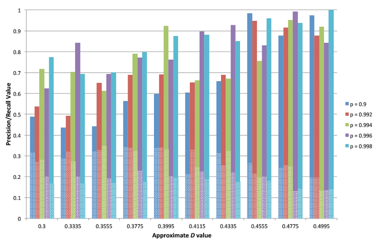


Fig. 1. The precision and recall values for the authors property dump with different values for D and p. The solid bars denote precision values, whilst the striped overlapped bars denote recall.

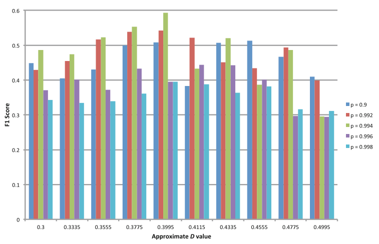


Fig. 2. The F1 score authors property dump with different values for D and p.

cells. Figure 2 represents the F1 score for the authors property dump manual experiment, showing an average of almost 43% for this harmonic mean score.

Sub-experiment #2 – Setting Approximate *D* Automatically. The same evaluated property was used in this experiment, where first an approximate D value was calculated first using the *simple* reservoir sampler and then using the *type-selective* reservoir sampler. A single host was chosen randomly from these reservoir samplers, together with a starting host location. The choice of a random data object will not affect the precision of the algorithm, as all data objects will be compared and mapped in suitable cells. From Fig. 3 we observe that the *type-selective* sampler outperforms its simpler counterpart for all p values with regard to the precision. One possible reason is due to the low approximate D values identified by the simple reservoir sampler. Low approximate D values mean that less data objects get mapped together in cells, since the approximate distance becomes smaller and data objects will be dispersed throughout the whole 2D space. This means that since less data objects are mapped in the same cell or surrounding cells, it would be more difficult to reach the $M + 1$ quota, and thus more cells will be marked as outliers. Therefore, whilst a low approximate D could lead for a decrease of *false positives* in non-outlier cells, it can also increase of *false negatives* (thus decreasing *true positives*), as objects that should not be marked as outliers could end up in outlier-marked cells. The main factors that affect the approximate D value are (1) the choice of the semantic similarity measure, and (2) the underlying schema (cf. limitations in Sect. 4.1). Furthermore, this approximate D value and the user-defined sensitivity threshold value (p) affect the precision and recall.

Following these experiments, in Fig. 4 we compared the *type-selective* precision and recall results for every p against the manual approach. For this comparison we used the manual scores that got the highest F1 measure for each p value, thus having a balance between the precision and recall. Figure 4 shows that the manual approach performed overall better than the automatic one in terms of

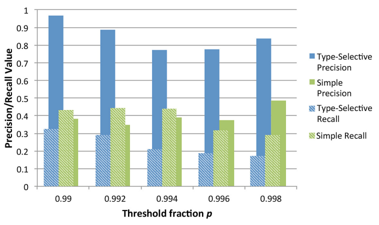


Fig. 3. The precision and recall values for the authors property dump with different values for p and a generated D value.

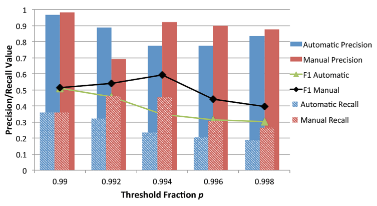


Fig. 4. Precision and recall values for the authors property dump comparing the manual results against the automatic results for multiple values of the fraction p.

the F1 measure. Nevertheless, in most cases, there are no large discrepancies between the two. The automatic approach resulted into a higher approximate D value than the manual approach. The approximation D value for the automatic approach was 0.482147, 0.0826 more than the given manual approximation D value with the highest F1 value (i.e. 0.3995 for threshold fraction p).

4.1 Discussion

The led evaluation is as yet not conclusive, since we only evaluated our approach with one property. This evaluation also showed that our approach produces a low recall value and thus a low F1 measure. A higher recall, without comprising the precision, would have been ideal, as with low recall we are missing a relevant data objects that should have been marked as outliers. One must also note that the choice of a semantic similarity measure will also affect the precision and recall values of such an approach, in a way that its results are the deciding factor where a data objects is mapped.

Nevertheless, our approach has a number of known limitations:

1. the approach is limited to knowledge bases without blank nodes, which can effect the degree of similarity, thus making this approach less robust and generic;
2. the approach does not fully exploit the semantics of typed annotations in linked datasets, since our approach assumes that an instance is a member of only one type, in particular the most specific type assigned to the resource;
3. the evaluated semantic similarity measures are limited to hierarchical '*is-a*' relations that might be more fitting to biomedical ontologies having deep hierarchies;
4. the sampled population might not reflect the actual diverse population of the data objects that have to be clustered in both sampler implementations. Thus, with both implementations we will not achieve the best representative sample, such as that obtained by stratified sampling [5];

5. whilst with the *simple* sampler outliers might occur in the sample population, with the *type-selective* sampler there is a 100% certainty that outlier data objects are present in the sample that determines the approximate D. Knorr et al. [6] had foreseen this problem and whilst suggesting that sampling provides a reasonable starting value for D, it cannot provide a high degree of confidence for D because of the unpredictable occurrence of outliers in the sample.

5 Conclusions

In this article we investigated the possibility of detecting potentially incorrect RDF statements in a dataset using a time and space efficient approach. More specifically, we applied a distance-based clustering technique [6] to identify outliers in a Linked Data scenario. While providing satisfactory results, our approach has a number of limitations that we are currently addressing. However, the preliminary results give us an indication on the research question set in the introduction. In the future, we aim to extend our experiments by using semantic relatedness measures instead of the semantic similarity measures, thus our distance based measure will also consider the semantic relationships between two terms, such as `owl:equivalentClass`.

References

1. Acosta, M., Zaveri, A., Simperl, E., Kontokostas, D., Auer, S., Lehmann, J.: Crowdsourcing linked data quality assessment. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013. LNCS, vol. 8219, pp. 260–276. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41338-4_17
2. Debattista, J., Auer, S., Lange, C.: Luzzu - a framework for linked data quality analysis. In: 2016 IEEE International Conference on Semantic Computing, Laguna Hills (2016)
3. Debattista, J., Londoño, S., Lange, C., Auer, S.: Quality assessment of linked datasets using the approximation. In: 12th European Semantic Web Conference Proceedings (2015)
4. Harispe, S., Ranwez, S., Janaqi, S., Montmain, J.: Semantic measures for the comparison of units of language, concepts or entities from text and knowledge base analysis, October 2013. arXiv abs/1310.1285
5. Hausman, J.A., Wise, D.A.: Stratification on endogenous variables and estimation: the gary income maintenance experiment. In: Manski, C.F., McFadden, D.L. (eds.) Structural Analysis of Discrete Data with Econometric Applications. MIT Press, Cambridge (1981)
6. Knorr, E.M., Ng, R.T., Tucakov, V.: Distance-based outliers: algorithms and applications. VLDB J. **8**(3–4), 237–253 (2000)
7. Mazandu, G.K., Mulder, N.J.: A topology-based metric for measuring term similarity in the gene ontology. Adv. Bioinf. **2012**, 1–17 (2012)
8. Paulheim, H., Bizer, C.: Improving the quality of linked data using statistical distributions. Int. J. Semant. Web Inf. Syst. **10**(2), 63–86 (2014)

9. Töpper, G., Knuth, M., Sack, H.: DBpedia ontology enrichment for inconsistency detection. In: Proceedings of the 8th International Conference on Semantic Systems, I-SEMANTICS 2012, pp. 33–40. ACM, New York (2012)
10. Waitelonis, J., Ludwig, N., Knuth, M., Sack, H.: WhoKnows? - evaluating linked data heuristics with a quiz that cleans up DBpedia. Int. J. Interact. Technol. Smart Educ. (ITSE) **8**(3), 236–248 (2011)
11. Wienand, D., Paulheim, H.: Detecting incorrect numerical data in DBpedia. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 504–518. Springer, Heidelberg (2014). doi:10.1007/978-3-319-07443-6_34
12. Zaveri, A., Kontokostas, D., Sherif, M.A., Bühmann, L., Morsey, M., Auer, S., Lehmann, J.: User-driven quality evaluation of DBpedia. In: Proceedings of the 9th International Conference on Semantic Systems, I-SEMANTICS 2013, pp. 97–104. ACM, New York (2013)
13. Zhou, Z., Wang, Y., Gu, J.: A new model of information content for semantic similarity in wordnet. In: FGCNS 2008 Proceedings of the 2008 Second International Conference on Future Generation Communication and Networking Symposia, vol. 3, pp. 85–89. IEEE Computer Society, December 2008

Information Retrieval and Knowledge Discovery

Linked Data Collection and Analysis Platform for Music Information Retrieval

Yuri Uehara(✉), Takahiro Kawamura, Shusaku Egami, Yuichi Sei, Yasuyuki Tahara, and Akihiko Ohsuga

Graduate School of Information Systems, University of Electro-Communications, Tokyo, Japan
{uehara.yuri,kawamura,egami.shusaku}@ohsuga.is.uec.ac.jp,
{seiuny,tahara,ohsuga}@uec.ac.jp

Abstract. There has been extensive research on music information retrieval (MIR), such as signal processing, pattern mining, and information retrieval. In such studies, audio features extracted from music are commonly used, but there is no open platform for data collection and analysis of audio features. Therefore, we build the platform for the data collection and analysis for MIR research. On the platform, we represent the music data with Linked Data, which are in a format suitable for computer processing, and also link data fragments to each other. By adopting the Linked Data, the music data will become easier to publish and share, and there is an advantage that complex music analysis will be facilitated. In this paper, we first investigate the frequency of the audio features used in previous studies on MIR for designing the Linked Data schema. Then, we build a platform, that automatically extracts the audio features and music metadata from YouTube URIs designated by users, and adds them to our Linked Data DB. Finally, the sample queries for music analysis and the current record of music registrations in the DB are presented.

Keywords: Linked data · Audio features · Music information retrieval

1 Introduction

Recently, there are a large number of studies on music. Music Information Retrieval (MIR) deals with music on computers and has been studied in various ways [1]. In these studies, audio features extracted from music are frequently used, however, there is no open platform for collecting data including the audio features for music analysis. Therefore, we propose the platform for MIR research in this paper.

On the platform, we used Linked Data format, since it is suitable for complex searches for audio features and songs-related metadata. Also, the music data in the Linked Data can be easily linked to the external databases (DBs) such as DBpedia[1], and then become more valuable when published and shared in

[1] http://wiki.dbpedia.org/.

public. Thus, we built a platform, that automatically extracts audio features of music, transforms them to graphs in the Linked Data, and then insert the graphs to our music Linked Data DBs with connections to the external DBs. Note that this platform is designed for music-related researchers and developers, who intend to analyze music information and create their own applications, e.g., recommendation mechanism. Use of a listener is beyond the scope of this paper.

The rest of the paper is organized as follows. Related works in terms of MIR research using audio features and Linked Data are shown in Sect. 2. In Sect. 3, we describe the schema design of music Linked Data. After the system for automatically analyzing audio features is proposed in Sect. 4, some examples of music analysis are described in Sect. 5. Finally, we conclude this paper with future works in Sect. 6.

2 Related Work

There is MusicBrainz, that is an open database about music. The MusicBrainz database[2] has the music data, such as song title, artist name, etc., which are described in Resource Description Framework (RDF). The data are created mainly by participants, and thus there is a mechanism of data registration that requires the verification and approval of the other participants in order to maintain the reliability of the data.

In addition, Music Ontology[3] offers the data model related to music. This provides vocabulary in the RDF model of the MusicBrainz data for describing the relationship of music information.

Also, in our previous research, the music recommendation using Linked Data is proposed in [2]. We constructed Linked Open Data (LOD) by the data retrieved from Last. fm, Yahoo! Local, Twitter, and Lyric Wiki. Then, they proposed a method for recommending songs according to associative relations in the LOD.

However, there is no data of audio features in the MusicBrainz database, the Music Ontology, and our previous LOD set. Therefore, we built a platform that provides audio features combined with music metadata in Linked Data format for open MIR research.

3 Schema Design of Music Information

In this section, designing Linked Data schema, including audio features and music metadata is described.

[2] https://musicbrainz.org/.
[3] http://musicontology.com/.

3.1 Selection of Audio Features

Audio features refer to the characteristics of the music, such as *Tempo* representing the speed of the track, tonality of track and quality of sound, the features used in MIR studies vary. For example, Osmalskyj et al. used *Tempo* and *Loudness* to identify cover songs [3]. Luo et al. used the audio features *Pitch*, *Zero crossing rate*, etc. to detect of common mistakes in novice violin playing [4]. Thus, it is necessary to survey which audio features should be prepared in the music schema.

Thus, we investigated the frequency of the audio features used in previous MIR studies. We collected 114 papers published in the International Society of Music Information retrieval (ISMIR)[4] in 2015, which is the top conference in the field of MIR. Table 1 shows the results.

Table 1. Number of using audio features

Audio Features	Count
Tempo	18
Pitch	10
MFCC	9
Beat	8
Loudness	7
Chord	5
Chroma, Key, Zero crossing rate, Roll off	3
Roughness, Timbre, Low energy, RMS energy, Brightness, Mode, Duration	2
Harmony, Volume, Articulation, Energy ratio, Swing ratio, Spectral irregularity, Inharmonicity, Vibrate, Rhythm, Dynamics	1

We found in Table 1 that *Tempo* representing the speed of music songs has been the most used in many studies, although there are some features which have been used just once. Then, we selected some of the audio features according to the following policies.

1. Features which are similar to each other can be integrated. (Beat, Swing ratio and Rhythm are integrated Tempo.)
2. Features appeared just once in the publications can be ignored. (Harmony, Volume, Articulation, Energy ratio, Spectral irregularity, Inharmonicity, Vibrate, Dynamics are deleted.)
3. Features, which cannot be extracted through a song, can be ignored, since a user's input is assumed to be song by song. Features in a series of songs can be extracted by querying for the resulted DB.(Pitch, Loudness, Chord, Chroma, Timbre, Duration are deleted.)
4. Features should be quantitative in numerical values, and qualitative ones like an emotional feature are not included.(MFCC is deleted.)

[4] http://www.ismir.net/.

3.2 Design of the Schema

Linked Data is an RDF format, in which data fragments are linked by any semantic relations.

We defined original properties for selected audio features, excluding *Key* and *Mode*, since there were no existing properties for them, or the properties are not appropriate for our purpose. In terms of *Key* and *Mode*, there are appropriate properties in the Music Ontology, and thus we used them. Then, we classified properties of audio features into some classes for making them easy to use. Table 2 shows the classes and properties corresponding to the audio features.

Table 2. Class and property of audio features

Class	Property	Audio Features	Count
Tempo	tempo	Tempo	28
Key	key	Key, Mode	5
Timbre	zerocross	Zero crossing rate	3
	rolloff	Roll off	3
	brightness	Brightness	2
Dynamics	rmsenergy	RMS energy	2
	lowenergy	Low energy	2

In Table 2, *Mode* can be included in the *Key* class, thus we used the same property. Based on these definitions, we designed the music schema and built the Linked Data set for music. *Tempo* means speed of the song, *Key* means tonality of song, *Mode* means volume difference of the major chord and a minor chord, *Zero crossing rate* means the rate at which the signal changes from positive to negative or back, *Roll off* means ratio of bass which accounts for 85 percent of the total, *Brightness* means ratio of high-range (more than 1500 Hz), *RMS energy* means the average of the volume (root mean square), *Low energy* means Ratio of sound low in volume. Figure 1 shows part of the Linked Data.

We designed the music schema with the video id (URI) of YouTube. In Fig. 1, the id: dvgZkm1xWPE indicates a song "Viva La Vida" by Coldplay. In the graph, the id node links to the classes of audio features and then links to each audio feature. Also, we added some degrees for categorizing numerical values in the features. The *lowenergy*, the *rmsenergy*, and the *brightness* have a class by 0.1, the *zerocross* has a class by 100, and the *rolloff* has a class by 1000. The *tempo* has tmarks based on tempo values[5], which is a measure of the speed marks: Slow means 39 or less bpm, Largo means 40–49 bpm, Lento means 50–55 bpm, Adagio means 56–62 bpm, Andante means 63–75 bpm, Moderato means 76–95 bpm, Allegretto means 96–119 bpm, Allegro means 120–151 bpm, Vivace

[5] http://www.sii.co.jp/music/try/metronome/01.html.

means 152–175 bpm, Presto means 176–191 bpm, Prestissimo means 192–208 bpm, Fast means over 209 bpm.

In addition, we extended the schema of music metadata, that includes not only song title, artist name, etc. in the Music Ontology, but also lyricist name, cd name for the complex search for music information. Figure 2 shows part of the metadata in our Linked Data. In the graph, the video id node links to the class of metadata, and then links to the detailed value, as well as the graph for the audio features. Also, some nodes such as the artist name are linked to the external DBs like DBpedia.

Figures 1 and 2 are the graphs of "Viva La Vida" by Coldplay, and thus the two graphs can be linked with the video id of YouTube.

4 Music Information Extracting

4.1 System Overview

The system architecture for our music information extraction is shown in Fig. 3, and its workflow is indicated by the number 1 to 11 as follows.

1. Download the video data from the YouTube video URI designated by a user in a web browser.
2. Call the MATLAB process that analyzes audio features in the video file.
3. Store the obtained audio features in an RDB, MySQL.
4. Call the RDF create program.
5. Obtain the music information for the video from the YouTube website.
6. Search the music metadata using Last.fm API.
7. Query the audio features of the video for MySQL.
8. Convert the metadata and audio features to RDF graphs, and store them in an RDF store, Virtuoso.
9. Notify the completion to the user.

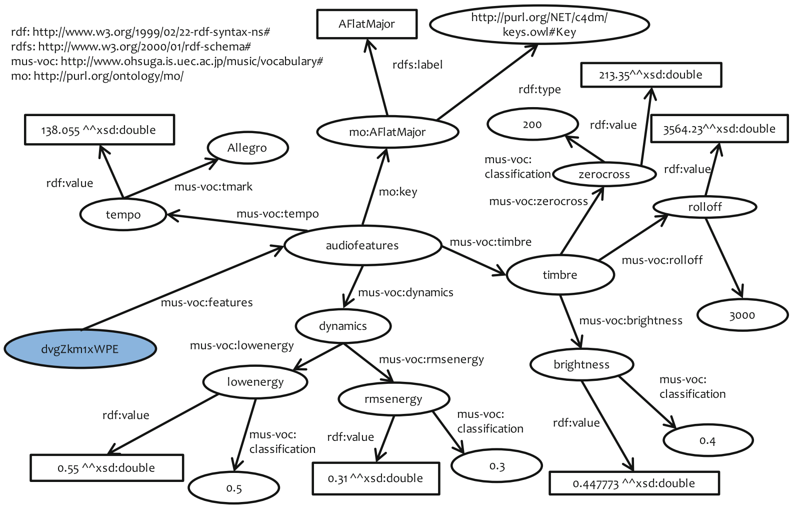


Fig. 1. Part of audio features in the Linked Data of music songs information

```
10. Submit a simple SPARQL query for confirmation.
11. Returns the evidence of the inclusion of new sub-graphs corresponding to the video.
```

Our system obtains videos to analyze audio features from YouTube, and so public users can easily extend the music information on the platform. However, we discard the video files after extracting the audio features, and thus we believe this process does not cause any legal or moral problems.

The workflow is divided into several phases. The first phase is for analyzing the audio features of the YouTube video, and the second phase is for acquiring the metadata of the YouTube video. Then, the third phase converts the metadata and audio features to RDF graphs, and the RDF graphs are stored in Virtuoso

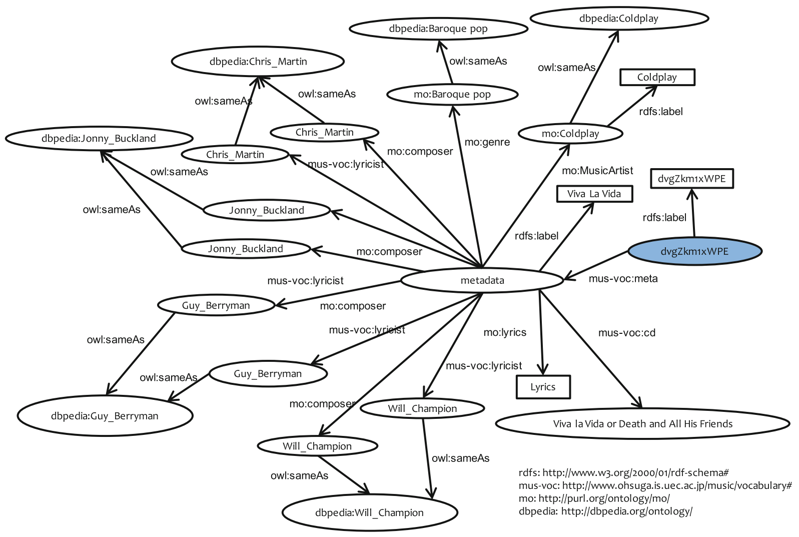


Fig. 2. Part of meta data in the Linked Data of music songs information

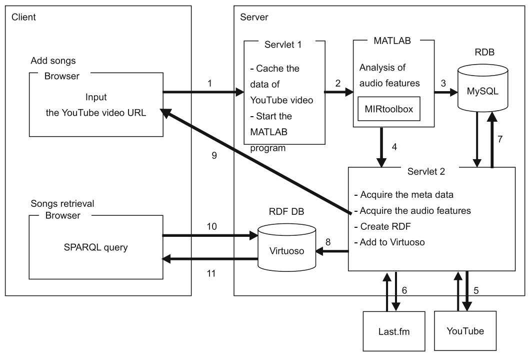


Fig. 3. Structure of the system

database. Then, the final phase is for the confirmation of newly added graphs. We describe the detail of each phase in the following sections.

4.2 Analyzing Audio Features

This phase (1–3 in Fig. 3) obtains and analyzes the audio features described in Sect. 3 from a specified video on the YouTube website. We used MIRtoolbox running on the MATLAB for analysis of the audio features. The MIRtoolbox includes several signal processing algorithms, which are commonly used in MIR.

First, a user inputs a URI of the YouTube video in the input form in a web browser. Then, the extract program downloads and caches the YouTube video data, and then it starts the MATLAB program that analyzes audio features of the video. Finally, the extracted audio features are temporally stored in MySQL database.

4.3 Searching Music Metadata

This phase (4–6 in Fig. 3) acquires music metadata, such as track title and artist name base on schema designed in Sect. 3. First, the video information is obtained from the YouTube. Then, we search the corresponding metadata, such as track title and artist name using Last.fm API for extending the related information.

4.4 Adding Linked Data

This phase (7 and 8 in Fig. 3) adds the audio features and the metadata to the music Linked Data.

First, the RDF create program gets the audio features from MySQL database, then it converts the audio features and the above metadata to RDF graphs based on the schema design. Finally, the RDF graphs are stored in Virtuoso.

4.5 Confirming Results

Figure 4 shows an example that a user added a new music "Doom and Gloom" by The Rolling Stones (the id: 1DWiB7ZuLvI) to the existing graph including "Applause" by Lady Gaga (the id: _bHhpufKRjs) in RDF DB. The two graphs are linked at a classification of the audio feature *Low energy* since they have the similar values in that feature. We visualize the RDF graph using the visualization tool the Visualization of RDF graph by ARC2[6]. Note that part of metadata and audio features are omitted for convenience.

[6] http://www.kanzaki.com/works/2009/pub/graph-draw.

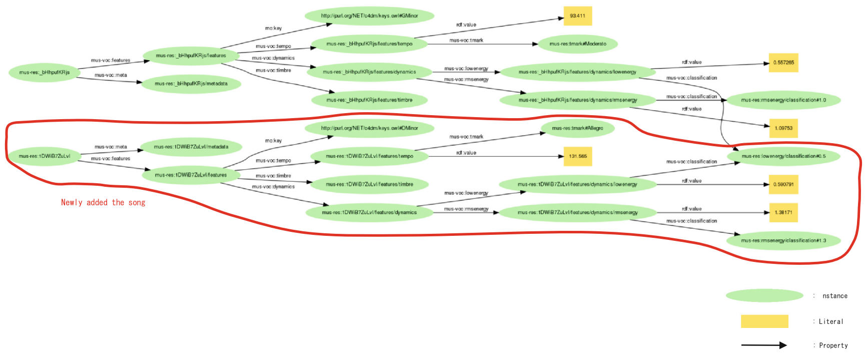

mus-res:_bHhpufKRjs
mus-voc:features
mus-voc:meta
mus-res:_bHhpufKRjs/features
mus-res:_bHhpufKRjs/metadata
mo:key
mus-voc:tempo
mus-voc:dynamics
mus-voc:timbre
http://purl.org/NET/c4dm/keys.owl#GMinor
mus-res:_bHhpufKRjs/features/tempo
mus-res:_bHhpufKRjs/features/dynamics
mus-res:_bHhpufKRjs/features/timbre
rdf:value
93.411
mus-voc:tmark
mus-res:tmark#Moderato
mus-voc:lowenergy
mus-voc:rmsenergy
mus-res:_bHhpufKRjs/features/dynamics/lowenergy
mus-res:_bHhpufKRjs/features/dynamics/rmsenergy
rdf:value
0.557265
mus-voc:classification
mus-voc:classification
mus-res:rmsenergy/classification#1.0
rdf:value
1.09753
mus-res:1DWiB7ZuLvl
mus-voc:meta
mus-voc:features
mus-res:1DWiB7ZuLvl/metadata
mus-res:1DWiB7ZuLvl/features
mo:key
mus-voc:tempo
mus-voc:timbre
mus-voc:dynamics
http://purl.org/NET/c4dm/keys.owl#DMinor
mus-res:1DWiB7ZuLvl/features/tempo
mus-res:1DWiB7ZuLvl/features/timbre
mus-res:1DWiB7ZuLvl/features/dynamics
mus-voc:tmark
mus-res:tmark#Allegro
rdf:value
131.565
mus-voc:lowenergy
mus-voc:rmsenergy
mus-res:1DWiB7ZuLvl/features/dynamics/lowenergy
mus-res:1DWiB7ZuLvl/features/dynamics/rmsenergy
mus-voc:classification
mus-res:lowenergy/classification#0.5
rdf:value
0.590791
rdf:value
1.38171
mus-voc:classification
mus-res:rmsenergy/classification#1.3
Newly added the song
: nstance
: Literal
: Property

Fig. 4. The graph added new data of music song

5 Example of Music Analysis

The current number of music registered in the platform is 1073 and the number of triples automatically extracted for representing the audio features and the metadata for that music is 20858. The platform is publicly available at http://www.ohsuga.is.uec.ac.jp/music/.

In this section, we show the results of some example queries on the platform, and how the music Linked Data can be used for MIR.In the SPARQL Query, we specified the audio feature *Brightness* of the song "Hello, Goodbye" by The Beatles, and search other songs, in which the value of the *Brightness* is similar to the specified song. As the result, we get 5 songs, in which the *Brightness* has the similar degree in Table 3.

```
[SPARQL Query]
PREFIX mus-voc:<http://www.ohsuga.is.uec.ac.jp/music/vocabulary#>
PREFIX mo:<http://purl.org/ontology/mo/>

SELECT ?artist_x ?title_x ?brightness_x
WHERE { ?metadata rdfs:label ?title .
        ?resource mus-voc:meta ?metadata .
        ?resource mus-voc:features ?features .
        ?features mus-voc:timbre ?timbre .
        ?timbre mus-voc:brightness ?brightnessc .
        ?brightnessc rdf:value ?brightness.
        ?brightnessc_x rdf:value ?brightness_x.
        ?timbre_x mus-voc:brightness ?brightnessc_x .
        ?features_x mus-voc:timbre ?timbre_x .
        ?resource_x mus-voc:features ?features_x .
        ?resource_x mus-voc:meta ?metadata_x .
        ?metadata_x rdfs:label ?title_x .
        ?metadata_x mo:MusicArtist ?MusicArtist_x .
        ?MusicArtist_x rdfs:label ?artist_x .
        FILTER regex(?title,"Hello Goodby") . }
ORDER BY (
  IF( ?brightness < ?brightness_x,
      ?brightness_x - ?brightness,
      ?brightness - ?brightness_x )
) LIMIT 5
```

Table 3. Result of submitting the SPARQL query

artist_x	title_x	brightness_x
The Beatles	Can't Buy Me Love	0.553889
Whitney Houston	Never Give Up	0.559786
Coldplay	Princess Of China Ft. Rihanna	0.560039
Lady Gaga	Judas	0.550279
The Beatles	Penny Lane	0.550221

6 Conclusion and Future Work

In this paper, we proposed a platform for providing audio features and the music metadata to MIR research. Thus, we designed the Linked Data schema for audio features and the metadata. Then, we built the platform for the music data collection and analysis and showed the current status of the dataset and a simple example of its use.

In future, we plan to provide more sophisticated examples and applications of music information analysis, which will encourage the expansion of the music Linked Data to music researchers and developers.

Acknowledgments. This work was supported by JSPS KAKENHI Grant Numbers 16K12411, 16K00419, 16K12533.

References

1. Kitahara, T., Nagano, H.: Advancing Information Sciences through Research on Music: 0. Foreword. IPSJ magazine. Joho Shori **57**(6), 504–505 (2016)
2. Wang, M., Kawamura, T., Sei, Y., Nakagawa, H., Tahara, Y., Ohsuga, A.: Context-aware Music Recommendation with Serendipity Using Semantic Relations. In: Proceedings of 3rd Joint International Semantic Technology Conference, pp. 17–32 (2013)
3. Osmalskyj, J., Foster, P., Dixon, S., Embrechts, J.J.: Combining features for cover song identification. In: Proceedings of the 16th International Society for Music Information Retrieval Conference, pp. 462–468 (2015)
4. Luo, Y.-J., Su, L., Yang, Y.-H., Chi, T.-S.: Real-time music tracking using multiple performances as a reference. In: Proceedings of the 16th International Society for Music Information Retrieval Conference, pp. 357–363 (2015)

Semantic Data Acquisition by Traversing Class–Class Relationships Over Linked Open Data

Atsuko Yamaguchi[1(✉)], Kouji Kozaki[2], Kai Lenz[3], Yasunori Yamamoto[1], Hiroshi Masuya[4,3], and Norio Kobayashi[3,4,5]

[1] Database Center for Life Science (DBCLS), Research Organization of Information and Systems, 178-4-4 Wakashiba, Kashiwa, Chiba 277-0871, Japan
{atsuko,yy}@dbcls.rois.ac.jp

[2] The Institute of Scientific and Industrial Research (ISIR), Osaka University, 8-1 Mihogaoka, Osaka, Ibaraki 567-0047, Japan
kozaki@ei.sanken.osaka-u.ac.jp

[3] Advanced Center for Computing and Communication (ACCC), RIKEN, 2-1 Hirosawa, Wako, Saitama 351-0198, Japan
{kai.lenz,norio.kobayashi}@riken.jp

[4] BioResource Center (BRC), RIKEN, 3-1-1, Koyadai, Tsukuba, Ibaraki 305-0074, Japan
hmasuya@brc.riken.jp

[5] RIKEN CLST-JEOL Collaboration Center, 6-7-3 Minatojima-minamimachi, Chuo-ku, Kobe 650-0047, Japan

Abstract. Linked Open Data (LOD), a powerful mechanism for linking different datasets published on the World Wide Web, is expected to increase the value of data through mashups of various datasets on the Web. One of the important requirements for LOD is to be able to find a path of resources connecting two given classes. Because each class contains many instances, inspecting all of the paths or combinations of the instances results in an explosive increase of computational complexity. To solve this problem, we have proposed an efficient method that obtains and prioritizes a comprehensive set of connections over resources by traversing class–class relationships of interest. To put our method into practice, we have been developing a tool for LOD exploration. In this paper, we introduce the technologies used in the tool, focusing especially on the development of a measure for predicting whether a path of class–class relationships has connected triples or not. Because paths without connected triples can be predicted and removed, using the prediction measure enables us to display more paths from which users can obtain data that interests them.

Keywords: Linked data · Class–class relationships · Data integration · Path finding

Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 136–151, 2016.
DOI: 10.1007/978-3-319-50112-3_11

1 Introduction

An important feature of Linked Data is to provide an efficient mechanism for linking different datasets published on the World Wide Web. The method enables users to mash up different data, and combinations of various datasets are expected to contribute to new innovations [1]. The Linked Open Data (LOD) Cloud (http://lod-cloud.net/) shows the evolution of LOD and many datasets published as Linked Data in a large variety of domains. In government, publishing government data as open data is promoted as an important policy in many countries. In science, open science is strongly promoted because open data publishing is essential for providing evidence for testing new theories. Particularly, in the life sciences, many Resource Description Framework (RDF) databases are published as LOD to provide foundations for data integration towards open science [2–5].

To use these databases published as LOD efficiently, users must be permitted to obtain data in a flexible manner according to their interests. An important case is to find paths of links between instances (resources) whose types are given two classes for integrative data analysis with semantics. For example, when biomedical researchers obtain molecular pathways through their experiments, they want to obtain a set of their IDs in the Reactome database[1], IDs of proteins related them, and their protein names. These paths can be obtained by retrieving chains of properties (links) that connect instances of classes such as Pathway, Protein and Protein Name. In other words, these paths can be obtained by traversing paths of class–class relationships over the LOD.

However, many users have difficulty specifying the appropriate path of class–class relations suitable for their search request, because each RDF dataset has different data schema that must be analyzed. Therefore, the technologies for exploring RDF datasets and displaying the summary of paths of class–class relationships are strongly required. As a method for exploring RDF datasets, we developed a metadata specification called SPARQL Builder Metadata (SBM) and a crawling tool to extract SBM through SPARQL endpoints. In addition, as a means of displaying paths, we developed some graph-based methods for expressing class–class relationships in what we call a class graph. In this paper, of these methods, we focus especially on the proposal of a measure for removing meaningless paths of class–class relationships through which a user cannot obtain any data.

The remainder of this paper is organized as follows: In Sect. 2, we describe existing work related to our approach. Section 3 introduces an application, SPARQL Builder, based on our proposed methods to make the explanations concrete. In addition, we explain two technologies, SBM and class graphs, that support our method, as mentioned above. Section 4 discusses a measure for avoiding meaningless paths and shows the performance of the proposed measure through computational experiments. Discussion and conclusion are provided in Sect. 5.

[1] https://www.ebi.ac.uk/rdf/services/reactome/.

2 Related Work

Some methods for obtaining paths from LOD have been proposed. To find a path or paths between two given resources, RelFinder [6] uses an algorithm that iteratively finds interim resources (sequentially related triples) by following RDF triples. Our approach differs from that of RelFinder in that ours accepts two classes of interest and finds paths between individuals that belong to each respective class. Because each class often contains thousands of individuals, and multiple paths between any two end individuals are highly probable, there are many cases with which RelFinder's algorithm cannot cope.

Another related application is Visor [7], which enables users to browse RDF datasets in the light of class–class relationships. For Visor, an exploratory search called Multi-Pivot, that extracts concepts and relationships from ontologies of interest to the user, was developed. The extracts are visualized and used for semantic searches among instances (data) associated with ontology terms. However, Visor provide no method for finding an end-to-end path through multiple resources.

As another approach for exploring RDF datasets using classes, a web based tool named Sparklis[2] [8] presents users with lists of classes in its target endpoints and allows them to construct queries through facet-based graphical user interfaces (GUIs). Faceted navigation, such as that in [9], is a very powerful approach to the browsing of datasets and finding specific resources. However, our approach focuses on obtaining comprehensive data having a class–class relationship that interests an user.

Related work regarding class graphs introduced in Subsect. 3.3, includes class association graphs [10]. Although for both class graphs and class association graphs nodes correspond to classes, edges of class association graphs correspond to the number of triples between classes, while edges of class graphs correspond to properties and their statistical values.

3 Data Acquisition Based on Class–Class Relationships

To discuss the technologies that are required to put our approach, which enables a user to understand RDF datasets by traversing class–class relationships, into practice, we first introduce an application called *SPARQL Builder*. The SPARQL Builder system generates a SPARQL query based on a path of class–class relationships over LOD. Then, we briefly explain two important technologies, *SPARQL Builder Metadata (SBM)* and *class graphs*, to realize the system.

3.1 SPARQL Builder

We have been developing a practical LOD search tool called SPARQL Builder for users who are not familiar with the SPARQL language to generate SPARQL

[2] http://www.irisa.fr/LIS/ferre/sparklis/osparklis.html.

queries without knowledge of SPARQL and RDF data schema [11]. The system is based on our proposed method, with which users can obtain their required data flexibly by traversing a path of class–class relationships over LOD.

The most important issue for the application is how to find candidate paths of class–class relationships for a user. In other words, how to compute candidate paths more efficiently and accurately is key for the application. To compute paths efficiently, the system obtains class–class relationships from LOD in advance, stores it as metadata using SBM, and constructs a labeled multigraph named a class graph as mentioned in Subsects. 3.2 and 3.3. In addition, we will discuss in Sect. 4 how to remove unnecessary paths to display paths more accurately.

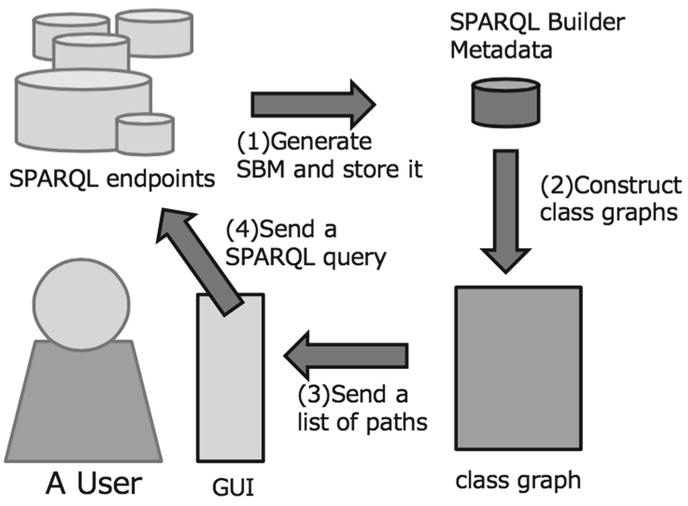


Fig. 1. An overview of the SPARQL builder system.

An overview of the SPARQL Builder system's architecture is shown in Fig. 1. (1) SPARQL Builder manages SBM generated by a crawler to access SPARQL endpoints of LOD in advance. (2) When a user accesses the SPARQL Builder system via a web browser as a GUI and selects two classes from a list of classes shown in the GUI initially, the class graph for the RDF dataset including the selected two classes is constructed using SBM, and possible paths between the selected two classes are computed and sent to the GUI. (3) Then, the SPARQL Builder GUI displays the list of paths in the user's web browser. If the user selects one path from the list, the system generates the SPARQL query corresponding to the path. (4) A user can obtain data by throwing the SPARQL query from the GUI.

Therefore, using the SPARQL Builder GUI, a user can explore RDF datasets of interest by specifying classes and a path. SPARQL Builder supports 38 SPARQL endpoints as of March 2016. Subsect. 3.2 relates to step (1) in Fig. 1. Subsect. 3.3 relats to step (2).

3.2 SBM

SBM is a summary of RDF datasets provided via a SPARQL endpoint. As described above, SBM is designed for describing a summary of class–class relationships in RDF datasets provided by SPARQL endpoints to enable SPARQL Builder to obtains necessary information quickly for computing paths of a sequentially connected class–class relationships.

SBM is defined as an extension of the VoID[3] and SPARQL 1.1 service description[4] with our original vocabulary of name space `sbm:`. For a SPARQL endpoint, SBM consists of summaries of the datasets provided as named graphs in the endpoint. For an individual RDF dataset, the summary for the dataset typed by `void:Dataset` includes a list of classes using the property `void:classPartition`, a list of properties using the property `void:propertyPartition`. For each class, the class URI with human-readable labels using `rdfs:label` and the number of instances using `void:entities` are described. Note that the summation of `void:entities` for all the classes is not the number of instances for the dataset because some instances might be typed to two or more classes and some instances might not be typed to any class.

For each property, in addition to statistical values related to the property such as the number of triples (`void:triples`), the numbers of distinct subjects (`void:distinctSubjects`) and objects (`void:distinctObjects`), and class–class relationships that are distinct pairs of subject classes, object classes/datatypes, are described using `sbm:classRelation` propety. For each class–class relationship, the subject class (`sbm:subjectClass`) and object class/datatype (`sbm:objectClass` or `sbm:objectDatatype`) are the class of subject instances and class/datatype of object instances/literals in triples associated with the concerned property. In addition, statistical values for a class–class relationship are described just as those for a property. For more information, see the web page[5] of the SBM specification.

To obtain SBM from RDF datasets, we implemented a crawler that generates SBM data by throwing lightweight but numerous SPARQL queries to the concerned SPARQL endpoints. The reason we used lightweight queries was to avoid time out response for a query. Because we found through our preliminary study that the size of the result for each query is not larger than the maximum size of result for a SPARQL endpoint, we focused on the response time for each query more than on the intermediate sizes of results by the queries to obtain information required for SBM. We have already crawled 38 endpoints including life-science SPARQL endpoints provided at EBI RDF Platform (https://www.ebi.ac.uk/rdf/platform) [2], Bio2RDF [3] Release 3 (http://download.bio2rdf.org/release/3/release.html) and Database Center for Life Science (http://dbcls.rois.ac.jp/en/services) as of April 2016. The crawled metadata in SBM are available through the web[6].

[3] https://www.w3.org/TR/void/.
[4] https://www.w3.org/TR/sparql11-service-description/.
[5] http://www.sparqlbuilder.org/doc/sbm_2015sep/.
[6] http://www.sparqlbuilder.org/sbm/.

Because SPARQL Builder discovers sequentially connected triples associated with a path specified by a user, the comprehensiveness of domain-range information for properties and class declarations for instances are important metadata. As described in [12], many datasets miss domain-range information for properties. In addition, a class is sometimes not typed as a class. Therefore, to make available as many RDF datasets as possible, the crawler extracts not only declared classes explicitly typed by `rdfs:Class` and domain–range relationships using the properties `rdfs:domain` and `rdfs:range`, but also classes expressed implicitly such as a subject or an object for the property `rdfs:subClassOf` and implicit class–class relationships found by triples with subjects and objects typed by some classes. By expanding classes and class–class relationships using such inferences, over 99.9% of instances of the 38 databases crawled are associated with classes.

Although SBM is designed for SPARQL Builder, statistical values gathered by the crawler can be applied to more general cases, such as evaluation of a SPARQL endpoint and RDF datasets in the endpoints. For example, a qualitative categorization with levels from 1 to 3 of SPARQL endpoints based on SBM is introduced in [11]: Level 1 corresponds to datasets that have domain-range declarations for all pf the properties and class declarations for all of the resources. Level 3 corresponds to datasets fpr which none of the resources is typed by any class. The other datasets with neither Level 1 or 3 are Level 2. Datasets with Level 2 can be further evaluated by the ratio of typed resources and the ratio of properties with domain-range declarations.

3.3 Class Graphs

To compute paths between two classes efficiently, we used a specialized graph whose nodes and edges correspond to the classes and the class–class relationships, respectively. We call such a graph a *class graph*.

Formally, a class graph is defined as follows: Given an RDF dataset R, we denote by C the set of all classes in R. A class graph $G_R = (V, E, c, p)$ of R is a directed labeled multigraph defined as follows: V is a $|C|$-sized set of nodes and c is a one-to-one mapping from V to a set of URLs of C. E is a multiset of directed edges between the nodes of V, and p maps E to a set of URLs of predicates in R. To construct E and p from R, we add to E a directed edge e_{pred} from node n_d to n_r, where $c(n_d) = class_d$ and $c(n_r) = class_r$, and define $p(e_{pred}) = pred$ if $pred$ satisfies either of the following two conditions: (1) both the triples "$pred$ rdfs:domain $class_d$" and "$pred$ rdfs:range $class_r$" exist in R for some classes $class_d$ and $class_r$; (2) there exist three triples "sub $pred$ ob", "sub rdf:type $class_d$", "ob rdf:type $class_r$" in R, where sub and ob are resources and $class_d$ and $class_r$ are classes.

A class graph can be constructed from SBM efficiently, because SBM include a list of all the classes and a list of all the class–class relationships. V corresponds to a set of objects of the property `void:classPartition`. We can define c by referring to objects of the property `void:class`. Each object of the property `sbm:classRelation` corresponds to edge e in E. From the subject of the property `sbm:classRelation`, we can find the property URI for the class–class relationship. Because inferred classes and class–class relationships are included in SBM as explained in Subsect. 3.2, nodes include inferred classes and edges include inferred class–class relationships as the definition of edges includes condition (2).

Given a class graph G_R, we define a *class path* p from a start class *start* to an end class *end* as a sequence $(n_0, e_1, n_1, e_2, \ldots, n_k)$, where the nodes n_i and edges e_i of G_R satisfy the following conditions: (1) $c(n_0) = start$, $c(n_k) = end$, (2) $c(n_i) \neq end$ for any $i \neq k$, and (3) e_i is a directed edge from n_{i-1} to n_i or from n_i to n_{i-1}. An edge e_i directed from n_{i-1} to n_i or from n_i to n_{i-1} is called *forward* or *reverse* directed, respectively. The length of a class path $(n_0, e_1, \ldots, n_k)$ is defined as k. To compute possible class paths between two classes in a practical time, the maximum length of class paths is given in advance and is currently set as 4.

A class path corresponds to a multi-step class–class relationship to obtain the instances of an end class from the instances of a start class by relating a sequence of predicates $p(e_i)$. By searching the possible class paths from the start class to the end class, we can obtain candidates of connections between data allowing a user to select a class path according to the interests of the user. Class paths between two classes can be found in a practically short time using an algorithm provided in [13].

4 Removal of Empty Paths

4.1 Measure to Remove Empty Paths

Although many paths can be computed efficiently using the technologies introduced in Sect. 3, we found through our preliminary investigation that some class paths have no sequence of instances obtained by traversing triples along the class paths. For example, as in Fig. 2, for a class path $(n_0, e_1, n_1, e_2, n_2)$ with forward edges e_1 and e_2, there might be no two triples of $(r_1\ p(e_1)\ r_2)$ and $(r_2\ p(e_1)\ r_3)$ such that r_1, r_2, and r_3 are instances of classes $c(n_0)$, $c(n_1)$, and $c(n_2)$, respectively. We call such a path an *empty path.* A user cannot obtain any data using an empty path because there are no connected triples from a start class to an end class. For example, if a user selects an empty path from a list of paths displayed in the SPARQL Builder GUI at step (3), the generated SPARQL query has no result at step (4). Therefore, it is important to present a method for removing as many such empty paths as possible from candidate paths.

An exact method for deciding whether a class path is an empty path involves using a SPARQL query with `ASK` corresponding to the path and deciding it is an empty path if the result is "false". Concretely, for a class path $(n_1, e_1, n_2, e_2, \ldots, n_m)$, using the following SPARQL query, we can decide whether a class path is an empty path.

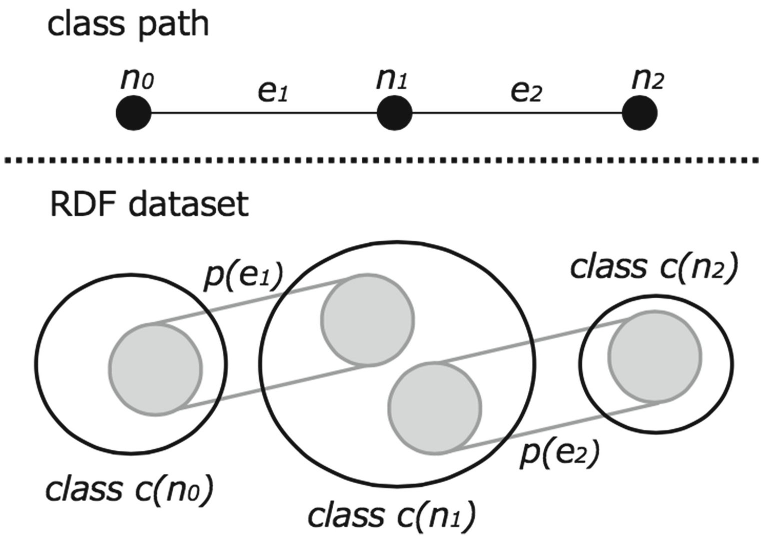


Fig. 2. Example of an empty path. Although two nodes n_0 and n_2 are connected in the class graph, there is no sequence of triples connecting instances in class $c(n_0)$ to those in class $n(n_2)$.

```
ASK {
    $?r_1\ p(e_1)\ ?r_2$. (or $?r_2\ p(e_1)\ ?r_1$. if $e_1$ is backward)
    $?r_2\ p(e_2)\ ?r_3$. (or $?r_3\ p(e_2)\ ?r_2$. if $e_2$ is backward)
    ...
    $?r_{m-1}\ p(e_{m-1})\ ?r_m$. (or $?r_m\ p(e_{m-1})\ ?r_m$. if $e_{m-1}$ is backward)
    $?r_1$ rdf:type $c(n_1)$.
    $?r_2$ rdf:type $c(n_2)$.
    ...
    $?r_m$ rdf:type $c(n_m)$
    }
```

If a SPARQL endpoint always returned the result for a query quickly, our system could extract only non-empty class paths from a set of class paths to show them to a user. However, using current triple stores, an `ASK` query sometimes consumes too much time. In addition, because the number of class paths is sometimes very large, the total time for computing a set of non-empty paths from a set of class paths tends to be very long. Furthermore, obtaining results for `ASK` in advance as we did for SBM is intractable through the Internet from SPARQL endpoints because results for all the paths with length four for all the combinations of classes are required. Therefore, it is not realistic in our approach at the moment to use `ASK` query to decide whether a class path is empty. In fact, although we had tried to use `ASK` query for SPARQL Builder to remove empty paths, it had often not worked even for relatively small RDF datasets. Therefore, we now introduce a measure using statistical values in SBM to predict whether a class path is an empty path.

The basic idea of the prediction measure is as follows: For each internal node n_i in a path, focusing on the probability of the overlap between two sets of instances in $c(n_i)$, one set is a set of objects for triples corresponding to edge (e_i), and the other set is a set of subjects for triples corresponding to edge (e_{i+1}). For example, because the class path shown in Fig. 2 has only one internal node n_1, the path is an empty path if and only if a set of objects for e_1 and a set of subjects for e_2 do not have overlapping parts. Therefore, we design our measure to be the probability that there exists an overlapping part for such an internal node in path. Then, we can predict a class path to be an empty path if the measure is small.

We here describe a measure for predicting empty paths. To explain the measure, we first introduce some notations. For an edge $e = (n_1, n_2)$ of a class graph, $T(e)$ is a set of triples (s, p, o) such that s is an instance of $c(n_1)$, $p = p(e)$, and o is an instance of $c(n_2)$. We define $S(e) = \{s|(s, p, o) \in T(e)\}$ and $O(e) = \{o|(s, p, o) \in T(e)\}$. Figure 3 shows a set $T(e)$ of triple and sets $S(e)$ and $O(e)$ of instances in an RDF dataset for an edge e of a class graph.

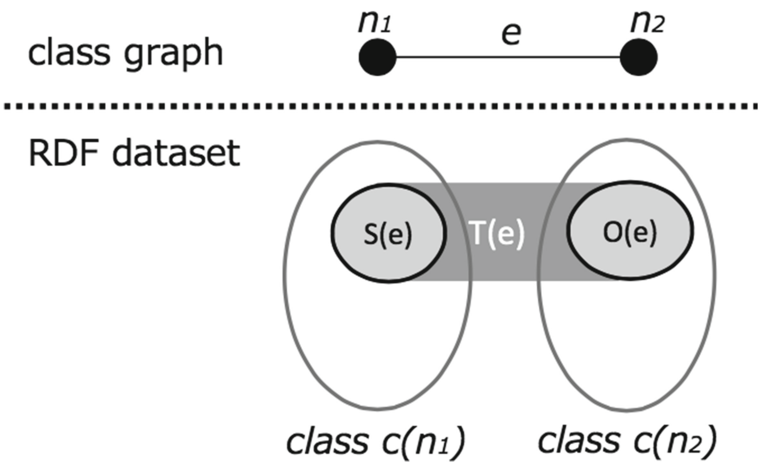


Fig. 3. Sets $T(e)$, $S(e)$, and $O(e)$ in an RDF dataset.

For a node n_i of a class graph, we denote by $|n_i|$ the number of instances in the class $c(n_i)$. Note that $|T(e)|$, $|S(e)|$, and $|O(e)|$ are described using the properties `void:triples`, `void:distinctSubject`, and `void:distinctObject`, respectively, from instances of `sbm:ClassRelation` located in `sbm:PropertyPartition` of SBM. Similarly, we can obtain $|n_i|$ for each n_i from SBM using `void:entities` from instances of `sbm:ClassPartition`.

For a class path $p = (n_0, e_1, n_1, \ldots, n_k)$, $I(p)$ is a set of sequences $(t_1, \ldots, t_k)$ of triples satisfying the following two conditions: (1) For $t_i = (s_i, p_i, o_i)$, if e_i is forward in p, then s_i is an instance of $c(n_{i-1})$, $p(e) = p_i$, and o_i is an instance of $c(n_i)$. If e_i is backward, then s_i is an instance of $c(n_i)$, $p(e) = p_i$, and o_i is an instance of $c(n_{i-1})$. (2) for two triples $t_{i-1} = (s_{i-1}, p_{i-1}, o_{i-1})$ and $t_i = (s_i, p_i, o_i)$, $r_{i-1} = r_i$ where $r_{i-1} = o_{i-1}$ if e_{i-1} is forward, $r_{i-1} = s_{i-1}$ if e_{i-1} is backward, $r_i = s_i$ if e_i is forward, and $r_i = o_i$ if e_i is backward. Note that for an empty path p, $I(p)$ is empty.

We now introduce our proposed measure Pr based on statistical values, such as $|S(e)|$ and $|O(e)|$ for edge e, included by SBM. To simplify the definition, we assume here that all the edges in $p = (n_0, e_1, n_1, \ldots, n_k)$ are forward. We then define $Pr(p) = (1 - (1 - |S(e_2)|/|n_1|)^{|O(e_1)|}) \times (1 - (1 - |S(e_3)|/|n_2|)^{|O(e_2)|}) \times \cdots \times (1 - (1 - |S(e_k)|/|n_{k-1}|)^{|O(e_{k-1})|})$. Note that if there is a backward edge e_i in p, $O(e_i)$ and $S(e_i)$ in the corresponding terms for e_i of the definition above must be changed to $S(e_i)$ and $O(e_i)$, respectively.

$Pr(p)$ is a very rough approximation of the probability of the existence of a triple sequence in $I(p)$, assuming that each instance in $S(e)$ and $O(e)$ is uniformly distributed independently of the other in instances of classes $c(n_1)$ and $c(n_2)$, respectively, for each edge $e = (n_1, n_2)$ appearing in p. Even though in reality, occurrences in instances between $c(n_1)$ and $c(n_2)$ might not be independent, because they are connected by triples, this assumption simplifies the definition of Pr. Under the assumption, if there are two edges e_i and e_{i+1} sharing one node n_i, for each instance in $O(e_i)$, the probability of a connecting instance in $S(e_{i+1})$ is $|S(e_{i+1})|/|n_i|$. Therefore, for all the instances in $O(e_i)$, because the probability of having no instances in $S(e_{i+1})$ is $(1 - |S(e_{i+1})|/|n_i|)^{|O(e_i)|}$, the probability of having at least one instance is $1 - (1 - |S(e_{i+1})|/|n_i|)^{|O(e_i)|}$. Because there are $k - 1$ such nodes in a path $p = (n_0, e_1, n_1, \ldots, n_k)$, Pr can be computed by multiplying the probabilities for the nodes.

Because $|O(e)|$, $|S(e)|$, and $|n_i|$ can be found in SBM as objects for the properties `void:distinctObjects`, `void:distinctSubjects`, void:entities, respectively, a class graph can hold these values with edges and nodes in a class graph when being constructed from SBM. Therefore, $Pr(p)$ can be computed in $O(A(n)B(m)k)$, where $A(n)$ is the look-up time of a node from n nodes, and $B(m)$ is the look-up time of an edge from m edges in a class graph. Because the length k of a path is always limited to four or five for the SPARQL Builder system, and $A(n)$ and $B(m)$ can be almost constant for n and m, respectively, through implementation using an efficient look-up structure such as a hash table, the computation time of $Pr(p)$ can be almost constant. Therefore, even for datasets with very large n and m, $Pr(p)$ can be computed in a short time using SBM.

4.2 Evaluation of the Measure Through Computational Experiment

To investigate the prediction performances of using the measure Pr, which is easily computable from SBM, we checked whether each class path p was an empty path, and we compared the results with the value of $Pr(p)$. We selected

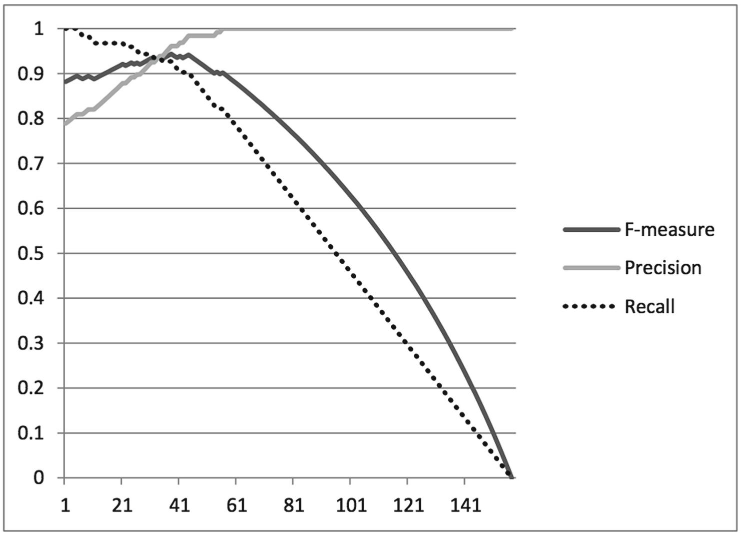


Fig. 4. The distribution of the F-measures with recalls and precisions for an obtained set of class paths by removing n paths of the smaller $Pr(p)$ for the Allie dataset.

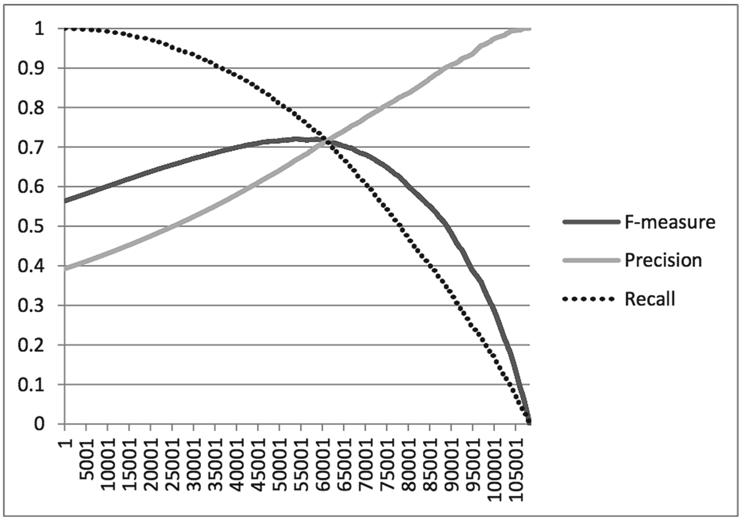


Fig. 5. The distribution of the F-measures with recalls and precisions for an obtained set of class paths by removing n paths of the smaller $Pr(p)$ for the Reactome dataset.

three datasets from Allie [14], Reactome from EBI RDF Platform [2], Affymetrix from Bio2RDF [3] as of April 2016 with various ratios, approximately 0.8, 0.4, and 0.25, respectively, of non-empty paths of all the paths between two classes.

We first processed the datasets to produce metadata written in SBM. We then set a maximum path length of four for the experiment. Then, we computed

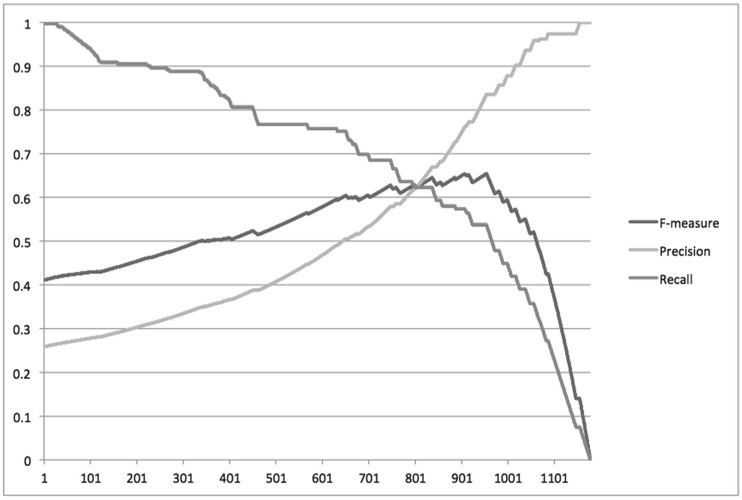


Fig. 6. The distribution of the F-measures with recalls and precisions for an obtained set of class paths by removing n paths of the smaller $Pr(p)$ for the Affymetrix dataset.

all of the class paths between every pair of classes using the algorithm proposed in [13] from the metadata previously computed for each dataset. For each class path p, we then computed $Pr(p)$ using the metadata.

To prepare exact positive and negative sets, i.e., to decide whether each class path $(n_1, e_1, n_2, e_2, \ldots, n_m)$ is an empty path, we downloaded the three datasets and checked all of the class paths with a maximum length of four to determine whether they were empty paths. Concretely, we uploaded these three datasets locally to the open source version of OpenLink Virtuoso 7.20.3215 on a CentOS 5.11 machine with 24 cores of Intel Xeon 2.53 GHz and accessed the triple store with Virtuoso Driver 4.0 for JDBC and Virtuoso Jena Driver 2.6.2 using the `ASK` SPARQL query described as an exact method in Subsect. 4.1.

To evaluate Pr as the measure for predicting an empty path, we aligned all of the paths in ascending order with respect to Pr, and computed the precision, recall, and F-measure of the performance when n paths of the smaller Pr are removed from the original set of paths. Figure 4 shows the distribution of the precisions, recalls, and F-measures for the Allie dataset. The x-axis corresponds to the number of removed paths. The precision for the number n of removed paths is the ratio of non-empty paths to all of the paths after the n paths from the smallest Pr are removed. Similarly, the recall for n is the ratio of non-empty paths to all of the non-empty paths after the n paths are removed. The best F-measure was 0.944 with precision 0.961, and recall 0.927 was found at the threshold 0.389 of Pr.

Similarly, Figs. 5 and 6 show the distribution of the precisions, racalls, and F-measures for the Reactome and Affymetrix datasets. For Reactome, the best F-measure was 0.721 with precision 0.667 and the recall 0.783 found at the

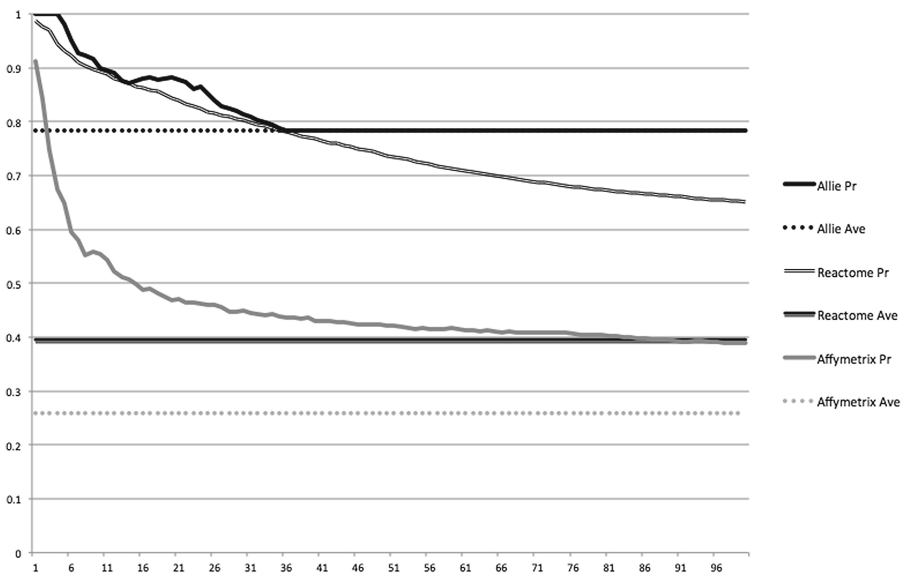


Fig. 7. Performance using Pr. Allie Pr, Reactome Pr and Affymetrix Pr show accuracy rates of the top n $(1 \leq n \leq 100)$ paths using Pr, i.e., average ratio of non-empty paths among top n path for every pair of classes, for Allie, Reactome and Affymetrix datasets, respectively. Allie Ave, Reactome Ave and Affymetrix Ave show the average non-empty path ratio for all the paths for Allie, Reactome and Affymetrix, respectively.

threshold 0.000252 of Pr. For Affymetrix, the best F-measure was 0.654 with precision 0.836 and recall 0.538 found at the threshold 0.121 of Pr.

From a practical point of view, because systems such as SPARQL Builder can show only a small number, for example, ten, of class paths at one time as a result of limited screen size, it is important that the top n paths include as many non-empty paths as possible. Therefore, we plotted the average ratio of non-empty paths in the top n paths, sorted using the measure Pr. In other words, we plotted the precision for non-empty paths after all of the paths were removed except for the top n paths. Figure 7 shows a comparison between the average ratios of non-empty paths in the top n paths sorted by Pr and the average ratio of non-empty paths to all paths for Allie, Reactome and Affymetrix datasets. The x-axis corresponds to the number n of paths from the largest Pr. Because the average non-empty path ratios for all of the paths are 0.783, 0.393, and 0.25 for Allie, Reactome and Affymetrix datasets, respectively, we expect it to be easiest for Allie and most difficult for Affymetrix to obtain non-empty paths from all the class paths among the three datasets. For all three dataset, we can see from Fig. 7 that the average ratios of non-empty paths for the top n paths are always more than those for all the class paths. For example, the average ratios of non-empty paths for the top ten paths are 0.9, 0.893, and 0.555 for Allie, Reactome and Affymetrix datasets, respectively.

Especially for the Affymetrix dataset, although the average non-empty path ratio for all of the paths is approximately 0.25, the top five paths have the average accuracy 0.85. The average non-empty path ratio for all of the paths corresponds to the expected accuracy of selecting n paths randomly. The top n paths from one to 100 always obtain better accuracy than the averages for the three dataset. Therefore, we can claim that Pr is useful in removing empty paths from a set of class paths.

5 Discussion and Conclusion

In this study, we discussed a novel LOD exploring methodology and its application to enabling practical LOD data discovery from a SPARQL endpoint. We achieved data acquisition using class paths of interest to a user based on class–class relationships, with paths of traversing class–class relations being computed and used for two classes given by a user. Therefore, our approach requires some technologies including SBM specification, crawling for obtaining SBM, an efficient algorithm for finding class paths, and a measure for predicting empty paths proposed in this paper. These technologies are strongly related to one another in realizing our approach. For example, because an efficient algorithm finds many paths, including empty paths, we need a prediction measure for empty paths. Computing the prediction measure in nearly constant time requires metadata written in SBM designed to provide multiple statistical values in machine-readable form. Obtaining SBM requires an efficient crawler for SBM. Because we showed that the performance of our proposed measure Pr is practically useful, our approach is now ready to be realized as a system such as SPARQL Builder for analysis of LOD.

As for the measure for predicting empty paths based on an approximation of the existence probability using statistical values in SBM, the proposed measure were successfully demonstrated by our evaluation experiment, using three life-science datasets. For the experiment shown in Fig. 7, even though the ratio of the top n paths for Allie and Reactome datasets are very similar, they are very different from that for the Affymetrix dataset. The ratio of the top n paths for the Affymetrix dataset decreases quickly as n increases. This might have resulted from the lowness of the average ratio of non-empty class paths to all class paths for the Affymetrix dataset. However, we believe there to be another reason for the difference of the performance because the performances for Allie and Reactome are similar, even though the average ratios of non-empty class paths to all class paths are more different between Allie and Reactome datasets than between Reactome and Affymetrix. By analyzing datasets in regard to many features, the measure might be able to improve its performance. In addition, although this paper proposed only a measure for predicting empty paths, a measure for estimating the size of data obtained by a path, corresponding to the size of $I(p)$, might also be useful because data are strongly connected through p if $I(p)$ is large.

Our approach has been evaluated using life-science datasets. Although we believe that our approach does not depend on domain-specific techniques, we would like to confirm it by using datasets from another domain. In addition, our future work will include an improvement of a federated search across large-scale SPARQL endpoints by applying our methodology to reduce search space and enhance the effectiveness of data traversing for searches that have not been realized. By introducing a federated search system, SPARQL Builder can support queries using class paths between classes from different datasets.

Acknowledgments. This work was supported by JSPS KAKENHI Grant Number 25280081, 24120002 and the National Bioscience Database Center (NBDC) of the Japan Science and Technology Agency (JST).

References

1. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology, 1st edn. 1: 1, 1–136. Morgan & Claypool (2011)
2. Jupp, S., Malone, J., Bolleman, J., Brandizi, M., Davies, M., Garcia, L., Gaulton, A., Gehant, S., Laibe, C., Redaschi, N., Wimalaratne, S.M., Martin, M., Le Novére, N., Parkinson, H., Birney, E., Jenkinson, A.M.: The EBI RDF platform: linked open data for the life sciences. Bioinformatics **30**(9), 1338–1339 (2014)
3. Belleau, F., Nolin, M.A., Tourigny, N., Rigault, P., Morissette, J.: Bio2RDF: towards a mashup to build bioinformatics knowledge systems. J. Biomed. Inf. **41**(5), 706–716 (2008)
4. Redaschi, N., UniProt Consortium: UniProt in RDF: tackling data integration and distributed annotation with the semantic web. Nat. Precedings (2009). doi:10.1038/npre.2009.3193.1
5. Fu, G., Batchelor, C., Dumontier, M., Hastings, J., Willighagen, E., Bolton, E.: PubChemRDF: towards the semantic annotation of PubChem compound and substance databases. J. Cheminformatics **7**(34) (2015). doi:10.1186/s13321-015-0084-4
6. Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., Stegemann, T.: RelFinder: revealing relationships in RDF knowledge bases. In: Chua, T.-S., Kompatsiaris, Y., Mérialdo, B., Haas, W., Thallinger, G., Bailer, W. (eds.) SAMT 2009. LNCS, vol. 5887, pp. 182–187. Springer, Heidelberg (2009). doi:10.1007/978-3-642-10543-2_21
7. Popov, I.O., Schraefel, M.C., Hall, W., Shadbolt, N.: Connecting the dots: a multi-pivot approach to data exploration. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011. LNCS, vol. 7031, pp. 553–568. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25073-6_35
8. Ferré, S.: Sparklis: a SPARQL endpoint explorer for expressive question answering. In: Proceedings of the ISWC 2014 Posters & Demonstrations Track, CEUR Workshop Proceedings 1272, Riva del Garda, Italy (2014)
9. Oren, E., Delbru, R., Decker, S.: Extending faceted navigation for RDF data. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 559–572. Springer, Heidelberg (2006). doi:10.1007/11926078_40
10. Qu, Y., Ge, W., Cheng, G., Gao, Z.: Class association structure derived from linked objects. In: Proceedings of the Web Science Conference (WebSci 2009: Society On-Line), Athens, Greece (2009)

11. Yamaguchi, A., Kozaki, K., Lenz, K., Wu, H., Kobayashi, N.: An intelligent SPARQL query builder for exploration of various life-science databases. In: The 3rd International Workshop on Intelligent Exploration of Semantic Data (IESD 2014), CEUR Workshop Proceedings 1279, Riva del Garda, Italy (2014)
12. Villalon, P., Suárez-Figueroa, M.C., Gómez-Pérez, A.: A double classification of common pitfalls in ontologies. In: Workshop on Ontology Quality (OntoQual 2010), Lisbon, Portugal (2010)
13. Yamaguchi, A., Kozaki, K., Lenz, K., Wu, H., Yamamoto, Y., Kobayashi, N.: Efficiently finding paths between classes to build a SPARQL query for life-science databases. In: Qi, G., Kozaki, K., Pan, J.Z., Yu, S. (eds.) JIST 2015. LNCS, vol. 9544, pp. 321–330. Springer, Heidelberg (2016). doi:10.1007/978-3-319-31676-5_24
14. Yamamoto, Y., Yamaguchi, A., Bono, H., Takagi, T.: Allie: a database and a search service of abbreviations and long forms. Database (2011). doi:10.1093/database/bar013

Estimation of Spatio-Temporal Missing Data for Expanding Urban LOD

Shusaku Egami[1(✉)], Takahiro Kawamura[1,2], and Akihiko Ohsuga[1]

[1] Graduate School of Informatics and Engineering,
The University of Electro-Communications, Tokyo, Japan
egami.shusaku@ohsuga.lab.uec.ac.jp, takahiro.kawamura@jst.go.jp,
ohsuga@uec.ac.jp

[2] Japan Science and Technology Agency, Tokyo, Japan

Abstract. The illegal parking of bicycles has been an urban problem in Tokyo and other urban areas. We have sustainably built a Linked Open Data (LOD) relating to the illegal parking of bicycles (IPBLOD) to support the problem solving by raising social awareness. Also, we have estimated and complemented the temporally missing data to enrich the IPBLOD, which consisted of intermittent social-sensor data. However, there are also spatial missing data where a bicycle might be illegally parked, and it is necessary to estimate those data in order to expand the areas. Thus, we propose and evaluate a method for estimating spatially missing data. Specifically, we find stagnation points using computational fluid dynamics (CFD), and we filter the stagnation points based on popularity stakes that are calculated using Linked Data. As a result, a significant difference in between the baseline and our approach was represented using the chi-square test.

Keywords: Linked open data · Urban problem · Illegally parked bicycles

1 Introduction

The illegal parking of bicycles have been an urban problem in Tokyo and other urban areas since the number of bicycles owned in Japan is large. An increased awareness of health problems [1] and energy conservation [2] led to a 2.6-fold increase in bicycle ownership in Japan from 1970 to 2013. In addition to the insufficient availability of bicycle parking spaces and public transportations such as city buses, the inadequate public knowledge on bicycle parking laws, has contributed to this problem. Illegally parked bicycles obstruct vehicles, cause road accidents, encourage theft, and disfigure streets.

In order to address this problem, we believe it would be useful to publish the distribution of illegally parked bicycles as Linked Open Data (LOD). For example, it would serve to visualize illegally parked bicycles, suggest locations for optimal bicycle parking spaces, assist with the removal of illegally parked bicycles, and assist with the urban design. Thus, we built the illegally parked bicycle

LOD (IPBLOD) based on social data after designing LOD schema [3]. Furthermore, we estimated and complemented temporal missing data using Bayesian networks, since the temporal missing data of the social sensor data is inevitable. Therefore, IPBLOD became a temporally enriched LOD, and it became possible to suggest the efficient timing of removal of illegally parked bicycles by the city, through the visualization of time-series changes in the distribution of illegally parked bicycles.

However, there are not only temporal missing data, but also spatial missing data where bicycles might be illegally parked. It is necessary to complement the spatial missing data in order to apply IPBLOD to various urban areas. However, it is not satisfied merely by social sensors when collecting observation points of illegally parked bicycles.

In this paper, we propose the method for geographically expanding LOD by estimating spatial missing data. We thought that observation points of illegally parked bicycles have spatial or geographic features common such as road width and building density. Thus, we considered the flow of people as the fluid, and we estimated spatial missing data in such a way as to find stagnation points of the fluid. Specifically, we first simulated airflow in urban area using computational fluid dynamics (CFD) and found stagnation points using stagnation point patterns defined by us. Next, we collected POI information around each of the stagnation points and calculate popularity stakes of the POIs using DBpedia Japanese. Then, we filtered stagnation points if their sum of the popularity stakes of POIs is less than the threshold. We considered the filtered stagnation points as estimated data and added the data to IPBLOD separately from real data. Therefore, our contributions are the geographical expansion of IPBLOD, development of an approach for estimating spatial missing data using CFD and Linked Data, and evaluation of this approach. We aim to collect new accurate data related to estimated observation points from social sensors, by raising social awareness through the visualization of estimated observation points.

The remainder of this paper is organized as follows. In Sect. 2, correcting data, building IPBLOD, estimating temporally missing data, and visualization of IPBLD are presented. In Sect. 3, the approach for estimating spatial missing data using CFD and DBpedia Japanese is described. Also, we evaluate our results. In Sect. 4, related works of data collection and urban LOD are described. Finally, Sect. 5 concludes this paper with future works.

2 Illegally Parked Bicycle LOD

We have sustainably built IPBLOD and applied them to Tokyo and other several urban areas. Managing urban problem data *joining* multiple tables in (distributed) RDBs is troublesome from the aspect of data interoperability and maintenance, since the urban problem is closely related to multiple domains, such as government data, legal data, and social data as we already incorporated POIs and weather data in this application, and also those have different schemata. Thus, Linked Data is a suitable format as the data infrastructure of not only illegally parked bicycles, but also urban problems in general, since Linked Data can have advantages of flexible linkability and schema.

We divided our approach of sustainable LOD construction into the following five steps. Steps (2) to (5) are executed repeatedly as more input data become available.

1. Designing LOD schema
2. Collecting observation data and factor data
3. Building the LOD based on schema
4. Using Bayesian networks to estimate the missing number of illegally parked bicycles at each location
5. Visualizing illegally parked bicycles using LOD.

2.1 Building IPBLOD

First, we designed IPBLOD based on the result of extracting domain requirements from Web articles related to illegally parked bicycles. Figure 1 shows an overview of IPBLOD schema.

Next, we collected tweets containing location information, pictures, hashtags, and the number of illegally parked bicycles. Furthermore, we collected information on POI using Google Places API[1] and Foursquare API[2]. Also, we obtained bicycle parking information from websites of municipalities and in

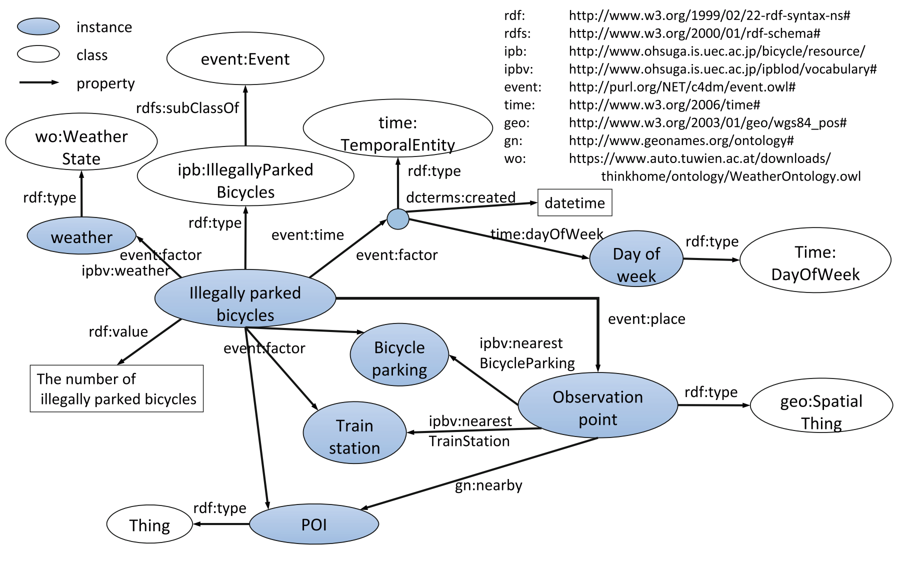

instance
class
property
event:Event
rdfs:subClassOf
wo:Weather State
time: TemporalEntity
ipb:IllegallyParked Bicycles
rdf:type
dcterms:created
datetime
weather
event:factor
ipbv:weather
event:time
event:factor
time:dayOfWeek
Day of week
Time: DayOfWeek
Illegally parked bicycles
event:place
rdf:value
event:factor
Bicycle parking
ipbv:nearest BicycleParking
The number of illegally parked bicycles
Observation point
geo:Spatial Thing
Train station
ipbv:nearest TrainStation
gn:nearby
Thing
POI
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs: http://www.w3.org/2000/01/rdf-schema#
ipb: http://www.ohsuga.is.uec.ac.jp/bicycle/resource/
ipbv: http://www.ohsuga.is.uec.ac.jp/ipblod/vocabulary#
event: http://purl.org/NET/c4dm/event.owl#
time: http://www.w3.org/2006/time#
geo: http://www.w3.org/2003/01/geo/wgs84_pos#
gn: http://www.geonames.org/ontology#
wo: https://www.auto.tuwien.ac.at/downloads/thinkhome/ontology/WeatherOntology.owl

Fig. 1. LOD schema containing instances

[1] https://developers.google.com/places/?hl=en.
[2] https://developer.foursquare.com/.

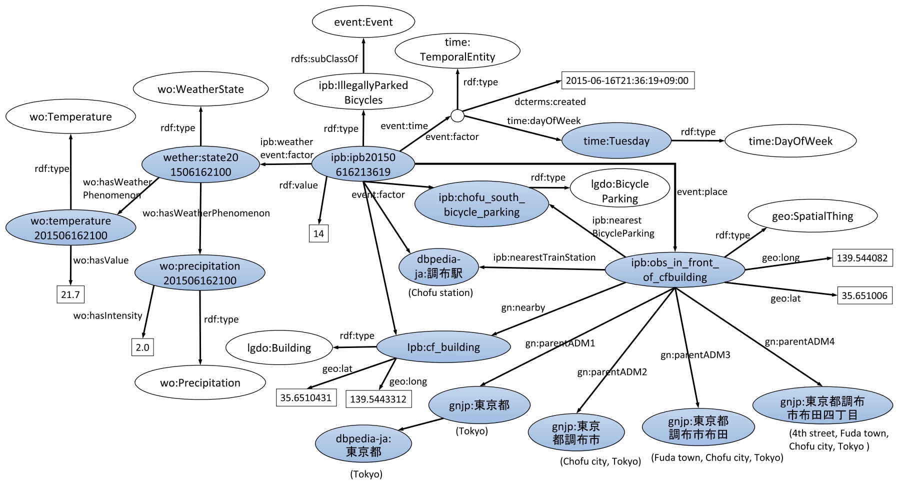


Fig. 2. Part of the integrated LOD

cooperation with the Bureau of General Affair of Tokyo[3]. The Bureau of General Affairs of Tokyo publishes Open Data on bicycle parking areas as CSV. The data contain names, latitudes, longitudes, addresses, capacities, and business hours. More information was collected from municipalities, for example, monthly parking fees and daily parking fees. Also, we retrieved weather information from the website of the Japanese Meteorological Agency (JMA)[4].

The collected data on illegally parked bicycles are converted to LOD based on the designed schema. First, the server program collects tweets containing the particular hash-tags, the location information, and the number of illegally parked bicycles in real time. The number of illegally parked bicycles is extracted from the text of tweets using regular expressions. Next, the server program checks whether there is an existing observation point within a radius of less than 30 m by querying our endpoint[5] using the SPARQL query. If there is no observation point on the IPBLOD, the point is added as a new observation point. In order to add new observation points, the nearest POI information is obtained using Google Places API and Foursquare API. The new observation point is generated based on the name of the nearest POI. It is possible to obtain the types of the POI from Google Places API and Foursquare API. We map the types of POI to classes in LinkedGeoData [14]. Thus, the POI is an instance of classes in LinkedGeoData. However, some POIs do not have a recognized types. Therefore, their types are decided by a keyword search with the name of the POI. Figure 2 shows part of the IPBLOD. The LOD are stored in Virtuoso[6] Open-Source Edition. Also, the RDF data set is published with CC-BY license on our website[7].

[3] http://www.soumu.metro.tokyo.jp/30english/index-en.htm.
[4] http://www.jma.go.jp/jma/indexe.html.
[5] http://www.ohsuga.is.uec.ac.jp/sparql.
[6] http://virtuoso.openlinksw.com/.
[7] http://www.ohsuga.is.uec.ac.jp/bicycle/dataset.html.

2.2 Complementing and Estimating Temporally Missing Values

Since we relied on the public to observe illegally parked bicycles, we did not have round-the-clock data for every place, and thus, missing data in the IPBLOD were inevitable. However, the number of the illegally parked bicycles should be influenced by several factors, thus we try to estimate these missing data using Bayesian networks. If the data is expanded in density through the estimation, it will serve, for example, as the suitable location of bicycle parking spaces, the decision on variable prices of the parking fee and efficient timing of removal of illegally parked bicycles by the city, and part of the references for future urban design.

Thus, we estimated the number of illegally parked bicycles, at observation points, where the number data are missing. We used the Bayesian network tool Weka[8] to estimate the unknown numbers of illegally parked bicycles. Suppose the aggregates of each factor are given by Location, Day={sun, mon,...,sat}, Hour={0,1,...,23}, Precipitation={0,1,...}, Temperature={...,-1,0, 1,...}, DailyFee ={0,1,...}, MonthlyFee={0,1,...}, Density={0,1,...}, Commuters={0,1,...}, POIs={0,1}, and Number (of illegally parked bicycles)={1,...,4}, then the observation data are stored as an aggregate O of vectors $o \in Location \times Day \times Hour \times Precipitation \times Temperature \times DailyFee \times MonthlyFee \times Density \times Commuters \times POIs \times Number$. In fact, POIs are divided to 46 elements, for example, restaurant, bar, supermarket, hospital, and school. The number of illegally parked bicycles is classified into four classes by Jenks natural breaks [13], which are often used in Geographic Information Systems (GISs). The range is 0 to 6, 7 to 17, 18 to 3, and 36 to 100. Therefore, the input data is a set O that consists of vectors with 56 elements, and the amount of the input data is 897. We used HillClimber as a search algorithm, and also used Markov blanket classifier. The maximum number of parent nodes was seven. The average estimation accuracy of ten times 10-fold cross validation became 70.9%, after random sampling with a 90 % rate.

Then, we examined the observation data in each observation point from the first observation date to the last observation date. If there are no data at 9 am or 9 pm, we estimated and complemented the number of illegally parked bicycles. Then, we added the estimated number and its probability to IPBLOD as follows.

```
@prefix ipb: <http://www.ohsuga.is.uec.ac.jp/ipblod/
    vocabulary#>
@prefix bicycle: <http://www.ohsuga.is.uec.ac.jp/bicycle/
    resource/>
bicycle:ipb_{observation point}_{datetime}
     ipb:estimatedValue [ rdf:value  "0-7" ;
         ipb:probability  "0.772"^^xsd:double ] .
```

[8] http://www.cs.waikato.ac.nz/ml/weka/.

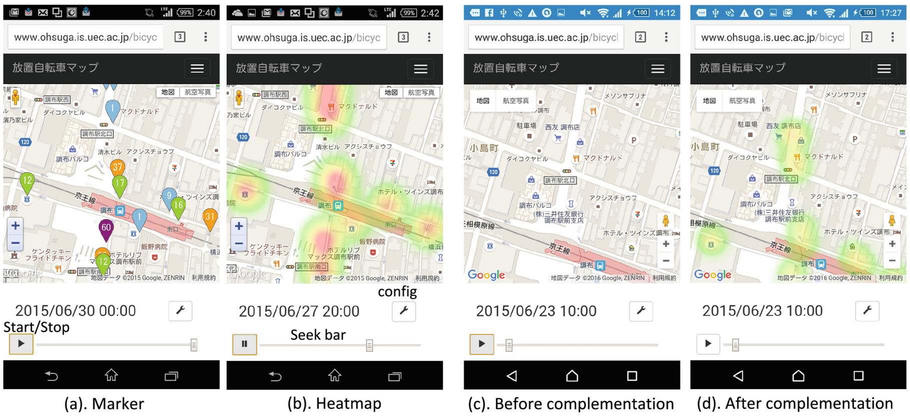


Fig. 3. Screenshots of the visualization application

2.3 Visualization of IPBLOD

Data visualization enables people to intuitively understand data contents. Thus, it can possibly raise the awareness of an issue among local residents. Furthermore, it is expected that we shall collect more urban data. In this section our visualization application of the IPBLOD is described.

As an example of the use of these data, we developed a Web application that visualizes illegally parked bicycles. The application can display time-series changes in the distribution of illegally parked bicycles on a map. Also, the application has a responsive design, so it is possible to use it on various devices such as PCs, smartphones, and tablets. When the start and end times are selected, and the play button is pressed, the time series changes of the distribution of the illegally parked bicycles are displayed. Figure 3(a) and (b) show screenshots of an Android smartphone, on which the Web application is displaying such an animation near Chofu Station in Tokyo using a heatmap and a marker UI.

The IPBLOD contain not only the data collected from Twitter, but also the data estimated by Bayesian networks. Therefore, time-series changes in the distribution of illegally parked bicycles become smoother than before estimating the missing values. Figure 3(c) and (d) show the comparison between the before and after complementation. The time-series changes after complementation are successive, whereas the time-series changes before complementation are intermittent.

3 Estimating Spatial Missing Data

There are spatio-temporal missing data in IPBLOD since the data is collected from social sensors. We estimated and complemented temporal missing data with 70.9% accuracy using Bayesian networks. However, spatial missing data

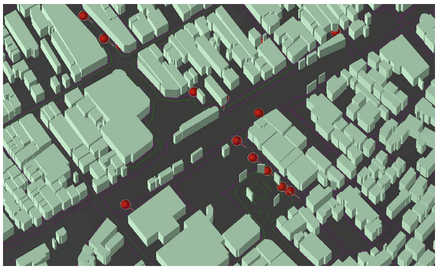


Fig. 4. The 3D map around of Chofu Sta.

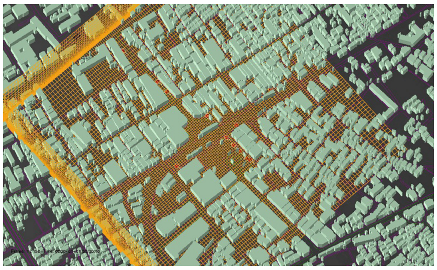


Fig. 5. The view of grid cells

(unobserved points where bicycles might be illegal parked) have not been complemented. In this study, we geographically expand IPBLOD by estimating and complementing the spatial missing data. We consider the flow of people to the fluid, and we find stagnation points of areas around train stations by airflow simulation using 3D maps and CFD. In Japan, since there are generally illegally parked bicycles around train stations, we selected those as our simulation areas. Then, we validate correlation of stagnation points and observation points of illegally parked bicycles. Furthermore, we filter stagnation points using DBpedia Japanese, and we regard these filtered points as new observation points. Therefore, in this section, we describe the hybrid approach using CFD and Linked Data for estimating spatial missing data.

3.1 Finding Stagnation Points Using CFD

There are wind tunnel test and CFD as the methods of airflow simulation. CFD is the method which observes the movements of fluid using computer simulation. A wind tunnel test requires expensive and large equipment, but CFD is easy to experiment with in different environments when using a computer. However, CFD can not produce exact copies of fluid movements, since CFD uses an approximate solution.

We first obtained maps of building from Geospatial Information Authority of Japan[9]. This data consists of 2D polygons. We converted the 2D maps to 3D maps using ArcGIS for Desktop[10]. Since we could not obtain information on the height of the buildings, we set the height of all buildings to 30 m. Figure 4 shows the 3D map around of Chofu Station in Tokyo. Red markers are observation points of illegally parked bicycles. We obtained observation points in a CSV format from the SPARQL endpoint of the IPBLOD, and then we imported the CSV to the 3D map. Next, we simulated the airflow around the station using Airflow Analyst[11], which is a simulation software run on ArcGIS. Figure 5 shows the grid cells which is set as the analysis range. We set the analysis range to

[9] http://www.gsi.go.jp/ENGLISH/index.html.
[10] http://www.esri.com/software/arcgis/arcgis-for-desktop.
[11] http://www.airflowanalyst.com/en/index.php.

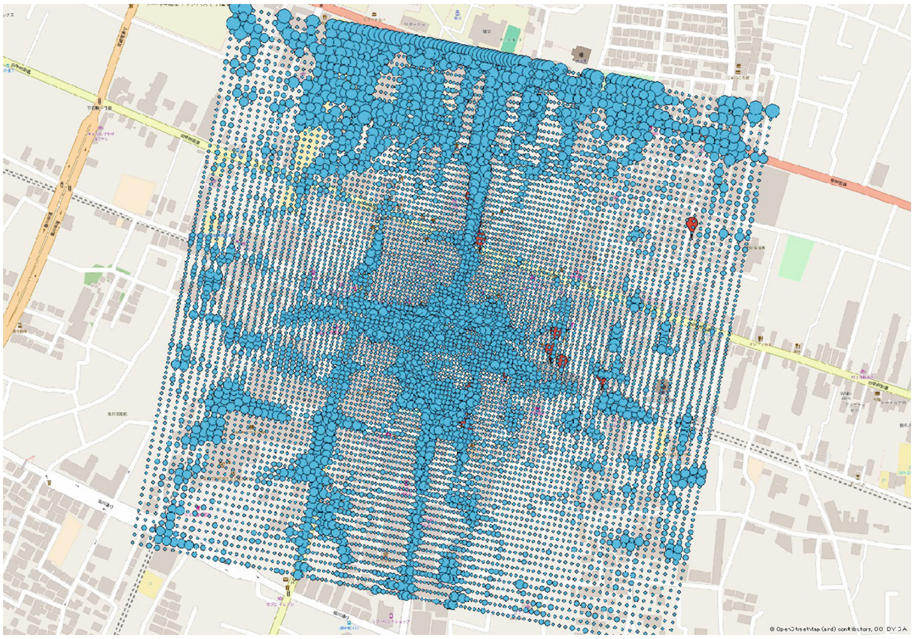


Fig. 6. The result of the airflow simulation

include all observation points around of Chofu Station. In Fig. 5, we selected 700×700 square meteres around Chofu Station as the analysis range. Also, the node spacing is 5 m, and the number of nodes is 10,000. The wind direction is set as being parallel to the road of the train station. Since it is considered that people come to the station from four directions in the case of Chofu Station, we simulated the airflow while changing the wind directions such as 11-degree, 109-degree, 190-degree, and 288-degree.

Figure 6 shows the visualization of the average wind velocity based on the results of the simulation, when the wind direction is 11-degree and the wind speed is 5 m/s. The size of the circle means the average wind velocity. We found stagnation points based on this numerical data. A stagnation point is a point where the velocity of the fluid is zero in the flow field. We tried to find stagnation points using patterns in Fig. 7. A black node is a node with average wind velocity $x > 0.1$. The white node is the node which is $x = 0$. The grey node is the node that $0 < x \leq 0.1$. In general, a stagnation point is a white node under these conditions. However, white nodes became buildings in our experiment. Therefore, we defined grey nodes as stagnation points. Table 1 shows the total accuracy of the findings of stagnation points around Chofu Station, Fuchu Station, and Shinjuku Station using the all patterns. The precision is the ratio of stagnation points within a 20-m radius from an observation point. The recall is the ratio of observation points that have stagnation points within a 20-m radius. As the result, the F-measure when we used the pattern (j) became the highest. Hence,

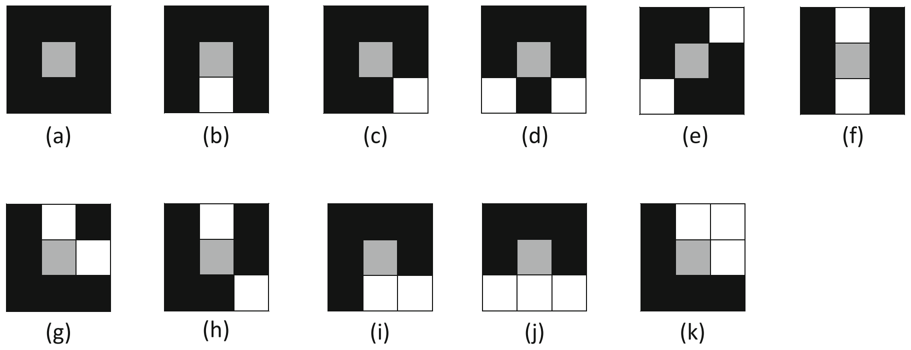


Fig. 7. Patterns of stagnation points

we use pattern (j) to find stagnation points in this study. Figure 8 shows the results of the findings stagnation points around of Chofu Station. This is the merged result of the simulation results of the four directions.

Table 1. Results of the findings stagnation points when we used patterns in Fig. 7

Pattern	Precision	Recall	F-measure
(a)	0.102	0.286	0.150
(b), (c)	0.0833	0.0357	0.0500
(d), (e), (f), (g), (h), (i)	0.000	0.000	0.000
(j)	0.0913	0.429	0.151
(k)	0.0746	0.107	0.0880

3.2 Filtering Stagnation Points Using DBpedia Japanese

We found the stagnation points, but, there were many noise points as can be seen in Fig. 8. We assumed that bicycles tend to be parked illegally at stagnation points having nearby POIs, whose popularity stakes are high. Therefore, we calculated the popularity stakes of the POIs around of the stagnation points and then filtered the stagnation points using the popularity stakes.

We first obtained the POIs information within a 20-m radius from the stagnation points using Google Places API. Then, we calculated the number of links from person resources to POIs on DBpedia Japanese. Also, we mapped the types of POIs to DBpedia Japanese resources. We considered the number of inbound links from person resources as the popularity stakes, and we obtained the number of links from instances of `foaf:Person` to types of POIs. Then, we calculated the sum of the popularity stakes of POIs, and we filtered stagnation points if the sum of the popularity stakes is less the threshold. We varied the threshold from

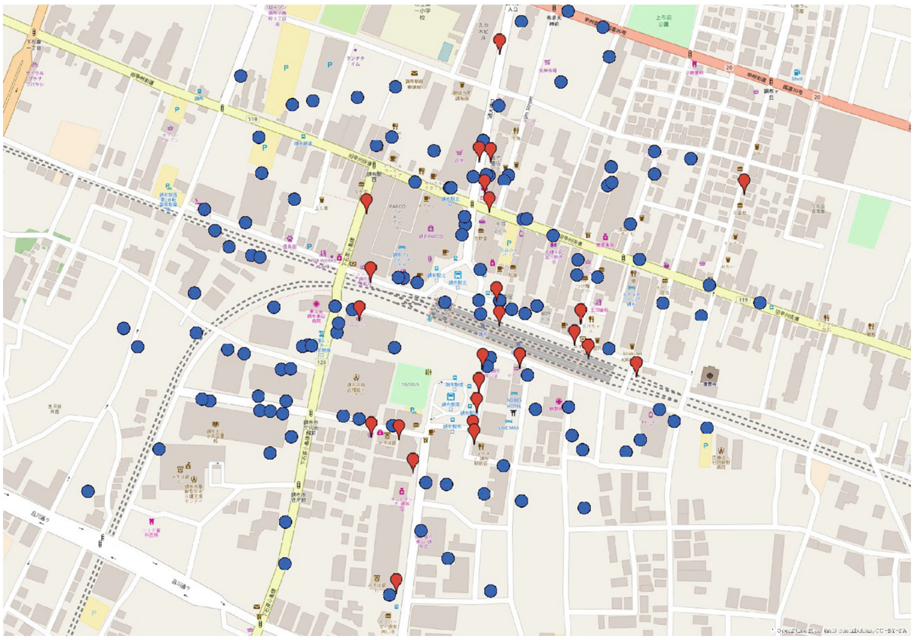


Fig. 8. The stagnation points around of Chofu station

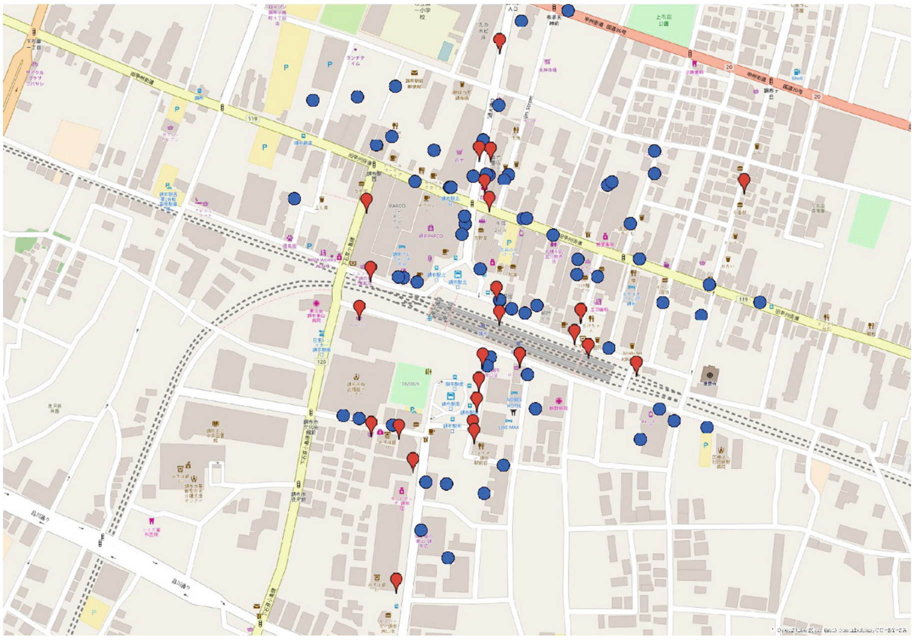


Fig. 9. The filtered stagnation points around of Chofu station

100 to 1,000, and we could achieve the best results when the threshold is 200. Hence, we set the threshold to 200. Figure 9 shows the results of the filtering.

Furthermore, we added estimated data to IPBLOD separately from real data as follows. The latitude and the longitude are obtained from ArcGIS. Address information is obtained from Yahoo! Reverse Geocoder API[12]. The POIs are also obtained from Google Places API.

```
@prefix ipb: <http://www.ohsuga.is.uec.ac.jp/ipblod/
   vocabulary#>
@prefix bicycle: <http://www.ohsuga.is.uec.ac.jp/bicycle/
   resource/>
@prefix geo:   <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix ogcgs: <http://www.opengis.net/ont/geosparql#> .
@prefix ngeo:  <http://geovocab.org/geometry#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix gn:    <http://www.geonames.org/ontology#> .
@prefix gnjp:    <http://geonames.jp/resource/> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
bicycle:estimated_obs_{timestamp}
    rdf:type    ipb:EstimatedObservationPoint ;
    geo:lat "latitude"^^xsd:double ;
    geo:long    "longitude"^^xsd:double ;
    gn:parentADM    gnjp:{Prefecture}
    gn:parentADM2   gnjp:{City, Prefecture} ;
    gn:parentADM3   gnjp:{Town, City, Prefecture} ;
    gn:parentADM4   gnjp:{Land lot, Town, City, Prefecture}
        ;
    ngeo:geometry   [ a ngeo:Geometry; ogcgs:asWKT  "POINT(
        latitude,longitude)"^^<http://www.openlinksw.com/
        schemas/virtrdf#Geometry> . ] ;
    gn:nearby   bicycle:{POI name} ;
    dcterms:created "datetime"^^xsd:dateTime .
```

3.3 Evaluation and Discussion

In this section, we describe the validation results whether there is a correlation of the data estimated from our approach and the observation points of illegally parked bicycles, and discuss the evaluation of the utility of our approach.

We carried out the experiments on Chofu Station, Fuchu Station, and Shinjuku Station which have multiple observation points of illegally parked bicycles. The total number of observation points was 56. We validated the utility of the proposed method by comparing the result of the baseline and the result of the proposed method. First, we compared the baseline and the stagnation point method as described in Sect. 3.1. Figure 10 shows the result of the baseline for the Chofu Station. The baseline estimates the spatial missing data at regular

[12] http://developer.yahoo.co.jp/webapi/map/openlocalplatform/v1/reversegeocoder.html.

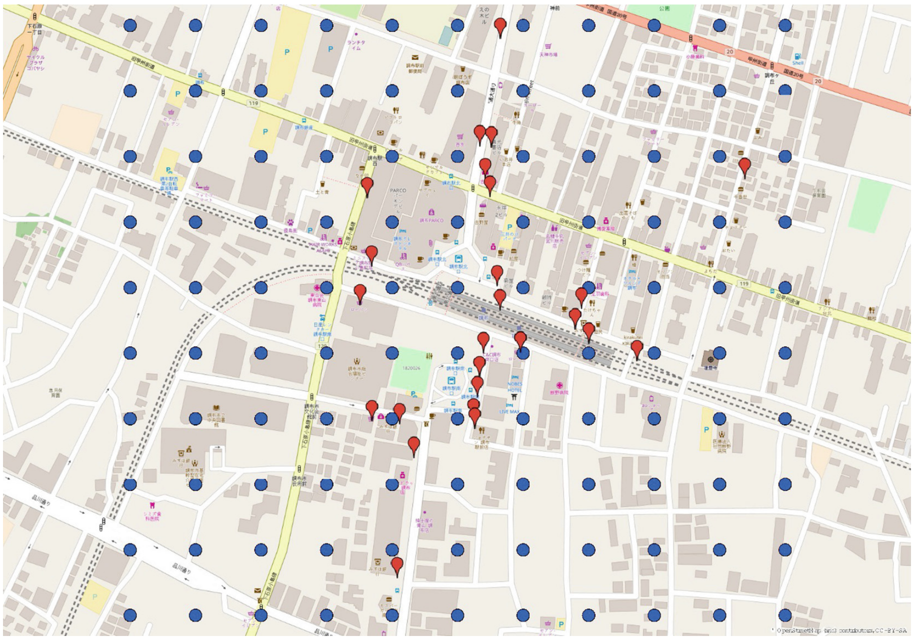


Fig. 10. The stagnation points of baseline

Table 2. Evaluation results of both baseline and stagnation point method

		Baseline	Stagnation point method
Chofu Sta.	Precision	0.0496	0.0726
	Recall	0.231	0.423
	F-measure	0.0816	0.152
Fuchu Sta.	Precision	0.125	0.188
	Recall	0.222	0.333
	F-measure	0.160	0.240
Shinjuku Sta.	Precision	0.0493	0.100
	Recall	0.190	0.476
	F-measure	0.0784	0.165
Total	Precision	0.0550	0.0913
	Recall	0.214	0.429
	F-measure	0.0876	0.151

Table 3. Evaluation results of both baseline and hybrid method

		Baseline	Hybrid method
Chofu Sta.	Precision	0.0469	0.121
	Recall	0.115	0.346
	F-measure	0.0667	0.180
Fuchu Sta.	Precision	0.125	0.250
	Recall	0.222	0.333
	F-measure	0.160	0.286
Shinjuku Sta.	Precision	0.0493	0.117
	Recall	0.190	0.476
	F-measure	0.0784	0.188
Total	Precision	0.0559	0.129
	Recall	0.161	0.393
	F-measure	0.0829	0.194

intervals, as many as the number of stagnation points. Table 2 shows the accuracy of both the baseline and the stagnation point method. As the result, the precision, the recall, and the F-measure of the stagnation point method became higher than the result of the baseline. Also, we validated the utility of the stagnation point method using the chi-square test. The null hypothesis is that there is no difference between the result (recall or precision) of the baseline and the result (recall or precision) of the stagnation point method, and we used a standard level of significance $p < 0.05$. As the result, the p-value of precision was 0.01591, and the p-value of recall was 2.244e-06. Hence, we found that there is a significant difference between the result of baseline and the result of the stagnation point method.

Next, we compared the baseline and the hybrid method (filtering stagnation points). Table 3 shows the accuracy of both the baseline and the hybrid method. As the result, the precision, the recall, and the F-measure of the hybrid method became higher than the result of the baseline. Also, as it is possible to see from Tables 2 and 3, the precision and the F-measure of the hybrid method became higher than the result of the stagnation point method. Therefore, there is the utility of the hybrid method using POIs and DBpedia Japanese. Also, we validated the utility of the hybrid method using the chi-square test. The null hypothesis is that there is no difference between the result of the baseline and the result of the hybrid method, and we used a standard level of significance $p < 0.05$. As the result, the p-value of precision was 7.393e-05, and the p-value of recall was 2.244e-06. Hence, we found that there is a significant difference between the result of the baseline and the result of the hybrid method. Although the accuracy is not high, the data of estimated points are considered to help to collect new data from social sensors.

The accuracy of the estimated data in this study was low for the following reasons. The number of observation points was less. There is a possibility that new observation points are found around the estimated points. Therefore, we should conduct a field survey in the future work; then we should evaluate our approach once again. Also, we could not exclude the stagnation points of the interior of the premises, since we could not obtain the data of building premises. The noise points were caused by this reason.

4 Related Work

In most cases, LOD sets have been built based on existing databases. However, there is little LOD available so far that describes urban problems. Thus, methods for collecting new data to build urban problem LOD are required. Data collection methods for building Open Data include crowdsourcing and gamification. A number of projects have employed these techniques. OpenStreetMap [5] is a project that creates an open map using crowdsourced data. Anyone can edit the map, and the data are published as Open Data. Similarly, FixMyStreet [6] is a platform for reporting regional problems such as road conditions and illegal dumping. Crowdsourcing to collect information in FixMyStreet has meant that regional problems are able to be solved more quickly than ever before. Zook et al. [7] reported a case, where crowdsourcing was used to link published satellite images with OpenStreetMap after the Haitian earthquake. A map of the relief efforts was created, and the data were published as Open Data. Celino et al. [9] proposed an approach for editing and adding Linked Data using a game with a purpose (GWAP) [8] and human computation. However, since the data relating to illegally parked bicycles are time-series data, it is difficult to collect data using these approaches. Therefore, we proposed a method to build IPBLOD while complementing the spatio-temporal missing data.

Also, there have been studies about building Linked Data for cities. Lopez et al. [10] proposed a platform that publishes sensor data as Linked Data. The platform collects streamed data from sensors and publishes Resource Description Framework (RDF) data in real time using IBM InfoSphere Stream and C-SPARQL [11]. The system is used in Dublinked2[13], which is a data portal of Dublin, Ireland, that publishes information about bus routes, delays, and congestion which is updated every 20 s. However, since embedding sensors are costly, this approach is not suitable for our study.

Furthermore, Bischof et al. [4] proposed a method for the collection, complementation, and republishing of data as Linked Data, as with our study. This method collects data from DBpedia [12], Urban Audit[14], United Nations Statistics Division (UNSD)[15], and U.S. Census[16] and then utilizes the similarity among such large Open Data sets on the Web. However, we could not find the

[13] http://www.dublinked.ie/.
[14] http://ec.europa.eu/eurostat/web/cities.
[15] http://unstats.un.org/unsd/default.htm.
[16] http://www.census.gov/.

corresponding data sets and thus could not apply the same approach to our study. Therefore, we estimated temporal missing data using Bayesian network, and we estimated spatial missing data using CFD and DBpedia Japanese.

In other areas, Bogárdi-Mészöly et al. [15] proposed a method for the detection of scenic leaves and blossoms viewing places. The proposed system collects images from Flickr[17], and then the system ranks scenic leaves and blossoms viewing places based on social features and image features. However, since we do have not enough amounts of the observation point's data and their images, this method is not suitable for our study. Furthermore, Hirota et al. [16] proposed a method for estimating missing metadata of images based on the image similarity, the photo-taking condition similarity, and the tag similarity. This method assumes that there is a sufficient amount of data for the estimation as well as Bogárdi-Mészöly et al.

5 Conclusion and Future Work

In this paper, we presented building IPBLOD while complementing temporal missing data, and we described geographically expansion of IPBLOD by estimating the spatial missing data. The mainly technical contribution is the proposal of a hybrid method using CFD and DBpedia Japanese for estimating the spatial missing data in LOD. Also, we evaluated our method using indicators such as precision, recall, and F-measure. Furthermore, we validated the utility of our method using the chi-square test. We expect that it will increase the social awareness of local residents regarding the illegally parked bicycle problem and encourage them to post more data over a wide area, through the visualization of estimated spatial data (new observation points).

In the future, we will estimate spatial missing data in more urban areas, and we will check true-false results to go to estimated points. Also, we will incorporate a new method such as machine learning to solve the problem that was described in Sect. 3.3. Furthermore, we will visualize estimated observation points and will design incentive for social sensors (workers of crowdsourcing), in order to collect more data related to illegally parked bicycles.

Acknowledgments. This work was supported by JSPS KAKENHI Grant Numbers 16K12411, 16K00419, 16K12533.

References

1. Nishi, N.: The 2nd Health Japan 21: goals and challenges. J. Fed. Am. Soc. Exp. Biol. 28(1), 632.19 (2014)
2. Ministry of Internal Affairs, Communications: Current bicycle usage and bicycle-related accident. http://www.soumu.go.jp/main_content/000354710.pdf. Accessed 10 September 2015 (Japanese)

[17] https://www.flickr.com/.

3. Egami, S., Kawamura, T., Ohsuga, A.: Building urban LOD for solving illegally parked bicycles in Tokyo. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (eds.) ISWC 2016. LNCS, vol. 9982, pp. 291–307. Springer, Heidelberg (2016). doi:10.1007/978-3-319-46547-0_28
4. Bischof, S., Martin, C., Polleres, A., Schneider, P.: Collecting, integrating, enriching and republishing open city data as linked data. In: Arenas, M., Corcho, O., Simperl, E., Strohmaier, M., d'Aquin, M., Srinivas, K., Groth, P., Dumontier, M., Heflin, J., Thirunarayan, K., Staab, S. (eds.) ISWC 2015. LNCS, vol. 9367, pp. 57–75. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25010-6_4
5. Haklay, M., Weber, P.: Openstreetmap: User-generated street maps. IEEE Pervasive Comput. **7**(4), 12–18 (2008)
6. King, S.F., Brown, P.: Fix my street or else: using the internet to voice local public service concerns. In: Proceedings of the 1st International Conference on Theory and Practice of Electronic Governance, pp. 72–80 (2007)
7. Zook, M., Graham, M., Shelton, T., Gorman, S.: Volunteered geographic information and crowdsourcing disaster relief: a case study of the Haitian earthquake. World Med. Health Policy **2**(2), 7–33 (2010)
8. von Ahn, L.: Games with a purpose. IEEE Comput. **39**(6), 92–94 (2006)
9. Celino, I., Cerizza, D., Contessa, S., Corubolo, M., Dell' Aglio, D., Valle, E.D., Fumeo, S., Piccinini, F.: Urbanopoly: collection and quality assesment of geo-spatial linked data via a human computation game. In: Proceedings of the 10th Semantic Web Challenge (2012)
10. Lopez, V., Kotoulas, S., Sbodio, M.L., Stephenson, M., Gkoulalas-Divanis, A., Aonghusa, P.M.: QuerioCity: A linked data platform for urban information management. In: Cudré-Mauroux, P., Heflin, J., Sirin, E., Tudorache, T., Euzenat, J., Hauswirth, M., Parreira, J.X., Hendler, J., Schreiber, G., Bernstein, A., Blomqvist, E. (eds.) ISWC 2012. LNCS, vol. 7650, pp. 148–163. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35173-0_10
11. Barbieri, D.F., Ceri, S.: C-SPARQL: SPARQL for continuous querying. In: Proceedings of the 18th International Conference on World Wide Web, pp. 1061–1062 (2012)
12. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A nucleus for a web of open data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76298-0_52
13. BlackJenks, G.F.: The data model concept in statistical mapping. Int. Yearb. Cartography **7**(1), 186–190 (1967)
14. Stadler, C., Lehmann, J., Höffner, K., Auer, S.: LinkedGeoData: A core for a web of spatialopen data. Semant. Web J. **3**(4), 333–354 (2012)
15. Bogárdi-Mészöly, A., Rövid, A., Yokoyama, S.: Detect scenic leaves and blossoms viewing places from flickr based on social and image features. In: Proceedings of the 5th IIAI International Congress on Advanced Applied Informatics, pp. 1162–1167 (2016)
16. Hirota, M., Fukuta, N., Yokoyama, S., Ishikawa, H.: A robust clustering method for missing metadata in image search results. J. Inf. Process. **20**(3), 537–547 (2012)

RDF and Query

ASPG: Generating OLAP Queries for SPARQL Benchmarking

Xin Wang[1(✉)], Steffen Staab[1,2], and Thanassis Tiropanis[1]

[1] Web and Internet Science Group, University of Southampton, Southampton, UK
xwang@soton.ac.uk

[2] Institute for Web Science and Technology, University of Koblenz-Landau, Mainz, Germany

Abstract. The increasing use of data analytics on Linked Data leads to the requirement for SPARQL engines to efficiently execute Online Analytical Processing (OLAP) queries. While SPARQL 1.1 provides basic constructs, further development on optimising OLAP queries lacks benchmarks that mimic the data distributions found in Link Data. Existing work on OLAP benchmarking for SPARQL has usually adopted queries and data from relational databases, which may not well represent Linked Data. We propose an approach that maps typical OLAP operations to SPARQL and a tool named ASPG to automatically generate OLAP queries from real-world Linked Data. We evaluate ASPG by constructing a benchmark called DBOBfrom the online DBpedia endpoint, and use DBOB to measure the performance of the Virtuoso engine.

Keywords: OLAP · Linked data · Benchmarking · Query generation · SPARQL · DBpedia

1 Introduction

Linked Data principles foster the provisioning and integration of a large amount of heterogeneous distributed datasets [2]. SPARQL 1.1 [11] has introduced aggregations that enable users to do basic analytics. Though limited, SPARQL 1.1 is expressive enough to implement Online Analytical Processing (OLAP) which is an approach to analysing and reporting multidimensional statistics from different perspectives and levels of granularity [3,5].

OLAP contains a rich set of combinations of analytical operations which generate a high workload on SPARQL engines that target the support of analytics queries. In fact, the scalability of SPARQL engines to execute OLAP queries is still rather limited owing further development and optimization. Such optimization and comparison of best developments critically depend on benchmarks that can measure the performance of SPARQL engines on analytic tasks from various perspectives. Several OLAP benchmarks for SPARQL have been proposed. For example Kämpgen and Harth [12] convert queries and data from the Star Schema Benchmark (SSB) to SPARQL and Linked Data using the RDF

© Springer International Publishing AG 2016
Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 171–185, 2016.
DOI: 10.1007/978-3-319-50112-3_13

Data Cube Vocabulary [6]. Since SSB is based on a relational database scenario, its data do not necessarily resemble common Linked Data structures. Another example, the BowlognaBench [8], uses data and queries based on the Bowlogna Ontology [7]. Similar to SSB for SPARQL, BowlognaBench covers a specific scenario which may not represent the heterogeneity and structure of Linked Data.

Görlitz et al. [9] propose a SPARQL query generator called SPLODGE to release benchmarks from pre-defined queries. Following the same direction we present a tool called Analytical SPARQL Generator (ASPG) that generates OLAP queries in SPARQL which can be used to construct benchmarks. ASPG takes an RDF graph as input and selects triples by semi-random walk. Selected triples are parametrised to generate basic graph patterns (BGPs) which are then extended with aggregations that resemble OLAP operations. Queries produced by ASPG are guaranteed to return results from the given RDF graph since they are parametrised from triples in the RDF graph. We construct an analytical benchmark based on DBpedia, referred to as DBpedia OLAP Benchmark (DBOB), using ASPG generated queries. We evaluate Virtuoso[1] using DBOB and present the results.

The remaining sections of this paper are organised as follows: technical details of ASPG are described in Sect. 2; queries and dataset of DBOB are presented in Sect. 3; experiment settings and evaluation result of DBOB are given in Sect. 4, and conclusions are given in Sect. 6. Due to page limit, a complete list of DBOB queries is given in http://xgfd.github.io/ASPG/.

2 Generating OLAP Queries from Linked Data

In this section we discuss the correspondence between typical OLAP operations and SPARQL components, and provide details of generating SPARQL queries from arbitrary RDF graphs that resemble typical OLAP operations.

SPARQL queries consist of basic graph patterns (BGPs) which can be viewed as graphs with variable nodes. When evaluating a BGP against a RDF graph, results are returned if and only if the BGP matches a sub-graph of the given RDF. Consequently, given an arbitrary RDF graph, we can construct BGPs that are guaranteed to return results by parametrising sub-graphs in the RDF. By controlling the structure of sub-graphs we can obtain BGPs that consist of chains or star-shaped triple patterns of arbitrary lengths. We simulate typical OLAP operations by summarising along properties (using GROUP BY) with randomly selected aggregate operations (e.g. SUM, COUNT, AVG etc.). In particular we discuss the challenges to generate queries from RDF graphs that are too large to fit in a single store and describe RDF summarising and sampling techniques to resolve those issues.

2.1 Background of OLAP Operations

OLAP queries operate on a multidimensional data model that is referred to as an OLAP cube. Each data point in the cube is associated with two types of

[1] http://virtuoso.openlinksw.com/.

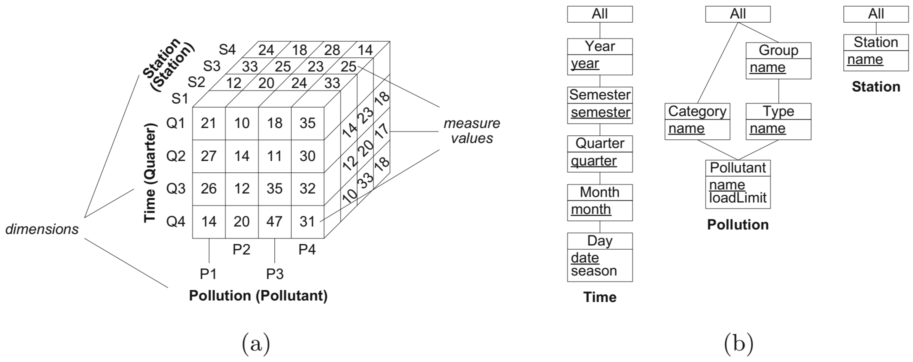


Fig. 1. A three-dimensional cube having dimensions **Time**, **Pollution**, and **Station**, and a measure **concentration**. Dimension hierarchies are shown on the right [4].

attributes, dimensions and measures. Dimensions identify data points and are usually organised hierarchically. Measures represents associated values of a data point and are usually operands of aggregations. An example of OLAP cube is shown in Fig. 1. There is no clear distinction between dimensions and measures. Any set of attributes that uniquely identifies a data point can be viewed as dimensions, and the remaining attributes are measures.

Typical OLAP operations defined on cubes include:

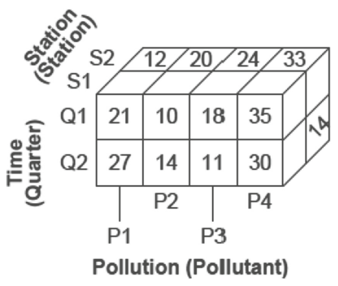
- Dice: Selecting a subset of an OLAP cube (Fig. 2a).
- Slice: Slice is a specific case of dice picking a rectangular subset of a cube by choosing a single value for one of its dimensions (Fig. 2b).
- Roll-up: Aggregating data by climbing up the hierarchy of a dimension (Fig. 2c).
- Drill-down: Aggregating data at a lower level of the dimension hierarchy (Fig. 2d). Drill-down is the reverse operation of roll-up.

In this paper we do not take into account operations that involve multiple OLAP cubes, such as drill-across [4], since multiple RDF graphs can be merged into one graph by taking their union.

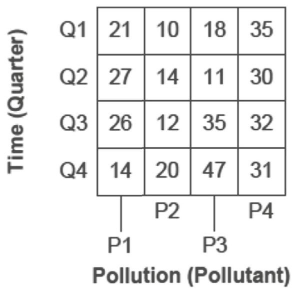
Kämpgen et al. [13] describe an approach to map OLAP queries into SPARQL queries with the RDF Data Cube (QB) vocabulary [6]. Since many Linked Data and SPARQL queries do not use QB, we examine the semantics of the above OLAP operations and propose a mapping between OLAP and SPARQL queries that are not limited to specific vocabularies.

2.2 Generating Dice and Slice Queries in SPARQL

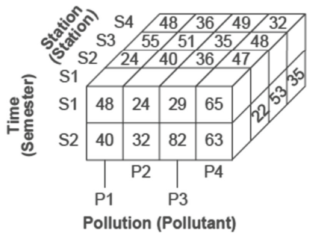
Dice and Slice select a subset of an OLAP cube while in SPARQL the same functionality is achieved by BGPs.



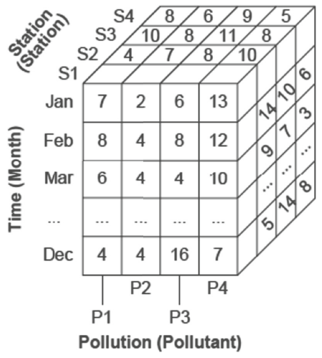
(a) Dice on **Station** = **‘S1’** or **‘S2’** and **Time.Quarter = ‘Q1’ or ‘Q2’**



(b) Slice on **Station** for **StationId = ‘S1’**



(c) Roll-up to the **Semester** level



(d) Drill-down to the **Month** level

Fig. 2. OLAP operations [4].

An OLAP data point and its attributes (dimensions and measures) correspond to a subject and its properties in a RDF graph[2]. Dice selects multiple data points in an OLAP cube, whereas in SPARQL it is analogous to a BGP with optional constraints on object values (using FILTER), as shown below:

Query 1.

```
  SELECT ?P ?Q ?S ?concentration
  WHERE
{ ?point  :Pollution    ?P ; # FILTER(?P ="P1"|| ?P ="P2")
          :Time         ?Q ; # FILTER(?Q ="Q1"|| ?Q ="Q2")
          :Station      ?S ; # FILTER(?S ="S1"|| ?S ="S2")
          :concentration :?concentration
}
```

[2] Mapping an OLAP data point to a subject is just one intuitive approach. An OLAP data point can be mapped to any RDF term.

Unlike in relational databases (where dimensions are usually keys that are distinguished from measures), we argue that dimensions and measures are indistinguishable in RDF and SPARQL. Thus any BGPs correspond to a valid Dice operation (with Slice as a special case) in OLAP. There may be difficulties to aggregate on certain types of values, since most aggregations in SPARQL are arithmetic. Meanwhile it is always possible to convert an arbitrary type to a numeric. For example a literal can be converted to its length (i.e. STRLEN), and a resource can be converted to its number of occurrences (i.e. COUNT). Queries generated with the above modifications may not be meaningful in a practical sense, they serve the purpose as far as benchmarks are concerned. In the rest of this paper we interchangeably use Dice query and BGP when no confusion is caused.

2.3 Generating Roll-Up and Drill-Down Queries in SPARQL

Roll-up and drill-down group measure values at a specific dimension level and aggregate values in each group using a given aggregate function. Without losing generality, we focus on the mapping of roll-up since drill-down is the reverse operation. In SPARQL Roll-up is achieved by GROUPing BY some variables (i.e. dimensions) in a query and aggregate on other variables (i.e. measures).

Given a Dice query (a BGP basically) that selects entities at the specified dimension levels (i.e. there is a triple pattern matching each of the specified dimension levels), simply GROUPing BY the specified dimension levels and applying an aggregate function on measure values (i.e. any variable object not appeared in GROUP BY) would simulate Roll-up in SPARQL. Taking Query 1 as an example, if we would like to know the concentration of each pollutant at each station averaged over all time points, we would GROUP BY variable *?P* and *?S* and apply AVG on variable *?concentration*, as shown in the query below:

Query 2.

```
  SELECT ?P ?S (AVG(?concentration) AS ?avgCon)
  WHERE
{ ?point  :Pollution     ?P ;
          :Time          ?Q ;
          :Station       ?S ;
          :concentration :?concentration
} GROUP BY ?P ?S
```

It is worth noticing that GROUPing BY all variables in a BGP does not change the result of the BGP[3]. Thus in SPARQL a Dice query can be trivially extended to a Roll-up query by appending a GROUP BY all variables clause at the end of its BGP.

A more involved case is when we are interested in dimension levels that do not explicitly appear in a Dice query. For example, in Query 2 instead of asking for

[3] It is enough to GROUP BY a subset of all variables that uniquely identifies an entity. Variables excluded from GROUP BY can be selected using the SAMPLE aggregation.

concentration per pollutant, we may ask for the same measure per Category in the Pollution hierarchy (shown in Fig. 1b). Depending on whether the hierarchy (dimension instance as in [4]) is explicitly stated in the RDF graph being queried, we use two different techniques to generate Roll-up queries.

Dimension hierarchy is explicit. Assuming the hierarchy is stated as triples, e.g. in the form

P_i *:rollupTo* C_j

where P_i is an instance of Pollutant, C_j is an instance of Category and :rollupTo states that its object is one level above its subject in the dimension hierarchy, we can add the triple pattern

?P a Pollutant; :rollupTo ?C. ?C a Category.

to Query 2 and GROUP BY *?C* (and *?S*) instead of *?P*.

Dimension hierarchy is absent. In this case values can be manually categorised in SPARQL using an IF expression

rdfTerm IF (boolean cond, rdfTerm expr1, rdfTerm expr2)

where the whole expression evaluates to the value of *expr*1 when *cond* evaluates to *true*, otherwise *expr*2. By nesting IF expressions, we can define a surjective (only) function

$$cat : rdfTerm \rightarrow rdfTerm$$

that maps a value to a category defined by users. For example, assuming both $P1, 2$ belong to $C1$, we can express *cat* in SPARQL as

$$cat(?P) := IF(?P = P1 || ?P = P2, C1, Other),$$

and convert Query 2 to the following query[4]

Query 3.

```
  SELECT ?C ?S (AVG(?concentration) AS ?avgCon)
  WHERE
{ ?point  :Pollution     ?P ;
          :Time          ?Q ;
          :Station       ?S ;
          :concentration :?concentration
} GROUP BY (cat(?P) AS ?C) ?S
```

[4] SPARQL 1.1 doesn't have the ability to define new functions, and therefore *cat* should be considered as a macro in Query 3.

This technique is more useful to categorise numerics (or elements of totally ordered sets) into different ranges. For example, we can define a *cat* to group numbers into ranges as

$$cat(x) := IF(x <= low, “Low'', IF(x <= high, “Medium'', “High''))$$

where *low* and *high* are numbers.

Given a BGP (i.e. a Dice query), ASPG adopts a naive heuristic to extend it to a Roll-up query: (1) It randomly selects a subset of all variables of the BGP as dimensions, and the remains as measures; (2) All dimensions are used in a GROUP BY clause; (3) If a measure is known to be numerical, it is aggregated using one of the set functions COUNT, MAX, MIN, AVG, SUM, GroupConcat. Otherwise, this measure is firstly converted to a literal with STR and then to an integer with STRLEN, and aggregated using a set function. This procedure is listed below:

```
queryGen(BGP)
    D, M ∈ vars(BGP)
    GroupBy ←“GROUP BY”
    for d ∈ D
        GroupBy ← concat(GroupBy, d)

    SELECT ←“SELECT”
    for m ∈ M
        AGG ∈ {COUNT, MAX, MIN, AVG, SUM, GroupConcat}
        if m is numerical
            SELECT ← concat(SELECT, AGG(m))
        else
            SELECT ← concat(SELECT, AGG(STRLEN(STR(m))))

    query ← concat(SELECT, BGP, GroupBy)
    return query
```

2.4 Generating Basic Graph Patterns

We generate BGPs by replacing nodes in RDF graphs with variables. A RDF graph (or a BGP) can be decomposed into star-shaped or chain-shaped sub graph patterns. Considering a triple (or a triple pattern) as an undirected edge between subject and object, we define the degree of a node as the number of edges connecting to this node. A star-shaped graph pattern has one and only one central node with a degree greater than 1 and all other nodes of degree 1. A chain only has nodes whose degree are no more than 2. We generate a sub-graph from a RDF graph by repeating two steps: (1) select one node in the RDF graph as root, (2) add an edge connected to the root to the sub-graph. A star-shaped graph pattern is generated by selecting the same node as root in every iteration, while a chain is generated as selecting as root the other node in the last added edge in each iteration. We generate a mix of stars and chains by controlling a branching probability of whether to select a different root in each step, as shown in the pseudo code below, where *RDF* is a RDF graph, *T* is a termination predicate function mapping a BGP to a boolean, *p* is branching probability, and *parametrise* maps a non-property IRI to a variable:

```
BGPGen(RDF, T, p)
    BGP ← {}
    root ∈ IRIs(RDF)
    while (!T(BGP))
        E ← getTriples(root)
        e ∈ E
        BGP ← parametrise(e) ∪ BGP
        if (random() < p)
            root ← root
        else
            root ← getObject(e)
    return BGP
```

The termination function is used to control the length of generated BGPs. In this paper we define the termination condition to result in true if either the BGP reaches 10 triple patterns or the longest path in the BGP reaches 5.

The above algorithm guarantees a non-empty result set when evaluating the generated BGP against the source RDF, but there is no guarantee about the size of the result. To avoid BGPs whose result size is too small for aggregation, we evaluate generated BGPs and filter out those whose result size is less than a threshold. This safe guard is not always necessary. Later we present a set of queries generated from DBpedia and none of the BGPs falls below a threshold of 100,000.

Generating BGP with large or remote RDF graphs. When using the above method, one may encounter difficulties when the RDF graph cannot be used as a direct input to *BGPGen*. For example, the graph may be too large to be traversed or it is only available as a SPARQL endpoint. In order to deal with such cases, we adopt techniques that combines ontology and triple sampling to convert large RDF graphs into smaller ones. We describe our techniques using DBpedia as an example, but the techniques can be applied to any graph.

To generate a BGP we need to know the connection between nodes. Such information is often captured in an ontology-like structure of a RDF graph that gather all instance level properties to their classes. For simplicity we still use ontology to refer to such structure. We can issue a SPARQL CONSTRUCT query to recover the ontology (assuming all instances in the RDF belong to some classes, i.e. all *rdf:type* are explicit). However, to construct the whole ontology in one query is likely to end with a time out. Instead, we first retrieve all classes, and then use a script to collect properties between any two classes using the query template below:

Query 4.

```
CONSTRUCT
{ dbo:$1 ?p dbo:$2 }
SELECT DISTINCT ?p
WHERE { [ a dbo:$1] ?p [ a dbo:$2]. }
```

where *$1* and *$2* are replaced by class names (e.g. Person, Event etc.). The ontology is the union of all graphs returned by Query 4. The ontology can be used as the input graph (i.e. the parameter G) in the BGP generation algorithm with some extra care taken. Since all nodes in the ontology are classes, they should all be replaced with variables in generated BGPs. In addition, when following a reflexive property, a new variable should be used as root. For example, *dbo:Person* has a reflexive property *foaf:knows*. When this property is included in a BGP, its subject and object should be two different variables.

Using the ontology instead of the original RDF graph significantly reduces the complexity of BGP generation. However it does not always guarantee that the generated queries have results against the original graphs. For example in DBpedia both *Athlete* and *Artist* are sub-classes of *Person*, an instance of either *Athlete* or *Artist* may also has a *rdf:type* property pointing to *Person*. As a result properties of both *Athlete* and *Artist* are gathered at *Person*. There is a chance that an *Athlete* property and an *Artist* are connected to the same node in a BGP, which may not match any triple in the original graph. This issue can be relieved by gathering properties only to the lowest class of an instance, however doing that in SPARQL is quite cumbersome[5].

When the above method is not applicable (e.g. generating BGPs from DBpedia), we employ triple sampling as an alternative approach to extract subsets of RDF graphs. By repetitively sampling sub-graphs of simple shapes, a more complex and larger sub-graph can be constructed. For example, in ASPG we sample DBpedia using triple chains of length 5, as show in Query 5.

Query 5. Chained triple sampling

```
CONSTRUCT
{
  ?s  ?p1 ?n1. ?n1 ?p2 ?n2.
  ?n2 ?p3 ?n3. ?n3 ?p4 ?n4.
  ?n4 ?p5 ?e.
}
WHERE
{
  ?s a dbo:$1. ?e a dbo:$2.
  ?s  ?p1 ?n1. ?n1 ?p2 ?n2.
  ?n2 ?p3 ?n3. ?n3 ?p4 ?n4.
  ?n4 ?p5 ?e.
}
```

[5] It requires to calculate the position of an item in a linked list and to identify the maximum item in a set. Refer to https://git.io/vwP0t for more details.

where *$1* and *$2* are replaced by class names. It is left to users to decide how many and what class pairs are used. For example, in the construction of DBOB we use the top 50 classes that have most instances, and it turns out that triple chains sampled by Query 5 intertwine with each other. The result graph is significantly smaller than DBpedia while its structure is rich enough to generate complex queries.

In addition we may also want to identify properties whose ranges are numerics, even it is always possible to convert an arbitrary type to a numeric in SPARQL. Such information enables us to identify variables of numerics to which aggregate functions can be directly applied.

2.5 Complexity Analysis

We examine the time complexity of aggregate functions used in ASPG, namely GROUP BY and set functions COUNT, MAX, MIN, AVG, SUM, GroupConcat (excluding SAMPLE).

GROUP BY can be realised by the application of a higher-order 'map' function on a constant time lower-order function and each set function can be mapped to a higher-order 'fold' function on a constant time arithmetic function. All aggregations used in ASPG have $O(n)$ time complexity, where n is the size of query result regardless of the grouping of the result. We exclude SAMPLE from ASPG since it is a $O(1)$ operation.

We conclude that the time complexity of aggregating on a BGP is linear in the number of aggregate functions and independent of the grouping. In other words, the time complexity of a query (generated by ASPG) can be characterised by its BGP and its number of aggregate functions.

3 DBOB: A Benchmark Constructed with ASPG

In order to evaluate ASPG, we construct an OLAP benchmark named DBOB from DBpedia's online endpoint. DBOB contains 12 queries, of which Q1–3 are real-world queries from online analysis and Q4–12 are generated with ASPG.

Query 4–12 are generated following the steps below:

1. Retrieving the top 50 classes from DBpedia having most instances.
2. Sampling from the DBpedia SPARQL endpoint using chains of length 5 whose endpoints are drawn from instances of the 50 classes.
3. Generating OLAP queries from the RDF graph gained from step 2.
4. Evaluating the query against DBpedia and filtering out those whose result size is less than 100,000.

Due to the page limit the complete list of DBOB queries is available at http://xgfd.github.io/ASPG/.

4 Evaluation

We evaluate ASPG from two perspectives to show that ASPG is able to generate non-trivial queries. Firstly we compare DBOB queries to OLAP4LD-SSB queries [12] with respect to query complexity and types of query patterns. Secondly we use DBOB to evaluate a Virtuoso engine and analyses the result.

4.1 DBOB Quereis Vs. OLAP4LD-SSB Queries

As stated in Sect. 2.5, the time complexity of a query can be decomposed into the complexity of its BGPs and the numbers of aggregate functions. We roughly measure the complexity of a BGP by its number of triple patterns[6] (Table 1).

Table 1. Comparison of DBOB and OLAP4LD-SSB queries.

DBOB	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
# of triple patterns	10	5	4	4	6	4	4	8	7	2	7	7
# of group by-s	1	1	3	1	1	1	1	1	1	1	1	1
# of set functions	2	3	3	4	5	4	3	8	7	1	7	7
OLAP4LD-SSB												
# of triple patterns	6	6	7	9	8	8	10	10	8	9	11	13
# of group by-s	0	0	0	1	1	1	1	1	1	1	1	1
# of set functions	1	1	1	1	1	1	1	1	1	1	1	1

Comparing to OLAP4LD-SSB, the number of triple patterns of ASPG queries vary a lot, as a result of random sampling. In addition, since ASPG does not focus on the semantic of queries, it can simply add as many aggregate functions as required. The ability of providing triple patterns and aggregate functions on demand makes ASPG a very flexible tool for benchmarking.

4.2 Evaluating Virtuosos with DBOB

We run DBOB on a DBPedia 3.9 endpoint hosted on a machine with the following settings: 4*2.9 GHz CPU, 16 G memory, Ubuntu 14.04.4, Virtuoso opensource 7.1.0.

We use the BSBM query driver[7] to execute all queries with 0 warm up and 20 runs.

[6] The complexity of a BGP is also affected by the number of intermediate results in each join. However the later requires detailed statistics to estimate which are not always available.

[7] http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/spec/BenchmarkRules/index.html#datagenerator.

The evaluation result is shown in Table 2, where QET stands for query execution time in seconds, #Rslt is the query result size before aggregation, #Trpl is the number of triple patterns, and #AF is the number of aggregate functions. We also calculate the correlation between QET and the number of triple patterns, result size and the number of aggregate functions respectively.

Table 2. DBOB evaluation result.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
#Rslt	600	39.4K	10.4K	95.5K	59.1K	65.3K	548.5K	120.8K	258.8K	81.2M	175.5K	5.0M
QET	0.65	1.19	1.31	0.10	0.72	0.31	2.90	0.08	1.43	19.33	2.74	1.73
	Correlation											
	#Rslt		#Trpl		#AF							
QET	0.99		0.07		0.38							

Most queries are finished in no more than 3 s. This may due to that queries with aggregation usually do not need to materialise all intermediate results. In addition we see the correlation between QET and the number of triple patterns is quite low. It is not surprising since QET of BGPs is mainly affected by the number of intermediate results which is not captured by only the number of triple patterns. At the same time the number of aggregate functions shows a higher impact on QET. One possible reason could be the high number of aggregate functions in ASPG queries. Alternatively as the contribution to QET from aggregation is liner to result size, the relatively higher impact from aggregation may just be a side effect of the high correlation between the result size and QET. It may be worth measuring only the execution time of aggregation, however such measure is usually difficult to obtain from outside of query engines.

5 Related Work

We divide related work into two categories: SPARQL query generators and SPARQL benchmarks.

5.1 Related Query Generators

ASPG generates queries from a RDF graph, which is similar to SPLODGE [9]. SPLODGE exploits query characteristics (e.g. join type, query type, variable pattern) and constructs queries from a federated RDF graph. While ASPG focuses on simulating OLAP queries, SPLODGE aims to generate queries for federated benchmarks. Both decompose queries into star-shaped or chained triple patterns. ASPG queries are generated by replacing nodes in a sub-RDF-graph with variables, while SPLODGE queries are generated from linked predicates (i.e. a pair of predicates sharing a common node). SPLODGE queries are not guaranteed to have results, but statistics are used to increase the chance.

FEASIBLE [16] represents a different approach to generate benchmark queries. Instead of generating queries from a RDF graph, it takes existing queries (from query logs) as prototypes and generates similar queries. Comparing to ASPG and SPLODGE, FEASIBLE queries are usually more close to real-world queries.

5.2 Related Benchmarks

To the best of our knowledge only two existing benchmarks are based in an OLAP scenario, namely BowlognaBench [8] and OLAP4LD [12]. We also review a few popular non-OLAP benchmarks.

– Lehigh University Benchmark (LUBM) [10] is designed with focus on inference and reasoning capabilities of RDF engines.
– SP^{2}Bench [17] has a focus of testing the performance of a variety of SPARQL features.
– The Berlin SPARQL Benchmark (BSBM) [1] mimics a e-commerce scenario and its dataset resembles a relational database.
– DBpedia SPARQL Benchmark (DBPSB) [14] uses (a sub set of) DBpedia as testing data and most used DBpedia queries as testing queries.
– BowlognaBench models an OLAP use case around the Bowlogna Ontology [7] and implements queries such as TopK, Max, Min, Path etc.
– OLAP4LD converts dataset and queries of the Star Schema Benchmark [15] into RDF and SPARQL. It resembles OLAP queries in relational databases.

We compare DBOB with aforementioned benchmarks in Table 3.

Table 3. Comparison of DBOB and existing benchmarks, adapted from [14]. Synthetic stands for artificially generated data; Real stands for real-world data; Mix stands for a mix of the former two types.

	LUMB	SP2Bench	BSBM	DBPSB	OLAP4LD	Bowlogna	DBOB
Dataset type	Synthetic	Synthetic	Synthetic	Real	Synthetic	Synthetic	Real
Query type	Synthetic	Synthetic	Synthetic	Real	Synthetic	Synthetic	Mix
Num. of classes	43	8	8	239	7	76	239
Num. of properties	32	22	51	1200	28	36	1200

6 Conclusions and Future Plan

In this paper we present ASPG that can be used to generate Dice, Slice, Roll-up and Drill-down queries in SPARQL. By exploiting ontologies and triple sampling techniques, ASPG is able to generate queries from large RDF graphs or graphs available as SPARQL endpoints. We further construct a benchmark called DBOB with ASPG and DBpedia to evaluate processing time of OLAP SPARQL queries.

Queries generated by ASPG usually have more complex BGPs compared to real-world queries. Perhaps human users are more likely to issue simple queries and combine their results afterwards, due to the lack of convenient query builders and constraints on query complexity from SPARQL endpoints. We argue that as far as query processing time is concerned, generated queries may give more insight on the performance of SPARQL engines than simple real-world queries. In addition, it is likely that the increasing demand of SPARQL analytics will foster better tools that enable users to generate complex queries. The Roll-up generation heuristic used by ASPG may contribute to the creation of such tools.

Currently ASPG queries only consist of one BGP and randomly selected aggregate functions, while real-world queries may also employ FILTERs and sub-queries (e.g. Q2 and Q3 of DBOB). As a result ASPG queries only represent some basic analytical needs. A future plan is to extend ASPG to generate multiple BGPs and sub queries that covers a broader range of analysis operations.

References

1. Bizer, C., Schultz, A.: The Berlin SPARQL benchmark. Int. J. Semant. Web Inf. Syst. (IJSWIS) - Special Issue on Scalability and Performance of Semantic Web Systems **5**(2), 1–24 (2009)
2. Capadisli, S., Auer, S., Riedl, R.: Linked Statistical Data Analysis. Semantic Web (2013)
3. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. ACM SIGMOD Record **26**(1), 65–74 (1997)
4. Ciferri, C., Ciferri, R., Gómez, L., Schneider, M., Vaisman, A., Zimányi, E.: Cube algebra: a generic user-centric model and query language for OLAP cubes. Int. J. Data Warehous. Min. **9**(2), 39–65 (2013)
5. Codd, E.F., Codd, S.B., Salley, C.T.: Providing OLAP (on-line Analytical Processing) to user-analysts: an IT mandate. Codd Date **32**, 3–5 (1993)
6. Cyganiak, R., Reynolds, D., Tennison, J.: The RDF Data Cube Vocabulary
7. Demartini, G., Enchev, I.: The bowlogna ontology: fostering open curricula and agile knowledge bases for Europe ' s higher education. Landscape **0**, 1–11 (2012)
8. Demartini, G., Enchev, I., Wylot, M., Gapany, J., Cudré-Mauroux, P.: BowlognaBench-Benchmarking RDF analytics. Data-Driven Process Discovery Anal. **116**, 82–102 (2011)
9. Görlitz, O., Thimm, M., Staab, S.: SPLODGE: systematic generation of SPARQL benchmark queries for linked open data. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012. LNCS, vol. 7649, pp. 116–132. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35176-1_8
10. Guo, Y., Pan, Z., Heflin, J.: LUBM: a benchmark for OWL knowledge base systems. Web Semant. **3**(2–3), 158–182 (2005)
11. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language (2013)
12. Kämpgen, B., Harth, A.: No size fits all – running the star schema benchmark with SPARQL and RDF aggregate views. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 290–304. Springer, Heidelberg (2013). doi:10.1007/978-3-642-38288-8_20

13. Kämpgen, B., ORiain, S., Harth, A.: Interacting with Statistical Linked Data via OLAP Operations. In: Simperl, E., Norton, B., Mladenic, D., Della Valle, E., Fundulaki, I., Passant, A., Troncy, R. (eds.) ESWC 2012. LNCS, vol. 7540, pp. 87–101. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46641-4_7
14. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo, A.-C.: DBpedia SPARQL benchmark – performance assessment with real queries on real data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011. LNCS, vol. 7031, pp. 454–469. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25073-6_29
15. Neil, P.O., Neil, B.O., Chen, X.: Star Schema Benchmark - Revision 3. Technical report, UMass/Boston (2009)
16. Saleem, M., Mehmood, Q., Ngonga Ngomo, A.-C.: FEASIBLE: a feature-based SPARQL benchmark generation framework. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 52–69. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25007-6_4
17. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP2Bench: a SPARQL performance benchmark. In: Proceedings of the International Conference on Data Engineering, pp. 222–233. IEEE (2009)

Towards Answering Provenance-Enabled SPARQL Queries Over RDF Data Cubes

Kim Ahlstrøm(✉), Katja Hose, and Torben Bach Pedersen

Department of Computer Science, Aalborg University, Aalborg, Denmark
{kah,khose,tbp}@cs.aau.dk

Abstract. The SPARQL 1.1 standard has made it possible to formulate analytical queries in SPARQL. While some approaches have become available for processing analytical queries on RDF data cubes, little attention has been paid to answering provenance-enabled queries over such data. Yet, considering provenance is a prerequisite to being able to validate if a query result is trustworthy. The main challenge for existing triple stores is the way provenance can be encoded in standard triple stores based on context values (named graphs). Hence, in this paper we analyze the suitability of existing triple stores for answering provenance-enabled queries on RDF data cubes, identify their shortcomings, and propose an index to handle the high number of context values that provenance encoding typically entails. Our experimental results using the Star Schema Benchmark show the feasibility and scalability of our index and query evaluation strategies.

1 Introduction

The rapid expansion of the Linked Open Data (LOD) cloud and the introduction of SPARQL 1.1 have created new possibilities for the integration of online data. It is natural to use this vast amount of linked data to answer analytical queries [1]. Several initiatives have already been started to facilitate analytics over the Semantic Web [9,14,18]. When querying data from remote sources, provenance data is essential to ensure that the results are interpreted in a correct manner. Provenance data is not limited to quality control, there are many more uses such as access control, result ranking, query optimization, and provenance filters [19]. Therefore, it is important not to limit the descriptive power of provenance data. Hence, we use the W3C PROV-O vocabulary [20].

The standard way to encode provenance data is using reification. However, due to the verbose nature of reification we use provenance identifiers to link the provenance data as suggested in [13]. The context value of a triple is used to store a provenance identifier, this identifier corresponds to the subject in a provenance triple, thus connecting one or more information triples to a provenance triple.

We observe the problem that standard triple stores, i.e., Jena TDB [2] and RDF4J Native [8], are not designed to support provenance data. Hence, to support it, we either need to find an encoding so that standard triple stores can support it or develop a new type of triple store (see related work in Sect. 8). In this paper we make the following contributions:

Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 186–203, 2016.
DOI: 10.1007/978-3-319-50112-3_14

– Analysis of the suitability of standard triple stores for answering provenance-enabled analytical queries.
– Two query processing strategies to enable provenance-enabled SPARQL queries over RDF data cubes.
– Proposing the Context Index to reduce the number of context values that have to be considered to answer a query.
– Evaluation of our strategies combined with the index using the Star Schema Benchmark.

The rest of this paper is structured as follows: we start with preliminaries in Sect. 2, here we also present our running example. Section 3 presents the baseline strategy for answering provenance-enabled analytical queries. In Sect. 4 we propose our novel Context Index. In Sect. 5 we combine the baseline strategy with the index. Next, in Sect. 6 the materialization strategy is presented and how it can be combined with the Context Index. In Sect. 7 we evaluate the proposed strategies. We conclude with related work in Sect. 8 followed by the conclusion and future work in Sect. 9.

2 Preliminaries

In this section, we present how provenance data is encoded, define provenance-enabled analytical queries, and present our running example of a provenance-enabled analytical query and an RDF data cube.

2.1 Encoding Provenance

Provenance data describes a piece of data, in terms of its origin, how it was created, when it was changed, and who created it. Reification is the standard (W3C) for expressing provenance information about triples. However, using the context value as a provenance identifier is gaining popularity [4,10,13,22,24]. To further define what provenance is, we need to define an RDF triple.

An *RDF triple* consists of a *subject*, *predicate*, and *object*, where the subject is related to the object through the relationship defined by the predicate. Formally we say a triple t is defined as $t = (s, p, o) \in (U \cup B) \times U \times (U \cup B \cup L)$, where U is a set of IRIs, B is a set of blank nodes, and L is a set of literals. An *RDF graph* contains a set of triples, each graph has a unique identifying IRI. When a triple is contained in an RDF graph, we write the triple as a *quad*, it consists of a subject, predicate, object, and *context value*. The context value contains the unique identifier of the graph.

Given a set of IRIs U, a set of blank nodes B, a set of literals L, and a set of query variables V, a *triple pattern* (TP) is defined as $TP = (s, p, o) \in (U \cup B \cup V) \times (U \cup V) \times (U \cup B \cup L \cup V)$. A *basic graph pattern* (BGP) consists of a set of triple patterns joined via shared query variables.

In this work, we distinguish between information triples, metadata triples, and provenance triples. An *information triple* represents a piece of information.

An example of an information triple is: `(ex:kim foaf:name "Kim")`, this triple encodes that `ex:kim` has the name "Kim". A *metadata triple* describes part of the structure of the RDF data cube, e.g., `(ex:City ex:rollUpTo ex:Country)`, this triple describes that the city level rolls up to the country level. A *provenance triple* describes an information triple; in this paper, we use the W3C PROV Ontology [20] (PROV-O) to describe the provenance data.

PROV-O defines provenance based on three core components: *Entities* are defined as conceptual or physical things. In the context of RDF, an entity represents a set of information triples. *Activities* express how entities are created or changed. *Agents* are actors that interact with activities or entities.

When we have a collection of connected provenance triples, we say that the provenance triples constitute a *provenance graph*. Figure 1 illustrates the provenance graph of an information triple using the PROV-O components. The provenance graph has two entities illustrated as ovals. There is one activity, illustrated by a rectangle, it uses one entity and generates another entity, the generated entity is marked as trusted. The agent, illustrated as a pentagon, is associated with the activity.

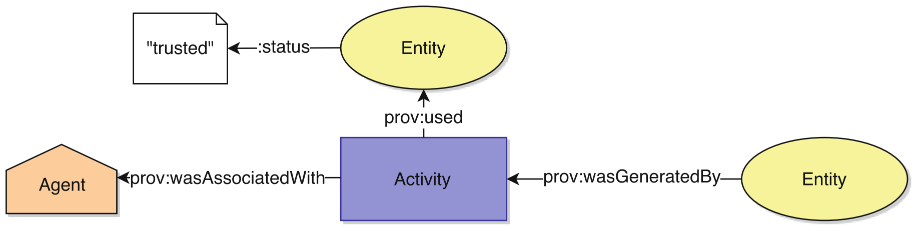


Fig. 1. Provenance graph

Information triples are linked to a provenance graph via an entity. The identifier of the entity is linked with the context value of the information triple. We will show a concrete example of this in the next section.

2.2 Provenance-Enabled RDF Data Cubes

In many ways, an RDF data cube is a traditional cube as defined in the context of relational databases [17]. However, the underlying data is formatted as RDF and instead of a schema an ontology is used; a full definition of the standard ontology for defining RDF data cubes is provided by the W3C [6]. An *RDF data cube* contains *observations*, these are the focus of the desired analysis, e.g., sales of books. An observation has a set of numerical attributes called *measures*. The observations are connected to a set of *dimensions* through a *dimension property*. Dimensions contain hierarchies of *levels* that are connected via *level properties*; each level may contain several attributes.

The RDF data cube in our running example has three dimensions: Date, Shop, and Location, the observations are sales of books and have one measure

`:price`. The Date dimension has three levels: Day, Month, and Year, the Book dimension has one level: Book, and the Location has three levels: Shop, City, and Country. The levels of these dimensions are described by attributes such as `:monthName` and `:yearNumber`. The structure of the RDF data cube is illustrated in Fig. 2. Throughout this paper, we use the QB4OLAP vocabulary[1] to define our cube. We choose this vocabulary because it builds upon the W3C standard vocabulary [6] and in addition defines levels, aggregate functions, cardinalities, and hierarchies. The strategies presented in this paper are not limited to the QB4OLAP vocabulary.

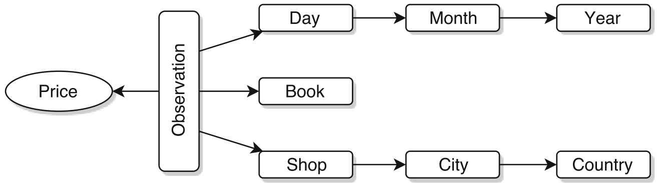


Fig. 2. Structure of the RDF data cube in running example

Table 1 shows a subset of the RDF data cube consisting of information triples and metadata triples; the data describes two book purchases made at two different dates. For brevity, the example does not contain any Book or Shop dimensions, or the QB4OLAP cube definition.

In this paper, we use provenance identifiers to link information triples to a provenance graph [13]. A *provenance identifier* is an IRI that is stored as the context value of the information triple and as the subject of a provenance triple from the corresponding provenance graph. In this example, we use capital letters to represent the IRI of the provenance identifiers, e.g., `:A`. Not all context values are provenance identifiers, all triples that define part of the cube structure, e.g., `(:january2016 skos:broader :2016)`, are stored in the graph `:Metadata`.

In Table 2, five provenance graphs are displayed. The provenance graphs describe the provenance identifiers: `:A`, `:B`, `:C`, `:D`, and `:E`. These provenance identifiers link the information triples to the provenance graphs. Each provenance graph consists of three provenance triples. The provenance identifier represents the PROV-O entity that has been generated by some activity and the activity "used" some source entity during the generation. The source entities have a status attribute that is either "trusted" or "unknown". All provenance triples are stored in the graph `:Provenance`.

[1] https://github.com/lorenae/qb4olap/blob/master/rdf/qb4olap.1.2.ttl.

Table 1. The RDF data cube *cube* showing two book sales made on two different dates

Subject	Predicate	Object	Context value
:observation1	:price	7	:A
:observation1	:date	:date31012016	:Metadata
:date31012016	skos:broader	:january2016	:Metadata
:january2016	:monthName	"January"	:B
:january2016	skos:broader	:2016	:Metadata
:2016	:yearNumber	2016	:C
:observation2	:price	12	:D
:observation2	:date	:date01022016	:Metadata
:date01022016	skos:broader	:february2016	:Metadata
:february2016	:monthName	"February"	:E
:february2016	skos:broader	:2016	:Metadata

Table 2. Provenance triples for *cube*

Subject	Predicate	Object	Context value
:A	prov:wasGeneratedBy	:BookExtractor	:Provenance
:BookExtractor	prov:used	:DBpedia	:Provenance
:DBpedia	:status	"trusted"	:Provenance
:B	prov:wasGeneratedBy	:DateExtractor1	:Provenance
:DateExtractor1	prov:used	:DateRepository1	:Provenance
:DateRepository1	:status	"trusted"	:Provenance
:C	prov:wasGeneratedBy	:CalenderExtractor	:Provenance
:CalenderExtractor	prov:used	:CSVFile	:Provenance
:CSVFile	:status	"trusted"	:Provenance
:D	prov:wasGeneratedBy	:WebTableExtractor	:Provenance
:WebTableExtractor	prov:used	:WebTable	:Provenance
:WebTable	:status	"unknown"	:Provenance
:E	prov:wasGeneratedBy	:DateExtractor2	:Provenance
:DateExtractor2	prov:used	:DateRepository2	:Provenance
:DateRepository2	:status	"trusted"	:Provenance

3 Processing Provenance-Enabled Analytical Queries

In this section, we define provenance-enabled analytical queries, propose our baseline strategy called "Native Querying Strategy", and conduct an analysis of shortcomings of standard triple stores in this context.

3.1 Provenance-Enabled Analytical Queries

A *provenance-enabled analytical query* is a SPARQL [12] query that is executed over an RDF data cube, where the data the query is evaluated over is filtered using a provenance query. Similar to [24] we adopt an approach where the analytical part and the provenance part of the query are seperated, both to enhance understandability and for performance reasons. Alternatively, it is possible to rewrite the queries such that they are combined into one, however, this is a complex process [24]. Using SPARQL 1.1 we express the analytical query: "find the average price of sold books per year", see Query 1. Note that in this query we use the `skos:broader` predicate to roll-up a level along a hierarchy, as defined by the QB4OLAP vocabulary [9].

```
SELECT ?year AVG(?price)
WHERE {
 ?fact :price ?price ;
      :date ?dayLevel .
 ?dayLevel skos:broader ?monthLevel .
 ?monthLevel skos:broader ?yearLevel .
 ?yearLevel :yearNumber ?year .
}
GROUP BY ?year
```

Query 1. Analytical query

A provenance query is characterized by always returning a set of context values, these correspond to the provenance identifiers from the information triples. Query 2 shows a provenance query that finds the context values of all triples that originate from trusted sources.

```
SELECT ?provenanceIdentifiers
FROM :Provenance
WHERE {
 ?provenanceIdentifiers prov:wasGeneratedBy ?activity .
 ?activity prov:used ?entity .
 ?entity :status ?status .
 FILTER (?status ="trusted") .
}
```

Query 2. Provenance query

3.2 Native Querying Strategy

As a baseline strategy, we propose the "Native Querying Strategy", henceforth referred to as "Native". Figure 3 illustrates the steps of this strategy.

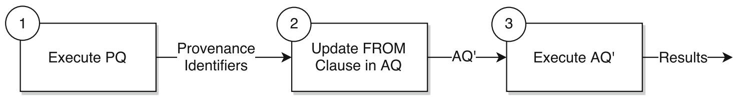


Fig. 3. Steps of the native strategy

First, the provenance query (PQ) is executed and the provenance identifiers of all matching provenance graphs are returned. When executing Query 2 over the provenance data presented in Table 2, the following provenance identifiers are returned: `:A`, `:B`, `:C`, and `:E`. Second, the analytical query (AQ) is updated by adding additional FROM clauses, we name this query $AQ^{'}$. For each provenance identifier, we add a FROM clause with the IRI of the provenance identifier. This ensures that only information triples with a provenance graph that matches the provenance query are considered when executing the AQ. Additionally, we need to make sure that we always have a valid cube. Therefore, a FROM clause with the `:Metadata` IRI is always added. In Query 3 we see $AQ^{'}$; four FROM clauses have been added, one for each of the provenance identifiers and one for the cube metadata.

```
SELECT ?year AVG(?price)
FROM :A
FROM :B
FROM :C
FROM :Metadata
WHERE {
  ?fact :price ?price ;
       :date ?dayLevel .
  ?dayLevel skos:broader ?monthLevel .
  ?monthLevel skos:broader ?yearLevel .
  ?yearLevel :yearNumber ?year .
}
GROUP BY ?year
```

Query 3. Updated analytical query ($AQ^{'}$)

Third, we execute $AQ^{'}$ over the RDF data cube. When executed over the example cube in Table 1 the query will evaluate to: “2016, 7”. This means that in 2016 the average price for sold books was 7.

3.3 Preliminary Analysis

In this section, we make a preliminary analysis of the Native strategy and how it performs on standard triple stores to determine its strengths and weaknesses. We hypothesize that standard triple stores, i.e. Jena TDB [2] Band RDF4J native [8], are not able to efficiently handle a large number of FROM clauses. This is important, because the Native strategy may have hundreds of FROM clauses, depending on the provenance query and the provenance data.

To test this, we create a small RDF data cube with 68,700 triples, where 65% are information triples, 19% are provenance triples, and 16% are metadata triples. We create three RDF data cubes such that the information triples are distributed over 1000, 500, and 100 provenance identifiers. We execute the same query on the three datasets rewritten such that the number of FROM clauses in the query matches the datasets, i.e., 1000, 500, and 100 FROM clauses. The experiment is conducted on the Jena TDB and RDF4J native triple stores, both stores are created with the GSPO, GPOS, GOSP, SPOG, POSG, and OSPG indices to ensure that the queries are evaluated in an efficient manner. Table 3 shows the results of this experiment. We see that, when the number of FROM

Table 3. Runtime of standard triple stores using the native strategy

	100 FROM clauses	500 FROM clauses	1000 FROM clauses
Jena TDB	1.4 s	7.5 s	13.7 s
RDF4J native	1.0 s	28.0 s	47.0 s

clauses increases, the query evaluation time increases. Based on this observation, we can conclude that the hypothesis holds and it is indeed a problem for standard triple stores to handle a high number of FROM clauses in SPARQL queries.

4 Context Index

To reduce the number of FROM clauses in the AQ, we propose the *Context Index*. Using this index it is possible to reduce the number of provenance identifiers that need to be added as FROM clauses. First, we explain the structure of the index, then how it is used, and last how it is constructed.

4.1 Structure

The Context Index is an unbalanced tree where each node corresponds to a predicate from the RDF data cube except the root node. Each child of the root is a measure or a dimension property; the node is named after the predicate of the corresponding triple. The following nodes are attributes or level properties, again named after the corresponding predicates. Each node that is an attribute is connected to one or more leaf nodes, the leaf nodes are named after provenance identifiers for that specific attribute.

Figure 4 illustrates the index constructed based on the data used in the running example, see Table 1. The root node has two children, the measure `:price` and the dimension property `:date`. This dimension property links to the bottom level of the Date dimension, the Day level. This level does not have any attributes, only a level property. Recall that QB4OLAP uses the predicate `skos:broader` to identify these. The month level has the attribute `:monthName` and the level property `skos:broader`. The year level has the attribute `:yearNumber`. The leaves are the property identifiers from Table 1, such that the provenance identifier of the observations with the predicate `:price` are the leaves of that predicate, i.e., `:A` and `:D`.

4.2 Lookup

The lookup is split into two steps. First, we analyze the analytical query.

The WHERE clause of the analytical query is traversed when path-shaped BGPs are extracted. A *path-shaped BGP* (PSB) is a non-circular chain of triple patterns $\{(s_1, p_1, o_1), (s_2, p_2, o_2), ..., (s_n, p_n, o_n), ..., (s_m, p_m, o_m)\}$, where

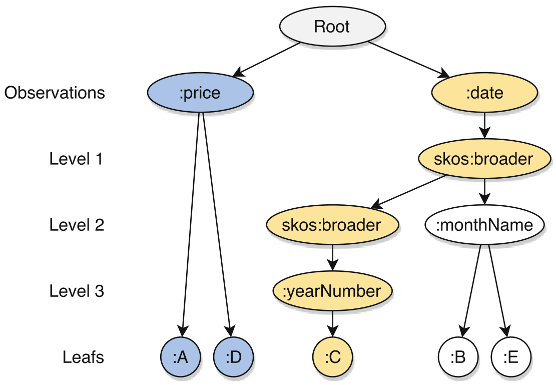


Fig. 4. The context index.

each triple pattern is linked to at most two triple patterns via object-subject joins, such that $o_i = s_{i+1}$.

If we analyze the analytical query, Query 1, introduced above we see that two PSBs are found. They are illustrated in Fig. 5.

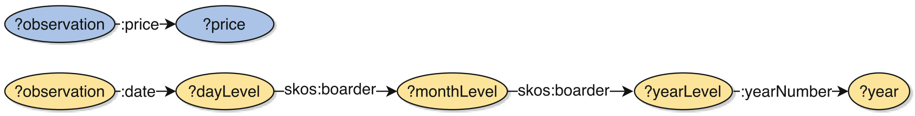


Fig. 5. Two path-shaped basic graph patterns (Color figure online)

Second, we use the PSB to look up the provenance identifiers in the context index. For each PSB, we perform a lookup in the index. By matching the predicates of the triple patterns with the predicates in the index, we identify a set of leaf nodes. This is the set of provenance identifiers that is needed to answer the analytical query. We have color-coded the two PSBs in Fig. 5 in blue and yellow, and correspondingly marked the lookup paths in Fig. 4. The result of the lookup consists of the three provenance identifiers: `:A`, `:D`, and `:C`. This is useful because any provenance identifier that is not found this way, can be discarded. This is because they will not be used when answering the analytical query. Therefore, they can be used to reduce the number of FROM clauses, we will elaborate on this in Sect. 5.

4.3 Construction

Now we explain how the index is built. The index is precomputed and updated when the data is updated. To build the index, a full scan of the RDF data cube is required. First, the RDF data cube definition is traversed; starting from the observations all measures are added as nodes, they are named after the predicate of the specific measure, e.g., `:price`. For each dimension, a new node is created, these nodes are named after the dimension property, e.g., `:date`. Second, in a depth-first manner each dimension is traversed one level at a time. All attributes are added as nodes to their corresponding level and named after the attribute predicate. Similarly, each parent level spawns a new node. This node is named after the level property, in QB4OLAP these are always called `skos:broader`. This continues until all dimensions have been traversed. Third, for each attribute a query is generated and issued that returns the provenance identifiers of that specific attribute. These are added as leafs to the attribute. Figure 4 illustrates a fully constructed context index.

5 Index-Based Native Strategy

In this section, we combine the Native Querying Strategy with the Context Index in order to address the problem of too many FROM clauses, as discussed in Sect. 3.3. In this strategy, we use the fact that the provenance query potentially finds more provenance identifiers than what are actually needed to answer the analytical query. Figure 6 illustrates the steps of the strategy.

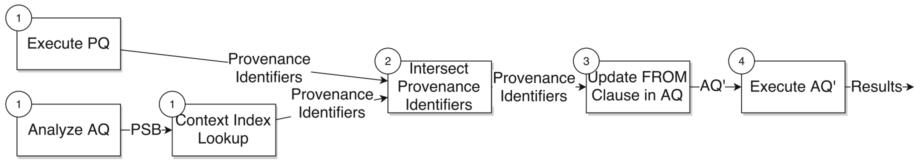


Fig. 6. Steps of the index-based native strategy

First, the provenance query is executed, as described in Sect. 3.2, and a set of provenance identifiers is returned. In our running example, this corresponds to the set: (`:A`, `:B`, `:C`, `:E`). In parallel to this the analytical query is analyzed and the path-shaped BGPs are identified. These are used to look up the set of provenance identifiers in the context index, as described in Sect. 4. This set of provenance identifiers is: (`:A`, `:D`,`:C`). Second, the intersection of these two sets is found by finding common IRIs. We say it is minimum because it contains the minimum set of provenance identifiers that is needed to answer the provenance-enabled analytical query. The intersection of the two aforementioned sets is: (`:A`, `:C`). This means that only the information triples with these provenance identifers are needed to answer the provenance-enabled analytical query. The next two steps are identical to the second and third step in the Native Strategy (Sect. 3.2). Third, the IRIs of the provenance identifiers are added as FROM clauses to the analytical query to produce the updated analytical query. Fourth, the updated analytical query is executed over the RDF data cube.

6 Materialization Strategy

In this section, we propose an additional strategy that relies on materialization. Further, we explain how it is combined with the Context Index.

6.1 Materialization Strategy

In this strategy, we materialize the subset of the RDF data cube based on the provenance identifiers and execute the analytical query over the materialized subset. Obviously, this strategy allows for reuse of the materialized cube when identical queries are issued in the fugure. However, this is not the main benefit of this strategy. As shown in Sect. 3.3, standard triple stores are not able to handle queries with a large number of FROM clauses. By first materializing the sub-cube and then executing the analytical query over that, we split a complex query into two simple queries, which is easier for the triple stores to handle efficiently.

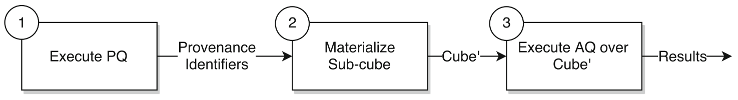


Fig. 7. Steps of the materialization strategy

Figure 7 illustrates the steps of this strategy. Similar to the native strategy, the first step is to execute the provenance query, to obtain a set of provenance identifiers. In our running example these are: `:A`, `:B`, `:C`, and `:E`. Second, a SPARQL CONSTRUCT query is created. This query creates the sub-cube, we call this `Cube'`. In the WHERE clause, we match all triple patterns and in the CONSTRUCT clause we insert the matching triple patterns but change the context value to `:MaterializedSubCube`. For each provenance identifier a FROM clause is created. Again we also add the `:Metadata` graph. This query creates a sub-cube that only contains information triples that satisfy the provenance query. In Query 4 the CONSTRUCT query for `Cube'` is shown. It creates a set of quads where the subject, predicate, and object remain unchanged but the context value is set to `:MaterializedSubCube`.

```
CONSTRUCT {
   GRAPH <:MaterializedSubCube> {?s ?p ?o }
}
FROM <:A>
FROM <:B>
FROM <:C>
FROM <:E>
FROM <:Metadata>
WHERE {
  ?s ?p ?o
}
```

Query 4. Query for materializing the sub-cube (`Cube'`)

The result of the CONSTRUCT query is shown in Table 4.

Table 4. The triples of the sub-cube (`Cube`′)

Subject	Predicate	Object	Context value
:observation1	:price	7	:MaterializedSubCube
:observation1	:date	:date31012016	:MaterializedSubCube
:date31012016	skos:broader	:january2016	:MaterializedSubCube
:january2016	:monthName	"January"	:MaterializedSubCube
:january2016	skos:broader	:2016	:MaterializedSubCube
:2016	:yearNumber	2016	:MaterializedSubCube
:observation2	:date	:date01022016	:MaterializedSubCube
:date01022016	skos:broader	:febuary2016	:MaterializedSubCube
:febuary2016	:monthName	"Febuary"	:MaterializedSubCube
:febuary2016	skos:broader	:2016	:MaterializedSubCube

Third, the analytical query is executed over the materialized sub-cube (`Cube′`). A single FROM clause is added with the identifier `:MaterializedSubCube`. When executed over `Cube′`, the query will evaluate to: "2016, 7".

6.2 Index-Based Materialization Strategy

The materialization strategy can further be improved by combining it with the context index. The steps are illustrated in Fig. 8.

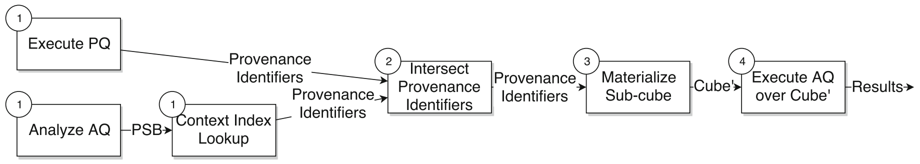


Fig. 8. Steps of the index-based materialization strategy

The first two steps are identical to the first two steps explained in Sect. 5. The last two steps are identical to step two and three from the materialization strategy explained above. Therefore, we will only give an example. First, the provenance query is executed yielding the set (`:A`, `:B`, `:C`, `:E`). The index is used and the set (`:A`, `:D`,`:C`) is output. Second, the intersection of the two sets are determined (`:A`, `:C`). Third, the CONSTRUCT query for the sub-cube is built and executed, resulting in the sub-cube `Cube′`, see Table 5. Fourth, the analytical query is executed over `Cube′`.

Table 5. Materialized cube where index is used

Subject	Predicate	Object	Context value
:observation1	:price	7	:MaterializedSubCube
:observation1	:date	:date31012016	:MaterializedSubCube
:date31012016	skos:broader	:january2016	:MaterializedSubCube
:january2016	skos:broader	:2016	:MaterializedSubCube
:2016	:yearNumber	2016	:MaterializedSubCube
:observation2	:date	:date01022016	:MaterializedSubCube
:date01022016	skos:broader	:febuary2016	:MaterializedSubCube
:febuary2016	skos:broader	:2016	:MaterializedSubCube

7 Experiments

To empirically evaluate our Native and Materialization strategies in combination with the context index, we conduct a series of experiments on the Jena TDB triple store. In order to test the scalability of our strategies we use the Star Schema Benchmark dataset and generate matching provenance data.

7.1 Experimental Environment

Hardware Platform. All experiments were run in a virtual machine with an AMD Opteron (TM) Processor 6274 (dual core), 16 GB of RAM, 500 GB harddisk, running Ubuntu 14.04.4 LTS.

SSB Dataset. For evaluating our strategies we use the Star Schema Benchmark [21]. This dataset refines the TPC-H benchmark to provide a realistic analytical benchmark [3]. The dataset has four dimensions: Supplier, Part, Customer, and Date, these describe the Lineorder observations. Table 6 shows the size of the datasets used in our experiments and the distribution of the triples.

Table 6. Overview of the four different sizes of cubes used in the experiments

Triples	Information triples	Provenance triples	Metadata triples	Unique provenance graphs
1,000,000	744,000	6,000	250,000	30,000
1,800,000	1,300,000	10,000	490,000	60,000
8,000,000	5,588,000	12,000	2,400,000	300,000
13,500,000	8,980,000	20,000	4,500,000	600,000

Provenance Generation. We generate provenance data for the SSB dataset. Each provenance graph consists of between 42 and 72 provenance triples with a varying number of entities, agents, and activities.

We use three different levels of granularity, unique, shared, and split. *Unique* is the finest level of granularity, each information triple is described by its own provenance graph. This means that each information triple has a unique provenance identifier. *Shared* is the coarsest level of granularity; all information triples share the same provenance graph, thus all information triples have the same provenance graph. *Split* has a varying level of granularity. When splitting we select an attribute, e.g., `:monthName`, which encodes the name of the month. Splitting on this attribute means that all dates from a given month, e.g., January, will share the same provenance graph. Depending on the number of distinct values for the attribute, the granularity may vary.

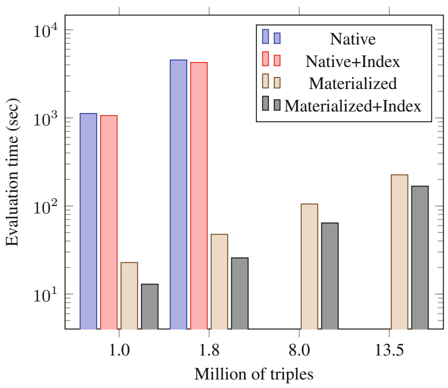
We vary the granularity level on each of the four dimensions and the observations. As default we choose to split the Lineorders by the attribute `:custkey`, this means that all Lineorders made by the same customer share the same provenance graph. All information triples from the same dimension share the same provenance graph, such that all information triples from the supplier dimension have the same provenance graph.

Workload. As workload we consider an analytical query and a set of synthetic provenance queries. The provenance queries we use return a slice of the RDF data cube, such that 10%, 20% ... 90% of the provenance identifiers are selected. This allows us to measure the performance of the strategies in a controlled manner. The slice is designed such that a valid RDF data cube will always be returned, thus ensuring that all provenance-enabled queries are valid over the RDF data cube.

7.2 Results

Analysis. The scalability and performance of our strategies are reported in Fig. 9. The x-axis shows the size of the RDF data cubes in millions of triples and the y-axis shows the execution time in seconds on a logarithmic scale. The provenance query in this experiment has 10% of the provenance identifiers. Note that the native and native+Index strategies are omitted for large cubes due to execution times exceeding one hour. Due to the poor scaling of the Native strategies, it is difficult to make any conclusions. We observe that the materialization strategies are faster than the native strategies, up to two orders of magnitude, if we consider the context index, then even more. We also observe that the Context Index reduces the execution time of both strategies. The index provides a constant improvement of 50% for the Materialization Strategy on all scales. For the Native Strategy this improvement is only 5%.

Figure 10 illustrates how the strategies scale when the number of provenance identifiers increase. On the x-axis is the number of FROM clauses and on the y-axis is the query evaluation time in seconds. In this experiment we use an RDF



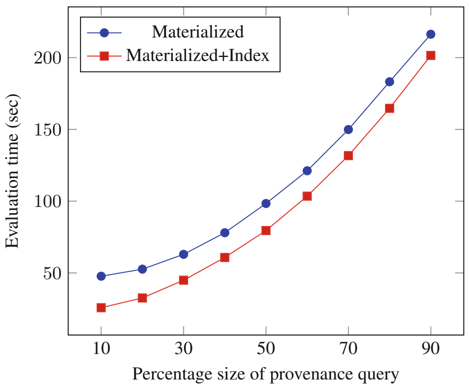


Fig. 9. Query execution over different sizes of RDF data cubes (log scale)

Fig. 10. Query execution with increasing number of FROM clauses

data cube of 1.8 million triples. Due to very large performance difference, the Native strategies have been omitted but they show a similar tendency. On the lower percentage, the Context Index gains a 100% speed up. As the percentage increases the performance of the index relatively decreases. Again this is because the index provides a constant improvement.

Discussion. As expected the Context Index reduces the number of FROM clauses and thereby reduces query time. Because of the synthetic nature of the provenance queries, we see a fixed performance improvement. However, because the index is constructed on load time the index has to be rebuilt when the data is updated. The Materialized strategy benefits the most from the Context Index. The index does not provide a substantial improvement for the Native Strategy, because it suffers from poor support in standard triples stores. The Materialization Strategy is up to 100 times faster than the Native Strategy. The price for this performance improvement lies in the additional storage cost for storing and updating the materialized cube. Reuse of the materialized cube was not part of the evaluation, but we expect that this would further improve this strategy. Combining the Context Index with the Materialized proves to be the best strategy.

8 Related Work

Several custom RDF provenance storage systems have been proposed in the literature. RDFProv [5] is based on a relational store for answering provenance queries over scientific workflows. By using mappings and translation algorithms on-the-fly SPARQL queries can be answered over the relational store. This approach is limited to workflow provenance queries. Our work addresses the problem of combining queries with provenance filters. In our work, we propose strategies for combining analytical queries and provenance queries.

Chebotko et al. [4] optimized Apache HBase to handle a high number of large provenance graphs, this is primarily done by using specialized indexes for select and join operations. Similar to PDFProv querying both the provenance and information triples are not considered.

TriplePROV [23] stores RDF data in molecule templates and custom physical storage models to enable fast retrieval of triples. When queries are evaluated a provenance polynomial is constructed that makes it possible to track the triples used for answering the query. While provenance polynomials are a powerful tool for some tasks, it is not possible to query the provenance polynomials using SPARQL. However, by using the context value to identify provenance triples, we do not suffer from this limitation. These custom RDF storage systems all have in common that their techniques cannot be applied to standard triple stores such as Jena TDB [2] or RDF4J Native [8]. The strategies we suggest are applicable in a storage independent manner.

In this work we build upon the query execution strategies proposed in [24]. However, it is not possible or sensible to directly apply these strategies to answer provenance-enabled analytical queries over RDF data cubes, because important metadata would be discarded. Additionally, unlike our strategies these are not applicable to standard triple stores.

The area of RDF data cubes is in constant growth. While there are several groups working towards how to best combine business intelligence and the Semantic Web [1,7,15,18,19], there is an agreement that this area carries a lot of potential for enabling web analytics. While some works on optimizing the execution of analytical queries [14,16] others work on adding spatial concepts [11]. Our work takes a step further by working towards how quality, security, and traceability can be enabled through provenance. In this paper, we address the problem of efficiently answering provenance-enabled analytical queries over RDF data cubes.

9 Conclusions and Future Work

In this paper, we work towards the problem of answering provenance-enabled analytical queries over RDF data cubes stored in standard triple stores. We observe that the main problem of evaluating such queries is handling SPARQL queries with a high number of FROM clauses. We propose the Native Strategy for answering provenance-enabled analytical queries and present two improvements. The first improvement is the novel Context Index that takes advantage of the cube structure to reduce the number of FROM clauses. The second improvement is the Materialization Strategy, it splits a provenance-enabled analytical query into a construction query and an analytical query, thus avoiding executing complex query over many graphs. Finally, we perform an empirical evaluation of our strategies using the Star Schema Benchmark augmented with provenance data. Our experimental evaluation confirms that the materialization strategy is efficient and scales for large RDF data cubes and the context index provides a consistent improvement for both strategies.

Building on the results of this work, we see a number of possible paths of future work. The current evaluation focuses on scale tests. However, using real-life data would help us further optimize the strategies and the index. Additionally, we would like to apply our strategies in a distributed setting, this which involves for a series of new challenges.

Acknowledgments. This research was partially funded by the Danish Council for Independent Research (DFF) under grant agreement No. DFF-4093-00301.

References

1. Abelló, A., Romero, O., Pedersen, T.B., Berlanga, R., Nebot, V., Aramburu, M.J., Simitsis, A.: Using semantic web technologies for exploratory OLAP: a survey. TKDE **27**(2), 571–588 (2015)
2. Apache software foundation. Jena TDB (3.1.0). https://jena.apache.org/
3. Bog, A., Plattner, H., Zeier, A.: A mixed transaction processing and operational reporting benchmark. ISF **13**(3), 321–335 (2011)
4. Chebotko, A., Abraham, J., Brazier, P., Piazza, A., Kashlev, A., Lu, S.: Storing, indexing and querying large provenance data sets as RDF graphs in apache HBase. In: Services, pp. 1–8 (2013)
5. Chebotko, A., Lu, S., Fei, X., Fotouhi, F.: RDFProv: a relational RDF store for querying and managing scientific workflow provenance. DKE **69**(8), 836–865 (2010)
6. Cyganiak, R., Reynolds, D.: The RDF data cube vocabulary. W3C recommendation, W3C, January 2014. http://www.w3.org/TR/2014/REC-vocab-data-cube-20140116/
7. Deb Nath, R.P., Hose, K., Pedersen, T.B.: Towards a programmable semantic extract-transform-load framework for semantic data warehouses. In: DOLAP, pp. 15–24 (2015)
8. Eclipse RDF4J. RDF4J (2.0.1). http://rdf4j.org/
9. Etcheverry, L., Vaisman, A., Zimányi, E.: Modeling and querying data warehouses on the semantic web using QB4OLAP. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 45–56. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10160-6_5
10. Flouris, G., Fundulaki, I., Pediaditis, P., Theoharis, Y., Christophides, V.: Coloring RDF triples to capture provenance. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 196–212. Springer, Heidelberg (2009). doi:10.1007/978-3-642-04930-9_13
11. Gür, N., Hose, K., Pedersen, T.B., Zimányi, E.: Modeling and querying spatial data warehouses on the semantic web. In: Qi, G., Kozaki, K., Pan, J.Z., Yu, S. (eds.) JIST 2015. LNCS, vol. 9544, pp. 3–22. Springer, Heidelberg (2016). doi:10.1007/978-3-319-31676-5_1
12. Harris, S., Seaborne, A.: SPARQL 1.1 query language. W3C recommendation, W3C, March 2013. http://www.w3.org/TR/2013/REC-sparql11-query-20130321/
13. Hartig, O., Thompson, B.: Foundations of an alternative approach to reification in RDF (2014). CoRR abs/1406.3399

14. Ibragimov, D., Hose, K., Pedersen, T.B., Zimányi, E.: Towards exploratory OLAP over linked open data - a case study. In: BIRTE, pp. 1–18 (2014)
15. Ibragimov, D., Hose, K., Pedersen, T.B., Zimányi, E.: Processing aggregate queries in a federation of SPARQL endpoints. In: Gandon, F., Sabou, M., Sack, H., d'Amato, C., Cudré-Mauroux, P., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9088, pp. 269–285. Springer, Heidelberg (2015). doi:10.1007/978-3-319-18818-8_17
16. Jakobsen, K.A., Andersen, A.B., Hose, K., Pedersen, T.B.: Optimizing RDF data cubes for efficient processing of analytical queries. In: COLD (2015)
17. Jensen, C.S., Pedersen, T.B., Thomsen, C.: Multidimensional Databases and Data Warehousing. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, San Rafael (2010)
18. Jovanovic, P., Romero, O., Simitsis, A., Abelló, A.: ORE: an iterative approach to the design and evolution of multi-dimensional schemas. In: DOLAP, pp. 1–8 (2012)
19. Laborie, S., Ravat, F., Song, J., Teste, O.: Combining business intelligence with semantic web: overview and challenges. In: INFORSID, pp. 99–114 (2015)
20. McGuinness, D., Lebo, T., Sahoo, S.: PROV-o: The PROV ontology. W3C recommendation, W3C, April 2013. http://www.w3.org/TR/2013/REC-prov-o-20130430/
21. O'Neil, P., O'Neil, B., Chen, X.: Star schema benchmark. Technical report, UMass/Boston, June 2019. http://www.cs.umb.edu/~poneil/StarSchemaB.PDF
22. Wang, H., Wu, T., Qi, G., Ruan, T.: On publishing Chinese linked open schema. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C., Vrandečić, D., Groth, P., Noy, N., Janowicz, K., Goble, C. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 293–308. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11964-9_19
23. Wylot, M., Cudre-Mauroux, P., Groth, P.: TripleProv: efficient processing of lineage queries in a native RDF store. In: WWW, pp. 455–466 (2014)
24. Wylot, M., Cudre-Mauroux, P., Groth, P.: Executing provenance-enabled queries over web data. In: WWW, pp. 1275–1285 (2015)

Data Analysis of Hierarchical Data for RDF Term Identification

Pieter Heyvaert(✉), Anastasia Dimou, Ruben Verborgh, and Erik Mannens

iMinds – IDLab – Ghent University, Ghent, Belgium
pheyvaer.heyvaert@ugent.be

Abstract. Generating Linked Data based on existing data sources requires the modeling of their information structure. This modeling needs the identification of potential entities, their attributes and the relationships between them and among entities. For databases this identification is not required, because a data schema is always available. However, for other data formats, such as hierarchical data, this is not always the case. Therefore, analysis of the data is required to support RDF term and data type identification. We introduce a tool that performs such an analysis on hierarchical data. It implements the algorithms, Daro and S-Daro, proposed in this paper. Based on our evaluation, we conclude that S-Daro offers a more scalable solution regarding run time, with respect to the dataset size, and provides more complete results.

1 Introduction

Data often originally resides in (semi-)structured formats. Tools [1,2] and mapping languages [3,4] allow to describe how Linked Data, via RDF triples, is generated based on the original data. Information structure modeling [5] (henceforth referred to as 'modeling') is required during the creation of these descriptions. This modeling includes the following tasks: (1) identify the candidate entities, their attributes and the relationships among these entities; (2) generate IRIs for the entities; and (3) define the data type of each attribute, if needed. For RDF these tasks align with RDF term identification. However, they can be fulfilled in different ways, and not every way results in the desired RDF triples. Additionally, current tools come short in fulfilling these tasks (semi-)automatically or do not provide the users with the required information to fulfill them manually. This information includes the data model, keys and data types. Though, this information can be found in the data schema, for hierarchical data the schema is not always available, nor always complete, as opposed to databases. Tools, such as XmlGrid[1] and FreeFormatter[2] for XML data, exist to generate these schemas.

The described research activities were funded by Ghent University, iMinds, the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research Flanders (FWO Flanders), and the European Union.

[1] http://xmlgrid.net/xml2xsd.html.

[2] http://www.freeformatter.com/xsd-generator.html.

However, they do not give all the aforementioned information, such as keys, and the data type information is not fine-grained enough when working with dirty data. Additionally, manually extracting this information is error-prone and time consuming, as the complete data source needs to be analyzed. In this paper, we introduce a tool[3] to obtain the required information of hierarchical data to address the three tasks. The tool implements two algorithms, Daro and S-Daro, to conduct the data analysis in a scalable way, as the dataset can become large. Based on theoretical analysis, the key discovery of Daro is not always complete, while for S-Daro it is. From our evaluation, we conclude that S-Daro has a better run time when the dataset size increases. The remainder of the paper is structured as follows. In Sect. 2, we discuss the related work. In Sect. 3, we explain, using an example, how the data analysis information can be used to fulfill the modeling tasks. In Sect. 4, we explain the two algorithms. In Sect. 5, we elaborate on the evaluation of the two algorithms. Finally, in Sect. 6, we conclude the paper.

2 Related Work

For XML the data model, keys and data types can be described via the XML schema. However, not in all cases is the schema available, nor complete. Tools exist that allow to generate a schema based on an XML file, such as XmlGrid and FreeFormatter. The same is applicable for JSON and the JSON schema [6]. The tool at http://jsonschema.net can be used to generate a JSON schema given a JSON input. These tools provide data model and data type information. However, the latter lacks detail as a single data type is given when certain data fractions might have different data types. Furthermore, these tools lack key discovery.

3 Example: RDF Term Identification Using Data Analysis

In most cases Linked Data is interpreted as a graph structure, as done by RDF, where the nodes (representing the entities and their attributes) are linked using edges (representing the relationships). Using the XML example in Listing 1.1, we execute the three aforementioned tasks (see Sect. 1) of the modeling process to identify the RDF terms, taking into account which information from the data analysis is used to fulfill each task. We aim to give one possible set of declarative statements of how these terms are generated, using the mapping language RML [4], based on the data model, keys and data types. Subsequently, these statements are used to generate RDF triples.

[3] https://github.com/RMLio/data-analysis-cli; available under the MIT license.

```
<person>
  <firstName>John</firstName>
  <lastName>Doe</lastName>
   <car id="0695-77968-33844">
     <brand>Peugeot</brand>
     <purchDate>12-01-2015</purchDate>
   </car>
</person>
<person>
  <firstName>Jane</firstName>
    <lastName>Doe</lastName>
    <car id="0695-77968-33897">
      <brand>Peugeot</brand>
      <purchDate>16-01-2015</purchDate>
    </car>
  </person>
</persons>
```

Listing 1.1. XML example with person metadata (`http://ex.com/persons.xml`)

Task 1: Identify Entities, Attributes and Relationships Using Data Model. RDF term identification is required to find the appropriate IRIs, blank nodes, and literals. It is supported by using the *data model*. The tree structure of these data sources allows determining possible entities, their literals and relationships by looking at the XML elements and XML attributes: parent elements (i.e., elements with child elements) are identified as entities (IRIs or blank nodes), and leaf elements (i.e., elements with no child elements) and attributes as the entities corresponding IRIs' or blank nodes' literals. Additionally, if a parent element has a parent element as a child, there exists a relationship between the corresponding entities. In the example, the parent elements are `<person>` and `<car>`. This leads to:

```
1  @prefix rr: <http://www.w3.org/ns/r2rml#> . @prefix rml:
2  <http://semweb.mmlab.be/ns/rml#> . @prefix xsd:
3  <http://www.w3.org/2001/XMLSchema#> .
4
5  <#PersonMapping>
6    rml:logicalSource [
7      rml:source "http://ex.com/persons.xml";
8      rml:referenceFormulation ql:XPath;
9      rml:iterator "/persons/person" ] .
10 <#CarMapping>
11   rml:logicalSource [
12     rml:source "http://ex.com/persons.xml";
13     rml:referenceFormulation ql:XPath;
14     rml:iterator "/persons/person/car" ] .
```

For each parent elements there is a triples map (lines 5 and 10). Each map requires a logical source, which includes the path to the parent element (lines 9 and 14). The leaf elements of `<person>` are `<firstName>` and `<lastName>`. Consequently, they can be identified as literals of the parent element's IRI or blank node, resulting in:

```
1  <#PersonMapping> rr:predicateObjectMap <#PreObjMapFirstName> .
2  <#PreObjMapFirstName> rr:objectMap [ rml:reference "firstName" ] .
3  <#PersonMapping> rr:predicateObjectMap <#PreObjMapLastName> .
4  <#PreObjMapLastName> rr:objectMap [ rml:reference "lastName" ] .
```

A predicate object map, with an object map, is added to the triples map for the `<firstName>` (lines 1 and 2) and `<lastName>` (lines 3 and 4). The same is the case for the parent element `<car>` and its leaf elements `<brand>`, `<purchDate>`, and the attribute `@id`, resulting in:

```
<#CarMapping> rr:predicateObjectMap <#PreObjMapBrand> .
<#PreObjMapBrand> rr:objectMap [ rml:reference "brand" ] .
<#CarMapping> rr:predicateObjectMap <#PreObjMapID> .
<#PreObjMapID> rr:objectMap [ rml:reference "@id" ] .
<#CarMapping> rr:predicateObjectMap <#PreObjMapPurchaseDate> .
<#PreObjMapPurchaseDate> rr:objectMap <#ObjMapPurchaseDate> .
<#ObjMapPurchaseDate> rml:reference "purchDate" .
```

Furthermore, we conclude that there is a relationship between these two entities, because `<car>` is a child element of `<person>`. This is done by adding a new predicate object map to the triples map for `<person>`, together with a parent triples map that refers to the triples map for `<car>`. This results in:

```
<#PersonMapping> rr:predicateObjectMap <#PreObjMapCar> .
<#PreObjMapCar> rr:objectMap [ rr:parentTriplesMap <#CarMapping> ] ] .
```

Task 2: Generate IRIs Using Keys. In most cases the IRIs have a specific structure, and certain elements of this structure are depended on the data. Additionally, each IRI has to represent at most one entity. This can be accomplished by using *keys* as part of the IRIs. Keys are data fractions that have a unique value for each entity in the original data. In the example, a key identified for the persons is `firstName`. A key identified for the cars is `@id`. This results in:

```
<#PersonMapping> rr:subjectMap [ rr:template "http://ex.com/person/{firstName}] .
<#CarMapping> rr:subjectMap [ rr:template"http://ex.com/car/{@id} ] .
```

A subject map is added to the triples map of each element together with a possible template to generate IRIs using the specified keys.

Task 3: Define Data Types. The *data types* of all values are string with exception of the purchase date (`<purchDate>`; lines 7 and 15), which is a date. This results in the following statement, where date data type is added to the object map corresponding with `<purchDate>`.

```
<#ObjMapPurchaseDate> rr:datatype xsd:date .
```

Subsequently, these statements can be used directly or via a tool, e.g., the RMLEditor [1], to provide the predicates to generate the desired triples.

4 Algorithms

Preliminaries. We structure hierarchical data using a *tree*, in which each node has a set of *properties*, regardless of the data format, e.g., XML or JSON. Each property points to one or more children or data values. For the example in Listing 1.1, the properties of `<person>` are given by the paths `firstName`, `lastName` and `car`. N is the set of all nodes in the tree. $\mathcal{P}$ is the set of all multi-level properties of a node. *Multi-level properties* are the properties of a node including all properties of that node's childnode trees. For the example in Listing 1.1, the multi-level properties of `<person>` are given by the paths `firstName`, `lastName`, `car/brand`, `car/id`, `car/purchaseDate`. P is used for a set of properties where $P \subseteq \mathcal{P}$. The value v of a node n for a certain (multi-level) property p is defined as $(n, p, v) \in N \times P \times V$, where V represents all values. Two nodes are distinguishable from each other given a set of properties if for at least one property the values of both nodes are not the same. This is formally given in Eq. 1.

$$\begin{aligned} dist(n, n', P) = \exists p \in P \ \wedge \ &\exists (n, p, v) \in N \times P \times V \ \wedge \\ &\exists (n', p, v') \in N \times P \times V : v \neq v' \end{aligned} \tag{1}$$

Daro. The first algorithm is based on the ROCKER algorithm, which uses a refinement operator for the discovery of keys, proposed by Soru et al. [7]. The operator refines which keys are worth checking, opposed to checking all possible keys. Originally, it was applied for key discovery on RDF datasets. Our version supports hierarchical data sources, and is called 'Data Analysis using the ROCKER Operator' (Daro). It uses a *scoring function* that gives the ratio of the number of nodes that is distinguishable given a set of properties over the total number of nodes ($score(P)$ in Eq. 2). P is a key if $score(P) = 1$, because that means that all nodes are uniquely identifiable using P. Additionally, the function $sortByScore(P)$ returns the properties of P ascendantly ordered using their score, i.e., $\forall p_i, p_j \in P : i \leq j \implies score(p_i) \geq score(p_j)$.

$$score(P) = \frac{|\{n \in N \mid \forall n' \in N : n \neq n' \Rightarrow dist(n, n', P)\}|}{|N|} \tag{2}$$

The refinement operator ($\rho(P)$ in Eq. 3) defines which sets of properties need to be checked next given a set of (previously checked) properties. It requires the properties of $\mathcal{P}$ to be ordered using $sortByScore(\mathcal{P})$.

$$\rho(P) = \begin{cases} \mathcal{P} \text{ if } P = \emptyset, \\ \{P \cup \{p_1\}, \ldots, P \cup \{p_i\}\} : & p'_0 \in sortByScore(P) \ \wedge \\ & (\exists p_j \in \mathcal{P} : p'_0 = p_j) \ \wedge (p_i \in \mathcal{P} : i < j) \end{cases} \tag{3}$$

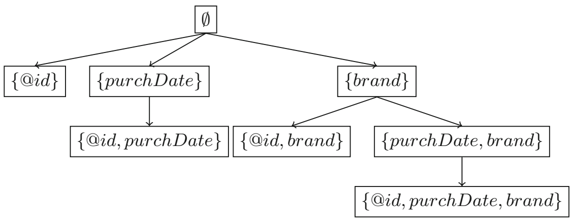


Fig. 1. Complete refinement operator tree for `<car>` of Listing 1.1

We explain the operator using `<car>` from Listing 1.1. In Fig. 1 you can see the complete refinement tree for the child elements and attributes of `<car>`, i.e.,

$\mathcal{P} = \{$*@id,purchDate,brand*$\}$. First, we start with an empty set of sets ($= \emptyset$), because we do not have a set of properties. Applying the operator on the empty set ($= \rho(\emptyset)$) results in the following sets of properties: $\{$*@id*$\}$, $\{$*purchDate*$\}$ and $\{$*brand*$\}$. This is visualized in the second level of the tree. The sets on the third and fourth level of the tree are generated by applying refinement operator on each element on the second and third level, respectively. The theorems and proofs regarding the operator are given in the original work by Soru et al. [7]. Additionally, we created a function to generate the data model and a method that analyzes the values of the properties in order to provide the data types.

Algorithm 1. Daro

```
1:  nodes ← xml.query(nodePath)
2:  foundKeys ← [ ]
3:  if ¬nodes.isEmpty() then
4:      paths ← getPaths(nodes)
5:      buildTreeAndIndex(nodes,paths)
6:      model ← getModel(paths)
7:      paths ← sortByScore(paths)
8:      if score(paths) = 1 then
9:          q ← new PriorityQueue()
10:         q.add(∅, 0)
11:         while ¬q.isEmpty() do
12:             P ← q.pop()
13:             P' ← ρ(P)
14:             for p in P' do
15:                 s ← score(p)
16:                 if s = 1 then
17:                     foundKeys.add(p)
18:                 else
19:                     q.add(p, s)
20:                 end if
21:             end for
22:         end while
23:     end if
24: end if
```

Algorithm 2. S-Daro

```
1:  nodes ← xml.query(nodePath)
2:  trees ← [ ]
3:  if ¬nodes.isEmpty() then
4:      paths ← getPaths(nodes)
5:      model ← getModel(paths)
6:      possibleKeys ← generateKeys(paths)
7:      for node in nodes do
8:          for key in possibleKeys do
9:              if ¬key.parent.valid() then
10:                 groups ← [ ]
11:                 for path in key do
12:                     value ← node.query(path)
13:                     analyze(value)
14:                     tree ← trees.search(path)
15:                     group ← tree.search(value)
16:                     groups.add(group)
17:                 end for
18:                 if groups.hasDupNode() then
19:                     key.valid(false)
20:                 end if
21:             end if
22:         end for
23:     end for
24: end if
```

The pseudo-code[4] of the algorithm can be found in Algorithm 1. The properties and the nodes are used to build a search tree of the nodes and an index over the values, during which also the data types are determined (line 5). The data model is generated using the properties (line 6). If the score of the set of all properties is 1, then all nodes are unique when taking into account all properties (line 8). Only when this is true, we continue the key discovery.

Keys that are supersets of already found keys will not be returned, because the algorithm only adds set of properties to the queue again when they are not keys. Therefore, the number of found keys might be smaller than the total number of keys. It depends on the data which keys will be found and which keys not, as the refinement operator is based on the scores of the properties. These scores are based on the actual values of the data. However, the algorithm always returns all keys consisting of one property, together with all the keys that contain

[4] For brevity, we did not include the code that allows users to determine the data model, keys, and data types separately.

a property that on itself is not a key. The reason is that the empty set (added on line 10) results in checking all possible keys consisting of one property, and properties that are not a key are used to generate new possible keys using the operator (line 19) until a key is found or none can be found.

Key discovery is the most expensive part of the analysis, because the different elements of the data have to be compared. The other elements of the data analysis only require a single pass over the data. However, they are done during the key discovery, because it is needed to iterate over the data in any case.

S-Daro. The second algorithm is called 'Scalable Data Analysis using the ROCKER Operator' (S-Daro). While building upon the ROCKER algorithm, it builds up an index for each property containing all possible values present in that dataset together with the nodes that have this value. Additionally, it does not use the scoring function to lower the run time. Algorithm 2 contains the pseudo code. Using the refinement operator of the previous algorithm, we determine all the possible sets of properties (line 6). They are all possible keys. Additionally, for each set we remember on which other set it was based, if applicable. In the refinement operator tree, this is the set on the lower level to which it connects. We call this set the parent set. A set is only evaluated if the parent set is not valid (i.e., not a key; line 9). If the parent set is valid than the current set stays valid, because the properties of the current set are a superset of the properties of the parent set [8]. If for all properties with those values there is one node (besides the current node) that is present (line 18), than the set is not a key. The current node and that specific node are indistinguishable using these properties.

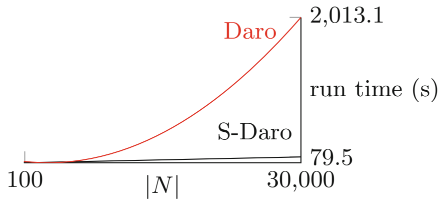
As opposed to Daro, this algorithm returns all keys, because, besides the keys that were marked valid during checking, also the keys that have a valid parent key are valid keys. Like for Daro, key discovery is the most expensive part of the analysis, because the different elements of the data have to be compared.

5 Evaluation

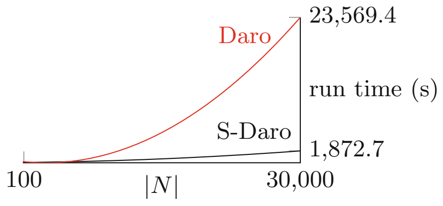
In this section, we elaborate on the evaluation conducted on Daro and S-Daro. The criterion of the evaluation is the run time, because the algorithms are only useful for practical purposes if they finish within a reasonable amount of time. We have evaluated[5] both algorithms using 4 sets of 240 artificially generated files[6]. These files have between 100 and 30,000 nodes, and have between 6 and 13 properties. Their data is about people and their jobs. In Fig. 2a and b plots of the fitted functions of the run times for both algorithms can be found for 6 and 13 properties, respectively. We see that S-Daro outperforms Daro, when the number of nodes becomes larger. The functions are polynomial of the second

[5] All experiments were conducted on a 64-bit Ubuntu 14.04 machine with 128 GB of RAM and a 24-core 2.40 GHz CPU. Each algorithm was run in a Docker container and was able to use at any moment a maximum of 8 GB of RAM and 1 CPU core.

[6] http://rml.io/data/ISWC16/ph/files.



(a) 6 properties/node



(b) 13 properties/node

Fig. 2. Daro vs S-Daro

degree for both algorithms. Nevertheless, the function for S-Daro rises slower than for Daro, because the coefficient of the quadratic number of nodes of S-Daro remains small when compared to the coefficient for Daro. However, the coefficient for S-Daro can still be fitted to an exponential function. The reason for this is the exponential growth of possible keys in function of the total number of properties [7]. Therefore, when the number of properties becomes too large even S-Daro might not be able to provide a result within a desired time frame.

6 Conclusion

Our tool implements the two algorithms Daro and S-Daro with support for XML data sources. However, they are applicable to other formats of hierarchical data, such as JSON. Although both algorithms benefit from the refinement operator regarding their run times, the evaluation showed that S-Daro outperforms Daro when the number of nodes becomes larger. Furthermore, the incompleteness of the key discovery of Daro drives the choice towards S-Daro when all keys are required. However, certain use cases might find the results of Daro sufficient.

References

1. Heyvaert, P., Dimou, A., Herregodts, A.-L., Verborgh, R., Schuurman, D., Mannens, E., Walle, R.: RMLEditor: a graph-based mapping editor for linked data mappings. In: Sack, H., Blomqvist, E., d'Aquin, M., Ghidini, C., Ponzetto, S.P., Lange, C. (eds.) ESWC 2016. LNCS, vol. 9678, pp. 709–723. Springer, Heidelberg (2016). doi:10.1007/978-3-319-34129-3_43
2. Pinkel, C., Schwarte, A., Trame, J., Nikolov, A., Bastinos, A.S., Zeuch, T.: DataOps: seamless end-to-end anything-to-RDF data integration. In: Gandon, F., Guéret, C., Villata, S., Breslin, J., Faron-Zucker, C., Zimmermann, A. (eds.) ESWC 2015. LNCS, vol. 9341, pp. 123–127. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25639-9_24
3. Das, S., Sundara, S., Cyganiak, R., R2RML: RDB to RDF mapping language. Working group recommendation, W3C. http://www.w3.org/TR/r2rml/
4. Dimou, A., Sande, M.V., Colpaert, P., Verborgh, R., Mannens, E., Rik Van de Walle, R.M.L.: A generic language for integrated rdf mappings of heterogeneous data. In: Workshop on Linked Data on the Web (2014)

5. Chen, P.P.-S.: The entity-relationship model - toward a unified view of data. ACM Trans. Database Syst. (TODS) **1**(1), 9–36 (1976)
6. Galiegue, F., Zyp, K., Json schema: core definitions and terminology. In: Internet Engineering Task Force (IETF) (2013)
7. Soru, T., Marx, E., Ngonga Ngomo, A.-C.: ROCKER - a refinement operator for key discovery. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1025–1033. International World Wide Web Conferences Steering Committee (2015)
8. Pernelle, N., Saïs, F., Symeonidou, D.: An automatic key discovery approach for data linking. Web Semant. Sci. Serv. Agents WWW **23**, 16–30 (2013)

PIWD: A Plugin-Based Framework for Well-Designed SPARQL

Xiaowang Zhang[1,3,4], Zhenyu Song[1,3], Zhiyong Feng[2,3(✉)], and Xin Wang[1,3]

[1] School of Computer Science and Technology, Tianjin University, Tianjin, China
[2] School of Computer Software, Tianjin University, Tianjin 300350, China
zyfeng@tju.edu.cn
[3] Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China
[4] Key Laboratory of Computer Network and Information Integration, Southeast University, Ministry of Education, Nanjing 211189, China

Abstract. In the real world datasets (e.g., DBpedia query log), queries built on well-designed patterns containing only AND and OPT operators (for short, WDAO-patterns) account for a large proportion among all SPARQL queries. In this paper, we present a plugin-based framework for all SELECT queries built on WDAO-patterns, named PIWD. The framework is based on a parse tree called *well-designed AND-OPT tree* (for short, WDAO-tree) whose leaves are basic graph patterns (BGP) and inner nodes are the OPT operators. We prove that for any WDAO-pattern, its parse tree can be equivalently transformed into a WDAO-tree. Based on the proposed framework, we can employ any query engine to evaluate BGP for evaluating queries built on WDAO-patterns in a convenient way. Theoretically, we can reduce the query evaluation of WDAO-patterns to subgraph homomorphism as well as BGP since the query evaluation of BGP is equivalent to subgraph homomorphism. Finally, our preliminary experiments on gStore and RDF-3X show that PIWD can answer all queries built on WDAO-patterns effectively and efficiently.

Keywords: SPARQL · BGP · Well-designed patterns · Subgraph homomorphism

1 Introduction

Resource Description Framework (RDF) [23] is the standard data model in the semantic web. RDF describes the relationship of entities or resources using directed labelling graph. RDF has a broad range of applications in the semantic web, social network, bio-informatics, geographical data, etc. [3,28,29]. The standard query language for RDF graphs is SPARQL [19]. Though SPARQL is powerful to express queries over RDF graphs [2], generally, the query evaluation of the full SPARQL is PSPACE-complete [18].

Currently, there are some popular query engines for supporting the full SPARQL such as Jena [7] and Sesame [6]. However, they become not highly efficient when they handle some large RDF datasets [33,34]. Currently, gStore [33,34] and RDF-3X [16] can highly efficiently query large datasets. But gStore and RDF-3X merely provide querying services of BGP. Therefore, it is very necessary to develop a query engine with supporting more expressive queries for large datasets.

Since the OPT operator is the least conventional operator among SPARQL operators [30], it is interesting to investigate those patterns extending BGP with the OPT operator. Let us take a look at the following example.

An RDF example in Table 1 describes the entities of bloggers and blogs. The relationship between a blogger and a blog is revealed in the property of *foaf:maker*. Both blogger and blog have some properties to describe themselves. Triples can be modeled as a directed graph substantially.

Table 1. bloggers.rdf

Subject	Predict	Object
id1	foaf:name	Jon Foobar
id1	rdf:type	foaf:Agent
id1	foaf:weblog	foobar.xx/blog
foobar.xx/blog	dc:title	title
foobar.xx/blog	rdfs:seeAlso	foobar.xx/blog.rdf
foobar.xx/blog.rdf	foaf:maker	id1
foobar.xx/blog.rdf	rdf:type	rss:channel

Example 1. Consider the RDF dataset G storing information in Table 1. Given a BGP $Q = ((?x, \mathit{foaf{:}maker}, ?y) \text{ AND } (?z, \mathit{foaf{:}name}, ?u))$, its evaluation over G is as follows:

$[\![Q]\!]_G =$

?x	?y	?z	?u
foobar.xx/blog.rdf	id1	id1	Jon Foobar

Consider a new pattern Q_1 obtained from Q by adding the OPT operator in the following way:
$Q_1 = (((?x, \mathit{foaf{:}maker}, ?y) \text{ OPT } (?y, \mathit{rdf{:}type}, ?v)) \text{ AND } (?z, \mathit{foaf{:}name}, ?u))$, the evaluation of Q_1 over G is as follows:

$[\![Q_1]\!]_G =$

?x	?y	?v	?z	?u
foobar.xx/blog.rdf	id1	foaf:Agent	id1	Jon Foobar

Consider another pattern $Q_2 = (((?x, \mathit{foaf{:}maker}, ?y) \text{ OPT } (?y, \mathit{rdf{:}type}, ?z)) \text{ AND } (?z, \mathit{foaf{:}name}, ?u))$, the evaluation of Q_2 over G is the empty set, i.e., $[\![Q_2]\!]_G = \emptyset$.

In the above example, Q_1 is a well-designed pattern while Q_2 is not a well-designed pattern [18].

In fact, we investigate that queries built on well-designed patterns are very popular in a real world. For example, in LSQ [20], a Linked Dataset describing SPARQL queries extracted from the logs of four prominent public SPARQL endpoints containing more than one million available queries shown in Table 2, queries built on well-designed patterns are over 70% [9, 22].

Table 2. SPARQL logs source in LSQ

Dataset	Date	Triple number
DBpedia	30/04/2010 to 20/07/2010	232,000,000
Linked Geo Data (LGD)	24/11/2010 to 06/07/2011	1,000,000,000
Semantic Web Dog Food (SWDF)	16/05/2014 to 12/11/2014	300,000
British Museum (BM)	08/11/2014 to 01/12/2014	1,400,000

Furthermore, queries with well-designed AND-OPT patterns (for short, WDAO-patterns) are over 99% among all queries with well-designed patterns in LSQ [9, 22]. In short, the fragment of WDAO-patterns is a natural extension of BGP in our real world. Therefore, we mainly discuss WDAO-patterns in this paper.

In this paper, we present a plugin-based framework for all SELECT queries built on WDAO-patterns, named PIWD. Within this framework, we can employ any query engine evaluating BGP for evaluating queries built on WDAO-patterns in a convenient way. The main contributions of this paper can be summarized as follows:

- We present a parse tree named *well-designed AND-OPT tree* (for short, WDAO-tree), whose leaves are BGP and all inner nodes are the OPT operator and then prove that for any WDAO-pattern, it can be translated into a WDAO-tree.
- We propose a plugin-based framework named *PIWD* for query evaluation of queries built on WDAO-patterns based on WDAO-tree. Within this framework, a query could be evaluated in the following three steps: (1) translating that query into a WDAO tree T; (2) evaluating all leaves of T via query engines of BGP; and (3) joining all solutions of children to obtain solutions of their parent up to the root.
- We implement the proposed framework PIWD by employing gStore and RDF-3X and evaluate the experiments on LUBM.

The rest of this paper is organized as follows: Sect. 2 briefly introduces the SPARQL, conception of well-designed patterns and OPT normal form. Section 3 defines the well-designed and-opt tree to capture WDAO-patterns. Section 4 presents PIWD and Sect. 5 evaluates experimental results. Section 6 summarizes our related works. Finally, Sect. 7 summarizes this paper.

2 Preliminaries

In this section, we introduce RDF and SPARQL patterns, well-designed patterns, and OPT normal form [18].

2.1 RDF

Let I, B and L be infinite sets of *IRIs*, *blank nodes* and *literals*, respectively. These three sets are pairwise disjoint. We denote the union $I \cup B \cup L$ by U, and elements of $I \cup L$ will be referred to as *constants*.

A triple $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an *RDF triple*. A *basic graph pattern* (BGP) is a set of triple patterns.

2.2 Semantics of SPARQL Patterns

The semantics of patterns is defined in terms of sets of so-called *mappings*, which are simply total functions $\mu \colon S \to U$ on some finite set S of variables. We denote the domain S of μ by $\mathrm{dom}(\mu)$.

Now given a graph G and a pattern P, we define the semantics of P on G, denoted by $[\![P]\!]_G$, as a set of mappings, in the following manner.

– If P is a triple pattern (u, v, w), then

$$[\![P]\!]_G := \{\mu \colon \{u, v, w\} \cap V \to U \mid (\mu(u), \mu(v), \mu(w)) \in G\}.$$

Here, for any mapping μ and any constant $c \in I \cup L$, we agree that $\mu(c)$ equals c itself. In other words, mappings are extended to constants according to the identity mapping.

– If P is of the form P_1 UNION P_2, then $[\![P]\!]_G := [\![P_1]\!]_G \cup [\![P_2]\!]_G$.

– If P is of the form P_1 AND P_2, then $[\![P]\!]_G := [\![P_1]\!]_G \bowtie [\![P_2]\!]_G$, where, for any two sets of mappings Ω_1 and Ω_2, we define

$$\Omega_1 \bowtie \Omega = \{\mu_1 \cup \mu_2 \mid \mu_1 \in \Omega_1 \text{ and } \mu_2 \in \Omega_2 \text{ and } \mu_1 \sim \mu_2\}.$$

Here, two mappings μ_1 and μ_2 are called *compatible*, denoted by $\mu_1 \sim \mu_2$, if they agree on the intersection of their domains, i.e., if for every variable $?x \in \mathrm{dom}(\mu_1) \cap \mathrm{dom}(\mu_2)$, we have $\mu_1(?x) = \mu_2(?x)$. Note that when μ_1 and μ_2 are compatible, their union $\mu_1 \cup \mu_2$ is a well-defined mapping; this property is used in the formal definition above.

– If P is of the form P_1 OPT P_2, then

$$[\![P]\!]_G := ([\![P_1]\!]_G \bowtie [\![P_2]\!]_G) \cup ([\![P_1]\!]_G \setminus [\![P_2]\!]_G),$$

where, for any two sets of mappings Ω_1 and Ω_2, we define

$$\Omega_1 \setminus \Omega_2 = \{\mu_1 \in \Omega_1 \mid \neg\exists \mu_2 \in \Omega_2 : \mu_1 \sim \mu_2\}.$$

– If P is of the form $\text{SELECT}_S(P_1)$, then $[\![P]\!]_G = \{\mu|_{S\cap \text{dom}(\mu)} \mid \mu \in [\![P_1]\!]_G\}$, where $f|_X$ denotes the standard mathematical notion of restriction of a function f to a subset X of its domain.
– Finally, if P is of the form P_1 FILTER C, then $[\![P]\!]_G := \{\mu \in [\![P_1]\!]_G \mid \mu(C) = \mathit{true}\}$.
  Here, for any mapping μ and constraint C, the evaluation of C on μ, denoted by $\mu(C)$, is defined in terms of a three-valued logic with truth values *true*, *false*, and *error*. Recall that C is a boolean combination of atomic constraints.
  For a bound constraint bound(?x), we define:

$$\mu(\text{bound}(?x)) = \begin{cases} \mathit{true} & \text{if } ?x \in \text{dom}(\mu); \\ \mathit{false} & \text{otherwise.} \end{cases}$$

  For an equality constraint $?x = ?y$, we define:

$$\mu(?x = ?y) = \begin{cases} \mathit{true} & \text{if } ?x, ?y \in \text{dom}(\mu) \text{ and } \mu(?x) = \mu(?y); \\ \mathit{false} & \text{if } ?x, ?y \in \text{dom}(\mu) \text{ and } \mu(?x) \neq \mu(?y); \\ \mathit{error} & \text{otherwise.} \end{cases}$$

  Thus, when $?x$ and $?y$ do not both belong to dom(μ), the equality constraint evaluates to *error*. Similarly, for a constant-equality constraint $?x = c$, we define:

$$\mu(?x = c) = \begin{cases} \mathit{true} & \text{if } ?x \in \text{dom}(\mu) \text{ and } \mu(?x) = c; \\ \mathit{false} & \text{if } ?x \in \text{dom}(\mu) \text{ and } \mu(?x) \neq c; \\ \mathit{error} & \text{otherwise.} \end{cases}$$

  A boolean combination is then evaluated using the truth tables given in Table 3.

Table 3. Truth tables for the three-valued semantics.

p	q	$p \wedge q$	$p \vee q$
true	*true*	*true*	*true*
true	*false*	*false*	*true*
true	*error*	*error*	*true*
false	*true*	*false*	*true*
false	*false*	*false*	*false*
false	*error*	*false*	*error*
error	*true*	*error*	*true*
error	*false*	*false*	*error*
error	*error*	*error*	*error*

p	$\neg p$
true	*false*
false	*true*
error	*error*

2.3 Well-Designed Pattern

A UNION-*free* pattern P is *well-designed* if the followings hold:

– P is safe;

– for every subpattern Q of form $(Q_1 \text{ OPT } Q_2)$ of P and for every variable $?x$ occurring in P, the following condition holds:

$$\text{If } ?x \text{ occurs both inside } Q_2 \text{ and outside } Q, \text{ then it also occurs in } Q_1.$$

Consider the definition of well-designed patterns, some conceptions can be explained as follows:

Remark 1. In the fragment of and-opt patterns, we exclude FILTER and UNION operators and it contains only AND and OPT operators at most. It is obvious that *and-opt* pattern must be UNION-*free* and safe.

We can conclude that WDAO-patterns are decided by variables in subpattern.

– **UNION-*free* Pattern**: P is UNION-free if P is constructed by using only operators AND, OPT, and FILTER. Every graph pattern P is equivalent to a pattern of the form denoted by $(P_1 \text{ UNION } P_2 \text{ UNION} \cdots \text{UNION } P_n)$. Each P_i $(1 \leq i \leq n)$ is UNION-free.
– **Safe**: If the form of (P FILTER R) holds the condition of $var(R) \subseteq var(P)$, then it is safe.

Note that the OPT operator provides really optional left-outer join due to the weak monotonicity [18]. A SPARQL pattern P is said to be weakly monotone if for every pair of RDF graphs G_1, G_2 such that $G_1 \subseteq G_2$, it holds that $[\![P]\!]_{G_1} \sqsubseteq [\![P]\!]_{G_2}$. In other words, we assume μ_1 represents $[\![P]\!]_{G_1}$, and μ_2 represents $[\![P]\!]_{G_2}$. Then there exists μ' such that $\mu_2 = \mu_1 \cup \mu'$. Weakly monotone is an important property to characterize the satisfiability of SPARQL [31]. For instance, consider the pattern Q_1 in Sect. 1, $(?y, \textit{rdf:type}, ?v)$ are really optional.

2.4 OPT Normal Form

A UNION-free pattern P is in *OPT normal form* [18] if P meets one of the following two conditions:

– P is constructed by using only the AND and FILTER operators;
– $P = (P_1 \text{ OPT } P_2)$ where P_1 and P_2 patterns are in OPT normal form.

For instance, the pattern Q aforementioned in Sect. 1 is in OPT normal form. However, consider the pattern $(((?x, p, ?y) \text{ OPT } (?x, q, ?z)) \text{ AND } (?x, r, ?z))$ is not in OPT normal form.

3 Well-Designed And-Opt Tree

In this section, we propose the conception of the well-designed and-opt tree (WDAO-tree), any WDAO-pattern can be seen as an WDAO-tree.

3.1 WDAO-tree Structure

Definition 1 (WDAO-tree). *Let P be a well-designed pattern in OPT normal form. A well-designed tree T based on P is a redesigned parse tree, which can be defined as follows:*

- *All inner nodes in T are labeled by the* OPT *operator and leaves are labeled by BGP.*
- *For each subpattern $(P_1 \text{ OPT } P_2)$ of P, the well-designed tree T_1 of P_1 and the well-designed tree T_2 of P_2 have the same parent node.*

For instance, consider a WDAO-pattern P[1]

$$P = (((p_1 \text{ AND } p_3) \text{ OPT}_2 \ p_2) \text{ OPT}_1 \\ ((p_4 \text{ OPT}_4 \ p_5) \text{ OPT}_5 \ (p_6 \text{ OPT}_6 \ p_7))).$$

The WDAO-tree T is shown in Fig. 1. As shown in this example, BGP - $(p_1 \text{ AND } p_3)$ is the exact matching in P, which corresponds to the non-optional pattern. Besides, in WDAO-tree, it is the leftmost leaf in T. We can conclude that the leftmost node in WDAO-tree means the exact matching in well-designed SPARQL query pattern.

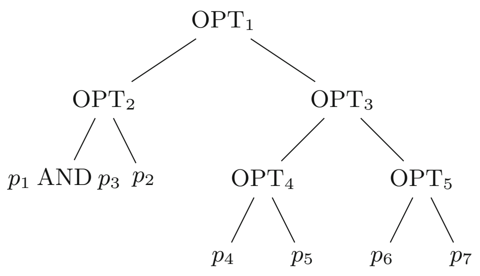


Fig. 1. WDAO-tree

3.2 Rewritting Rules over WDAO-tree

As described in Sect. 1, WDAO-tree does not contain any OPT operator in its leaves. In this sense, patterns as the form of Q_1 in Sect. 1 cannot be transformed into WDAO-tree since it is not OPT normal form.

Proposition 1. *[18, Theorem 4.11] For every UNION-free well-designed pattern P, there exists a pattern Q in OPT normal form such that P and Q are equivalent.*

[1] We give each OPT operator a subscript to differentiate them so that readers understand clearly.

In the proof of Proposition 1, we apply three rewriting rules based on the following equations: let P, Q, R be patterns and C a constraint,

- $(P \text{ OPT } R) \text{ AND } Q \equiv (P \text{ AND } Q) \text{ OPT } R$;
- $P \text{ AND } (Q \text{ OPT } R) \equiv (P \text{ AND } Q) \text{ OPT } R$;
- $(P \text{ OPT } R) \text{ FILTER } C \equiv (P \text{ FILTER } C) \text{ OPT } R$.

Intuitively, this lemma states that AND operator can forward and OPT operator can backward in a well-designed pattern with preserving the semantics. The above three rules can be deployed on a WDAO-tree. For each WDAO-tree T, there exists T' corresponding to T after applying rewriting rules.

Figures 2 and 3 have shown that the process of rewriting rules after generating grammar tree and finally WDAO-tree can be obtained. Clearly, WDAO-tree has less height than the grammar tree.

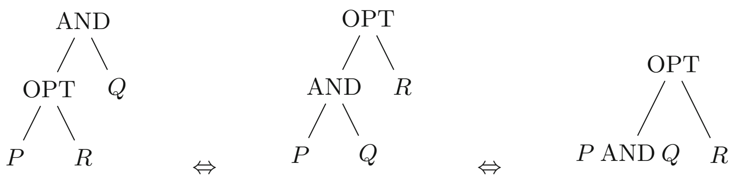


Fig. 2. Rewritting rule-1

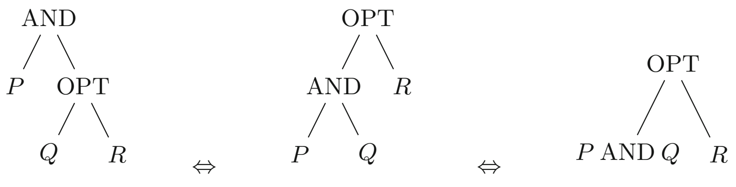


Fig. 3. Rewritting rule-2

3.3 WDAO-tree Construction

Before constructing WDAO-tree, we recognize query patterns and attachments at first. Then we rewrite query patterns by rewritting rules, which leads to a new pattern. Based on this new pattern, we construct WDAO-tree in the principle of Definition 1.

In the process of the WDAO-tree construction, we firstly build the grammar tree of SPARQL patterns, whose inner node is either AND operator or OPT operator. This process is based on recursively putting the left pattern and right pattern of operator in the left node and right node respectively until the pattern does not contain any operator. Then we apply the rewritting rules to the

Algorithm 1. rewritting rules

```
Input: GrammarTree with $Root$;
Output: RewriteTree with $Root$;
 1: while not all AND.child IS OPT do
 2: Procedure ReWriteRules($Root$)
 3: if $Root$ IS AND then
 4:     if $Root$ IS OPT then
 5:         swap($Root.left$,$Root.right.left$);
 6:         swap($Root.right$,$Root.right$);
 7:         swap($Root.left.left$,$Root$);
 8:         swap($Root.left.right$,$Root.left.right$);
 9:     end if
10:     if $Root.right$ IS OPT then
11:         swap($Root.left$,$Root.left$);
12:         swap($Root.right$,$Root.left.left$);
13:         swap($Root.left.left$,$Root$);
14:         swap($Root.left.right$,$Root.left.right$);
15:     end if
16: end if
17: Procedure ReWriteRules($Root.left$)
18: End Procedure
19: Procedure ReWriteRules($Root.right$)
20: End Procedure
21: End Procedure
22: end while
23: return $Root$;
```

grammar tree in Algorithm 1 to build rewriting-tree whose only leaf node is single triple pattern. Different rewritting rules are adopted depending on OPT operator are AND operator's left child or right child. Since WDAO-tree's inner nodes only contain AND operators, After getting rewriting-tree, we merge the AND operators only containing leaf child nodes with its child nodes into new nodes in order to get a WDAO-tree.

The WDAO-tree construction can be executed in PTIME. Given a pattern containing n ANDs and m OPTs, the construction of the grammar tree and rewriting tree have $O(n + m)$ time complexity and $O(nm)$ time complexity, respectively. Furthermore, the merge of nodes whose parent is AND has $O(n)$ time complexity.

4 PIWD Demonstration

In this section, we introduce PIWD, which is a plugin-based framework for well-designed SPARQL.

4.1 PIWD Overview

PIWD is written in Java in a 2-tier design shown in Fig. 4. The bottom layer consists of any BGP query framework which is used as a black box for evaluating BGPs. Before answering SPARQL queries, the second layer provides the rewriting process and left-outer join evaluation, which lead to the solutions.

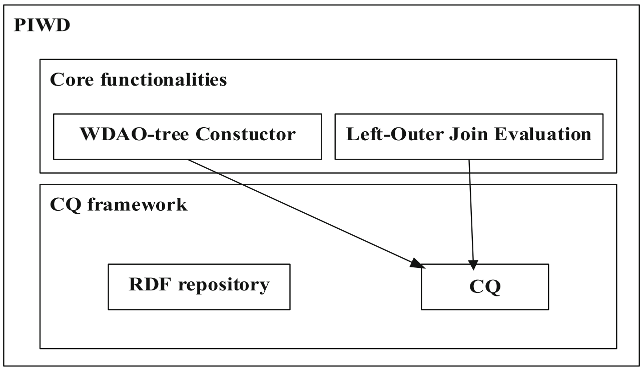


Fig. 4. PIWD architecture

BGP query framework supports both query and RDF data management, such as gStore, RDF-3X and so on, which solve the problem of subgraph isomorphism. PIWD provides the left-outer join between the BGPs. That is, the problem of answering well-designed SPARQL has been transformed into the problem of subgraph isomorphism and left-outer join between triple patterns.

4.2 Answering Queries over PIWD

The query process over PIWD can be described as follows:

Firstly, WDAO-tree is built after rewriting rules on the grammar tree. Secondly, post-order traversal is applied on WDAO-trees. The traversal rule is: If the node is a leaf node without the OPT operator, BGP query framework is deployed on it to answer this query and return solutions which is stored in a stack. If the node is an inner node labeled by the OPT operator, we get the top two elements in the stack and left-outer join them. We repeat this process until all of WDAO-tree's nodes are visited. Finally, only one element in the stack is the final solutions.

In the querying processing, BGP query framework serves as a query engine to support queries from leaves in WDAO-trees. OPT operators take an essential position in the query processing. Users receive optional solutions based on OPT operators which contribute to the semantic abundance degree since optional solutions are considered in this sense. In other words, OPT operators lead to the explosive growth of the solution scale.

The query process is described in Algorithm 2.

Algorithm 2. Query Processing over PIWD

```
Input: WDAO-tree with $Root$; Prefix $prefix$; $Stack$ to store subresults;
Output: Query result $result$;
 1: Procedure TraverseTree($Root$)
 2: if $root$ is not null then
 3:     Procedure TraverseTree($Root \rightarrow Lnode$)
 4:     End Procedure
 5:     Procedure TraverseTree($Root \rightarrow Rnode$)
 6:     End Procedure
 7:     if $node$ is not $OPTIONAL$ then
 8:         $subquery$=AssembleQuery($prefix$,$node$);
 9:         $subresult$=QueryIngStore($subquery$);
10:         Push($Stack$ , $subresult$);
11:     else
12:         $r$=Pop($Stack$);
13:         $l$=Pop($Stack$);
14:         $result$=$l \ltimes\!\!\!\!\bowtie r$;
15:         Push($Stack$ , $result$);
16:     end if
17: end if
18: End Procedure
19: $list$=ConvertToList($Stack$);
20: return $list$;
```

5 Experiments and Evaluations

This section presents our experiments. The purpose of the experiments is to evaluate the performance of different WDAO-patterns.

5.1 Experiments

Implementations and running environment. All experiments were carried out on a machine running Linux, which has one CPU with four cores of 2.40GHz, 32GB memory and 500GB disk storage. All of the algorithms were implemented in Java. gStore [33,34] and RDF-3X [16] are used as the underlying query engines to handle BGPs. In our experiments, there is no optimization in our OPT operation.

gStore and RDF-3X. Both gStore and RDF-3X are SPARQL query engines for subgraph matching. gStore stores RDF data in disk-based adjacency lists, whose format is *[vID,vLabel,adjList]*, where *vID* is the vertex ID, *vLabel* is the corresponding URI, and *adjList* is the list of its outgoing edges and the corresponding neighbor vertices. gStore converts an RDF graph into a data signature graph by encoding each entity and class vertex. Some different hash functions such as BKDR and AP hash functions are employed to generate signatures, which compose a novel index (called VS*-tree). A filtering rule and efficient search algorithms are developed for subgraph queries over the data signature graph in order to speed up query processing. gStore can answer exact SPARQL queries and queries with wildcards in a uniform manner. RDF-3X engine is a RISC-style architecture for executing SPARQL queries over large repositories of RDF triple. Physical design is workload-independent by creating appropriate indexes over a single giant triples table in RDF-3X. And the query processor is RISC-style by relying mostly on merge joins over sorted index lists. gStore and RDF-3X have good performances in BGPs since their query methods are based on subgraph matching.

Dataset. We used LUBM[2] as the dataset in our experiments to investigate the relationship between query response time and dataset scale. LUBM, which features an ontology for the university domain, is a standard benchmark to evaluate the performance of semantic Web repositories, In our experiments, we used LUBM1, LUBM50, LUBM100, LUBM150 and LUBM200 as our query datasets. The LUBM dataset details in our experiments are shown in Table 4.

Table 4. LUBM Dataset Details

Dataset	Number of triples	RDF NT File Size(bytes)
LUBM1	103,104	14,497,954
LUBM50	6,890,640	979,093,554
LUBM100	13,879,971	1,974,277,612
LUBM150	20,659,276	2,949,441,119
LUBM200	27,643,644	3,954,351,227

SPARQL queries. The queries over LUBM were designed as four different forms, which corresponds to different WDAO-trees. The details of queries are described in Table 5. Clearly, OPT nesting in Q_2 is the most complex among four forms. Furthermore, we build the AND operator in each query.

Table 5. SPARQL queries Details

QueryID	Pattern	OPT amount
Q_1	$(P_1$ AND P_2 AND $P_3)$ OPT P_4	1
Q_2	$((P_1$ AND P_2 AND $P_3)$ OPT $P_4)$ OPT $(P_5$ OPT $P_6)$	3
Q_3	$((P_1$ AND P_2 AND $P_3)$ OPT $P_4)$ OPT P_5	2
Q_4	P_1 OPT $((P_2$ AND P_3 AND $P_4)$ OPT $P_5)$	2

[2] http://swat.cse.lehigh.edu/projects/lubm/.

5.2 Evaluation on PIWD

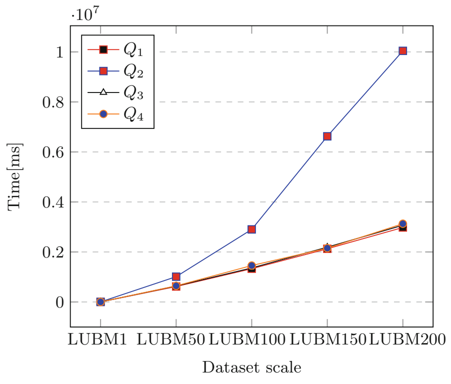
The variation tendencies of query response time are shown in Tables 6 and 7 and Fig. 5. Query efficiency is decreased with higher response time when OPT nesting becomes more complex. Furthermore, there has been a significant increase in query response time when the dataset scale grows up. For instance, we observe Q_2, which corresponds to the most complex pattern in our four experimental SPARQL patterns. When the dataset is ranging from LUBM100 to LUBM200, its query response time extends more than five times even though the dataset scale extends two times. In this sense, OPT nesting complexity in WDAO-patterns influences query response time especially for large dataset scale.

Table 6. Query Response Time[ms] on gStore

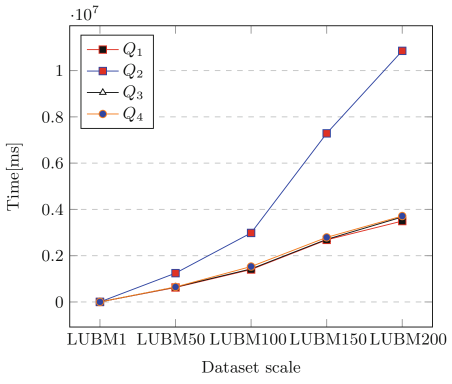
	LUBM1	LUBM50	LUBM100	LUBM150	LUBM200
Q_1	1,101	617, 642	1,329,365	2,126,383	2, 978, 237
Q_2	1,870	1, 010, 965	2,901,295	6,623,806	10, 041, 836
Q_3	1,478	637, 128	1,359,315	2,191,356	3, 068, 692
Q_4	1,242	644, 155	1,456,232	2,151,811	3, 129, 246

Table 7. Query Response Time[ms] on RDF-3X

	LUBM1	LUBM50	LUBM100	LUBM150	LUBM200
Q_1	1,231	625, 703	1,401,782	2,683,461	3, 496, 156
Q_2	1,900	1, 245, 241	2,983,394	7,286,812	10, 852, 761
Q_3	1,499	640, 392	1,427,392	2,703,981	3, 672, 970
Q_4	1,316	648, 825	1,531,547	2,791,152	3, 714, 042



(a) Performance on gStore



(b) Performance on RDF-3X

Fig. 5. Query response time over LUBM

6 Related Works

In this section, we survey related works in the following three areas: BGP query evaluation algorithms, well-designed SPARQL and BGP query evaluation frameworks.

BGP query algorithms have been developed for many years. Existing algorithms mainly focus on finding all embedding in a single large graph, such as ULLmann [24], VF2 [14], QUICKSI [21], GraphQL [11], SPath [32], STW [25] and TurboIso [10]. Some optimization method has been adapted in these techniques, such as adjusting matching order, pruning out the candidate vertices. However, the evaluation of well-designed SPARQL is not equivalent to the BGP query evaluation problem since there exists inexact matching.

It has been shown that the complexity of the evaluation problem for the well-designed fragment is coNP-complete [18]. The quasi well-designed pattern trees (QWDPTs), which are undirected and ordered, has been proposed [12]. This work aims at the analysis of containment and equivalence of well-designed pattern. Efficient evaluation and semantic optimization of WDPT have been proposed in [4]. Sparm is a tool for SPARQL analysis and manipulation in [13]. Above-mentioned all aim at checking well-designed patterns or complexity analysis without evaluation on well-designed patterns. Our WDAO-tree is different from QWDPTs in structure and it emphasizes reconstructing query plans. The OPT operation optimization has been proposed in [15], which is different from our work since our work aims to handle a plugin in any BGP query engine in order to deal with WDAO-patterns in SPARQL queries.

RDF-3X [16], TripleBit [27], SW-Store [1], Hexastore [26] and gStore [33,34] have high performance in BGPs. RDF-3X create indexes in the form of B+ tree, as well as TripleBit in the form of ID-Chunk. All of them have efficient performance since they concentrate on the design of indexing or storage. However, they can only support exact SPARQL queries, since they replace all literals (in RDF triples) by ids using a mapping dictionary. In other words, they cannot support WDAO-patterns well. Virtuoso [8] and MonetDB [5] support open-source and commercial services. Jena [7] and Sesame [6] are free open source Java frameworks for building semantic web and Linked Data applications, which focus on SPARQL parse without supporting large-scale date. Our work is independent on these BGP query frameworks, and any BGP query engine is adaptable for our plugin.

7 Conclusion

In this paper, we have presented PIWD, which is a plugin adaptable for any BGP query framework to handle WDAO-patterns. Theoretically, PIWD rebuilds the query evaluation plan based on WDAO-trees. After employing BGP query framework on WDAO-trees, PIWD supports the left-outer join operation between triple patterns. Our experiments show that PIWD can deal with complex and multi-level nested WDAO-patterns. In the future, we will further handle other non-well-designed patterns and deal with more operations such as UNION. Besides, we will consider OPT operation optimization to improve efficiency of PIWD and implement our framework on distributed RDF graphs by applying the distributed gStore [17].

Acknowledgments. This work is supported by the programs of the National Key Research and Development Program of China (2016YFB1000603), the National Natural Science Foundation of China (NSFC) (61502336), and the open funding project of Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education (K93-9-2016-05). Xiaowang Zhang is supported by Tianjin Thousand Young Talents Program and the project-sponsored by School of Computer Science and Technology in Tianjin University.

References

1. Abadi, D.J., Marcus, A., Madden, S.R., Hollenbach, K.: SW-store: a vertically partitioned DBMS for semantic web data management. VLDB J. **18**(2), 385–406 (2009)
2. Angles, R., Gutierrez, C.: The expressive power of SPARQL. In: Sheth, A., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 114–129. Springer, Heidelberg (2008). doi:10.1007/978-3-540-88564-1_8
3. Arenas, M., Pérez, J.: Querying semantic web data with SPARQL. In: Proceedings of SIGMOD 2011, pp. 305–316 (2011)
4. Barcelo, P., Pichler, R., Skritek, S.: Efficient evaluation and approximation of well-designed pattern trees. In: Proceedings of PODS 2015, pp. 131–144 (2015)
5. Boncz, P.A., Zukowski, M., Nes, N.J.: MonetDB/x100: Hyper-pipelining query execution. In: Proceedings of CIDR 2005 (2005)
6. Broekstra, J., Kampman, A., Harmelen, F.: Sesame: a generic architecture for storing and querying RDF and RDF schema. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 54–68. Springer, Heidelberg (2002). doi:10.1007/3-540-48005-6_7
7. Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K.: Jena: implementing the semantic web recommendations. In: Proceedings of WWW 2004, pp. 74–83 (2004)
8. Erling, O., Mikhailov, I.: RDF support in the virtuoso DBMS. Studies in Computational Intelligence, pp. 7–24 (2009)
9. Han, X., Feng, Z., Zhang, X., Wang, X., Rao, G.: On the statistical analysis of practical SPARQL queries. In: Proceedings of WebDB 2016, Article 2(2016)
10. Han, W.S., Lee, J., Lee, J.H.: Turbo ISO: Towards ultrafast and robust subgraph isomorphism search in large graph databases. In: Proceedings of SIGMOD 2013, pp. 337–348 (2013)
11. He, H., Singh, A.K.: Query language and access methods for graph databases. In: Aggarwal, C.C., Wang, H. (eds.) Managing and Mining Graph Data. Springer, Heidelberg (2010)
12. Letelier, A., Pérez, J., Pichler, R., Skritek, S.: Static analysis and optimization of semantic web queries. Proceedings of PODS **38**(4), 84–87 (2012)
13. Letelier, A., Pérez, J., Pichler, R., Skritek, S.: SPAM: a SPARQL analysis and manipulation tool. Proc. VLDB **5**(12), 1958–1961 (2012)

14. Luigi, P.C., Pasquale, F., Carlo, S., Mario, V.: A (sub)graph isomorphism algorithm for matching large graphs. IEEE Trans. Pattern Anal. Mach. Intell. **26**(10), 1367–1372 (2004)
15. Medha, A.: Left bit right: for SPARQL join queries with OPTIONAL patterns (left-outer-joins). In: Proceedings of SIGMOD 2015, pp. 1793–1808 (2015)
16. Neumann, T., Weikum, G.: The RDF3X engine for scalable management of RDF data. VLDB J. **19**(1), 91–113 (2010)
17. Peng, P., Zou, L., Özsu, M.T., Chen, L., Zhao, D.: Processing SPARQL queries over distributed RDF graphs. VLDB J. **25**(2), 243–268 (2016)
18. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Trans. Database Syst. **34**(3), 30–43 (2009)
19. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C Recommendation (2008)
20. Saleem, M., Ali, M.I., Hogan, A., Mehmood, Q., Ngomo, A.-C.N.: LSQ: the linked SPARQL queries dataset. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9367, pp. 261–269. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25010-6_15
21. Shang, H., Zhang, Y., Lin, X., Yu, J.X.: Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. Proc. VLDB **1**(1), 364–375 (2008)
22. Song, Z., Feng, Z., Zhang, X., Wang, X., Rao, G.: Efficient approximation of well-designed SPARQL queries. In: Song, S., Tong, Y. (eds.) WAIM 2016. LNCS, vol. 9998, pp. 315–327. Springer, Heidelberg (2016). doi:10.1007/978-3-319-47121-1_27
23. Swick, R.R.: Resource description framework (RDF) model and syntax specification. In: W3C Recommendation (1998)
24. Ullmann, J.R.: An algorithm for subgraph isomorphism. J. ACM **23**(1), 31–42 (1976)
25. Wang, H.: Efficient subgraph matching on billion node graphs. In: Proc. of VLDB 2012, 5: article 9 (2012)
26. Weiss, C., Karras, P., Bernstein, A.: Hexastore: sextuple indexing for semantic web data management. Proc. VLDB **1**, 1008–1019 (2008)
27. Yuan, P., Liu, P., Wu, B., Jin, H., Zhang, W., Liu, L.: Triplebit: a fast and compact system for large scale RDF data. Proc. VLDB **6**(7), 517–528 (2013)
28. Zhang, X., Feng, Z., Wang, X., Rao, G., Wu, W.: Context-free path queries on RDF graphs. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (eds.) ISWC 2016. LNCS, vol. 9981, pp. 632–648. Springer, Heidelberg (2016). doi:10.1007/978-3-319-46523-4_38
29. Zhang, X., Van den Bussche, J.: On the power of SPARQL in expressing navigational queries. Computer J. **58**(11), 2841–2851 (2015)
30. Zhang, X., Van den Bussche, J.: On the primitivity of operators in SPARQL. Inf. Process. Lett. **114**(9), 480–485 (2014)
31. Zhang, X., Van den Bussche, J.: On the satisfiability problem for SPARQL patterns. J. Artif. Intell. Res. **56**, 403–428 (2016)
32. Zhao, P., Han, J.: On graph query optimization in large networks. Proc. VLDB **3**(1–2), 340–351 (2010)
33. Zou, L., Mo, J., Chen, L., Özsu, M.T., Zhao, D.: gStore: answering SPARQL queries via subgraph matching. Proc. VLDB **4**(8), 482–493 (2011)
34. Zou, L., Özsu, M.T., Chen, L., Shen, X., Huang, R., Zhao, D.: gStore: a graph-based SPARQL query engine. VLDB J. **23**(4), 565–590 (2014)

Knowledge Graph

Non-hierarchical Relation Extraction of Chinese Text Based on Scalable Corpus

Xiaoheng Su[1], Hai Wan[1(✉)], Ruibin Chen[1], Qi Liu[1], Wenxuan Zhang[1], and Jianfeng Du[2]

[1] School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China
wanhai@mail.sysu.edu.cn,
{suxh8,chenrb6,liuq99,zhangwx26}@mail2.sysu.edu.cn

[2] Guangdong University of Foreign Studies, Guangzhou 510006, China
dududjf@gmail.com

Abstract. As for ontology construction from Chinese text, the non-hierarchical relation extraction is harder than the concept extraction and its extraction effect is still not satisfactory. In this paper, we put forward a scalable corpus model, which uses Tongyici Cilin and word2vec to calculate terms' similarity and add the qualified candidate terms to the corpora. In this way we can expand the scalable corpus while extracting non-hierarchical relations. In turn, the scalable corpus that has been expanded with the new terms will facilitate the non-hierarchical relation extraction further. We carry out the experiment with Chinese texts in the domain of Computer, whose results show that with expansion of the corpus, the extraction effect will be better and better.

Keywords: Relation extraction · Scalable corpus · Chinese text

1 Introduction

Maedche et al. defined the ontology structure as a five tuple: $O := \{C, R, H_C, rel, A_O\}$, where O, C, R and A_O indicated the ontology, the set of concepts, the set of relations and the set of axioms respectively. And H_C denotes a set of hierarchical relations among concepts with inheritance, for example, $H_C(C_1, C_2)$ expresses C_1 is a subconcept of C_2; $rel : R \rightarrow C \times C$ is a function, $rel(R) = (C_1, C_2)$ can also written in $R(C_1, C_2)$, denoting a set of non-hierarchical relations, such as $capital(China, Beijing)$ [1]. Therefore the main tasks of ontology construction from the Chinese text are concept and relation extractions, where the non-hierarchical relation extraction is a harder problem. The difficulty lies in unstructured organization of Chinese text and various relations occurring in Chinese sentences. Aiming at the non-hierarchical relation extraction, two kinds of methods are focused on. One is based on lexical rules matching, which is mainly suitable for English, while not working well in Chinese. The other is based on association rule analysis, which can determine whether

Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 231–238, 2016.
DOI: 10.1007/978-3-319-50112-3_17

there exist relations between two terms, however it is difficult to extract the exact relations. In this case, the paper tries to establish a robust non-hierarchical relation extraction model, making relations extraction in Chinese texts accurate and stable. In the model, firstly initialize the two scalable corpora including the concept corpus and the relation corpus, then use Density Extraction Algorithm proposed in the paper to extract the core concepts from the Chinese text, and seek for high quality of domain terms through core concepts, then add them into scalable corpora, next take advantage of improved association rule analysis to capture term pairs based on similarity analysis including word2vec and Tongyici Cilin, and extract relations after pruning sentences, finally expand corpus again. The rest of this paper is organized as follows: Sect. 2 introduces research status of ontology construction. Section 3 elaborates the main ideas of our method. Section 4 exhibits the experimental design and result analysis. Finally, Sect. 5 concludes our work.

2 Related Work

Nowadays ontology construction has received widespread attention from researchers. Generally, ontology construction algorithms are based on statistical methods and lexical syntactic patterns, and the idea of combining multiple methods has become a mainstream [2]. In the study of the concept extraction, Roberto et al. presented a novel method for filtering uncorrelated terms from candidate terms based on Domain Relevance(DR) and Domain Consensus (DC), their methods made the extracted concepts more representative for consistency of terms in the same domain [3]. On this basis, Haitao He et al. presented a method of the domain concept extraction based on semantic rules and association rules [4]. And for the relation extraction, there are two major approaches, lexical syntactic patterns [5] and association rule analysis [6]. By the limitations of these two methods, Jun Gu et al. proposed a improved association rule to acquire relations, which applied association rule to two terms and a verb at the same time, and could get pretty triples, however it ignored the case that noun expressions act as relations [7]. In order to extract more relations including nouns, Fan Yu raised a set of grammatical rules, which paid attention to the elimination of pseudo verbs and nouns, and experiments proved that this method helped getting subjects, predicates and objects precisely [8]. To improve the quality of extraction, Yufang Zhang et al. integrated similarity analysis based on the context into above methods [9]. Compared with literatures, our method does not require plenty of data to identify the domain terms and maintains the higher F1-value of relation extraction. More important is that we develop a robust extraction model, which can rigorously expand the scalable corpora and find more concepts and non-hierarchical relations from corpora in turn.

3 Extraction Based on Scalable Corpora

The scalable corpus is a set of domain terms that are the most common expressions in this domain and can be divided into the scalable concept corpus and the scalable relation corpus. The scalable concept [1] corpus refers to a set of concepts or instances mainly including nouns and noun phrases, and the scalable relation corpus is composed by a set of relations mainly in verbs or nouns. Our proposed extraction model includes the following two parts. But firstly we need establish a higher purity of initial corpora, this step is introduced in Part 4 detailedly.

3.1 Part One: Expand the Scalable Corpora

Since the scalable corpora are not complete and some domain terms in the document may not be in initial corpora, the primary objective of this part is to expand corpora with key terms of the document preliminarily, which lays the foundation of association rule analysis in the second part. Figure 1 depicts the work flow of the first part.

Firstly, segment the document and extract core concepts with Density Extraction Algorithm, and add them to the concept corpus. Secondly, locate the context of the core concepts and extract nouns and verbs around them. After that, filter out nouns that are not suitable for concepts through TF-IDF, and put the remainders keeping higher scores to the concept corpus. Last, filter out nouns and verbs that are unlikely to be relations with similarity analysis based on existent corpora, and add the remaining verbs or nouns to the relation corpus.

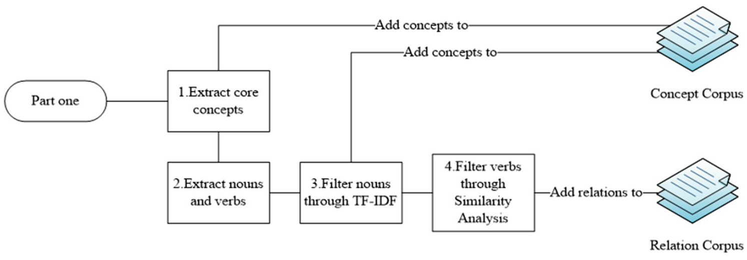


Fig. 1. Flow diagram of the first part

3.2 Part Two: Extract Relations

After the first part, the corpora have filled with key terms of this document, which can be regarded as the clue to extraction of concepts and relations further. The main goal in this part is to extract more non-hierarchical relations and concepts and expand the corpora again. Figure 2 shows the whole process.

Firstly, use the improved Apriori algorithm and the scalable concept corpus to get term pairs which may have the strong relations. Next, prune sentences where term pairs are and extract predicates as candidate relations. Then, filter out the relations and concepts using similarity analysis including word2vec and Tongyici

[1] In the paper, the concept refers to some concept or the instance of some concept.

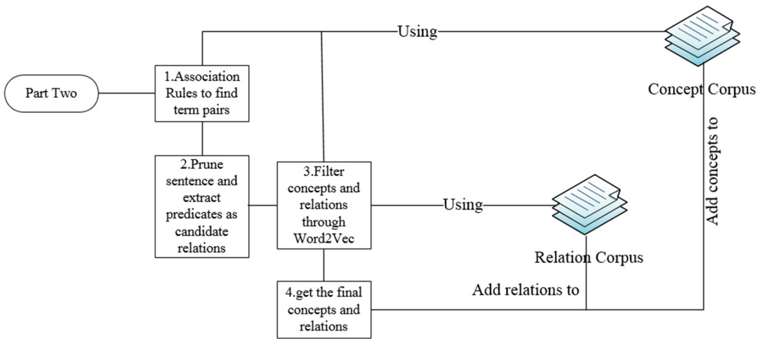

Part Two
1.Association Rules to find term pairs
2.Prune sentence and extract predicates as candidate relations
3.Filter concepts and relations through Word2Vec
4.get the final concepts and relations
Using
Using
Concept Corpus
Relation Corpus
Add concepts to
Add relations to

Fig. 2. Flow diagram of the second part

Cilin. If the pair of terms are both in concept corpus, then extract relations with the maximum similarity; if either of the pair of terms is in concept corpus, choose candidate relations whose similarity is greater than the preset threshold as the relation and the other term as the new concept. Finally, obtain the all relations and concepts from the previous step, and add them into corpora respectively.

3.3 Density Extraction Algorithm

The core concepts are the central words mainly depicted by a document. Generally a core concept represents a part of knowledge in the document. All core concepts together organize the content of the whole document. Besides, the document is the description of core concepts and distributed around core concepts. Based on this, we put forward the following assumptions: (1) the content of the document is spread out with core concepts; (2) the relevant part of knowledge about each core concept are distributed into the corresponding block in the document. In the paper, we propose Density Extraction Algorithm that is used to extract the core concepts from a single document based on the line density and uniformity.

Firstly, the document needs to have a segmentation including removing stop words and to be processed in the form of one sentence in one line. Then calculate the line density of each word, with the way that the number of total occurrences is divided by the number of spanning lines. Next, consider uniformity of each word, which refers to uniform distribution of each word in the corresponding block. In order to achieve the word uniformity, we divide the corresponding block of each word into consecutive several bands with equality, at the same time every band is made up with consecutive several rows. To this end each core concept must appear at least specified number of times in all bands. In order to enforce this method perfectly, we need to choose different arguments including the number of bands, the number of consecutive lines in one band and the specified density value according to the length of the specified Chinese document to process.

3.4 Improved Apriori Algorithm

Apriori Algorithm is a kind of realization of association rule analysis, aiming at getting some items that may have correlations between them [11]. The basic idea is that if items often appear in the same sentences, there probably be relations between them. Here we apply it to the term pairs extraction from documents, then find the concrete relations between term pairs.

Firstly, define a terms set $T = \{t_1, t_2, ..., t_m\}$ which contains the basic terms, $S = \{s_1, s_2, ..., s_n\}$ is a sentence set where the s_i expresses the ith sentence in S and is a set that contains some terms in T, $s_i \subseteq T$.

Then define an association rule $A \Rightarrow B$, where $A \subseteq T$, $B \subseteq T$, $A \cap B = \emptyset$. The rule is implemented by support and confidence.

Support of $A \Rightarrow B$ is the ratio of sentences containing both A and B to all sentences referred to as $\mathrm{Sup}(A \Rightarrow B)$. It is also the probability $P(A \cup B)$.

Confidence of $A \Rightarrow B$ is the ratio of sentences containing A and B to sentences containing A referred to as $\mathrm{Conf}(A \Rightarrow B)$. It is also the probability $P(\frac{A \cup B}{A})$.

In order to get the higher quality of term pairs, we make some improvements in choosing sentences. A sentence must contain at least a term in the concept corpus. From this perspective, it is beneficial to expand the corpora through core concepts beforehand in Part 1.

3.5 Prune Sentences

In a sentence, some nouns are not always used as subject or objects and the verbs are not for predicates. We call these words pseudo nouns and pseudo verbs [8]. For example, noun in preposition + noun. In the paper we take the following methods to prune sentences such that the sentence only includes standard nouns and verbs.

Firstly, merge some consecutive nouns and consecutive verbs because a complete sense of phrase may be split by the segmentation tool. Secondly, as some special words in Chinese which have a collocation with ordinary words will change the part of speech, we need to remove these collocations or change their parts of speech. Next, delete all adjectives and time words and these collocations: preposition + verb, noun + conjunction, conjunction + noun, preposition + noun, noun + particle. Finally, delete all remaining non-nouns and non-verbs.

3.6 Similarity Analysis

In order to expand our concept and relation corpus but no introduction of impurity, we use similarity analysis method including Tongyici Cilin and word2vec to expand our corpora.

The first similarity calculation method is based on synonyms, using the coding and structural features of the extended version of Tongyici Cilin [10]. Tongyici Cilin classification adopts hierarchy system with 5 layers of structure. With increasing of the level, the semantic description is closer and closer, and the similarity and correlation are higher.

Literature [12] presented a kind of semantic method based on context, but selected words must involve cooccurring words in all sentences. Here we find that word2vec is a natural tool for semantic similarity, which converts a word to a vector based on context using deep learning [13]. We compare the terms' similarity by computing the cosine value of their vectors. In addition, word2vec can also compute the similarity between relations.

4 Experiment and Analysis

Purity of the initial corpora is critical to extract the new concepts and relations from the specified domain documents. Initially, We crawls pages in the domain of Computer from Baidu Encyclopedia and extracts concepts and relations according to HTML tags to initialize the corpora. But we find that there is still a few impurities in corpora, so secondary purification is required.

Owing to special nature of Chinese text, we need to carry on the Chinese word segmentation. Here we choose "Jieba"[2] Chinese text segmentation tool. "Jieba" allows user to load a customer's dictionary of segmentation, we combine the relation corpus and the concept corpus as a synthetic term dictionary of "Jieba", to some extent it can solve a part of long tail concepts problem. Then we use the above crawled documents to train a word2vec model. Afterwards pick out 300 concepts from the initial concept corpus manually that can represent the domain of Computer. Then compute the average similarity distance between each term in initial corpora and 300 concepts though word2vec, finally filter the concepts lower than preset threshold that can be viewed as impurities. Similarly for relations. Eventually about 8000 pure concepts and 3000 relations are obtained.

In the paper, the precision rate, recall rate and F1-value are adopted to measure the results of extraction. Specially, we just evaluate the extracted relations from Chinese documents. Here we choose 50 documents from the Chinese version of "Computer Culture" written by June Jamrich Parsons as experiment documents, contents of these documents involve computer memory, operating system, CPU, computer network, database, software engineering and so on.

We mainly discuss two kinds of experiments with different expansion of corpora. One is expansion inside a single document, namely, all documents adopt the same initial corpora. And the other is continuous expansion within all documents. That is, the current document may adopt the corpora produced from the previous document. Here we choose the first 10 documents and compute the assessment values. Concrete results are shown in Fig. 3.

The average F1-value of the first expansion reaches to 76.9 %, even more exciting is that the F1-value of the second expansion reaches to 82.7 %. Besides, for the same document the precision value of the first expansion is generally greater than the second, stable at 85.3 %, and the recall value of the second expansion is commonly greater than the first, as high as 89.6 %. In fact the first expansion just follows the basic procedure depicted by our model, and concepts

[2] https://github.com/fxsjy/jieba.

or relations in the front of the document may not be extracted because initial corpora are smaller and incomplete. Therefore the precision is not too high but stable in the first case. Unlike the first expansion, the second expansion is a continuous process beneficial to find more concepts and relations. For example, a domain concept does not exist in the concept corpus, but after extraction of the last document, the concept is captured and added into the corpus, so in this round, we can extract relative relations and concepts about the concept. We continue to conduct the same experiment in the following 40 documents, the results of the experiments are not quite different.

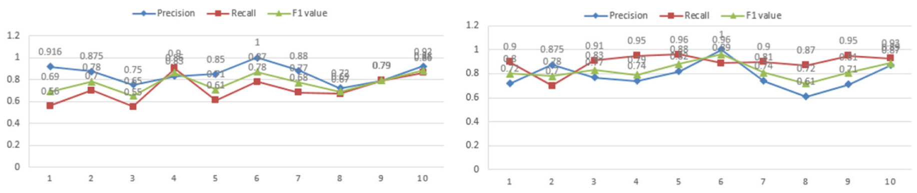


Fig. 3. The first expansion (left) and the second expansion (right)

Here we compare our method with the relevant method [8], which also uses association rule analysis and sentence pruning technique, but adopts different strategies. Detailed comparison is seen in the following table.

Table 1. Comparison with other method

Methods	Precision	Recall	F1
Literature [8] (2013)	77.1 %	71.4 %	72.6 %
Our method	78.6 %	89.6 %	82.7 %

Table 1 shows that all evaluation values of our method are higher than Literature [8] on the same test documents. Although we utilize method of Literature [8], our scalable corpora are expanding while extracting relations, more domain terms not just simple results of text segmentations are extracted.

5 Conclusion

The paper puts forward a new ontology extraction model by setting up two scalable corpora. In this model, expanding corpora and extracting new concepts or relations occur at the same time. Scalable corpora facilitate concept and relation extractions which enrich the corpora in turn. The result of experiment indicates that overall F1-value of this method is 82.7 %, and it is better than related methods. More importantly, our method can perform better and better when extraction continues all the way. Besides, it has a strong portability, that is to say, concepts and relations in other domain can be extracted in the same way without a large number of modification.

Acknowledgments. Hai Wan's research was in part supported by the National Natural Science Foundation of China under grant 61573386, Natural Science Foundation of Guangdong Province under grant 2016A030313292, Guangdong Province Science and Technology Plan projects under grant 2016B030305007, and Sun Yat-sen University Young Teachers Cultivation Project under grant 16lgpy40.

References

1. Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intell. Syst. **16**(2), 72–79 (2001)
2. Jia, X., Wen, D.: A survey of ontology learning from text. Comput. Sci. **34**(2), 181–185 (2007)
3. Navigli, R., Velardi, P.: Learning domain ontologies from document warehouses and dedicated web sites. Comput. Linguist. **30**(2), 151–179 (2004)
4. He, H., Shanhong, Z., et al.: Research on domain ontology the concept extraction based on association rule and semantic rules. J. Jilin Univ. (Info. Sci. Edt.) **32**(06), 657–663 (2014)
5. Hearst, M.A.: Automatic acquisition of hyponyms on large text corpora. In: Proceedings of the 14th International Conference on Computational Linguistics, pp. 539–545, Nantes, France (1992)
6. Buitelaar, P., Daniel, O., et al.: A Protege plug-in for ontology extraction from text based on linguistic analysis. In: Proceedings of the 1st European Semantic Web Symposium (2004)
7. Gu, J., Yan, M., et al.: Research on ontology relation acquisition based on improved association rule. Info. Stud. Theo. Appl. **34**(12), 121–125 (2011)
8. Yu, F., Cheng, H., et al.: Non-hierarchical relations extraction of chinese texts based on grammar rules and improved association rules. Lib. Info. Ser. **57**(22), 126–131 (2013)
9. Zhang, Y., Yang, F., et al.: Study on context based domain ontology the concept extraction and the relation extraction. Appl. Res. Comput. **27**(1), 74–76 (2010)
10. Tian, J., Zhao, W.: The method of word similarity calculation based on synonym word lin. J. Jilin. Univ. **28**(6), 602–608 (2010)
11. Agrawal, R., Ramakrishnan, S.: Fast algorithms for mining association rule in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499. VLDB (1994)
12. Zhang, Y., Yang, F.: Study on context based domain ontology the concept extraction and the relation extraction. Appl. Res. Comput. **27**(1), 74–76 (2010)
13. Mikolov, T., Chen et al.: Efficient estimation of word representations in vector space. arXiv preprint arxiv:1301.3781 (2013)

Entity Linking in Web Tables with Multiple Linked Knowledge Bases

Tianxing Wu(✉), Shengjia Yan, Zhixin Piao, Liang Xu, Ruiming Wang, and Guilin Qi

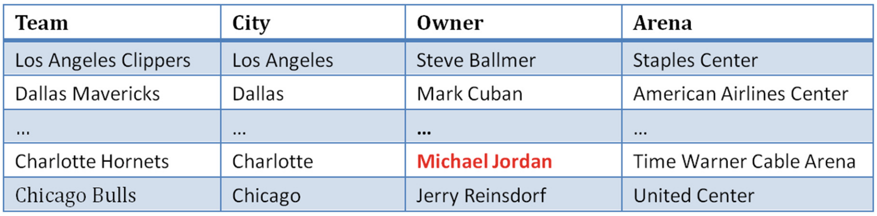
School of Computer Science and Engineering, Southeast University, Nanjing, China
{wutianxing,sjyan,piaozhx,liang.xu,wangruiming,gqi}@seu.edu.cn

Abstract. The World-Wide Web contains a large scale of valuable relational data, which are embedded in HTML tables (i.e. Web tables). To extract machine-readable knowledge from Web tables, some work tries to annotate the contents of Web tables as RDF triples. One critical step of the annotation is entity linking (EL), which aims to map the string mentions in table cells to their referent entities in a knowledge base (KB). In this paper, we present a new approach for EL in Web tables. Different from previous work, the proposed approach replaces a single KB with multiple linked KBs as the sources of entities to improve the quality of EL. In our approach, we first apply a general graph-based algorithm to EL in Web tables with each single KB. Then, we leverage the existing and newly learned "`sameAs`" relations between the entities from different KBs to help improve the results of EL in the first step. We conduct experiments on the sampled Web tables with Zhishi.me, which consists of three linked encyclopedic KBs. The experimental results show that our approach outperforms the state-of-the-art table's EL methods in different evaluation metrics.

Keywords: Entity linking · Web tables · Linked knowledge bases

1 Introduction

The current World-Wide Web contains a large scale of relational data in the form of HTML tables (i.e. Web tables), which have already been viewed as an important kind of sources for knowledge extraction on the Web. To realize the vision of Semantic Web, various efforts [9–12,19,20] have been made to interpret the implicit semantics of Web tables by annotating their contents as RDF triples. One critical step of such annotation is entity linking (EL), which refers to map the string mentions in table cells to their referent entities in a given knowledge base (KB). For example, in the third column of the Web table in Fig. 1, EL aims to link the string mention "`Michael Jordan`" to the entity "`Michael Jordan (American basketball player)`" in a given KB. Without correct identified entities, the annotation on Web tables is hard to get accurate RDF triples. Thus, in this paper, we focus on studying the problem of EL in Web tables.



Team	City	Owner	Arena
Los Angeles Clippers	Los Angeles	Steve Ballmer	Staples Center
Dallas Mavericks	Dallas	Mark Cuban	American Airlines Center
...	...	...	...
Charlotte Hornets	Charlotte	**Michael Jordan**	Time Warner Cable Arena
Chicago Bulls	Chicago	Jerry Reinsdorf	United Center

Fig. 1. An example of web table describing the information of NBA teams

There exist two main problems in previous work for EL in Web tables as follows. **(1)** Many work [9–11,19,21,22] strongly relies on the features based on specific information, such as column headers (e.g. "`Team`", "`City`", etc. in the first row of Fig. 1) in Web tables, entity types in the target KB, and so on. Therefore, it is obvious that these approaches can not work well when the given Web table or KB contains no or few such information. **(2)** Most of the existing approaches [2,9,10,16,19,21,22] only consider linking string mentions in table cells to a single KB, which can not ensure good coverage of EL in Web tables. This problem is also presented in [15] when performing EL in natural language text.

To overcome the above problems, we propose a new general approach for EL in Web tables with multiple linked KBs. The proposed approach contains two steps. We first apply a graph-based algorithm without using any specific information to EL in Web tables with each single KB. Then, we present three heuristic rules leveraging the existing and newly learned "`sameAs`" relations between the entities from different KBs to improve the results of EL in the first step. The second step of our approach can not only reduce the errors generated by EL with each single KB, but also improve the coverage of the EL results. In experiments, we map the string mentions in the cells of sampled Web tables to the entities in Zhishi.me [14], which is the largest Chinese linked open data and composed of three Chinese linked online encyclopedic KBs: Chinese Wikipedia[1], Baidu Baike[2] and Hudong Baike[3]. The evaluation results show that our approach outperforms two state-of-the-art systems (i.e. TabEL [2] and LIEGE [16]) in terms of MRR (i.e. Mean Reciprocal Rank[4]), precision, recall and F1-score.

The rest of this paper is organized as follows. Section 2 outlines some related work. Section 3 introduces the proposed approach in detail. Section 4 presents the experimental results and finally Sect. 5 concludes this work and describes the future work.

[1] https://zh.wikipedia.org.
[2] http://baike.baidu.com.
[3] http://www.baike.com.
[4] https://en.wikipedia.org/wiki/Mean_reciprocal_rank.

2 Related Work

In this section, we review some related work regarding semantic annotation on Web tables, which usually tackles three tasks: entity linking (EL), column type inference and relation extraction between the entities in the same row but different columns. After Cafarella et al. [6] reported that there are more than 150 million Web tables embedded with high-quality relational data, lots of researchers realized that Web tables are important sources that can be used for many applications, such as information extraction and structured data search. Hence, there emerged various work about semantic annotation on Web tables.

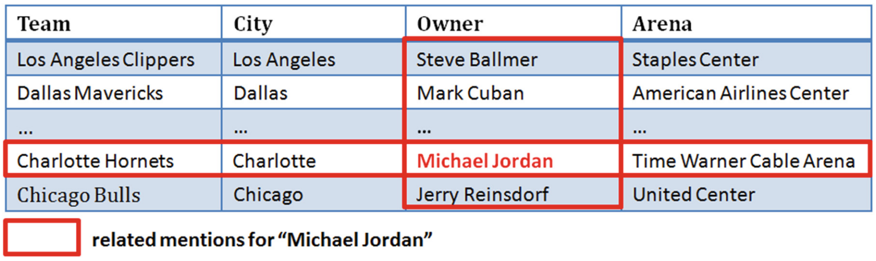
Hignette et al. [9] proposed an aggregation approach to annotate the contents of Web tables using vocabularies in the given ontology. It first annotates cells, then columns, finally relations between those columns. Similarly, Syed et al. [19] also presented a pipeline approach, which first infers the types of columns, then links cell values to entities in the given KB, finally selects appropriate relations between columns. Zhang [22] designed a tool called TableMiner for annotating Web tables. TableMiner only focuses on column type inference and EL, and can not extract relations from Web tables. Afterwards, Zhang [21] also proposed some strategies to improve TableMiner. Limaye et al. [10] and Mulwad et al. [11] described two approaches which can respectively jointly model the EL, column type inference and relation extraction tasks for Web tables. The main difference between our approach and these work is that we do not use any specific information for the task of EL, such as column headers and captions of Web tables, entity types in KBs, semantic markups in Web pages, and so on.

There also exists some work in specific scenarios about semantic annotation on Web tables without the step of EL. In the work of Venetis et al. [20], their approach weakens the impacts of EL, and directly infers the types of columns and determines the relationships by the frequency of different patterns in large scale isA and relation databases, which are both built from Web pages but usually unavailable to most of the researchers. Besides, Muñoz et al. [12] proposed an approach to mine RDF triples from Wikipedia tables. In this work, they can directly identify the entities in Wikipedia with internal links and article titles.

The closest work to our approach is done by Shen et al. [16] and Bhagavatula et al. [2]. Shen et al. [16] tried to link the string mentions in list-like Web tables (multiple rows with one column) to the entities in a given KB. Bhagavatula et al. [2] presented TabEL, a table entity linking system, which uses a collective classification technique to collectively disambiguate all mentions in a given Web table. Both of these two work do not use any specific information for EL, and can be applied to any KB. Here, we focus on EL with multiple linked KBs instead of a single KB, in order to improve the quality of EL in Web tables.

3 Approach

In this section, we introduce our proposed approach for entity linking (EL) in Web tables, which consists of two main steps: EL with any single KB and improving EL using "`sameAs`" links between multiple linked KBs.



Team	City	Owner	Arena
Los Angeles Clippers	Los Angeles	Steve Ballmer	Staples Center
Dallas Mavericks	Dallas	Mark Cuban	American Airlines Center
...	...	...	...
Charlotte Hornets	Charlotte	Michael Jordan	Time Warner Cable Arena
Chicago Bulls	Chicago	Jerry Reinsdorf	United Center

□ **related mentions for "Michael Jordan"**

Fig. 2. An example of related mentions in the same row or column

3.1 Entity Linking with a Single KB

Candidate Generation. For each string mention in table cells, we first need to identify its candidate referent entities in the given KB. Here, we segment each mention in word level, so each mention can be represented by a set of words. If an entity e in the given KB or one of e's synonyms in BabelNet [13] (a Web-scale multilingual synonym thesaurus) contains at least one word of some mention m, then e is taken as one candidate referent entity of the mention m. For example, the mention "`Charlotte`" has candidate referent entities such as "`Charlotte, North Carolina`", "`Charlotte, Illinois`" and "`Charlotte Hornets`". The results of candidate generation is that each mention may correspond to a set of candidate entities.

Entity Disambiguation. In entity disambiguation, we aim to choose an entity from the candidate set as each mention's referent entity in the given KB. As shown in Fig. 2, we can easily find that mentions in the same row or column tend to be related. In other words, there exists some potential association between any two mentions appearing in the same Web table. Therefore, we choose to jointly disambiguate all the mentions in one table using a graph-based algorithm:

(a) Firstly, for each given table, we build an *Entity Disambiguation Graph* only using mentions and their candidate referent entities as the graph nodes.
(b) Secondly, in each constructed *Entity Disambiguation Graph*, we compute the initial importance of each mention for joint disambiguation and the semantic relatedness between different nodes as the **EL impact factors** to decide whether an entity is the referent entity of a given mention.
(c) Finally, with iterative probability propagation using the **EL impact factors** until convergence, each entity gets its probability to be the referent entity of the given mention and our algorithm makes the EL decisions based on these probabilities.

In the following part of this section, we describe the above three steps in detail.

(a) Building *Entity Disambiguation Graph*. For each given table, we build an *Entity Disambiguation Graph*, which consists of two kinds of nodes and two kinds of edges introduced as follows.

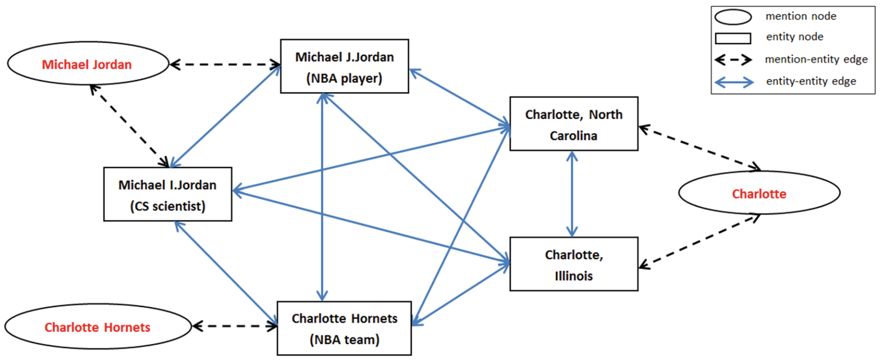


Fig. 3. An example of constructed *Entity Disambiguation Graph*

– **Mention Node**: These nodes refer to the mentions in Web tables.
– **Entity Node**: These nodes represent mentions' candidate referent entities in the given KB.
– **Mention-Entity Edge**: A mention-entity edge is an undirected edge between a mention and one of its candidate referent entities.
– **Entity-Entity Edge**: An entity-entity edge is an undirected edge between entities.

An example of the constructed *Entity Disambiguation Graph* is given in Fig. 3. Due to the limited space, it is only the part of the constructed *Entity Disambiguation Graph* for the Web table in Fig. 2, and lots of nodes and edges are not shown in Fig. 3. Note that each mention such as "`Michael Jordan`" should has a mention-entity edge linking to any of its candidate entities "`Michael J. Jordan (NBA player)`" and "`Michael I. Jordan (CS scientist)`". Entity-entity edges should also be created between all the entity nodes in the graph.

(b) Computing the EL Impact Factors. After constructing the *Entity Disambiguation Graph* for the given Web table, each node or edge is assigned with a probability. For the entity nodes, their probabilities refer to the possibilities of they being the referent entities of mentions, and are initialized as 0 before affected by the **EL impact factors**, which are actually (1) the probabilities of mention nodes, and they can be viewed as the importance of mentions for joint disambiguation; (2) the probabilities assigned to edges, and they are semantic relatedness between nodes. In this paper, we equally treat each mention, so when there exist k mentions in the Web table, the importance of each mention is initialized to $1/k$. Since there are two kinds of edges in each constructed *Entity Disambiguation Graph*, entity-entity edges and mention-entity edges should be respectively associated with the semantic relatedness between entities and that between mentions and entities.

For the semantic relatedness between mentions and entities, we use two features to measure it as follows.

– **String Similarity Feature.** If a mention m and an entity e are of similar strings, it is possible that e is m's the referent entity in the given KB. Hence, we define the string similarity feature $strSim(m, e)$ as

$$strSim(m, e) = 1 - \frac{EditDistance(m, e)}{\max\{|m|, |e|\}} \quad (1)$$

where $|m|$ and $|e|$ are the string lengths of the mention m and entity e, respectively. $EditDistance(m, e)$ means the edit distance[5] between m and e, and it is a way to quantify how dissimilar two strings are. In other words, the more similar in string level mention m and entity e are, the higher the value of $strSim(m, e)$ is.

– **Mention-Entity Context Similarity Feature.** Given a mention m and one of its candidate referent entities e, if they are semantic related, they tend to share similar context. Here, for obtaining the context of the given mention m, we first collect other mentions in the row or column where m locates. Then, we segment each collected mention into a set of words. Finally, we take all the words as the context of m and it is denoted by $menContext(m)$. For the context of the entity e, we first collect all the RDF triples which e exists in, and then segment each object (when e is the subject) or each subject (when e is the object) into a set of words. These words are also treated as e's context $entContext(e)$. To calculate the *mention-entity context similarity feature* $contSim_{me}(m, e)$ between the mention m and the entity e, we apply the Jaccard Similarity[6] as follows:

$$contSim_{me}(m, e) = \frac{|menContext(m) \cap entContext(e)|}{|menContext(m) \cup entContext(e)|} \quad (2)$$

Given a mention m and an entity e, to integrate the *string similarity feature* $strSim(m, e)$ with the *mention-entity context similarity feature* $contSim_{me}(m, e)$, we define the **Mention-Entity Semantic Relatedness** $SR_{me}(m, e)$ as follows:

$$SR_{me}(m, e) = 0.99 \times (\alpha_1 \cdot strSim(m, e) + \beta_1 \cdot contSim_{me}(m, e)) + 0.01 \quad (3)$$

where both α_1 and β_1 are set to 0.5 in this work. $SR_{me}(m, e)$ at least equals 0.01, in order to keep the connectivity of the *Entity Disambiguation Graph* during the subsequent process of probability propagation.

For the semantic relatedness between entities, we also define following two features to measure it.

– **Triple Relation Feature.** If two entities are in the same RDF triple, they are obviously semantic related. Thus, we compute the *triple relation feature* $IsRDF(e_1, e_2)$ between the entity e_1 and the entity e_2 as

$$IsRDF(e_1, e_2) = \begin{cases} 1, & e_1 \textit{ and } e_2 \textit{ are in the same RDF triple} \\ 0, & \textit{otherwise} \end{cases} \quad (4)$$

[5] https://en.wikipedia.org/wiki/Edit_distance.
[6] https://en.wikipedia.org/wiki/Jaccard_index.

– **Entity-Entity Context Similarity Feature.** Similar to the idea introduced in the *mention-entity context similarity feature*, i.e. semantic related entities may be of similar context, and we use the same process for extracting the context of each entity. Given an entity e_1 and an entity e_2, we also use Jaccard Similarity to compute the *entity-entity context similarity feature* $contSim_{ee}(e_1, e_2)$ between their respective context $entContext(e_1)$ and $entContext(e_2)$ as

$$contSim_{ee}(e_1, e_2) = \frac{|entContext(e_1) \cap entContext(e_2)|}{|entContext(e_1) \cup entContext(e_2)|} \tag{5}$$

To acquire the semantic relatedness between an entity e_1 and an entity e_2, we compute the **Entity-Entity Semantic Relatedness** $SR_{ee}(e_1, e_2)$ integrating *triple relation feature* $IsRDF(e_1, e_2)$ with the *entity-entity context similarity feature* $contSim_{ee}(e_1, e_2)$ as follows:

$$SR_{ee}(e_1, e_2) = 0.99 \times (\alpha_2 \cdot IsRDF(e_1, e_2) + \beta_2 \cdot contSim_{ee}(e_1, e_2)) + 0.01 \tag{6}$$

where both α_2 and β_2 are also set to 0.5.

(c) Iterative Probability Propagation. To combine different EL impact factors for the EL decisions, we utilize iterative probability propagation to compute the probabilities associated with entity nodes (i.e. the probabilities for entities to be the referent entities of mentions) until convergence. The detailed process of our proposed iterative probability propagation on each *Entity Disambiguation Graph* is described as follows.

Given an *Entity Disambiguation Graph* $\boldsymbol{G = (V, E)}$ containing $\boldsymbol{n}$ nodes (with $\boldsymbol{k}$ mention nodes and $\boldsymbol{l}$ entity nodes), each node is assigned to an integer index from $\boldsymbol{1}$ to $\boldsymbol{n}$. We use these indexes to represent the nodes, and an $n \times n$ adjacency matrix of the *Entity Disambiguation Graph* $\boldsymbol{G}$ is denoted as $\boldsymbol{A}$, where $\boldsymbol{A}_{ij}$ refers to the transition probability from the node i to the node j and $\boldsymbol{A}_{ij} = \boldsymbol{A}_{ji}$. Since the edge between the node i to the node j has been associated with a probability, which is the semantic relatedness (defined in Eqs. 3 and 6) between different nodes, we define $\boldsymbol{A}_{ij}$ as

$$\boldsymbol{A}_{ij} = \begin{cases} \frac{SR_{me}(i,j)}{SR_{me}(i,*)}, & \text{if } i \neq j \text{ and } i \text{ represents a mention node} \\ \gamma \times \frac{SR_{ee}(i,j)}{SR_{ee}(i,*)}, & \text{if } i \neq j \text{ and } i, j \text{ represent two entity nodes} \\ (1-\gamma) \times \frac{SR_{me}(i,j)}{SR_{me}(i,*)}, & \text{if } i \neq j,\ i \text{ is an entity node and } j \text{ is a mention node} \\ 0, & \text{if } i = j \end{cases} \tag{7}$$

where $SR_{me}(i, j)$ is the *mention-entity semantic relatedness* between a mention node and an entity node (defined in Eq. 3), $SR_{me}(i, j) = SR_{me}(j, i)$, $SR_{ee}(i, j)$

is the *entity-entity semantic relatedness* between entity nodes (defined in Eq. 6), $SR_{ee}(i,*)$ means the total *entity-entity semantic relatedness* between i and its adjacent entity nodes and γ is set to 0.5. If i represents a mention node, $SR_{me}(i,*)$ is the total *mention-entity semantic relatedness* between i and all of its adjacent nodes. If i denotes an entity node, $SR_{me}(i,*)$ is the total *mention-entity semantic relatedness* between i and its adjacent mention nodes.

Here, we finally define a notation, i.e. an $n \times 1$ vector $\boldsymbol{r}$ for all the nodes, where $\boldsymbol{r}(i)$ means the probability for the node i to be the referent entity of some mention (if i is an entity node). To compute $\boldsymbol{r}$ with iterative probability propagation, we first set its initial value $\boldsymbol{r}^0$. As introduced before, if the node i is a mention node, $\boldsymbol{r}^0(i)$ is set to the initial importance of i, i.e. $1/k$. If i is an entity node, $\boldsymbol{r}^0(i) = 0$. Then, we update $\boldsymbol{r}$ in the process of iterative probability propagation using other EL impact factors, i.e. *mention-entity semantic relatedness* and *entity-entity semantic relatedness* encoded in the matrix $\boldsymbol{A}$. In this way, the recursive form of $\boldsymbol{r}$ is given as follows:

$$\boldsymbol{r}^{t+1} = ((1-d) \times \frac{\boldsymbol{E}}{n} + d \times \boldsymbol{A}) \times \boldsymbol{r}^t \tag{8}$$

where t is the number of iterations and $\boldsymbol{E}$ is an $n \times n$ square matrix of all 1's. In this formula, to ensure the matrix $\boldsymbol{A}$ is aperiodic to converge, we add a special kind of undirected edges from each node to all the other nodes and give each edge a small transition probability controlled by the damping factor d. In other words, during the process of iterative probability propagation, there exists a probability that the EL impact factors are propagated by neither the defined *mention-entity edges* nor *entity-entity edges*, but the above special kind of edges associated with small transition probabilities. Since the process of iterative probability propagation is similar to the PageRank algorithm [5], we apply the same setting that $d = 0.85$. After the iterative probability propagation, given a mention m and its corresponding set of candidate referent entities $ESet(m) = \{e_1, e_2, ..., e_s\}$, we pick the entity which is of the highest probability in $ESet(m)$ as the referent entity of m.

Different from other methods, our approach for EL in Web tables with a single KB does not rely on any specific information but only general RDF triples. Thus, it can be applied to any KB containing RDF triples in Linking Open Data[7], including DBpedia [1,3], Yago [17,18], Freebase [4], Zhishi.me [14] and etc.

3.2 Improving Entity Linking with Multiple Linked KBs

Entity Linking (EL) in Web tables only with a single KB can not always ensure a good coverage. One solution is to respectively perform the task of EL with different KBs so that we can improve the coverage of the EL results. However, the problem is that there may exist conflicts among the results of EL with different KBs. In this paper, we have done a test on Zhishi.me consisting of three largest Chinese linked online encyclopedic KBs, i.e. Chinese Wikipedia, Baidu Baike and

[7] http://linkeddata.org/.

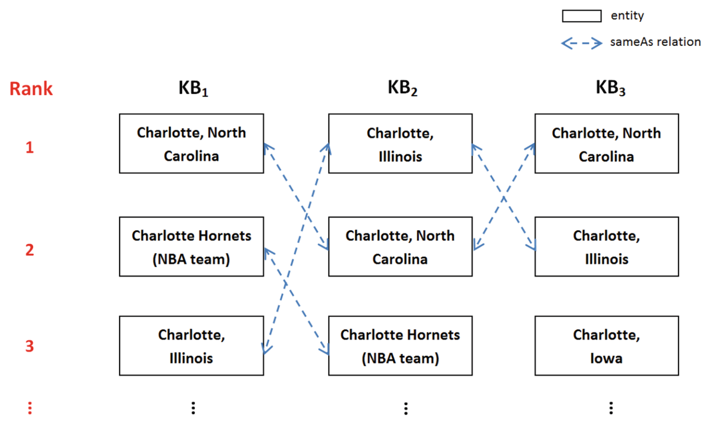

entity
sameAs relation
Rank
KB$_1$
KB$_2$
KB$_3$
1
2
3
Charlotte, North Carolina
Charlotte Hornets (NBA team)
Charlotte, Illinois
Charlotte, Illinois
Charlotte, North Carolina
Charlotte Hornets (NBA team)
Charlotte, North Carolina
Charlotte, Illinois
Charlotte, Iowa

Fig. 4. An example of EL results: the ranking lists of entities in different KBs for the mention "`Charlotte`" in Fig. 2

Hudong Baike. We first apply our proposed approach for EL with a single KB to our extracted Web tables (more than 70 thousand) with Chinese Wikipedia, Baidu Baike and Hudong Baike, respectively. Then, given a mention in a Web table and its identified entities in three KBs, if two identified entities have the "`sameAs`" relation, they can be considered as the same individual, otherwise they are different, i.e. there exists a conflict. According to the statistics, the conflicts exist in totally 38.94 % EL results (one result refers to a mention with its identified entities from different KBs). After that, we observe the EL results of the above test and analyse the reasons for such conflicts among the results of EL with different KBs as follows:

– **Reason 1**: For some KBs, the EL results are really incorrect, that is to say, some potential correct referent entities do not rank the highest.
– **Reason 2**: The "`sameAs`" relations are incomplete between KBs, i.e. there does not exist the "`sameAs`" relations between some equivalent entities from different KBs.

Based on these two reasons, we present our approach in detail to solve the conflicts of EL with different KBs in the following part of this section.

Suppose that there are n different linked KBs. For each given KB, we first apply our proposed approach to EL with a single KB to the given Web table. Then, for each mention, we can get its n ranking lists of referent entities. Afterwards, with the "`sameAs`" relations between entities in different KBs, we group the entities representing the same individual into different sets. For example, in Fig. 4, we can get 4 sets as follows:
(1) Set_1 = {"`Charlotte, North Carolina`" (KB_1), "`Charlotte, North Carolina`" (KB_2), "`Charlotte, North Carolina`" (KB_3)};
(2) Set_2 = {"`Charlotte, Illinois`" (KB_1), "`Charlotte, Illinois`" (KB_2), "`Charlotte, Illinois`" (KB_3)};

(3) $Set_3 = \{$"`Charlotte Hornets`" (KB_1), "`Charlotte Hornets`" (KB_2) $\}$;
(4) $Set_4 = \{$"`Charlotte, Iowa`" $(KB_3)\}$.
After that, we compute the average ranking, the highest ranking and the number of the entities in each set. For example, for the entities in set_1, the average ranking is computed as $(1 + 2 + 1)/3 = 1.33$, the highest ranking is 1 and the number is 3. Finally, we propose three rules as follows to solve the conflicts by choosing one set as the final EL results for the given mention.

- **Rule 1**: If both the average ranking and the highest ranking of the entities in a set rank the highest, and the number of the entities in this set is not less than half of the number of KBs, then we choose this set as the final EL results for the given mention.
- **Rule 2**: If there exist two or more sets that the average ranking and the highest ranking of the sets' corresponding entities are the same and rank the highest, also the number of the entities in each of these sets is not less than half of the number of KBs, then we choose one set at random as the final EL results for the given mention.
- **Rule 3**: If the number of the entities in each set is less than half of the number of KBs, the original EL results of the given mention remain unchanged.

To obtain the global and local optimal EL results at the same time, we consider not only the average ranking and the highest ranking of the entities in each set, but also the number of times that each individual (represented by a set of entities) occurs in different KBs. If the number of the entities in a set is less than half of the number of KBs, it means that the individual represented by these entities is covered by few KBs, so the average ranking is not convincing and it is not reasonable to take this set of entities to solve the conflicts among the results of EL with all the KBs.

According to *Reason 2*, if there exist more "`sameAs`" relations between the entities from different KBs, we may better solve the conflicts with our proposed rules. Here, in order to learn new "`sameAs`" relations, we define three features and train a supervised learning classifier Support Vector Machine (SVM), which is of the best performance in most situations [8]. The proposed features are introduced as follows:

- **Synonym Feature**. This feature tries to detect that whether the strings of two entities may represent synonyms. We input the strings of two entities e_1 and e_2 into BabelNet [13], if these two strings may represent synonyms in BabelNet, the synonym feature $isSyn = 1$, otherwise $isSyn = 0$.
- **String Similarity Feature**. This feature captures the linguistic relatedness between entities. It is denoted by $strSim(e_1, e_2)$, where e_1 and e_2 are entities. We use Eq. 1 to compute this feature using edit distance.
- **Entity-Entity Context Similarity Feature**. For two given entities in different KBs, this feature measures the similarity between the extracted context of entities and is already defined in Eq. 5.

After completing the "`sameAs`" relations with this SVM classifier, we also utilize our proposed rules to decide the final EL results. It is to verify whether the performance is improved compared with that of the rules only using the existing "`sameAs`" relations.

4 Experiments

In this section, we evaluated our approach on the sampled Web tables with three linked KBs (i.e. Chinese Wikipedia, Baidu Baike and Hudong Baike) in Zhishi.me, and compared our approach with two state-of-the-art systems for EL in Web tables and two degenerate versions of our approach.

4.1 Data Set and Evaluation Metrics

Since entity linking in Web tables with multiple linked KBs is a new task, we do not have any existing benchmark. Therefore, we need to generate ground truths by ourselves. We have extracted more than 70 thousand Web tables containing relational data from the Web. We randomly sampled 200 Web tables and invited five graduate students to manually map each string mention in table cells to the entities in each KB of Zhishi.me. The labeled results is based on majority voting and are publicly available[8]. Besides, in order to train the SVM classifier to learn new "`sameAs`" relations between different KBs, we also need to manually generate the labeled data. We first randomly selected 500 existing "`sameAs`" relations as the positive labeled data. Then, we random selected 3,000 entity pairs, each of which consists of the entities from different KBs. Finally, we also asked the five graduate students to label them and 3,000 entity pairs were all labeled as negative.

For each sampled Web table, we performed EL with our approach and the designed comparison methods. We evaluated the results with four metrics, which are Precision, Recall, F1-score and MRR (Mean Reciprocal Rank [7]). F1-score is the harmonic mean of precision and recall. Mean Reciprocal Rank is used for evaluating the quality of the ranking lists. For a mention m, the reciprocal rank in EL is the multiplicative inverse of the rank for m's referent entity. For example, if the correct referent entity of m is in the second place in the ranking lists generated by some EL algorithm, the reciprocal rank is 1/2.

4.2 Comparsion Methods

We compared our approach with the following methods.

- ***TabEL***: TabEL [2] is the current state-of-the-art system for EL in Web tables, and it uses a collective classification technique with several general features to collectively disambiguate all mentions in a given Web table. Besides, any KB can be used for EL in Web tables with TabEL.

[8] https://github.com/jxls080511/MK-EL.

Table 1. The overall EL results evaluated with each single KB

Knowledge base	Approach	Precision	Recall	F1-score	MRR
Chinese Wikipeida	TabEL	0.823	0.809	0.816	0.858
	LIEGE	0.778	0.747	0.762	0.813
	Our-s	0.830	0.797	0.813	0.860
	Our-m-e	0.861	0.821	0.841	0.881
	Our-m-(e+n)	**0.873**	**0.828**	**0.850**	**0.887**
Baidu Baike	TabEL	0.659	0.628	0.643	0.707
	LIEGE	0.629	0.576	0.601	0.670
	Our-s	0.696	0.652	0.673	0.725
	Our-m-e	0.758	0.705	0.731	0.746
	Our-m-(e+n)	**0.774**	**0.727**	**0.750**	**0.776**
Hudong Baike	TabEL	0.681	0.649	0.665	0.780
	LIEGE	0.661	0.632	0.646	0.751
	Our-s	0.708	0.642	0.673	0.768
	Our-m-e	0.729	0.700	0.714	0.787
	Our-m-(e+n)	**0.744**	**0.708**	**0.726**	**0.796**

– ***LIEGE***: LIEGE [16] is a general approach to link the string mentions in list-like Web tables (multiple rows with one column) to the entities in a given KB. It proposes an iterative substitution algorithm with three features to EL in Web lists. This approach can also be applied to EL in Web tables with any KB.
– ***Our-s***: It is a degenerate version of our approach. It only uses our proposed approach for EL with a single KB and does not utilize the rules with "`sameAs`" relations to improve the EL results.
– ***Our-m-e***: It is also a degenerate version of our approach. After performing EL with each single KB, it only uses existing "`sameAs`" relations (without newly learned "`sameAs`" relations) to improve the EL results.

4.3 Result Analysis

In the whole version of our proposed approach (denoted as ***Our-m-(e+n)***), we first apply a graph-based algorithm without using any specific information to EL in Web tables with each single KB, and then we leverage the existing and newly learned "`sameAs`" relations between the entities from different KBs to help improve the results of EL. Table 1 gives the overall results of our approach and the designed comparison methods evaluated with each single KB, and we can see that:

– Our approach for EL with a single KB, i.e. *Our-s*, is comparable to the state-of-the-art system TabEL and outperforms LIEGE, which reflects the effectiveness of our proposed graph-based algorithm.

- *Our-m-e* is always better than *Our-s* in precision, recall, F1-score and MRR. It shows the value of our proposed heuristic rules for improving the EL results of *Our-s*.
- The whole version of our approach, i.e. *Our-m-(e+n)* outperforms all the other comparison methods, which verifies the superiority of our approach for EL in Web tables with multiple linked KBs. Compared with *Our-m-e*, the better performance of *Our-m-(e+n)* demonstrates that the newly learned "`sameAs`" relations are beneficial to solve the conflicts among the EL results of *Our-s* with different KBs.

Besides, we also calculated the precision, recall and F1-score as the evaluation results of our approach (i.e. *Our-m-(e+n)*) on the whole Zhishi.me. The precision is 0.831, recall is **0.903** and F1-score is 0.866. The most important thing is that the recall is significantly improved, which shows EL in Web tables with multiple linked KBs can really ensure a good coverage.

5 Conclusions and Future Work

In this paper, we presented a new approach for EL in Web tables with multiple linked KBs. We first proposed an algorithm based on graph-based iterative probability propagation to perform EL with each single KB. In order to improve the EL results generated by the first step, we then applied three heuristic rules leveraging the existing and newly learned "`sameAs`" relations between the entities from different KBs. The experimental results showed that our approach not only outperforms the designed state-of-the-art comparison methods in different evaluation metrics, but also can use any single KB or linked KBs for EL in the Web tables.

As for the future work, we first will build more benchmarks of other languages for the new task of EL in Web tables with multiple linked KBs and further verify the effectiveness of our approach in other languages, especially English. We then plan to provide APIs or tools as the programming interface of our proposed approach. Finally, we also consider to extend our approach to cross-lingual entity linking in Web tables with multiple linked KBs.

Acknowledgements. This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant No. 61272378, the 863 Program under Grant No. 2015AA015406 and the Research Innovation Program for College Graduates of Jiangsu Province under Grant No. KYLX16_0295.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC - 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76298-0_52

2. Bhagavatula, C.S., Noraset, T., Downey, D.: TabEL: entity linking in web tables. In: Arenas, M., et al. (eds.) ISWC 2015. LNCS, vol. 9366, pp. 425–441. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25007-6_25
3. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: Dbpedia-a crystallization point for the web of data. Web Seman. Sci. Serv. Agents WWW **7**(3), 154–165 (2009)
4. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD, pp. 1247–1250 (2008)
5. Brin, S., Page, L.: Reprint of: the anatomy of a large-scale hypertextual web search engine. Comput. Netw. **56**(18), 3825–3833 (2012)
6. Cafarella, M.J., Halevy, A., Wang, D.Z., Wu, E., Zhang, Y.: Webtables: exploring the power of tables on the web. PVLDB **1**(1), 538–549 (2008)
7. Craswell, N.: Mean reciprocal rank. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, p. 1703. Springer, Heidelberg (2009)
8. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. **15**(1), 3133–3181 (2014)
9. Hignette, G., Buche, P., Dibie-Barthélemy, J., Haemmerlé, O.: Fuzzy annotation of web data tables driven by a domain ontology. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 638–653. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02121-3_47
10. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. PVLDB **3**(1–2), 1338–1347 (2010)
11. Mulwad, V., Finin, T., Joshi, A.: Semantic message passing for generating linked data from tables. In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8218, pp. 363–378. Springer, Heidelberg (2013). doi:10.1007/978-3-642-41335-3_23
12. Muñoz, E., Hogan, A., Mileo, A.: Using linked data to mine RDF from wikipedia's tables. In: WSDM, pp. 533–542 (2014)
13. Navigli, R., Ponzetto, S.P.: Babelnet: building a very large multilingual semantic network. In: ACL, pp. 216–225 (2010)
14. Niu, X., Sun, X., Wang, H., Rong, S., Qi, G., Yu, Y.: Zhishi.me - weaving chinese linking open data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011. LNCS, vol. 7032, pp. 205–220. Springer, Heidelberg (2011). doi:10.1007/978-3-642-25093-4_14
15. Pereira, B.: Entity linking with multiple knowledge bases: an ontology modularization approach. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8797, pp. 513–520. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11915-1_33
16. Shen, W., Wang, J., Luo, P., Wang, M.: Liege: link entities in web lists with knowledge base. In: SIGKDD, pp. 1424–1432 (2012)
17. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: WWW, pp. 697–706 (2007)
18. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a large ontology from wikipedia and wordnet. Web Seman. Sci. Serv. Agents WWW **6**(3), 203–217 (2008)
19. Syed, Z., Finin, T., Mulwad, V., Joshi, A.: Exploiting a web of semantic data for interpreting tables. In: WebSci, vol. 5 (2010)
20. Venetis, P., Halevy, A., Madhavan, J., Paşca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering semantics of tables on the web. PVLDB **4**(9), 528–538 (2011)

21. Zhang, Z.: Learning with partial data for semantic table interpretation. In: Janowicz, K., Schlobach, S., Lambrix, P., Hyvönen, E. (eds.) EKAW 2014. LNCS (LNAI), vol. 8876, pp. 607–618. Springer, Heidelberg (2014). doi:10.1007/978-3-319-13704-9_45
22. Zhang, Z.: Towards efficient and effective semantic table interpretation. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 487–502. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11964-9_31

Towards Multi-target Search of Semantic Association

Xiang Zhang[1,2](✉) and Yulian Lv[3]

[1] School of Computer Science and Engineering, Southeast University, Nanjing, China
x.zhang@seu.edu.cn
[2] Key Laboratory of Data Engineering and Knowledge Services,
Nanjing University, Nanjing, China
[3] College of Software Engineering (Suzhou), Southeast University, Suzhou, China
lvyulian@seu.edu.cn

Abstract. Semantic association represents group relationship among objects in linked data. Searching semantic associations is complicated, which involves the search of multiple objects and the search of their group relationships simultaneously. In this paper, we propose this kind of search as a multi-target search, and we compare it to traditional search tasks, which we classify as single-target search. A novel search model is introduced, and the notion of virtual document is used to extract linguistic information of semantic associations. Multi-target search is finally fulfilled by a PageRank-like ranking scheme and a top-*K* selection policy considering object affinity. Experiments show that our approach is effective in improving retrieval precision on semantic associations.

Keywords: Linked data · Semantic association · Multi-target search

1 Introduction

Linked data provides good practice for connecting and sharing objects by URI and RDF. An important knowledge we can discovered in linked data is the explicit or hidden relationship among objects, which is named as semantic associations. Stated in [1], semantic association is defined as group relationships among multiple objects.

At present, massive semantic associations can be discovered efficiently from large volume of linked data. But few studies have been done on searching these semantic associations and making use of them in a friendly and efficient manner. It was proposed to transform semantic associations into text-based structure and searching based on keywords in [2]. However, this approach still lacks considerations on the group relationships. Traditional keyword-based search only consider the hit of the keywords in single information target in a dataset, which can be a single document or a web page. We name it as single-target search. In our cases, the search model involves the search of multiple objects and the group relationships simultaneously. For example, a search of associations could be a set of keywords: "Tim Berners-Lee", "Ted Nelson" and "Doug Engelbart". Each set of keywords in this search hits an object. User wants to find out the group relationships among these three scientists. Thus, searching semantic associations is a complicated combination of multiple single-target searches.

In this paper, we propose a multi-target search model of semantic associations in Sect. 4. The notion of virtual document is used also in Sect. 4 to extract linguistic information of semantic associations. In Sect. 5, multi-target search is fulfilled by a PageRank-style ranking scheme and a top-K selection policy considering object affinity. Experiments are discussed in Sect. 6, showing that our approach is feasible in improving retrieval precision.

2 System Architecture

An overview of the architecture of multi-target search is given in Fig. 1. **Association Miner** discovers meaningful semantic associations in multiple linked data, and then passes the mining results to ***Association Indexer***. The latter uses a ***VDoc Extractor*** to build the linguistic information of each association. Once a user poses a multi-target search on indexed associations, the search will be divided into multiple single-target searches, each performed by a corresponding ***Single-Target Query Processor***. The ***Association Ranker*** will sort the search result based on a ***Static Ranker***, which rank the results using a PageRank-style ranking algorithm to filter out important associations, and also on a ***Top-K Selector***, which utilizes an ***Object Affinity*** **Assessor** to control the size of result set and makes a reasonable combination of the multiple single-target search into a multi-target one. The final search result will return to user through ***Multi-Target Query Processor***.

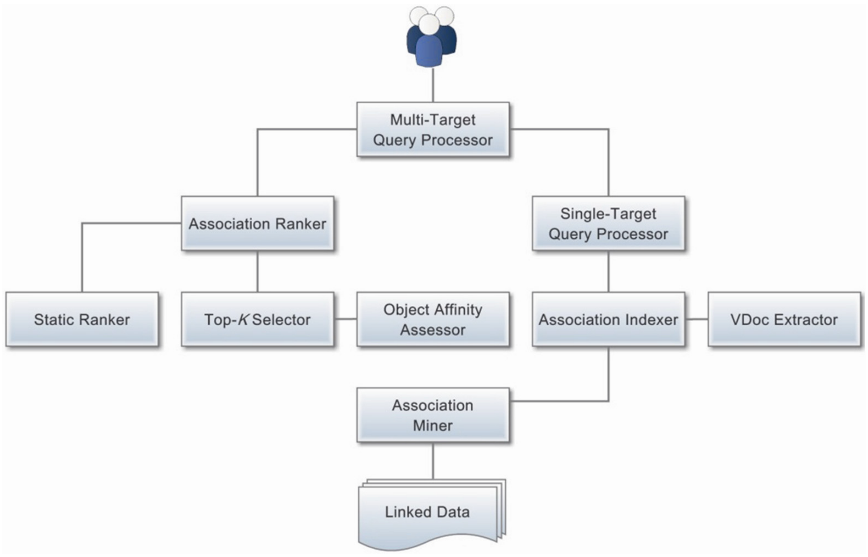


Fig. 1. The architecture of searching system of semantic association

3 Discovering Semantic Associations

Early definitions of semantic association are based on RDF paths between two objects. In [3], Aleman declared that, if two objects are semantically connected by a semantic path then they are semantically associated. In [4], Kochut used defined directionality path to characterize semantic associations. But path-based definition of semantic associations is not able to characterize complex group relationships among multiple objects. Thus, a graph-based definition was proposed in [1], in which a semantic association is a graph instantiation of a frequent subgraph called link pattern.

Figure 2(b) shows a link pattern discovered in real-world linked data. It represents a paper-presenting association among "Author", "Publication" and "Conference". This pattern is discovered by pattern-growth frequent subgraph mining algorithm, assuring the pattern is typical and meaningful. Figure 2(a) presents two semantic associations. One is a paper-presenting event in WWW2004 conference, the other is on ISWC2003 conference. Both are instantiations of the link pattern shown in Fig. 2(b). A discussion was given in [1] on mining link patterns and semantic associations.

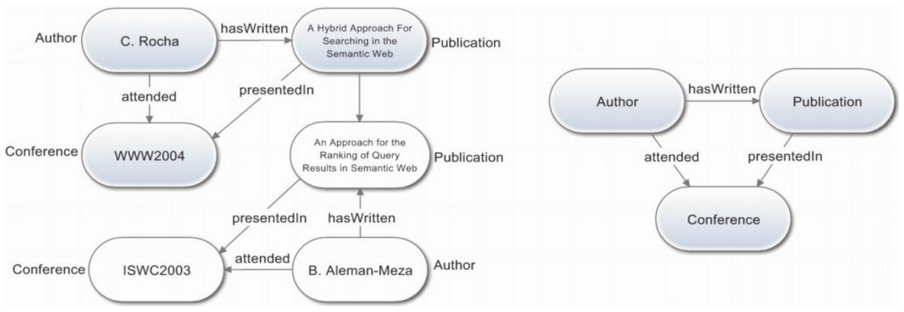


Fig. 2. (a) An example of semantic associations (b) corresponding link pattern

4 Searching Semantic Association

4.1 Search Model

Traditional keyword-based search only consider the hit of the keywords in single information target in a dataset, which can be a single document or a web page. Single-target search is not applicable in the context of semantic association search, as explained in previous section. A typical search in our context will be composed of several separated sub-query, in which each indicates a user need expecting a certain object, and the expected semantic associations are the group relationship among these expected objects.

Definition 1 (Multi-target Search Model): A multi-target search model is a quadruple: $\langle O, A, Q, F \rangle$, where:

(1) O is the set of objects in linked data;
(2) A is a set of semantic associations discovered in linked data;

(3) Q is the query set. A query $q \in Q$ is a set of sub-queries: $q = \{q_1, q_2, \ldots q_k\}$ in which each sub-query is a bag of query keywords, and will be posed to hit a set of objects in O.

(4) $F = \langle F_o, F_Q \rangle$ is the query function comprising two sub-functions. F_o is a function mapping a sub-query to a set of objects. For a sub-query $q_i \in q$, $F_o(q_i) = O_{q_i} \subseteq O$ is the set of hit objects. F_Q is a function mapping a query to a set of semantic associations. For a query $q \in Q$ and $q = \{q_1, q_2, \ldots q_k\}$, $F_o(q) = A_q \subseteq A$ is the research result of q, in which each association contains at least one object from $F_o(q_1)$ to $F_o(q_k)$.

Multi-target search model is a two-step search model. Given a semantic association query containing sub-queries, the first step of the search model aims at finding separated sets of objects using sub-queries. The second step of the model utilizes the sets of objects hit in the first step to find appropriate semantic associations. The sub-queries in multi-target search model enable a search with finer granularity comparing to single-target search model.

4.2 Search Process

The graph structure of the semantic association brings a great barrier to the process of searching. Transforming the graph into text-based structure is a good idea. Therefore, we borrowed the notion of virtual document referred in [5].

Briefly, for a literal node, the description is a collection of words derived from the literal itself; for a URI, it is a collection of words extracted from the local name, *rdfs*:*label*, *rdfs*:*comment* and other possible annotations.

Given an object o, its linguistic information is a bag of words defined in Eq. 1:

$$info(o) = \bigcup \{LN(o), RL(o), RC(o), OA(o)\} \tag{1}$$

Here, $LN(o)$ represents the bag of words in the local name of o, $RL(o)$ represents the bag of words in the *rdfs:label* of o, $RC(o)$ represents the bag of the *rdfs:comment* of o and $OA(o)$ represents the bag of the other possible annotations. The operator $\cup$ stands for merging several sets of words together.

When a multi-target search is posed, it will be divided into single-target ones and each single-target search will look for corresponding objects, then a resulted set of associations will be built using hit objects, shown in Algorithm 1.

Algorithm 1 . The process of multi-target search

```
Input: A query $q = \{q_1, q_2, \dots q_k\}$, in which $q_i$ is a sub-query.
1. for $i = 1$ to $k$:
    1) find target objects $o_1, o_2, \dots o_x$ that match keywords in $q_i$;
    2) for each target objects $t_j$, find semantic associations $a_{j1}, a_{j2}, \dots a_{jy}$ that
       involve target object $o_j$; then add these associations into an association
       set: $s_i$, which is the result set of the corresponding sub-query.
2. for $i = 1$ to $k$:
    1) quick sort all associations in each $s_i$ with their internal id;
    2) find $s_m, (1 \le m \le k)$, which contains least semantic associations;
3. for each semantic association $a_l$ in $s_m$:
    1) for $j = 1$ to $k$: use binary search to check whether $s_j$ contains $a_l$;
    2) if every $s_j$ $(j \neq m)$ contains $a_l$, put $a_l$ into result set $R$;
Output: Resultset $R = \{a_1, a_2, \dots a_n\}$
```

5 Ranking Scheme

The time complexity of finding the intersection of semantic association will be $O(h * N \log N)$, when the keywords are frequent words, the amount of hit objects may be very large and response time may be unacceptable for user. So a top-*K* selection policy is used to control the volume of results, and meanwhile to sort objects according to mutual affinity.

A PageRank-like static ranking scheme is firstly performed on resulted objects, to evaluate their importance and authority. For object o_i, its importance is calculated as Eq. 2, in which N is total number of objects, $L(o_j)$ is the number of objects that connected with o_j and the value of damping factor d is 0.85.

$$PageRank(o_i) = \frac{1-d}{N} + d \sum_{o_j} \frac{PageRank(o_j)}{L(o_j)} \tag{2}$$

Next, for each group of hit objects in a single-target search, top-*2 K* objects are selected according to their PageRank value. Further, top-*K* objects are selected based on the evaluation of their affinity to objects in other groups. For object o_i and o_j, the affinity between them is as Eqs. 3 and 4. Here a_m represents a semantic association.

$$affinity(o_i, o_j) = \sum_{sa_i} \text{connection}(a_m, o_i, o_j) \tag{3}$$

$$\text{connection}(a_m, o_i, o_j) = \begin{cases} 0, & a_m \text{ dones't contain } o_i \text{ or } o_j \\ 1, & a_m \text{ contains both } o_i \text{ and } o_j \end{cases} \tag{4}$$

6 Evaluation

6.1 Dataset

DBpedia is used as dataset in this paper. It is a widely-used linked data on structured information extracted from Wikipedia. As of September 2014, the entire DBpedia dataset contains about 30 billion triples. A part of it is used as our dataset, which includes 54,046,420 triples. 14,608 linked patterns and 47,535,150 semantic associations are discovered from this dataset.

6.2 Evaluation Method

Since we don't find any other similar research about multi-target search of semantic association, we performed experiments comparing our multi-target search model to a pure single-target search model as stated in [2]. Both response time and retrieval precision are evaluated between two search models based on 500,000 randomly selected semantic associations. The size of result is also considered in the experiment because it will affect the time cost for user to understand the results.

A set of keywords are randomly picked to construct 10 variable-length different queries. A human ground truth is built by semantic web experts for the evaluation of precision. For each query, we select top 20 records from two models respectively as a potential set of records. Since the two models may produce the result overlap, the size of set may less than 40. Then we invite experts to pick up the top 10 useful semantic associations these match the query keywords as possible from the set. The statistical information of the amount of hit objects is shown in Table 1, the amount that each group of keywords hit is separated by 'l', the amount of result of each query will be shown in Sect. 6.4

Table 1. Statistical information of hit objects

Query 1	Query 2	Query 3	Query 4	Query 5
179 l 209	11 l 11 l 4	102 l 1 l 13	71 l 120 l 5	1 l 8
Query 6	Query 7	Query 8	Query 9	Query 10
102 l 1 l 6 l 12	7 l 9 l 8	2 l 1	223 l 1	31 l 86

6.3 Evaluation on Response Time and Precision

The horizontal axis in Figs. 3 and 4 present 10 different queries. The vertical axis is the response time and precision of each query in Figs. 3 and 4 respectively. The blue column presents the single-target search model and the red column presents the multi-target search model.

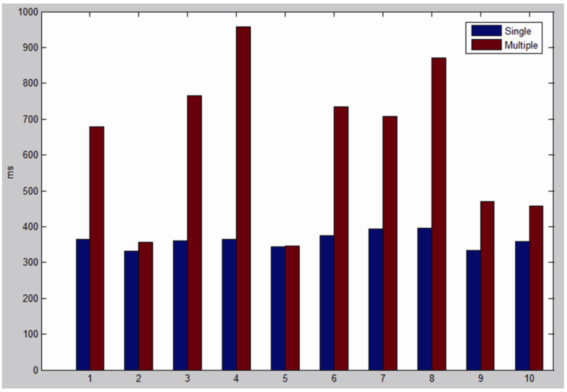


Fig. 3. Evaluation on response time (Color figure online)

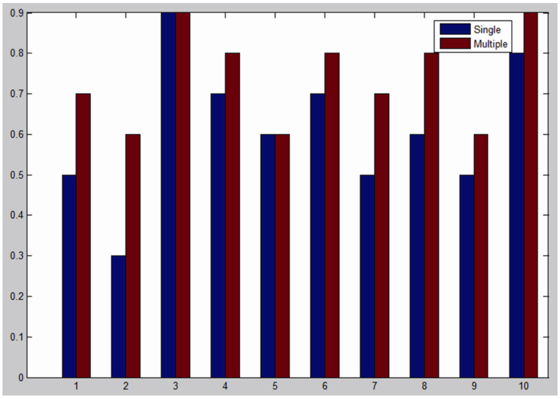


Fig. 4. Evaluation on precision (Color figure online)

Figure 3 shows that, the response time of single-target search model is quite stable, the average response time is about 400 ms, but that of multi-target search vary a lot. When users search with popular keywords, it may hit large quantities of objects, and then get a large size of result. The search process may take a long time. Figure 4 shows that the average precision of single target search model is 61%, the average of multi-target search model is 74%.

6.4 Evaluation on Result Size

Figure 5 clearly presents the significant difference of result sizes with or without top-K selection. The average value of result size with top-K selection is 35, while it is 109 without top-K selection. The result size with top-K selection is only about 12%–72% of that without top-K selection. A top-K selection policy can control the volume of results effectively, and it will also take user less time to distinguish useful information.

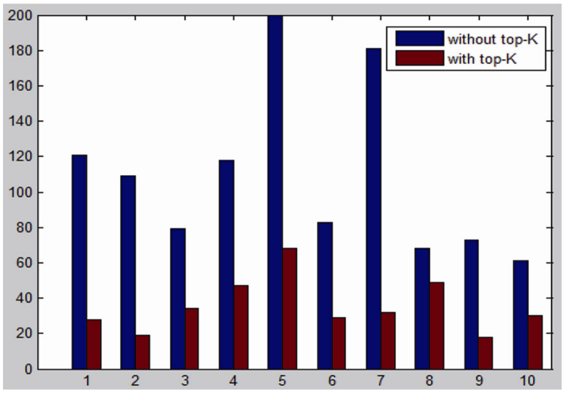


Fig. 5. Evaluation on result size (Color figure online)

7 Related Work

Traditional definition of semantic association is paths connecting two objects. In [3], if two objects are semantically connected by a semantic path then they are semantically associated. In [6], Myungjin proposed a semantic association search methodology that consists of how to find relevant information for a given user's query. In [7], Viswanathan presented a method to find relevant semantic association paths through user-specific intermediate entities.

Virtual document is constructed and used for ontology matching in [8]. The size of virtual document is easy to control, can be applied to search. Therefore we transform the semantic association into text-based structure in [2]. In order to solve the problem of short text, we propose k-step virtual document.

Top-K algorithm is widely used in RDF research. In [9], Tran proposed a novel algorithm for the exploration of top-K matching subgraphs. In [10], Huiying Li present an algorithm for searching top-K answers on RDF data. In our paper, top-K algorithm is used to control the size of result and search space.

8 Conclusions and Future Works

In this paper, a multi-target search model is proposed for searching semantic association. Comparing to single-target search model, our model consider the search of multiple objects and their group relationship simultaneously. The notion of virtual document is used to extract and represent linguistic information of objects and semantic associations. A PageRank-style ranking scheme and a top-K selection policy considering object affinity are used. Experiments show that our approach is feasible in improving retrieval precision.

In our future wok, the type of object will be taken into consideration to achieve higher retrieval accuracy. The search system will be evaluated on its efficiency on large-volume semantic associations.

Acknowledgements. The work was supported by the National High-Tech Research and Development (863) Program of China (No. 2015AA015406), the Open Project of Jiangsu Key Laboratory of Data Engineering and Knowledge Service (No. DEKS2014KT002), and National Natural Science Foundation of China (No. 61472077). We would like to thank Xing Li for his efforts in implementation and evaluations.

References

1. Zhang, X., Zhao, C., Wang, P., Zhou, F.: Mining link patterns in linked data. In: Gao, H., Lim, L., Wang, W., Li, C., Chen, L. (eds.) WAIM 2012. LNCS, vol. 7418, pp. 83–94. Springer, Heidelberg (2012)
2. Wang, C., Zhang, X., Lv, Y., Ji, L., Wang, P.: Searching semantic associations based on virtual document. In: Qi, G., Tang, J., Du, J., Pan, J.Z., Yu, Y. (eds.) CSWS 2013. CCIS, vol. 406, pp. 62–75. Springer, Heidelberg (2013)
3. Aleman-Meza, B., Halaschek-Wiener, C., Arpinar, I.B., Sheth, A.P.: Context-aware semantic association ranking, vol. 1, no. 3, pp. 33–50 (2003)
4. Kochut, K.J., Janik, M.: SPARQLeR: extended SPARQL for semantic association discovery. Semant. Web Res. Appl. **4519**, 145–159 (2007)
5. Le, B.T., Dieng-Kuntz, R., Gandon, F.: On ontology matching problems. In: Proceedings of the International Conference on Enterprize Information Systems, pp. 236–243 (2003)
6. Lee, M., Kim, W.: Semantic association search and rank method based on spreading activation for the semantic web. In: Proceedings of the International Conference on Industrial Engineering and Engineering Management, pp. 523–1527 (2009)
7. Viswanathan, V., Krishnamurthi, I.: Finding relevant semantic association paths through user-specific intermediate entities. Hum. Centric Comput. Inf. Sci. **2**(1), 1–11 (2012)
8. Qu, Y., Hu, W., Cheng, G.: Constructing virtual documents for ontology matching. In: Proceedings of the International Conference on World Wide Web, pp. 23–31 (2006)
9. Tran, T., Wang, H., Rudolph, S., Cimiano, P.: Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In Proceedings of the IEEE International Conference on Data Engineering, pp. 405–416 (2009)
10. Li, H., Wang, Y.: Ranked keyword query on semantic web data. In: Proceedings of the International Conference on Fuzzy Systems and Knowledge Discovery, pp. 2285–2289 (2010)

Linking Named Entity in a Question with DBpedia Knowledge Base

Huiying Li([✉]) and Jing Shi

School of Computer Science and Engineering, Southeast University,
Nanjing 210096, People's Republic of China
{huiyingli,220151530}@seu.edu.cn

Abstract. The emerging Linked Open Data provides an opportunity to answer the natural language question based on knowledge bases (KB). One challenge of the question answering (QA) problem is to link the entity mention in the question with the entity in the existing knowledge base. This study proposes an approach to link entity mention with a DBpedia entity. We propose an entity-centric indexing model to help search candidate entities in KB. After obtaining the candidate entities, we expand the context of the entity mention with WordNet and ConceptNet, we compute the context similarity between the expanded context and the property value of the candidate entity and the popularity of the candidate entity. Finally, we rerank the candidate entities by leveraging these features. Evaluations are performed on DBpedia version 2015, the evaluation tests show that our approach is promising in dealing with linking named entity in DBpedia.

1 Introduction

Linked Open Data (LOD) aims to publish structured data to enable the interlinking of such data and therefore enhance their utility [1]. It shares information that can be read automatically by computers and allows data from different sources to be connected and queried. LOD consists of an unprecedented volume of structured datasets currently amounts to 50 billion facts that are represented as Resource Description Framework (RDF) triples on the web. Recently, many large scale publicly available knowledge bases including DBpedia [2] and YAGO [3] have emerged.

The large amount of Linked Data has become an important resource to support question answering. However, many challenges are encountered in returning a right answer based on the knowledge base for a natural language question (utterance). The following example represents a question and some snippets in DBpedia.

Sample question: "what town was martin luther king assassinated in?".

Sample knowledge base: A snippet of DBpedia dataset is shown in Fig. 1, which lists three different instances named "martin luther king".

Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 263–270, 2016.
DOI: 10.1007/978-3-319-50112-3_20

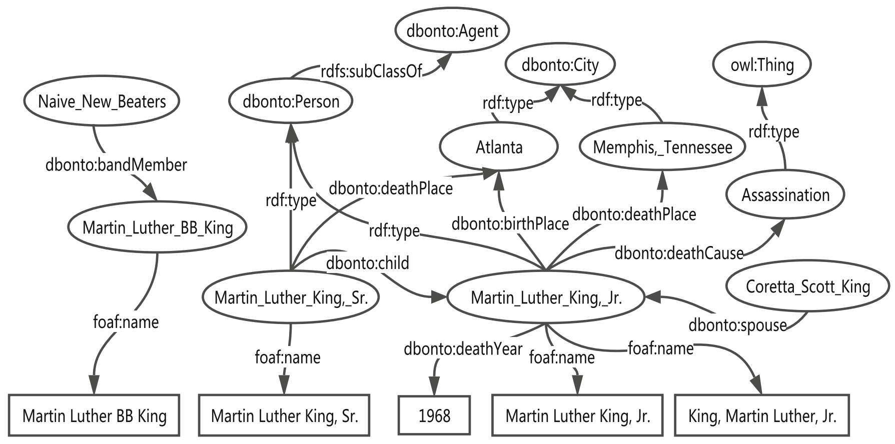


Fig. 1. RDF snippet Example

In examining the sample question and the sample knowledge base, we find that to answer the assassinated town of the civil rights leader Martin Luther King, we should map the queried "martin luther king" to the civil rights leader, then retrieve the assassinated town of this entity from the knowledge base directly.

Hence, the primary task for question answering is to locate the entity mention in the question and link it with an entity in the knowledge base. Locating entity mention (Named Entity Recognition) is out of the scope of our paper. We focus on the entity linking task in this paper. We propose an approach to link the entity mention in a question with an entity in the knowledge base. We index the surface forms for every entity in DBpedia by Lucene. Using this index, we can generate a ranked candidate entity list for each entity mention. Furthermore, the context similarity and entity popularity of the candidate entities are calculated, then we rerank the candidate entities with the combination of these measures.

The remainder of this article is organized as follows: Sect. 2 introduces the related work. Section 3 proposes the method to generate candidate entities. Section 4 presents the named entity linking approach. Section 5 details the experimental results of our approach. Section 6 concludes the study.

2 Related Work

The emergence of large scale knowledge bases like DBpedia and YAGO has spurred great interests in the entity linking task, which maps the textual entity mention to its corresponding entity in the knowledge base. [5] is the first work that considers Wikipedia as an information source for named entity disambiguation. The disambiguation is performed using an SVM kernel that compares the lexical context around the ambiguous named entity to the context of the candidate Wikipedia page. The subsequent work on Wikification [6–8] recognize the global document-level topical coherence of the entities. [6] addresses the entity linking problem through maximizing the agreement between the text of the mention document and the context of the Wikipedia entity, as well as the agreement among the categories associated with the candidate entities. [7] defines the semantic context as a set of unambiguous surface forms in the text, and uses the Normalized Google Distance (NGD) [9] to compute the relatedness. [8] formalizes the Disambiguation to Wikipedia task as an optimization problem with local and global variants, and analyzes their strengths and weaknesses. LINDEN [10] is a framework to link named entities in text with a knowledge base unifying Wikipedia and Word-Net, by leveraging the rich semantic knowledge embedded in the Wikipedia and the taxonomy of the knowledge base. The semantic associativity and semantic similarity are considered based on the constructing of semantic Network. A probabilistic approach is proposed in [11], entity mentions are disambiguated jointly across an entire document by combining a document-level prior of entity co-occurrences with local information captured from mentions and their surrounding context.

3 Candidate Entity Generation

Given an entity mention m, we generate the set of candidate entities E_m in this section. Generally, the candidates in E_m should have the name of the surface form of m. To solve this problem, we need to build an index for all entities in the knowledge base. We also find that the information in the KB is usually entity-centric, and many triples describe the property (attribute and relation) value pairs of an entity. Based on this observation, we consider the entity as the basic index unit.

We group the properties into multiple categories to reduce the fields number and preserve some of the original structure. Totally, we group RDF properties into five fields for a given entity e, and these fields and their values are listed as follows:

- *Name*: The *Name* field collects the name attributes of entity e, and we consider the name attribute as $foaf : name$, $dbonto : alias$, $rdfs : label$ or the attribute ends with "name", "label", or "title".
- *Type*: The *Type* field represents the $rdf : type$ attribute of entity e.
- *Attribute*: The *Attribute* field collects the literal attributes except for the name attributes. The field values are the corresponding attribute values, and the attribute names are also indexed for QA.
- *OutRelation*: The *OutRelation* field collects the object attributes from entity e to other entities e_i. The field values are a list of items, and each item represents an object attribute information for entity e. The item is composed of the object attribute name, the name of entity e_i, and the class and super class type of entity e_i.
- *InRelation*: The *InRelation* field collects the object attributes from other entities e_i to entity e. The field values are a list of items, and each item is composed of the object attribute name, the name of entity e_i, and the class and super class type of entity e_i.

Based on the proposed index method, we index all the entities in KB by Lucene. For each mention m, we look up the index and search for the mention m directly in the *name* field. We take the entity mention as the keyword to perform a search in Lucene. The entities returned by Lucene are considered as candidate entities. Each returned entity is assigned a relevance score by Lucene. We represent this relevance score by $LS(m, e)$, it means the Lucene score for entity e.

For entity mention "martin luther king" in question "what town was martin luther king assassinated in?", three entities in Fig. 1 are returned as candidate entity. The entity with highest Lucene score is *Martin_Luther_BB_King*, then *Martin_Luther_King,_Sr.*, finally *Martin_Luther_King,_Jr.* Based on the context in the question, entity *Martin_Luther_King,_Jr.* should be the linked entity. We find that although our index can help search candidate entities and rank them by Lucene score $LS(m, e)$. It is not enough for named entity linking.

4 Named Entity Linking

In this section, we discuss how to rerank the candidate entities generated in Sect. 3.

4.1 Context Similarity

Our guiding premise is that the property of the target entity should be similar to the context of the entity mention. Given an entity mention m, E_m is the set of candidate entities generated in Sect. 3. Based on the guiding premise, one reranking factor is the cosine similarity between the context of the entity mention and the properties of the candidate entity.

$$CosSim(m, prop) = \frac{m.T \bullet prop.T}{||m.T|| \bullet ||prop.T||} \tag{1}$$

For every property *prop* of candidate entity e, *prop.T* contains the words of the property, the words of the property value, and the words of the value type. Where $m.T$ contains all words occurring around the entity mention in the question. Meanwhile, to expand the context of the entity mention, we import its synonyms and related words with the help of WordNet and ConceptNet. WordNet is used to obtain the synonyms, and ConceptNet is used to obtain the related words and similar words. The *prop.T* and $m.T$ are represented in the standard vector space model, where each component corresponds to a term, and the term weight is the frequency of the term.

$$CS(m, e) = \sum_{prop \in e} CosSim(m, prop) \tag{2}$$

The value of *Context similarity* for each candidate entity e is defined as the sum of the *CosSim* to each property.

4.2 Entity Popularity

Given an entity mention m, E_m is the set of candidate entities. The other reranking factor is the entity popularity. Each $e \in E_m$ containing the same surface form m has different popularity, some entities are rare and some entities are popular for the given surface form. A popular entity means that there are many entities which points to it. Based on the *inRelation* field index in Sect. 3, we can calculate the number of links which point to an entity.

For example, for the entity mention "martin luther king", the entity *Martin_Luther_BB_King* and *Martin_Luther_King,_Sr.* are much rarer than *Martin_Luther_King,_Jr.*, and in most cases when people mention "martin luther king", they mean the civil rights leader rather than other two entities.

Hence, we define the popularity score $PS(e)$ for entity e as:

$$PS(e) = \frac{num(e)}{\sum_{ce \in E_m} num(ce)} \tag{3}$$

where $num(e)$ is the number of links which points to entity e.

4.3 Candiates Reranking

Based on the three score factors introduced above, we can rerank the candidate entities. We consider using a linear reranking function as follows:

$$\hat{e} = \underset{e \in E_m}{\operatorname{argmax}}\, S(m, e) \tag{4}$$

where $S(m, e)$ is the reranking score for each $e \in E_m$, it can be calculated by:

$$S(m, e) = W \bullet X_m(e) \tag{5}$$

The feature vector $X_m(e)$ is generated for each $e \in E_m$ where $X_m(e) =< LS(m, e), CS(m, e), PS(e) >$. These features affect the reranking score, different features have different degrees of importance. The weight vector W, namely, $W =< w_1, w_2, w_3 >$, which gives different weights for each feature in $X_m(e)$. It can be learned by training on the question answer dataset. Therefore, the candidate entities can be reranked according the their score $S(m, e)$, the $top-1$ entity $\hat{e}$ is selected as the predicted mapping entity for mention m.

To learn the weight vector W by training on the question answer dataset, we use Support Vector Machine(SVM) based on maximum interval. The SVM is completed by Libsvm toolkit[1].

4.4 Detecting Unlinkable Mention

The disambiguation method discussed above implicitly assumes that DBpedia contains all entities that the entity mention refers to. In practice, there may be

[1] http://www.csie.ntu.edu.tw/~cjlin/libsvm/#nuandone.

contexts where entity mention m refers to an entity e that is not covered in DBpedia, especially when e is not a popular entity. We call this entity mention as unlinkable mention. To deal with the unlinkable mention, we adopt a simple method and learn a threshold τ to validate the top one entity e for entity mention m. If $S(m, e)$ is less than the threshold τ, we consider entity mention m as unlinkable.

5 Experimental Study

We use the WEBQUESTIONS dataset [12], which consists of 5,810 question/answer pairs, as test questions. The questions are split into training and testing sets, which contain 3,778 questions and 2,032 questions, respectively. We learn the weight vector W by training on the training dataset, and test our approach on the testing dataset. The real-world RDF dataset DBpedia 2015 is selected as the KB to answer the questions.

We adopt the evaluation measure precision which is used in most work about entity linking. The precision is calculated as the number of correctly linked entity mentions divided by the total number of all mentions. Since the entities are returned by ranked score, we evaluate the precisions at different k (k means how many entities are returned). The precision of $top-k$ is calculated as the number of entity mentions, which is correctly linked in $top-k$ entities, divided by the total number of all mentions.

Table 1. Feature set effectiveness over the "What" questions

Feature Set	*top*-1		*top*-2		*top*-3	
	Number	Precision	Number	Precision	Number	Precision
LS	452	0.40	522	0.46	560	0.50
LS+CS	512	0.61	563	0.67	589	0.70
LS+PS	508	0.60	536	0.63	538	0.63
LS+CS+PS	545	**0.65**	595	**0.71**	600	**0.72**

To evaluate the effectiveness of our approach on different questions, we consider three typical questions in the testing sets of the WEBQUESTIONS, "What" questions, "Who" questions, and "Where" questions.

Table 1 shows the precisions of the proposed approach on "What" questions. We analyze the effectiveness of different feature sets, it shows the precision and the number of correctly linked mentions obtained by our approach with different feature sets. It can be seen that every feature has a positive impact on the performance of our approach, and with the combination of all features our approach can obtain the best result. The precisions of LS are the results that only Lucene score is considered. The improvement achieved by adding CS (context similarity) feature to LS is greater than that can be achieved by adding PS

(popularity score) feature, which means that the CS feature is quite useful to deal with entity linking problem in "What" questions.

It shows the precisions at different k (k = 1, 2, and 3 respectively). For a tested question, the result is considered to be right when the right target entity is returned in the *top-k* results. The best result is 0.65 when only *top*-1 entity is considered. It is nature the precision increases to 0.72 when *top*-3 entities are considered.

Table 2. Feature set effectiveness over the "Who" questions

Feature Set	*top*-1		*top*-2		*top*-3	
	Number	Precision	Number	Precision	Number	Precision
LS	168	0.63	201	0.75	209	0.78
LS+CS	164	0.62	187	0.70	200	0.75
LS+PS	192	**0.72**	202	**0.76**	209	**0.78**
LS+CS+PS	179	0.67	196	0.74	202	0.76

But the precision of unlinkable mention is only 0.18, it means that the simple threshold setting method is not effective for detecting unlinkable mention. Although the precision of unlinkable mention can be increased by changing the threshold, but it decreases the precision of entity linking.

Table 3. Feature set effectiveness over the "Where" questions

Feature Set	*top*-1		*top*-2		*top*-3	
	Number	Precision	Number	Precision	Number	Precision
LS	138	0.56	169	0.69	180	0.73
LS+CS	143	0.58	167	0.68	173	0.70
LS+PS	172	**0.70**	187	**0.76**	188	**0.76**
LS+CS+PS	154	0.63	176	0.72	177	0.72

Table 2 shows the precisions on "Who" questions. The PS feature has a positive impact on the performance of our approach, but the CS feature has a slight improve on the performance, sometimes it has a negative impact on the contrary. It leads that our approach only obtains a little improve of precision with the combination of all features.

Table 3 shows the precisions on "Where" questions. The results are similar to those in Table 2, the PS feature obtains the best result.

6 Conclusions

In this study, we propose a named entity linking approach on the DBpedia dataset. We set up an entity-centric indexing, candidate entities are returned according to the LS score given an entity mention. Then we consider a set of useful features for entity linking. The CS feature measures the cosine similarity between the context of the mention and the property values of the candidate entity. The PS feature measures the popularity of the candidate entity. Finally, a SVM re-ranker is used to score each candidate entity. The evaluation tests show that our approach is promising in dealing with entity linking problem on Linked Data.

Acknowledgments. The work is supported by the Natural Science Foundation of Jiangsu Province under Grant BK20140643 and the National Natural Science Foundation of China under grant No. 61502095.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. IJSWIS **5**(3), 1–22 (2009)
2. Zaveri, A., Kontokostas, D., Sherif, M.A., Bühmann, L., Morsey, M., Auer, S., Lehmann, J.: User-driven quality evaluation of DBpedia. In: 9th International Conference on Semantic Systems (I-SEMANTICS 2013), pp. 97–104 (2013)
3. Suchanek, F., Kasneci, G., Weikum, G.: Yago: a large ontology from wikipedia and wordnet. J. Web Semant. **6**(3), 203–217 (2008)
4. Miller, G.A.: WordNet: a lexical database for english. Commun. ACM **38**(11), 39–41 (1995)
5. Bunescu, R., Pasca, M.: Using encyclopedic knowledge for named entity disambiguation. In: Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, pp. 9–16 (2006)
6. Cucerzan, S.: Large-scale named entity disambiguation based on wikipedia data. In: Proceedings of EMNLP-CoNLL, pp. 708–716 (2007)
7. Milne, D., Witten, I.H.: Learning to link with wikipedia. In: Proceedings of the 17th Conference on Information and Knowledge Management, pp. 509–518 (2008)
8. Ratinov, L., Roth, D., Downey, D., Anderson, M.: Local, global algorithms for disambiguation to wikipedia. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 1375-1384 (2011)
9. Cilibrasi, R.L., Vitanyi, P.M.B.: The google similarity distance. IEEE Trans. Knowl. Data Eng. **19**(3), 370–383 (2007)
10. Shen, W., Wang, J., Luo, P., Wang, M.: LINDEN: linking named entities with knowledge base via semantic knowledge. In: Proceedings of WWW, pp. 449–458 (2012)
11. Ganea, O., Ganea, M., Lucchi, A., Eickhoff, C., Hofmann, T.: Probabilistic bag-of-hyperlinks model for entity linking. In: Proceedings of WWW (2016)
12. Berant, J., Chou, A., Frostig, R., Liang, P.: Semantic parsing on Freebase from question-answer pairs. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1533–1544 (2013)

Applications of Semantic Technologies

Hypercat RDF: Semantic Enrichment for IoT

Ilias Tachmazidis[1](✉), John Davies[2], Sotiris Batsakis[1], Grigoris Antoniou[1], Alistair Duke[2], and Sandra Stincic Clarke[2]

[1] University of Huddersfield, Huddersfield, UK
i.tachmazidis@hud.ac.uk
[2] British Telecommunications, Ipswich, UK

Abstract. The rapid growth of sensor networks and smart devices has led to the generation of an increasing amount of information. Such information typically originates from various sources and is published in different formats. One of the key prerequisites for the Internet of Things (IoT) is interoperability. The Hypercat specification defines a lightweight JSON-based hypermedia catalogue, and is tailored towards the existing needs of industry. In this work, we propose a semantic enrichment of Hypercat, defining an RDF-based catalogue. We propose an ontology that captures the core of the Hypercat RDF specification and provides a mapping mechanism between existing JSON and proposed RDF properties. Finally, we propose a new type of search, called *Semantic Search*, which allows SPARQL-like queries on top of semantically enriched Hypercat catalogues and discuss how this semantic approach offers advantages over what was previously available.

1 Introduction

In 2014, Innovate UK (the UK's innovation agency) funded the Internet of Things Ecosystem Demonstrator programme. Eight industry-led projects were funded to deliver IoT 'clusters', each centred around a data hub to aggregate and expose data feeds from multiple sensor types.

A major objective of the programme was to address interoperability and this led to Hypercat, a standard for representing and exposing Internet of Things data hub catalogues [6] over web technologies, to improve data discoverability and interoperability. The idea is to enable distributed data repositories (hubs) to be used jointly by applications through making it possible to query their catalogues in a uniform machine-readable format. This enables applications to identify and access the data they need, whatever the data hub in which they are held.

As described in the specification of Hypercat (Beart et al. [3]), this is achieved through employing the same principles on which linked data and the web are built: data accessible through standard web protocols and formats (HTTPS, JSON, REST); the identification of resources through URIs; and the establishment of common, shared semantics for the descriptors of datasets. From this perspective, Hypercat represents a pragmatic starting point to solving the issues

© Springer International Publishing AG 2016
Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 273–286, 2016.
DOI: 10.1007/978-3-319-50112-3_21

of managing multiple data sources, aggregated into multiple data hubs, through linked data and web approaches. It incorporates a lightweight, JSON-based approach based on a technology stack used by a large population of web developers and as such offers a very low barrier to entry.

Each Hypercat catalogue lists and annotates any number of URIs (which typically identify data sources), each having a set of relation-value pairs (metadata) associated with it. In this way, Hypercat allows a server to provide a set of resources to a client, each with a set of semantic annotations. Importantly, there is only a small set of core mandatory metadata relations which a valid Hypercat catalogue must include, thus implementers are free to use any set of annotations to suit their needs. A Hypercat developer community is emerging, with open source tools becoming available[1]. Hypercat provides a standard, machine-processable means for resource discovery, which enables an interoperable ecosystem.

The complexity and diversity of IoT data sets is one of the main reasons why they have emerged as a key use case for linked data and semantic technologies recently[2]. Linked data enable the integration of data into a common, browsable and accessible knowledge graph, while leaving data distributed and managed in different systems, under the control of different contributors. The use of linked data technologies has been effective in many cases where information from different sources needs to be put together in a generic way, to enable a variety of applications, without the need to encode the constraints of the applications in the data model. Semantic web technologies add to this the ability to apply meaningful data models (ontologies) both to improve interoperability between systems, and to enable improved data analysis (see e.g. Lecue et al. [8]). It is therefore natural to consider how the Hypercat specification could be serialised in a semantic language and to investigate the benefits that could accrue from such a materialisation and that is the subject of this paper. One can envisage a more expressive, richer catalogue where data policies/licences, as well as the data flows that relate to them (see d'Aquin et al. [5]) are represented as machine readable information, enabling the implementation of inference rules to support automated reasoning in tasks such as data discovery and policy validation. This can be achieved using Semantic Web technologies [1] such as RDF and OWL that allow for representation of the meaning of data.

In this paper, we propose a semantic enrichment of the Hypercat specification that further increases interoperability by defining an RDF-based catalogue. Catalogue information is published based on a well-defined ontology that: (a) captures the core of the Hypercat RDF specification and (b) provides a mapping mechanism between existing JSON and proposed RDF properties. We describe how existing Hypercat JSON catalogues can be systematically translated into Hypercat RDF catalogues. We then propose a new type of search,

[1] https://hypercatiot.github.io/.

[2] See for example the "Semantic Cities" - http://research.ihost.com/semanticcities14/ - series of workshops.

which allows SPARQL-like queries on top of semantically enriched Hypercat catalogues that capture the semantic hierarchy of classes and properties.

The rest of the paper is organized as follows. Section 2 provides background on Semantic Web technologies. Section 3 presents the current Hypercat 3.00 specification. Section 4 introduces the Hypercat ontology, while Sect. 5 describes the translation of a JSON-based catalogue into an RDF-based catalogue. Section 6 presents the Hypercat RDF specification, while Sect. 7 introduces *Semantic Search*. We conclude in Sect. 8.

2 Background

The Semantic Web [1] evolved out of the Web with the aim to represent Web content in a form that is machine understandable and processable. Today, Web content, in HTML format, retrieved using search engines is typically suitable for human consumption, while content that is generated automatically from databases is usually presented without the original structural information of a given database. Formats such as JSON[3] are used for data exchange in a structured way, enabling machines to parse and generate data. However, even these formats do not address the following problem: the meaning of Web content is not machine-accessible. Semantic Web technologies are used to represent the semantics of Web content in a machine readable form in order to apply intelligent methods (i.e., reasoning) and automate tasks that are currently handled manually by users. Automating tasks is even more important today for the proliferation of connected devices that are part of the *Internet of Things* (IoT) [7].

Machine readable semantics of concepts of an application domain can be defined using Semantic Web standards. Specifically, an *ontology* is an explicit and formal specification of a conceptualization. An ontology consists of definitions of concepts (classes of objects) of the domain and relationships between these concepts (e.g., class hierarchies), and can be defined using the Web Ontology Language *OWL* [2]. OWL is a W3C standard and the current version is OWL 2[4]. Facts about application domain objects and their relations can be asserted using the RDF[5] format. Using RDF, Web resources are connected using a labelled graph representation, and simple ontologies containing descriptions of these resources can be defined using RDF Schema or RDF/S[6].

Using Semantic Web standards such as RDF and OWL, Web resources are represented with machine readable semantics, allowing for automatically interfering implied facts about these resources. Retrieving information represented using RDF format can be achieved using the SPARQL query language[7], which is a W3C standard and the current version is SPARQL 1.1[8]. Furthermore, reasoning and querying can be combined for retrieving not only explicitly asserted

[3] http://www.json.org/.
[4] http://www.w3.org/TR/owl2-overview/.
[5] http://www.w3.org/RDF/.
[6] http://www.w3.org/TR/rdf-schema/.
[7] http://www.w3.org/TR/rdf-sparql-query/.
[8] http://www.w3.org/TR/sparql11-query/.

facts, but also implied facts based on asserted facts and concept definitions and their relations into an ontology. Automatic inference and retrieval is very important when data is voluminous and changes fast (e.g., streaming data) which is a typical case in Internet of Things application scenarios.

3 Hypercat 3.00 Specification

In this section, we provide the basic notions of the Hypercat 3.00 specification[9]. Hypercat is a lightweight JSON-based hypermedia catalogue format for exposing collections of URIs, with each URI having any number of RDF-like triple statements about it.

By definition, a Hypercat catalogue is a file representing an unordered collection of resources on the web, with each item in the catalogue referring to a single resource by its URI. Thus, a Hypercat catalogue may expose a collection of resources, such as data feeds, and provide links to external Hypercat catalogues. Although the definition of a catalogue within a catalogue is not allowed, catalogues may be linked by referring to other catalogue URIs. In addition, a given catalogue may provide metadata about itself and each catalogue item.

The structure of a Hypercat catalogue is defined based on a *Catalogue Object*, which is a JSON object. A given *Catalogue Object* contains the following properties: (a) *items*, which is a list of items (JSON array of zero or more *Item Objects*), and (b) *catalogue-metadata*, which is an array of *Metadata Objects* describing the catalogue object (JSON array of *Metadata Objects*).

An *Item Object* (from the *items* array) is a JSON object, which contains the following properties: (a) *href*, which is an identifier for the resource item (URI as a JSON string), and (b) *item-metadata*, which is an array of *Metadata Objects* describing the resource item (JSON array of *Metadata Objects*).

A *Metadata Object* is a JSON object, which describes a single relationship between the parent object (either the catalogue or catalogue item) and some other entity or concept denoted by a URI, such a relationship is applicable to both the catalogue itself and each catalogue item. A *Metadata Object* contains the following properties: (a) *rel*, which is a relationship between the parent object and a target noun, expressed as a predicate (URI of a relationship as a JSON string), and (b) *val*, the entity (noun) to which the *rel* property applies (JSON string, optionally the URI of a concept or entity).

The structure that is described above constitutes the basic core of any given Hypercat catalogue. However, the Hypercat 3.00 specification defines a far more detailed model compared to the aforementioned description. Thus, in the remainder of this paper, we explore each aspect of the Hypercat 3.00 specification while providing the corresponding semantically enriched solution based on an OWL ontology, which is asserted in RDF format.

[9] http://www.hypercat.io/uploads/1/2/4/4/12443814/hypercat_specification_3.00rc1-2016-02-23.pdf.

4 Hypercat Ontology

In this section, we provide the definition of an OWL ontology that captures the aforementioned Hypercat structure, thus providing a translation mechanism from a JSON-based to an RDF-based catalogue. The proposed Hypercat ontology is available with the uri

http://portal.bt-hypercat.com/ontologies/hypercat

and captures the core properties that would enable the development of RDF-based catalogues. Namespaces for the Hypercat ontology can be written prefixing concepts and properties with "hypercat:". Currently, it is part of an IoT Data Hub[10], while as a next step it will be proposed to the Hypercat community for standardization. We believe that providing catalogues in RDF, based on a well-defined ontology, would further increase interoperability and offer intelligent reasoning capabilities.

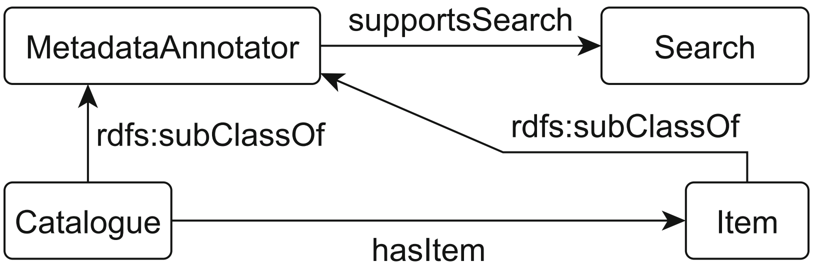


Fig. 1. The hypercat ontology.

The Hypercat ontology consists of a class hierarchy that is depicted in Fig. 1, a range of properties that are included in Tables 1, 2, and 3, and a set of individuals that are described in Table 4. The core hierarchy is rich enough to capture the corresponding constructs of the JSON-based catalogue while providing the flexibility for further extensions.

As described above, a *Metadata Object* is applicable to both the catalogue itself and each catalogue item. Thus, we define class *MetadataAnnotator*, which captures metadata properties that are applicable to both *Catalogue Objects* and *Item Objects*. Note that the Hypercat 3.00 specification defines certain properties as applicable to either *Catalogue Objects* or *Item Objects*, but not both, as such properties cannot be included in the definition of class *MetadataAnnotator*.

Subsequently, class *MetadataAnnotator* has two subclasses, namely class *Catalogue* and class *Item*. In essence, class *Catalogue* models a *Catalogue Object* defining properties that are applicable only to the catalogue's metadata, while class *Item* models an *Item Object* defining properties that are applicable only

[10] http://portal.bt-hypercat.com/.

Table 1. *MetadataAnnotator* properties mapped to existing JSON properties

JSON-based	RDF-based
urn:X-hypercat:rels:hasDescription:en	rdfs:comment
urn:X-hypercat:rels:supportsSearch	hypercat:supportsSearch
urn:X-hypercat:rels:isContentType	hypercat:isContentType
urn:X-hypercat:rels:hasHomepage	hypercat:hasHomepage
urn:X-hypercat:rels:containsContentType	hypercat:containsContentType
urn:X-hypercat:rels:hasLicense	hypercat:hasLicense
urn:X-hypercat:rels:acquireCredential	hypercat:acquireCredential

Table 2. *Catalogue* properties mapped to existing JSON properties

JSON-based	RDF-based
urn:X-hypercat:rels:eventSource	hypercat:eventSource
urn:X-hypercat:rels:hasRobotstxt	hypercat:hasRobotstxt

to an item's metadata. In order to build a complete RDF-based catalogue, class *Catalogue* is related to class *Item* through property *hasItem*, which means that a given *Catalogue* may contain a collection of *Items*. Class *Search* models the various types of searches that are supported by a given *Catalogue*, with the two classes being related through property *supportsSearch*. The set of currently supported searches is defined through individuals of class *Search*. Note that the details of supported search types will be covered below.

5 Hypercat JSON to Hypercat RDF

Prior to exploring each aspect of the Hypercat 3.00 specification, we provide a mapping of existing JSON properties and proposed RDF relations/individuals for each defined class. Tables 1, 2, 3 and 4 can be used in order to create a translator from a JSON-based catalogue to an RDF-based catalogue. Note that this work is in line with recent developments in the Semantic Web community, namely the translation of JSON data to RDF using RML[11].

Table 3. *Item* properties mapped to existing JSON properties

JSON-based	RDF-based
urn:X-hypercat:rels:accessHint	hypercat:accessHint
urn:X-hypercat:rels:lastUpdated	hypercat:lastUpdated

[11] http://rml.io/.

Table 4. *Search* individuals mapped to existing JSON properties

JSON-based	RDF-based
urn:X-hypercat:search:simple	hypercat:SimpleSearch
urn:X-hypercat:search:geobound	hypercat:GeoboundSearch
urn:X-hypercat:search:lexrange	hypercat:LexrangeSearch
urn:X-hypercat:search:multi	hypercat:MultiSearch
urn:X-hypercat:search:prefix	hypercat:PrefixSearch

For example, for the JSON-based catalogue with the uri

http://portal.bt-hypercat.com/cat

the following *catalogue-metadata*, namely metadata about the catalogue itself:

"rel":"urn:X-hypercat:rels:isContentType"
"val":"application/vnd.hypercat.catalogue+json"

will be translated into the following RDF triple, using Table 1:

<http://portal.bt-hypercat.com/cat-rdf>
<http://portal.bt-hypercat.com/ontologies/hypercat#isContentType>
"application/n-triples".

Note that we choose to represent our RDF-based catalogue in N-Triples[12] format. Thus, we define that our RDF-based catalogue is of MIME type "application/n-triples". In addition, the URI of the catalogue needs to be changed from http://portal.bt-hypercat.com/cat to

http://portal.bt-hypercat.com/cat-rdf

since the RDF-based catalogue will be stored in a different location.

This translation pattern applies to all properties of Tables 1 and 2 except for property *supportsSearch* where, for the JSON-based catalogue, both *rel* and *val* contain URIs. Thus, for the JSON-based catalogue http://portal.bt-hypercat.com/cat, the following *catalogue-metadata*:

"rel":"urn:X-hypercat:rels:supportsSearch"
"val":"urn:X-hypercat:search:simple"

will be translated in the following RDF triple, using Tables 1 and 4:

<http://portal.bt-hypercat.com/cat-rdf>
<http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch>
<http://portal.bt-hypercat.com/ontologies/hypercat#SimpleSearch>.

Note that Table 4 provides the mapping from JSON-based URIs that represent the various types of searches to the RDF-based individuals of class *Search*, which represent semantically the various types of searches.

[12] http://www.w3.org/TR/n-triples/.

Finally, the following *Item Object* (in *items*):

"href":"http://api.bt-hypercat.com/sensors/feeds/UUID"

and "item-metadata" containing

"rel":"urn:X-hypercat:rels:lastUpdated"
"val":"2015-12-01T00:00:00Z"

will be translated in the following RDF triple, using Table 3:

<http://api.bt-hypercat.com/sensors/feeds/UUID>
<http://portal.bt-hypercat.com/ontologies/hypercat#lastUpdated>
"2015-12-01T00:00:00Z".

For properties that are not defined in Tables 1, 2, 3 and 4, Hypercat RDF publishers are encouraged to develop their own OWL ontology by extending the one proposed in this work. Thus, the newly defined ontology would capture the meaning of their catalogue, by defining additional properties, and would enable the full translation of their JSON-based catalogue into an RDF-based catalogue. Alternatively, Hypercat RDF publishers could translate and publish only standardized properties (using Tables 1, 2, 3 and 4). Even though in this case the RDF-based catalogue would contain less information compared to the JSON-based catalogue, it would still be a valid Hypercat RDF catalogue.

In order to fully translate the JSON-based catalogue of the BT Data Hub, an extension of the core ontology has been developed and made available with the uri

http://portal.bt-hypercat.com/ontologies/bt-hypercat

Namespaces for the BT Hypercat ontology can be written prefixing concepts and properties with "bt-hypercat:". In this way, "item metadata" containing

"rel":"urn:X-bt:rels:feedTitle",
"val":"Met Office Datapoint Observations"

that could not be translated using the core ontology, can now be translated in the following RDF triple, using the BT Hypercat ontology:

<http://api.bt-hypercat.com/sensors/feeds/UUID>
<http://portal.bt-hypercat.com/ontologies/bt-hypercat#feed_title>
"Met Office Datapoint Observations".

6 Hypercat RDF Specification

In this section, we examine each aspect of the Hypercat RDF specification by following the structure of (the JSON-based) Hypercat 3.00 specification.

Hypercat File Format Specification: We have already provided a description of the OWL ontology and how an RDF-based catalogue should be developed. In addition, we have presented all standard semantic properties and their correspondence to JSON properties. However, we need to elaborate on several aspects that have not been covered. Thus, each instance of class *MetadataAnnotator* must include the mandatory property *rdfs:comment*, and may include the optional properties *hypercat:isContentType*, *hypercat:hasHomepage*, *hypercat:containsContentType* and *hypercat:supportsSearch*. In addition, each instance of class *Catalogue* must include the mandatory property *hypercat:isContentType*.

As described above, we define each RDF-based catalogue in N-Triples format. Thus, RDF-based catalogues are of MIME type "application/n-triples". In terms of extensibility, we follow the Hypercat 3.00 specification:

– An unknown metadata relationship should be ignored.
– New search method supported by a catalogue server may be added.
– Human readable descriptions may be added in any language.
– Old style catalogues may point to new style and vice versa without version ambiguity.
– A catalogue may contain any number of other properties and classes as developers see fit as long as they are defined in an ontology.

Hypercat Server API Specification: An RDF-based Hypercat server follows the JSON-based Hypercat server specification with several minor adjustments. Every RDF-based Hypercat server must provide a publicly readable "*/cat-rdf*" endpoint serving a Hypercat document asserted in RDF. Requests to an RDF-based Hypercat server, such as insert and delete, can be implemented in a similar fashion as for a JSON-based Hypercat server, while the response will be an RDF-based catalogue instead of a JSON-based catalogue.

A Simple Search Mechanism is implemented in a similar way on top of RDF-based catalogues. For a given RDF-based catalogue http://portal.bt-hypercat.com/cat-rdf, we can advertise that it supports the simple search mechanism by including the following triple:

<http://portal.bt-hypercat.com/cat-rdf>
<http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch>
<http://portal.bt-hypercat.com/ontologies/hypercat#SimpleSearch>.

All query parameters must be URL encoded and are all optional. If multiple search parameters are supplied, the server must return the intersection of items where search parameters match in a single item, combining parameters with boolean AND.

Simple search supports the following parameters: (a) *s*, which is the N-Triple's subject, (b) *p*, which is the N-Triple's predicate, and (c) *o*, which is the N-Triple's object. Note that each parameter should be inserted in exactly the same form as it would appear in the RDF-based catalogue. For example, we could query a given catalogue based on the following query strings (even though queries must be URL encoded, for readability we present them as plain text):

?s=<http://portal.bt-hypercat.com/ItemID>
?p=<http://portal.bt-hypercat.com/ontologies/hypercat#isContentType>
?p=<http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch>&
o= <http://portal.bt-hypercat.com/ontologies/hypercat#SimpleSearch>
?o="2015-12-01T00:00:00Z"

Hypercat Subscription: The Hypercat 3.00 specification describes a simple subscription system, providing an API for polling catalogues. A client subscribed to a stream of events from a Hypercat server will receive a stream of events, with each event containing an event name and a body. By first fetching a catalogue and then accumulating catalogue events, a client may keep a synchronised local copy of a given catalogue.

The existing Hypercat subscription mechanism can be used, with minor changes, for an RDF-based catalogue. An RDF-based catalogue which can be used for subscribing to, must be annotated with the property *hypercat:eventsource*. For events concerning a specific catalogue item within a catalogue, the event name is the (unique) N-Triple's subject of all RDF triples for the specified item. Moreover, the event body for an item update event is a set of N-Triples (related to the specified item), while for an item deletion event, the event body is an empty string.

Hypercat Resource Subscription: Hypercat provides the ability to link to resources through URIs. Such resources may contain real-time data required by various applications. Use-cases where client applications require real-time data feeds from devices, hubs or other services are very common in the field of IoT. A possible solution where all data are placed directly into Hypercat catalogues was considered in the past, but was inapplicable due to the simple data model of a JSON-based catalogue. On the other hand, an RDF-based catalogue could serve as a solution for importing data directly into a Hypercat catalogue, given that imported data is semantically enriched and is expressed as N-Triples. However, we believe that such a decision should be part of a wider discussion within the Hypercat community since incorporating data into a catalogue will result in RDF-based catalogues providing a functionality that will not be supported by a JSON-based catalogue.

Hypercat Signing: Hypercat Signing for an RDF-based catalogue remains an open issue as *JSON Web Signature* is not applicable to RDF. Thus, we defer Hypercat signing, based on a well-accepted standard, to future work. However, the same intuition is applicable to an RDF-based catalogue, namely we can create a signature for the entire catalogue or a specific item based on all triples in the catalogue or triples that correspond to a specific item respectively.

Hypercat Security Access Hints: Although systems supporting Hypercat should provide open data with traversable links whenever possible, many systems will potentially provide resources or catalogues only to authenticated clients. Thus, where resources or catalogues are discoverable, but not accessible without authentication, authentication information can be provided to clients.

An item that requires authentication, should point at a machine or human readable description of the authentication method, using property *hypercat:accessHint*. Note that in case multiple *hypercat:accessHint* declarations are present, the client should assume that the resource can be accessed using multiple authentication systems.

Hypercat Security Credential Acquisition: A Hypercat catalogue may support various methods of acquiring access credentials in order to access catalogues and resources. Thus, a catalogue or item, should point at a self-describing web page or resource helping the client acquire credentials, using property *hypercat:acquireCredential*. Note that when multiple *hypercat:acquireCredential* declarations are present, the client should assume that credentials can be acquired in multiple ways.

Hypercat Geographic Bounding Box Search: A geographic search allows for filtering items that fall within a geographic region, which is defined by a bounding box. A given RDF-based catalogue may reuse the following properties:

http://www.w3.org/2003/01/geo/wgs84_pos#lat
http://www.w3.org/2003/01/geo/wgs84_pos#long

which are well-defined by an external ontology. In addition, a given RDF-based catalogue can inform a client that geographic bounding box search is supported, by including the following triple:

<http://portal.bt-hypercat.com/cat-rdf>
<http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch>
<http://portal.bt-hypercat.com/ontologies/hypercat#GeoboundSearch>.

Geographic search supports the following parameters: (a) *geobound-minlat*, which is the inclusive lower bound of latitude of bounding box, (b) *geobound-maxlat*, which is the inclusive upper bound of latitude of bounding box, (c) *geobound-minlong*, which is the inclusive lower bound of longitude of bounding box, and (d) *geobound-maxlong*, which is the inclusive upper bound of longitude of bounding box. Geographic search queries are submitted by providing the aforementioned parameters.

Hypercat Lexicographic Range Search: Lexicographic search allows searching for items which, when sorted lexicographically, fall between a minimum and maximum. A given RDF-based catalogue can inform a client that lexicographic range search is supported, by including the following triple:

<http://portal.bt-hypercat.com/cat-rdf>
<http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch>
<http://portal.bt-hypercat.com/ontologies/hypercat#LexrangeSearch>.

The property *hypercat:lastUpdated* could be used for dates and time.

Lexicographic search supports the following parameters: (a) *lexrange-p*, which is the N-Triple's predicate to search on, (b) *lexrange-min*, which is the lower bound of range to return (inclusive), and (c) *lexrange-max*, which is the upper bound of range to return (non-inclusive). Lexicographic search queries are submitted by providing the aforementioned parameters.

Hypercat Robots Exclusion Search: A given Hypercat RDF catalogue can contain information for the client about an associated *robots.txt* file, which is used by websites to communicate with web crawlers and other web robots, using property *hypercat:hasRobotstxt*. Note that if a robots exclusion file is provided, it must be located at the *BASE_URL* of a catalogue and it must be named *robots.txt*, namely N-Triple's object should contain a URI of the form "[BASE_URL]/robots.txt".

Hypercat Multi-Search: Hypercat supports several different search extensions, which are mainly variations of the simple search, and thus, only allow for simple interactions with a catalogue. Combining geographic search and lexicographic range search could be done by submitting two independent queries and then processing the results accordingly. However, such a solution is inefficient.

Multi-search allows a client to combine single or multiple search mechanisms supported by a server so as to retrieve only the items of interest. A given RDF-based catalogue can inform a client that multi-search is supported, by including the following triple:

<http://portal.bt-hypercat.com/cat-rdf>
<http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch>
<http://portal.bt-hypercat.com/ontologies/hypercat#MultiSearch>.

In order to retain compatibility between JSON-based and RDF-based catalogues, a multi-search query, for a given RDF-based catalogue, could be submitted as a single JSON object. Multi-search supports the following parameters: (a) *query*, which is a JSON string, holding a URL query string as passed to underlying search mechanism, (b) *intersection*, which is a JSON array of objects that could contain *query*, *intersection* or *union*, and (c) *union*, which is a JSON array of objects that could contain *query*, *intersection* or *union*. Searches may be nested to allow complex mixing of union and intersection.

Hypercat Prefix Match Search: Prefix match search allows searching for items where the N-Triple's object specified in the query is a prefix match of the N-Triple's object in a triple describing a catalogue item. As with simple search, any N-Triple's predicate can be used. A given RDF-based catalogue can inform a client that prefix match search is supported, by including the following triple:

<http://portal.bt-hypercat.com/cat-rdf>
<http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch>
<http://portal.bt-hypercat.com/ontologies/hypercat#PrefixSearch>.

If multiple search parameters are supplied, the server must return the intersection of items where search parameters match a single item, combining parameters with boolean AND. Prefix match search supports the following parameters: (a) *prefix-s*, which is a prefix of the N-Triple's subject, (b) *prefix-p*, which is a prefix of the N-Triple's predicate, and (c) *prefix-o*, which is a prefix of N-Triple's object.

Hypercat Linked Data rel: The newly introduced Hypercat JSON *rel* for specifying that the item at hand is an instance of an RDF class, namely http://www.w3.org/1999/02/22-rdf-syntax-ns#type is part of the RDF concepts vocabulary. Thus, we do not need to include a new property.

Hypercat License rel: Hypercat catalogues or linked resources may be available under a specific license. Thus, in order to allow clients to determine the license under which the data is released, we specify the property *hypercat:hasLicense*, which should point at a machine or human readable version of a license. Where multiple *hypercat:hasLicense* declarations are present, the client should assume that the resource is available under multiple licenses.

7 Semantic Search

Semantic search allows SPARQL-like queries on top of semantically enriched Hypercat catalogues, providing a searching mechanism that captures the underlying semantic hierarchy. Given a query where *rel* (or N-Triple's predicate) is *rdf:type* and *val* (or N-Triple's object) is *bt-hypercat:Feed*, with BT catalogue's ontology defining that *bt-hypercat:SensorFeed* is subclass of *bt-hypercat:Feed*, semantic search will return all catalogue items that are instances of both *bt-hypercat:Feed* and *bt-hypercat:SensorFeed* - this would not be possible without the use of the Hypercat RDF to encode the subclass relationship.

A given JSON-based catalogue can inform a client that semantic search is supported, by including the following *rel val* pair:

"rel" : "urn:X-hypercat:rels:supportsSearch"
"val" : "urn:X-hypercat:search:semantic"

while a given RDF-based catalogue must include the following triple:

<http://portal.bt-hypercat.com/cat-rdf>
<http://portal.bt-hypercat.com/ontologies/hypercat#supportsSearch>
<http://portal.bt-hypercat.com/ontologies/hypercat#SemanticSearch>.

If multiple search parameters are supplied, the server must bind them to a single triple pattern and run a SPARQL query, including reasoning based on both catalogue's ontology and catalogue itself. Semantic search supports the following parameters for JSON-based (resp. RDF-based) catalogues: (a) *sem-href* (resp. *sem-s*), which is a resource URI (resp. the N-Triple's subject), (b) *sem-val* (resp. *sem-p*), which is a semantic metadata relation (resp. N-Triple's predicate), and (c) *sem-rel* (resp. *sem-o*), which is a semantic metadata value (resp. N-Triple's object).

In terms of implementation, reasoning over the given Hypercat ontology can be performed using standard reasoners such as Pellet[13] and HermiT[14],

[13] http://clarkparsia.com/pellet/.
[14] http://hermit-reasoner.com/.

while querying can be based on the query engine of Apache Jena[15]. Alternatively, an OBDA approach proposed by Botoeva et al. [4] can be considered in order to perform SPARQL queries over JSON data.

8 Conclusion

In this work, we presented a semantic enrichment of the Hypercat specification, which allows the definition of an RDF-based catalogue. We proposed an ontology that captures the core of the Hypercat RDF specification. In addition, we showed how existing JSON-based catalogues can be translated into RDF-based catalogues in an automated fashion. Finally, we proposed a new type of search, called *Semantic Search*, which allows SPARQL-like queries on top of semantically enriched catalogues.

In future work, we plan to propose and standardize the Hypercat RDF specification by working closely with the Hypercat community. In addition, we intend to collaborate with existing partners in order to provide a publicly available converter from JSON-based to RDF-based catalogues, and a publicly available implementation of *Semantic Search*. In this way, richer Hypercat catalogues will provide a higher degree of interoperability.

References

1. Antoniou, G., van Harmelen, F.: A Semantic Web Primer. Cooperative Information Systems, 2nd edn. The MIT Press, Cambridge (2008)
2. Antoniou, G., van Harmelen, F.: Web ontology language: OWL. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. IHIS, pp. 67–92. Springer, Heidelberg (2004)
3. Beart, O.: Hypercat 3.00 specification (2016)
4. Botoeva, E., Calvanese, D., Cogrel, B., Rezk, M., Guohui Xiao, O., Relational, B.: DBs: a study for MongoDB. In: Proceedings of the 29th International Workshop on Description Logics, Cape Town, South Africa, 22–25 April 2016 (2016)
5. d'Aquin, M., Adamou, A., Daga, E., Liu, S., Thomas, K., Motta, E.: Dealing with diversity in a smart-city datahub. In: ISWC, pp. 68–82 (2014)
6. Davies, J., Fisher, M.: Internet of things - why now? J. Inst. Telecommun. Prof. **7**(3), 36–42 (2015)
7. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
8. Lécué, F., Tucker, R., Bicer, V., Tommasi, P., Tallevi-Diotallevi, S., Sbodio, M.: Predicting severity of road traffic congestion using semantic web technologies. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8465, pp. 611–627. Springer, Heidelberg (2014). doi:10.1007/978-3-319-07443-6_41

15 https://jena.apache.org/index.html.

Enabling Spatial OLAP Over Environmental and Farming Data with QB4SOLAP

Nurefşan Gür[1(✉)], Katja Hose[1], Torben Bach Pedersen[1], and Esteban Zimányi[2]

[1] Department of Computer Science, Aalborg University, Aalborg, Denmark
{nurefsan,khose,tbp}@cs.aau.dk

[2] Department of Computer and Decision Engineering, Université Libre de Bruxelles, Bruxelles, Belgium
ezimanyi@ulb.ac.be

Abstract. Governmental organizations and agencies have been making large amounts of spatial data available on the Semantic Web (SW). However, we still lack efficient techniques for analyzing such large amounts of data as we know them from relational database systems, e.g., multi-dimensional (MD) data warehouses and On-line Analytical Processing (OLAP). A basic prerequisite to enable such advanced analytics is a well-defined schema, which can be defined using the QB4SOLAP vocabulary that provides sufficient context for spatial OLAP (SOLAP). In this paper, we address the challenging problem of MD querying with SOLAP operations on the SW by applying QB4SOLAP to a non-trivial spatial use case based on real-world open governmental data sets across various spatial domains. We describe the process of combining, interpreting, and publishing disparate spatial data sets as a spatial data cube on the SW and show how to query it with SOLAP operators.

1 Introduction

In late 2012, the Danish government joined the Open Data movement by making several raw digital data sets [3] freely available at no charge. These data sets span domains such as environmental data, geospatial data, business data from transport to tourism, fishery, forestry, and agriculture. GovAgriBus Denmark[1] was an initial effort in 2014 to make Danish government Open Data from various domains available as Linked Open Data (LOD) [2] on the Semantic Web in order to pose queries across domains. If the corresponding domains can be related through space and location, spatial attributes of these data sets become particularly interesting as we can derive spatial joins and containment relationships that were not encoded in the original data sets. Danish government organizations and agencies continue publishing data sets for new domains and update the corresponding data sets regularly on a yearly basis, which brings opportunities in querying the expanding spatial data with analytical perspectives on the Semantic Web. Responding to such queries is a complex task, which requires well-defined schemas to facilitate OLAP operations on the Semantic Web. QB4SOLAP [7]

[1] https://datahub.io/dataset/govagribus-denmark.

Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 287–304, 2016.
DOI: 10.1007/978-3-319-50112-3_22

aims to support intelligent multidimensional querying in SPARQL by providing context to SOLAP and its elements on the SW. However, QB4SOLAP has not been applied on complex real-world data yet. This could bring particular challenges with the use of real-world spatial data. In this paper, we address the challenging problem of multidimensional querying with SOLAP operations on the SW by applying QB4SOLAP on real-world open governmental data sets from various domains. These domains span from livestock farming to environment, where many of them have spatial information.

In this paper, we design a spatial data cube schema with data from livestock farming, environment, and geographical domains. Every data set is downloaded from different governmental sources in various formats. The downloaded data is prepared and conciliated with a spatial data cube schema in order to publish it on the SW with QB4SOLAP. We use the common SOLAP operators [8] on the spatial data cube for advanced analytical queries. These analytical queries give perspective on the use case data sets that are linked and published with QB4SOLAP as a unified spatial data cube. Having the use case data sets with spatial attributes also allows us to reveal patterns across the use case domains that were not possible before. We share our experiences with the best practices and methods together with the lessons learned. Finally, we show how to formulate and execute SPARQL queries with individual and nested SOLAP operations.

The remainder of this paper is structured as follows. Section 2 presents the background and motivation, Sect. 3 discusses related work and presents the state-of-the-art spatial data cubes on the SW. Section 4 presents the data sources for the use case while Sect. 5 describes how to annotate and publish the use case data as a spatial data cube on the SW. Section 6 presents SOLAP operators and their SPARQL implementation. Section 7 presents a brief overview of the process and reflects on the problems and improvements. Finally, Sect. 8 concludes the paper with an outlook to future work.

2 Background and Motivation

The Semantic Web supports intelligent querying via SPARQL with active inference and reasoning on the data in addition to capturing its semantics. Linked Open Data on the Semantic Web is an important source to support Business Intelligence (BI). Multidimensional data warehouses and OLAP are advanced analytical tools in analyzing complex BI data. State-of-the-art SW technologies support advanced analytics over *non-spatial* SW data. QB4SOLAP supports intelligent multidimensional querying in SPARQL by providing context to spatial data warehouses and its concepts. Variety of the data is an intriguing concept on both the Semantic Web and in complex BI systems. The variety of the data and heterogeneous representation formats (e.g., CSV, JSON, PDF, XML, and SHP) require underlying conceptualizations and data models with well-defined spatial (and temporal) dependencies, which can be modeled with QB4SOLAP in order to answer complex analytical queries. Complex queries cannot be answered from within one domain alone but span over multiple disciplines and various data sources. As a result, this paper is driven by the motivation of using QB4SOLAP as a proof of concept for spatial data warehouses on the Semantic Web by using open (government) data of various domains from different sources, which creates a non-trivial spatial use case.

3 State of the Art

Data warehouses and OLAP technologies have been successful for analyzing large volumes of data [1], including integrating with external data such as XML [16]. Combining DW/OLAP technologies with RDF data makes RDF data sources more available for interactive analysis. Kämpgen et al. propose an extended model [11] on top of the RDF Data Cube Vocabulary (QB) [4] for interacting with statistical linked data via OLAP operations directly in SPARQL. In OLAP4LD [10], Kämpgen *et al.* suggest enhancing query performance of OLAP operations expressed as SPARQL queries by using RDF aggregate views. The W3C published a list of RDF cube implementations [19]. However, they all have inherent limitations of QB and thus cannot support OLAP dimensions with hierarchies and levels, and built-in aggregate functions.

Etcheverry et al. [6] introduce QB4OLAP as an extended vocabulary based on QB, with a full MD metamodel, supporting OLAP operations directly over RDF data with SPARQL queries. Matei *et al.* [12] use QB and QB4OLAP as a basis to support OLAP queries in Graph Cube [20] with the IGOLAP vocabulary. Jakobsen *et al.* [9] study OLAP query optimization techniques over QB4OLAP data cubes. However, none of these approaches and vocabularies support *spatial* DWs.

QB4SOLAP(v1) [7] is the first attempt to model and query spatial DWs on the SW, and QB4SOLAP(v2) [8] is a foundation for spatial data warehouses and SOLAP operators on the SW, which is currently under submission with completely revised formal semantics of SOLAP operators and SPARQL query generations algorithms. QB4SOLAP is an extension of QB4OLAP with spatial concepts. The QB4OLAP vocabulary is compatible with the QB vocabulary. Therefore QB4SOLAP provides backward compatibility with other statistical or MD data cube vocabularies in addition to providing spatial context for querying with SOLAP. Figure 1 depicts the QB4SOLAP(v2) vocabulary. Capitalized terms with non-italic font represent RDF classes, capitalized terms with italic font represent RDF instances, and non-capitalized terms represent RDF properties. Classes in external vocabularies are depicted in light gray background and font. QB, QB4OLAP, and QB4SOLAP classes are shown with white, light gray, and dark gray backgrounds. Original QB terms are prefixed with `qb:`[2]. QB4OLAP and QB4SOLAP terms are prefixed with `qb4o:`[3] and `qb4so:`[4]. Spatial classes are prefixed with `geo:`[5], where the spatial extension to QB4SOLAP is based on the GeoSPARQL [15] standard from the Open Geospatial Consortium (OGC) for representing and querying geospatial linked data on the SW.

[2] RDF Cube: http://purl.org/linked-data/cube#.
[3] QB4OLAP: http://purl.org/qb4olap/cubes#.
[4] QB4SOLAP: http://w3id.org/qb4solap#.
[5] GeoSPARQL: http://www.opengis.net/ont/geosparql#.

QB4SOLAP is a promising approach for modeling, publishing, and querying spatial data warehouses on the SW. However, it has only been validated with a synthetic use case. Andersen et al. [2] consider publishing/converting open Danish governmental spatial data as Linked Open Data without considering the MD aspects of geospatial data. In this paper, however, we validate QB4SOLAP with a non-trivial use case, which is created as a spatial data cube from open Danish government spatial data. Furthermore, we show how to exploit multidimensional spatial linked data on the SW, which is not solely about adding semantics and linking disparate data sets on the SW, but also about enabling analytical queries by modeling them as spatial data cubes.

4 Source Data

In order create a spatial data cube of livestock holdings in Danish farms, we have gathered data that is published by different agencies in Denmark. We have found these domains to be particularly interesting as they represent a non-trivial use case that covers spatial attributes and measures, which can be modeled in a spatial data cube for multidimensional analysis. In the following we first give a brief overview of the flat data and their sources, and then represent the whole use case data set as a spatial data cube in Sect. 5.

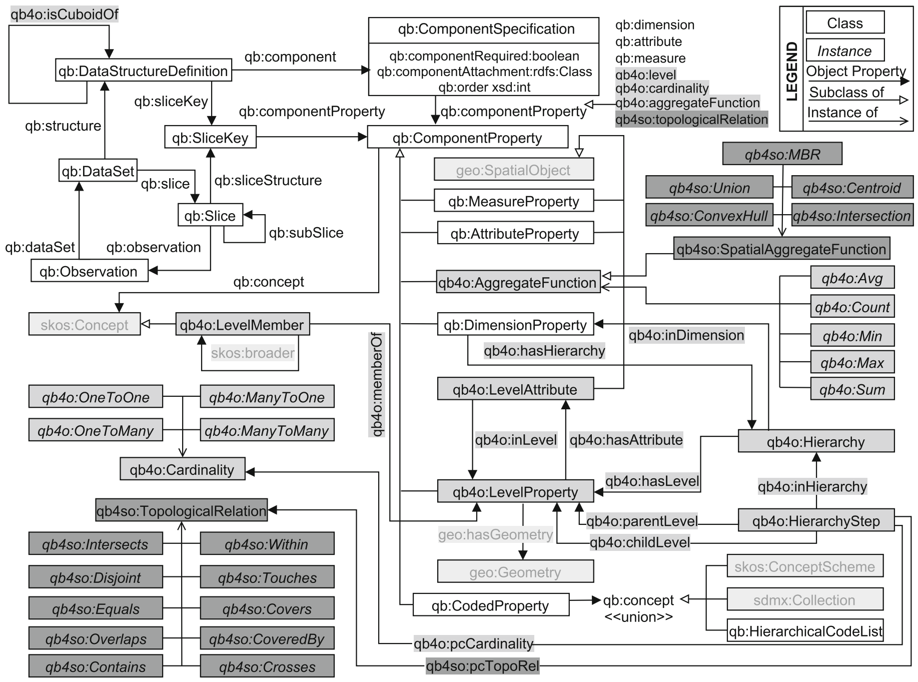


Fig. 1. QB4SOLAP vocabulary

The environmental protection agency under the Ministry of Environment and Food of Denmark regulates the livestock units (DE)[6] per area in order to keep nitrate leaching under control in vulnerable areas. Prohibition rules against the establishment of livestock farms and the siting of animal housing are determined with respect to livestock units and distance to specific natural habitats (e.g., ammonia vulnerable areas, water courses, and water supply facilities etc.) [13].

Livestock Farming (CHR) Data. The Ministry of Environment and Food of Denmark (http://en.mfvm.dk) publishes the central husbandry (livestock) registry (CHR) data, which is the central database used for registration of holdings and animals. We refer to this set of data as *CHR data*. We have downloaded several relevant data sets from http://jordbrugsanalyser.dk in livestock farming domain. The CHR data collection is downloaded in SHP format as 6 data sets, where each data set represents the state of the farms for a year between the years 2010 to 2015. SHP format is used for *shapefile*s, which store geometry information of the spatial features in a data set. In each shapefile, there is information about more than 40,000 farms. In total, the CHR data collection contains around 240,000 records. Farm locations are given as `(X,Y)` point coordinates. Each data set has 24 attributes in which the important ones are: *CHR - Central Husbandry (animal) Registry (holding) number, CVR - Central Company Registry number (owner company of the holding), DE (Livestock unit), Address of the holding (Postnr and Commune), Geographical position of the holding (X and Y coordinates), Different type of normalized herds, Number of animals for each herd, Animal code and label, Animal usage code and purpose.*

Environmental Data. Public environmental data is published on Denmark's environment portal http://www.miljoeportal.dk/, where we can find information about nitrate catchment areas and vulnerable sites. The soil measurements contain data from 2008 to 2015 [14]. We downloaded the data sets in SHP format, which have recently become available on the portal. The files record measurements of the soil quality across Denmark. The environmental data collection contains 3 data sets about nitrogen reduction potentials and phosphor and nitrate classifications of the soil. Temporal validity of the soil measurement data is recorded in the attributes with timestamps. Each data set keeps records of polygon areas. In total, the environmental data collection contains around 30,000 records. Datasets have attribute fields about the area of the polygons, CVR number of the data provider agency or company, responsible person name, etc. The important attributes, which record the soil measurement data are: *Nitrate class type, Nitrogen reduction potentials, Phosphor class type.*

Geographical (Regions) Data. The primary use case data is built around livestock farming (CHR) and environmental data as mentioned above. In sorder to pursue richer analysis upon this use case we enrich the spatiality of the use

[6] Livestock units are used to produce statistics describing the number of livestocks in farms.

case data by adding two geographical data sets; parishes and drainage areas of Denmark. These data sets are spatially and topically relevant since we have found pre-aggregated maps created by the Ministry of Environment and Food of Denmark at parish and drainage area levels for livestock farming data. We downloaded parishes and drainage areas of Denmark as SHP files from http://www.geodata-info.dk/. The total number of records of the geographical data collection are 2,300. These data sets have attribute fields such as: *Drainage area name, Parish name, Total area, etc.*

Central Company Registry (CVR) Data. Danish companies, agencies and industries are registered in the Central Company Register (CVR). Every livestock holding is owned by a company and has a CVR number. Environmental data also records the CVR number of the corresponding data provider agencies. Through this CVR number, we can access detailed information of the companies and contact details of the responsible people. This collection allows evaluating interesting queries with the selected domains given above. The CVR data is published at http://cvr.dk and can be accessed via a web service with a Danish social security number log-in. We accessed and downloaded only the data in CSV format that are accredited for publishing. This data includes attributes such as: *Company name, Phone number, and Address etc.*

5 Publishing Spatial Data Cubes with QB4SOLAP

The QB4SOLAP vocabulary allows to define *cube schemas* and *cube instances*. A cube schema defines the structure of a cube as an instance of the class `qb:DataStructureDefinition` in terms of dimension levels, measures, aggregation functions (e.g., SUM, AVG, and COUNT) on measures, spatial aggregation functions[7] on spatial measures, fact-level cardinality relationships, and topological relationships. The properties used to express these relationships are: `qb4o:level`, `qb:measure`, `qb4o:aggregateFunction`[8], `qb4o:cardinality`, and `qb4so:topologicalRelation` respectively (Fig. 1). These schema level metadata are used to define MD data sets in RDF. Cube instances are the members of a cube schema that represent level members, facts and measure values. We describe the cube schema elements in Sect. 5.1 and the cube instances in Sect. 5.2 with their examples.

5.1 GeoFarmHerdState Cube Schema in RDF

As our use case we create a spatial data cube of livestock holdings that we refer to as *GeoFarmHerdState*. The use case data cube is created from the flat data sets of the livestock farming (CHR) data, the environmental data, the

[7] Spatial aggregation functions aggregate two or more spatial objects and return a new spatial object, e.g., union, buffer, and convexHull etc.

[8] SpatialAggregateFunction is a subclass of AggregateFunction. Thus, measures and spatial measures use the same property `qb4o:aggregateFunction`.

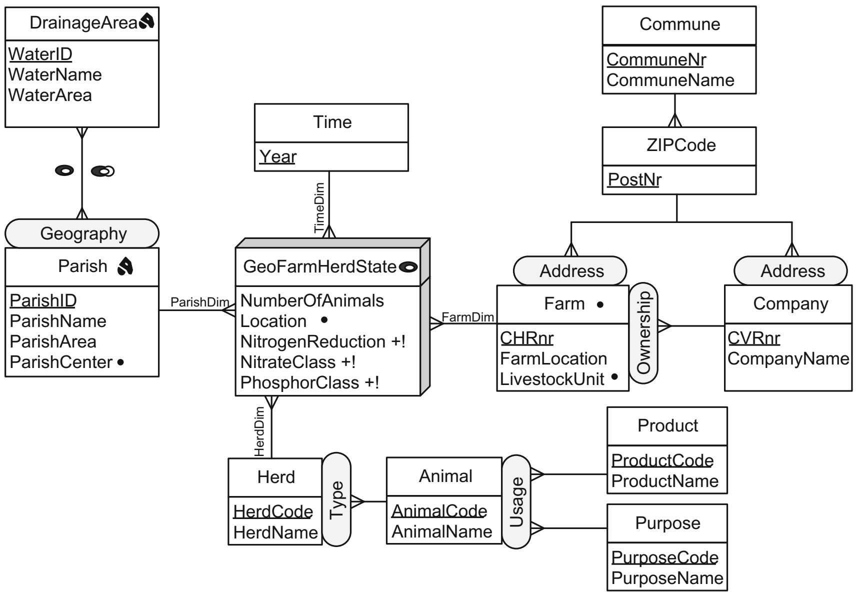


Fig. 2. GeoFarmHerdState – conceptual MD schema of livestock holdings data

geographical (regions) data, and the company registry (CVR) data collections, which are explained in Sect. 4. We create this data cube by thoroughly analyzing the attributes of the flat data sets from the collected relevant domains and conciliating them by foreign keys or by overlaying the SHP files of the spatial data sets and deriving new attributes from the intersected areas. After deriving useful spatial information across use case domains, we generalize the tabular data of the several use case data sets into the GeoFarmHerdState data cube. Figure 2 shows the multidimensional conceptual schema of the GeoFarmHerdState spatial cube. The multidimensional elements of the cube are explained in Remarks 1–7 followed by their examples in RDF. The underlying syntax for RDF examples is given in Turtle. We prefix the schema elements of the GeoFarmHerdState cube with `gfs:`.

Remark 1 (Dimensions). *Dimensions* provide perspectives to analyze the data. The GeoFarmHerdState cube has four dimensions, in which the two of them are *spatial* (FarmDim, ParishDim). All dimensions in the cube are defined with `qb:DimensionProperty`. A dimension is spatial if it has at least one spatial level (See Remark 3). Dimension hierarchies are defined with `qb4o:hasHierarchy` property. Hierarchies and their types are explained later in Remark 2.

Example 1. We give two spatial dimensions as an example.

```
gfs:farmDim rdf:type qb:DimensionProperty; qb4o:hasHierarchy gfs:ownership , gfs:address.
gfs:parishDim rdf:type qb:dimensionProperty; qb4o:hasHierarchy gnw:geography.
```

Remark 2 (Hierarchies). *Hierarchies* allow users to aggregate measures at various levels of detail. Hierarchies are composed of levels. A hierarchy is *spatial* if it has at least one spatial level (See Remark 3). Hierarchies of the GeoFarmHerdState cube are given in ellipses (Fig. 2). Each hierarchy is defined with `qb4o:Hierarchy` and linked to its dimension with the `qb4o:inDimension` property. Levels that belong to the hierarchy are defined with the `qb4o:hasLevel` property.

Example 2. We present the most interesting hierarchies from the GeoFarmHerdState cube as an example.

```
gfs:geogprahy rdf:type qb4o:Hierarchy; qb4o:inDimension gfs:parishDim; qb4o:hasLevel gfs:drainageArea.
gfs:usage rdf:type qb4o:Hierarchy; qb4o:inDimension gfs:animalDim; qb4o:hasLevel gfs:product , gfs:purpose.
gfs:address rdf:type qb4o:Hierarchy; qb4o:inDimension gfs:farmDim; qb4o:hasLevel gfs:zipCode , gfs:commune.
```

The Geography hierarchy is a *non-strict* spatial hierarchy. A spatial hiearchy is non-strict if it has at least one $(n - n)$ relationship between its levels. In the Geography hierarchy (Fig. 2) the $(n-n)$ cardinality represents that a parish may belong to more than one drainage area. Usually, non-strict spatial hierarchies arise when a partial containment relationship exists, which is given as *Intersects* in our use case. Usage hierarchy is a *generalized* hierarchy with non-exclusive paths to splitting levels (Product and Purpose) and has no joining level but the top level *All*. Finally, the Address and Ownership hierarchies are *parallel dependent* hierarchies. Parallel hierarchies arise when a dimension has several hierarchies sharing some levels. Note that the Address hierarchy has different paths from the Company and Farm levels (Fig. 2).

Remark 3 (Levels). *Levels* have a set of attributes (See Remark 4) that describes the characteristics of the level members (See Remark 9). Levels are defined with the `qb4o:LevelProperty` and their attributes are linked with the `qb4o:hasAttribute` property. A level is *spatial* if it has an associated geometry. Therefore, spatial levels have the property `geo:hasGeometry`, which defines the geometry of the spatial level in QB4SOLAP.

Example 3. We present a spatial level (Parish) as an example with its attributes. Attributes and spatial attributes of levels are further described in Remark 4.

```
gfs:parish rdf:type qb4o:LevelProperty; qb4o:hasAttribute gfs:parishID;
    qb4o:hasAttribute gfs:parishName; qb4o:hasAttribute gfs:parishArea;
    qb4o:hasAttribute gfs:parishCenter; geo:hasGeometry gfs:parishPolygon.
```

Note that the Parish level is defined as a spatial level because it has an associated polygon geometry (`gfs:parishPolygon`), which is specified with the `geo:hasGeometry` property. Some other spatial characteristics of the levels can be recorded in the spatial attributes of the level such as the center point of the parish (`gfs:parishCenter`).

Remark 4 (Attributes). Attributes and *spatial* attributes are defined with the `qb4o:LevelAttribute` property and linked to their levels with the `qb4o:inLevel` property. An attribute is spatial if it is defined over a spatial domain. Attributes are defined as ranging over XSD literals[9] and spatial attributes must be ranging over spatial literals, i.e., well-known text literals (WKT) from OGC schemas[10]. Spatial attributes are a sub-property of the `geo:Geometry` class. Further, the domain of the spatial attribute should be specified with `rdfs:domain`, which must be a geometry. Finally, the spatial attribute must be specified as an instance of `geo:SpatialObject` with the `rdfs:subClassOf` property. Examples of attributes are given in the following.

Example 4. We present spatial and some non-spatial attributes of the Parish level.

```
gfs:parishID rdf:type qb4o:LevelAttribute; qb4o:inLevel gfs:parish; rdfs:range xsd:positiveInteger.
gfs:parishName rdf:type qb4o:LevelAttribute; qb4o:inLevel gfs:parish; rdfs:range xsd:string.
gfs:parishCenter rdf:type qb4o:LevelAttribute; rdfs:subPropertyOf geo:Geometry; qb4o:inLevel gfs:parish;
    rdfs:domain geo:Point; rdfs:subClassOf geo:SpatialObject; rdfs:range geo:wktLiteral , virtrdf:Geometry.
```

We have mentioned in Remark 3 that spatial levels are defined through their associated geometries, which are not given as a level attribute. For the Parish level we present the following example of the corresponding geometry.

```
gfs:parishPolygon rdf:type geo:Geometry; rdfs:domain geo:MultiSurface;
    rdfs:subClassOf geo:SpatialObject; rdfs:range geo:wktLiteral , virtrdf:Geometry.
```

Remark 5 (Hierarchy Steps). Hierarchy steps define the structure of the hierarchy in relation to its corresponding, levels. A hierarchy step entails a roll-up relation between a lower (child) level and an upper (parent) level with a cardinality. The cardinality $(n - n, 1 - n, n - 1, n - n)$ relationship describes the number of members in one level that can be related to a member in the other level for both child and parent levels. A hierarchy step is *spatial* if it relates a spatial child level and a spatial parent level, in which case it entails a topological relationship between these spatial levels. Both spatial and non-spatial hierarchy steps are defined as a blank node with the `qb4o:HierarchyStep` property and linked to their hierarchies with the `qb4o:inHierarchy` property. The parent and child levels are linked to hierarchy steps with the `qb4o:childLevel` property and the `qb4o:parentLevel` property. The cardinality of a hierarchy step is defined by the `qb4o:pcCardinality` property. And finally, the topological relationship[11] of a hierarchy step is defined by the `qb4so:pcTopoRel` property.

Example 5. The following illustrates the hierarchy steps of the spatial hierarchy Geography and non-spatial hierarchy Address as it has different paths from child levels Farm and Company.

```
## Geography hierarchy structure ##
_:geography_hs1 rdf:type qb4o:HierarchyStep; qb4o:inHierarchy gfs:geography;
    qb4o:childLevel gfs:parish; qb4o:parentLevel gfs:drainageArea;
    qb4o:pcCardinality qb4o:ManyToMany; qb4so:pcTopoRel qb4so:Intersects, qb4so:Within.
## Address hierarchy structure ##
```

[9] XML Schema Definition: http://www.w3.org/TR/xmlschema11-1/.
[10] OGC Schemas: http://schemas.opengis.net/.
[11] Topological relations are Boolean predicates that specify how two spatial objects are related to each other, e.g., within, intersects, touches, and crosses etc.

```
_:farm_address_hs1 rdf:type qb4o:HierarchyStep; qb4o:inHierarchy gfs:address;
    qb4o:childLevel gfs:farm; qb4o:parentLevel gfs:zipCode;
    qb4o:pcCardinality qb4o:ManyToOne.
_:farm_address_hs2 rdf:type qb4o:HierarchyStep; qb4o:inHierarchy gfs:address;
    qb4o:childLevel gfs:zipCode; qb4o:parentLevel gfs:commune;
    qb4o:pcCardinality qb4o:ManyToOne.
_:company_address_hs1 rdf:type qb4o:HierarchyStep; qb4o:inHierarchy gfs:address;
    qb4o:childLevel gfs:company; qb4o:parentLevel gfs:zipCode;
    qb4o:pcCardinality qb4o:ManyToOne.
_:company_address_hs2 rdf:type qb4o:HierarchyStep; qb4o:inHierarchy gfs:address;
    qb4o:childLevel gfs:zipCode; qb4o:parentLevel gfs:commune;
    qb4o:pcCardinality qb4o:ManyToOne.
```

Remark 6 (Measures). Measures record the values of a phenomena being observed. Measures and *spatial* measures are defined with `qb:MeasureProperty`. A measure is spatial if it is defined over a spatial domain. Similarly to attributes (Remark 4), measures are defined ranging over XSD literals and spatial measures must be ranging over spatial literals.

Example 6. The following shows an example of a spatial measure (Location) and a non-spatial measure (NumberOfAnimals).

```
gfs:location rdf:type qb:MeasureProperty; rdfs:subPropertyOf sdmx-measure:obsValue;
    rdfs:subClassOf geo:SpatialObject; rdfs:domain geo:Point;
    rdfs:range geo:wktLiteral , virtrdf:Geometry.
gfs:numberOfAnimals rdf:type qb:MeasureProperty;
    rdfs:subPropertyOf sdmx-measure:obsValue; rdfs:range xsd:decimal.
```

Remark 7 (Fact). Fact defines the data structure (DSD) of the cube with `qb:DataStructureDefinition`. The dimensions are given as `components` and defined with the `qb4o:level` property as the dimensions are linked to the fact at the lowest granularity level. A fact is *spatial* if it relates two ore more spatial levels. Similarly, measures are given as `components` of the fact and are defined with the `qb:measure` property. Aggregation functions on measures and spatial aggregation functions on spatial measures are also defined in the DSD with `qb4o:aggregateFunction`. Fact-level cardinality relationships and topological relationships are defined with `qb4o:cardinality` and `qb4so:topologicalRelation` in DSD, respectively (Fig. 1).

Example 7. The following shows the data structure definition of the cube GeoFarmHerdState, which is defined with corresponding measures and dimensions.

```
# – GeoFarmHerdState Cube Definition of the Fact FarmHerdState
gfs:GeoFarmHerdState rdf:type qb:DataStructureDefinition;
    # Lowest level for each dimensions in the cube
    qb:component [qb4o:level gfs:herd; qb4o:cardinality qb4o:ManyToOne ];
    qb:component [qb4o:level gfs:time; qb4o:cardinality qb4o:ManyToOne ];
    qb:component [qb4o:level gfs:farm; qb4o:cardinality qb4o:ManyToOne;
        qb4so:topologicalRelation qb4so:Equals ];
    qb:component [qb4o:level gfs:parish; qb4o:cardinality qb4o:ManyToMany;
        qb4so:topologicalRelation qb4so:Within ];
    # Measures in the cube
    qb:component [qb:measure gfs:numberOfAnimals; qb4o:aggregateFunction qb4o:Sum];
    qb:component [qb:measure gfs:location; qb4o:aggregateFunction qb4so:ConvexHull];
    qb:component [qb:measure gfs:nitrogenReduction; qb4o:aggregateFunction qb4o:Avg];
    qb:component [qb:measure gfs:nitrateClass; qb4o:aggregateFunction qb4o:Avg];
    qb:component [qb:measure gfs:phosphorClass; qb4o:aggregateFunction qb4o:Avg].
```

5.2 GeoFarmHerdState Cube Instances in RDF

Cube instances are the members of a cube schema that represent level members and facts (members), which are explained in Remarks 8 and 9 below. We prefix the instances of the GeoFarmHerdState cube with `gfsi:`.

Remark 8 (Fact members). Fact members (i.e., facts of FarmHerdState) are instances of the `qb:Observation` class. Each fact member is related to a set of dimension *base* level members and has a set of measure values. Every fact member has a unique identifier (IRI) which is prefixed with `gfsi:`.

Example 8. The following shows an example of a single fact member, which represents the state of a farm with CHR no. 39679 in the year 2015 that has the herd code 15.

```
gfsi:farm_39679_2015 rdf:type qb:Observation;
## Dimension levels and base level members associated with the fact member
    gfs:herdCode gfsi:herd_15; gfs:year gfsi:year_2015;
    gfs:chrNumber gfsi:farm_39679; gfs:parishID gfsi:parish_8311;
## Measures associated with the fact member
    gfs:numberOfAnimals "100.0"^^xsd:decimal; gfs:nitrateClass "3"^^xsd:integer;
    gfs:nitrogenReduction "0.75"^^xsd:decimal; gfs:phosporClass "3"^^xsd:integer;
    gfs:location "POINT(8.3713 56.7912)"^^geo:wktLiteral.
```

Remark 9 (Level members). Level members are defined with `qb4o:LevelMember`. They are linked to their corresponding levels from the schema with the `qb4o:memberOf` property. For each level member there is a set of attribute values. Due to the roll-up relations between levels of hierarchy steps (Remark 5), the `skos:broader` property relates a child level member to its parent level member.

Example 9. The following shows an example of a child level member in the Parish level and one of its parent level members in the DrainageArea level from the Geography dimension. Figure 3 presents a map snapshot for fact members and level members. Parish level member "Astrup" is highlighted and DrainageArea level member "Mariager Inderfjord" is marked with red borders. Note that Astrup intersects another drainage area "Langerak", therefore it links to two parent level members via `skos:broader`.

```
## Parish level member
gfsi:parish_8648 rdf:type gfs:parish;
    qb4o:memberOf gfs:parish; skos:broader gfsi:water_3710, gfsi:water_159;
    gfs:parishID 8311; gfs:parishName"Astrup"; gfs:parishArea 46,118;
    gfs:parishCenter"POINT(8.2552, 56.8176)"^^geo:wktLiteral;
    gfs:parishPolygon"POLYGON((8.4038 56.7963, 8.3984 56.7721, 8.3689 56.7410, 8.3411 56.7372, 8.3078
    56.7281, 8.2987 56.7601, 8.2563 56.7763, 8.3112 56.8087, 8.3511 56.8137, 8.4038 56.7963))"^^geo:wktLiteral.
## DrainageArea level member
gfsi:water_159 rdf:type gfs:drainageArea;
    qb4o:memberOf gfs:drainageArea; gfs:waterID 159;
    gfs:waterName"Mariager Inderfjord"; gfs:waterArea 267,477;
    gfs:drainageGeo "POLYGON((8.6048 56.9843, 8.5908 56.8969, 8.5707 56.8664,
    8.5975 56.8519, 8.5215 56.8483, 8.3959 56.7625, 8.3938, 56.7340, 8.3613 56.6802,
    8.2584 56.7764, 8.2475 56.7051, 8.2175 56.7232, 8.3121 56.8441, 8.2806 56.8659,
    8.3602 56.9569, 8.4786 56.9713, 8.5474 56.9905, 8.6048 56.9843))"^^geo:wktLiteral.
```

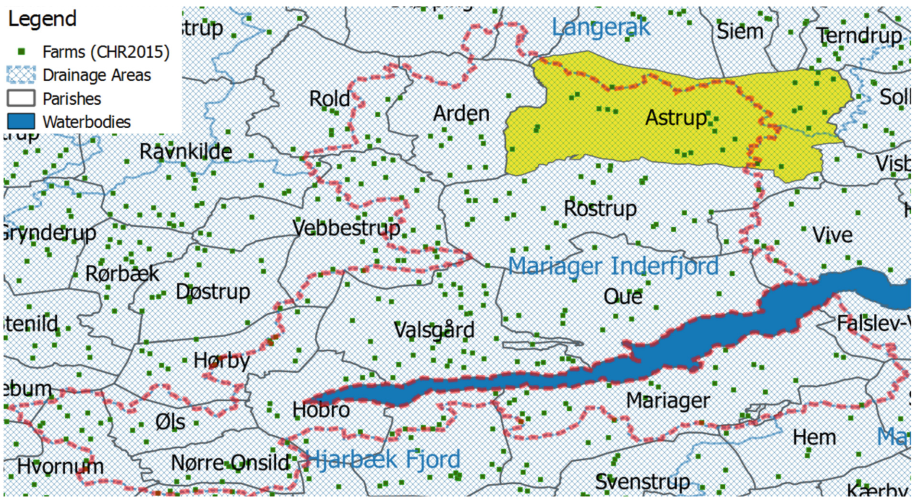


Fig. 3. GeoFarmHerdState – fact members and level members of Example 9 marked

6 SOLAP Operators over GeoFarmHerdState Cube

Spatial OLAP (SOLAP) operates on spatial data cubes. SOLAP increases the analytical capabilities of OLAP by taking into account the spatial information in the cube. SOLAP operators involve spatial conditions or spatial functions. Spatial conditions specify constraints (i.e., spatial Boolean predicates) on the geometries associated to cube members or measures, while spatial functions derive new data from the cube, which can be used, e.g., to derive dynamic spatial hierarchies.

6.1 SOLAP Operators

In what follows we present common SOLAP operators and examples of these operators on GeoFarmHerdState cube.

Remark 10 (S-Slice). The s-slice operator removes a dimension from a cube by choosing a single spatial value in a spatial level. It returns a cube with one dimension less.

Example 10. We can perform an s-slice operation in different ways.
1. Slice on farms (state of the farms) of the largest parish.
2. Slice on farms (state of the farms) of the drainage area containing `"POINT(10.43951 55.47006)"`.

The first one applies a spatial function call (for finding the *largest* parish by area) on a spatial level Parish and performs the slice. The second one applies a spatial predicate (for finding where a given point is *within* a particular drainage area) in a spatial level DrainageArea and performs the operation. The corresponding SPARQL queries are:

```
# 1 – s-slice with spatial function #
SELECT ?obs WHERE {
    ?obs rdf:type qb:Observation;
        gfs:parishID ?parish.
    ?parish gfs:parishPolygon ?parishGeo.
# Inner select for finding the largest parish
    { SELECT ?x (MAX(?area) as ?maxArea) WHERE{
    ?obs rdf:type qb:Observation;
        gfs:parishID ?parish.
    ?parish gfs:parishPolygon ?x.
    BIND (bif:st_area(?x) as ?area)}}
FILTER ?parishGeo = ?x) }
```

```
# 2 – s-slice with spatial predicate #
SELECT ?obs WHERE {
    ?obs rdf:type qb:Observation;
        gfs:parishID ?parish.
    ?parish qb4o:memberOf gfs:parish;
        skos:broader ?drainageArea.
    ?drainageArea gfs:drainageGeo ?drainageGeo.
FILTER (bif:st_within("POINT(10.43951 55.47006)",
?drainageGeo)) }
```

Remark 11 (S-Dice). The s-dice operator keeps the cells of the cube that satisfy the spatial predicate over dimension levels, attributes, or measures. It returns a subset of the cube with filtered members of the cube.

Example 11. In the following we show two examples of the s-dice operator.
1. Filter the farms located within 5 Km buffer from the center of a drainage area.
2. Filter the farms located within 2 Km distance from the center of their parish, which are in the nitrate class I areas.

In the first s-dice operation, initially, a spatial function is applied on level members of the DrainageArea level to get the *center* of their polygon geometries. Then, the level members of the Farm level are filtered with a spatial Boolean predicate with respect to the farm locations that are *within* a 5 Km buffer area of the center of the drainage areas. In the second s-dice operation, a spatial function is applied to the spatial measure farm location to get the *distance* of the farms from the center of their parish, which is followed by Boolean predicates; to filter the farms that are less than 2 Km away from the center of their parishes and are on nitrate class I areas.

```
# 1 – s-dice on dimension levels #
SELECT ?obs WHERE {
    ?obs rdf:type qb:Observation;
        gfs:farmID ?farm;
        gfs:parishID ?parish.
    ?farm gfs:farmLocation ?farmGeo.
    ?parish qb4o:memberOf gfs:parish;
        skos:broader ?drainageArea.
    ?drainageArea gfs:waterPolygon ?drainagePoly.
BIND (bif:st_centroid (?drainagePoly) as ?drainageCenter)
FILTER (bif:st_within(?drainageCenter, ?farmGeo, 5)) }
```

```
# 2 – s-dice on measures #
SELECT ?obs WHERE {
    ?obs rdf:type qb:Observation;
        gfs:location ?farmLocation;
        gfs:nitrateClass ?nitClass;
        gfs:parishID ?parish.
    ?parish gfs:parishCenter ?parishCent.
BIND (bif:st_distance (?farmLocation, ?parishCent)
AS ?distance)
FILTER (?distance < 2 && ?nitClass = 1)}
```

Remark 12 (S-Roll-up). The s-roll-up operator aggregates measures of a given cube by using an aggregate function and a spatial function along a spatial dimension's hierarchy. It returns a cube with measures at a coarser granularity for a given dimension.

Example 12. In the following, we present two examples of the s-roll-up operator.
1. Total amount of animals in the farms, which are closest to their parishes' center.
2. Average percentage of nitrogen reduction potentials in the parishes that are within and/or intersect the drainage area "Nibe-Bredning".

In the first s-roll-up operator, measures are aggregated to the Parish level after selecting the farms with respect to their proximity to the center of the parish with a spatial function. In the second s-roll-up operator, measures are aggregated to a specified drainage area ("Nibe-Bredning") at the DrainageArea level. We select all the possible topological cases where a parish intersects or within the drainage area, which means measures from the farms that are outside Nibe-Bredning are also aggregated to the level of this drainage area. In order to prevent this, the query needs to include an s-drill-down operator (Remark 13) to farms from Parish level and apply a spatial Boolean predicate to select the farms *within* the drainage area and then aggregate.

```
# 1 – s-roll-up #
SELECT ?parish (SUM(?animalCount) AS ?totalAnimals)
WHERE { ?obs rdf:type qb:Observation;
    gfs:numberOfAnimals ?animalCount;
    gfs:farmID ?farm;
    gfs:parishID ?parish.
?farm gfs:farmLocation ?farmGeo.
?parish gfs:parishCenter ?parishCent.
# Inner select for finding the
# closest farms to the parish centers #
    {SELECT ?farm1 (MIN(?distance) AS
    ?minDistance) WHERE
    { ?obs rdf:type ab:Observation;
        gfs:farmID ?farm1;
        gfs:parishID ?parish1.
    ?farm1 gfs:farmLocation ?farm1Geo.
    ?parish1 gfs:parishCenter ?parish1Cent.
    BIND (bif:st_distance (?farm1Geo, parish1Cent)
    AS ?distance) } GROUP BY ?farm1 }
    FILTER (?farm = ?farm1 && bif:st_distance
    (?farmGeo, ?parishCent) = ?minDistance )}
GROUP BY ?parish
```

```
# 2 – s-roll-up #
SELECT ?drainageArea (AVG(?nitRed) AS ?avgNitRed)
WHERE { ?obs rdf:type qb:Observation;
    gfs:location ?farmLocation;
    gfs:nitrogenReduction ?nitRed;
    gfs:parishID ?parish.
    ?parish qb4o:memberOf gfs:parish;
        gfs:parishPolygon ?parishGeo;
        skos:broader ?drainageArea.
    ?drainageArea gfs:memberOf gfs:drainageArea;
        gfs:waterPolygon ?drainageGeo;
        gfs:waterName ?drainageName.
FILTER (bif:st_within(?parishGeo, ?drainageGeo)
|| bif:st_intersects(?parishGeo, ?drainageGeo)
&& ?drainageName ="Nibe-Bredning")}
GROUP BY ?drainageArea
```

Remark 13 (S-Drill-down). The s-drill-down operator disaggregates measures of a given cube by using an aggregate function and a spatial function along a spatial dimension's hierarchy. It is the inverse operator of s-roll-up, therefore s-drill-down disaggregates the previously summarized data to a child level in order to obtain measures at a finer granularity.

6.2 Nested SOLAP Operations

A nested set of SOLAP operators can be designed with the pattern $(s\text{–}dice_2(s\text{–}roll\text{–}up_1(\ldots s\text{–}roll\text{–}up_k(s\text{–}slice_1(\ldots s\text{–}slice_n(s\text{–}dice_1(DataCube)))))))$. Initially a sub-cube is selected from the (spatial) data cube with the first s-dice. Afterwards, a number of s-slices can be applied, which is followed by a series of s-roll-ups. Finally, the expression ends with another s-dice for getting the final sub-cube at a coarser granularity by filtering the aggregated measures. In the following, we present a nested SOLAP operation example for the running case GeoFarmHerdState spatial data cube.

Example 13. $({}^3s\text{–}roll\text{–}up({}^2s\text{–}slice({}^1s\text{–}dice(GeoFarmHerdState))))$: This pattern represents a typical nested SOLAP operation that can be paraphrased for the running use case as follows: ^{1}Filter the farm states located within a 2 Km distance from the center of their parish and ^{2}slice on the parish which has the most number of topological relations (intersects, within) with a drainage area, ^{3}average the nitrogen reduction potential of the drainage areas intersecting with the parish.

7 Discussion and Perspectives

In the following, we give and evaluate the steps of our process with respect to the guidelines for publishing governmental linked data [18]. We discuss the particular challenges that we encountered and possible future improvements.

(1) Specification. The first step is to specify the scope of the data by identifying and analyzing the data sources. We identified the data sources for the domains of CHR data, Environmental data, Geographical data, and CVR data as described in Sect. 4. In order to find the correct relations between these domains we had to search documentations (i.e., [13,14]) and acquire knowledge about the domains' interests. As the purpose is to publish open data, the definition of an Open Data license is also required at this level.

(2) Modeling. We used the spatially extended MultiDim model [17] for designing the MD conceptual schema of the use case spatial data cube (Fig. 2) from the collected flat data sets. This process requires good knowledge of spatial data warehouses and its concepts. In order to model the spatial data cube in RDF, QB4SOLAP provides the state-of-the-art semantic spatial data cubes. Therefore, we annotate the designed use case conceptual schema with QB4SOLAP.

Modeling the RDF data with QB4SOLAP provides all the core concepts of spatial data warehouses (i.e., spatial dimensions, spatial levels, and spatial hierarchies) for spatial data on the SW. Therefore, QB4SOLAP conveniently handles the conceptual modeling process of DWs on the SW and clearly describes the certain relations that should be considered during the logical modeling process (e.g., cardinality and topological relationships to create integrity constraints for ER models).

(3) Generation. This step of the overall process involves the most complex tasks. In order to fully generate a spatial data cube in RDF the following subprocesses are performed: transformation and data conciliation.

(3.1) Transformation. The RDF triples were generated with ad-hoc C# code for mapping from relational CSV files to RDF. In total, 12 CSV files were organized based on the relational representation (snowflake schema) of an MD conceptual model such that: We obtained *one* fact table with foreign keys of the related dimension (base) levels and measures, *four* tables with each dimensions' base level, and *seven* tables for the remaining levels along the hierarchies. These 12 tables are related by referential integrity constraints. Every level table also records the level attributes and attribute values. In order to create this relational CSV files, we pursued a number of data conciliation activities as described in the following item.

(3.2) Data Conciliation. Initial data sets are downloaded in different formats i.e., SHP format for CHR, Environmental, and Geographical data; CSV format for CVR data. In order to create the desired relational implementation of the use case spatial data cube, we used the unique identifiers (i.e., CVR and CHR numbers) or utilized spatial joins by joining attributes from one geometry feature to another based on the spatial containment relationship. For instance, we overlaid the point coordinates of the farms from CHR data and polygon coordinates of three environmental data sets in order to intersect and find the soil quality measurements for NitrogenReduction, NitrateClass and PhosphorClass of each farm. Another interesting spatial join is utilized for relating the Parish level members with DrainageArea level members. Since there is an $(n-n)$ cardinality relationship, some parishes intersect with more than one drainage area, thus we used topological relationships (intersects and within) to find the related child and parent level members. For interacting and handling the spatial data, we used QGIS with integration to PostGIS[12]. We used PostgreSQL to create and export relational tables.

The lack of tools for mapping a spatial multidimensional model to the relational model has been an impediment since we have to use topological relationships, where there is an $(n-n)$ cardinality relationship. Therefore, semantic ETL for data warehouses [5] is an important research topic, which requires improvements also for spatial data. Semi-automated tool support of geo-semantic ETL for publishing data warehouses on the SW is a promising improvement for handling the above processes such that the spatial joins can be processed efficiently. Before publishing the final RDF data, a comprehensive data cleansing step is essential for removing redundant columns and cleaning the noise due to unescaped characters, denormalized spatial literals, and encoding problems.

(4) Publication. In order to store and publish the RDF data we chose the Virtuoso Universal Server as a triple store. The details about the SPARQL endpoint can be found on the project page http://extbi.cs.aau.dk/GeoFarmHerdState.

Publication of metadata in Danish and English languages should be completed. Also for enabling efficient discovery of published spatial data cubes, adding an entry of the data in the CKAN repository (`datahub.io`) is required.

(5) Exploitation. The goal of our research is to re-use open government data and publish it as spatial data cubes on the SW for advanced multidimensional analysis. Therefore, we show how to query in SPARQL with SOLAP operators.

We recognize the need for non-expert SW users to write their spatial analytical queries in our high-level SOLAP language instead of the lower-level complex SPARQL language. Thus, a query system with a GUI that can interpret spatial data cube schemas for allowing users to perform high level SOLAP operations is ongoing work. Performing SOLAP queries in SPARQL to work over multiple RDF cubes with *s-drill-across* and supporting spatial aggregation (*s-aggregation*) over spatial measures are other important improvements on exploitation of spatial data cubes.

[12] QGIS: http://www.qgis.org/ PostGIS: http://postgis.net/.

8 Conclusion and Future Work

The need for spatial analytical queries on the Semantic Web increases constantly with regularly published open government data, but there is a lack of effective solutions and efficient models. As a first attempt to publish spatial data cubes from open data, we have shown that the QB4SOLAP vocabulary can be used to link Danish government data that is published in different domains. First, we have studied the use case data sets thoroughly with corresponding regulations and requirements in order to satisfy cross-domain interests (e.g., tracking soil quality in livestock farms and farm animals density on drainage areas etc.). Second, we have conciliated the flat data sets in order to model the MD concepts of a spatial data cube. Third, we described the most popular individual SOLAP operators and a nested SOLAP operation pattern with examples and their SPARQL implementation.

In this paper, the QB4SOLAP vocabulary is validated by a non-trivial spatial use case. As a proof of concept, we showed that linking spatial (governmental open) data on the Semantic Web can be achieved at an advanced level, not solely linking spatial open data on the SW but also modeling this data for advanced analytical queries with SOLAP operations.

Several directions are interesting for future research: developing a geo-semantic ETL tool to support the process of creating spatial data cubes on the SW, a GUI for non-expert users to perform SOLAP operations on SW spatial cubes, extending the use case and implementing advanced SOLAP queries, such that; as s-drill-across and s-aggregation on the SW.

Acknowledgments. This research is partially funded by “The Erasmus Mundus Joint Doctorate in Information Technologies for Business Intelligence Doctoral College (IT4BI-DC)”.

References

1. Abelló, A., Romero, O., Pedersen, T.B., Aramburu, M.J., et al.: Using semantic web technologies for exploratory OLAP: a survey. TKDE **27**, 571–588 (2014)
2. Andersen, A.B., Gür, N., Hose, K., Jakobsen, K.A., Pedersen, T.B.: Publishing danish agricultural government data as semantic web data. In: Supnithi, T., Yamaguchi, T., Pan, J.Z., Wuwongse, V., Buranarach, M. (eds.) JIST 2014. LNCS, vol. 8943, pp. 178–186. Springer, Heidelberg (2015). doi:10.1007/978-3-319-15615-6_13
3. Arendt, J.B.: Denmark releases its digital raw material, Ministry of Finance of Denmark, October 2012. http://uk.fm.dk/news/
4. Cyganiak, R., Reynolds, D., Tennison, J.: The RDF Data Cube Vocabulary (2014)
5. Nath, D.R.P., Hose, K., et al.: Towards a Programmable Semantic Extract-Transform-Load Framework for Semantic Data Warehouses. In: DOLAP, pp. 15–24 (2015)
6. Etcheverry, L., Vaisman, A., Zimányi, E.: Modeling and querying data warehouses on the semantic web using QB4OLAP. In: Bellatreche, L., Mohania, M.K. (eds.) DaWaK 2014. LNCS, vol. 8646, pp. 45–56. Springer, Heidelberg (2014). doi:10.1007/978-3-319-10160-6_5

7. Gür, N., Hose, K., Pedersen, T.B., Zimányi, E.: Modeling and querying spatial data warehouses on the semantic web. In: Qi, G., Kozaki, K., Pan, J.Z., Yu, S. (eds.) JIST 2015. LNCS, vol. 9544, pp. 3–22. Springer, Heidelberg (2016). doi:10.1007/978-3-319-31676-5_1
8. Gür, N., Pedersen, T.B., Zimányi, E., Hose, K.: A foundation for spatial data warehouses on the semantic web. Journal paper, under submission (2016)
9. Jakobsen, K.A., Andersen, A.B., Hose, K., Pedersen, T.B.: Optimizing RDF data cubes for efficient processing of analytical queries. In: COLD (2015)
10. Kämpgen, B., Harth, A.: OLAP4LD – a framework for building analysis applications over governmental statistics. In: Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I., Tordai, A. (eds.) ESWC 2014. LNCS, vol. 8798, pp. 389–394. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11955-7_54
11. Kämpgen, B., O'Riain, S., Harth, A.: Interacting with statistical linked data via OLAP operations. In: Simperl, E., Norton, B., Mladenic, D., Della Valle, E., Fundulaki, I., Passant, A., Troncy, R. (eds.) ESWC 2012. LNCS, vol. 7540, pp. 87–101. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46641-4_7
12. Matei, A., Chao, K.-M., Godwin, N.: OLAP for multidimensional semantic web databases. In: Castellanos, M., Dayal, U., Pedersen, T.B., Tatbul, N. (eds.) BIRTE 2013-2014. LNBIP, vol. 206, pp. 81–96. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46839-5_6
13. Danish Ministry of the Environment. Consolidated Act on Livestock Farming Environmental Approvals (2012). http://eng.mst.dk/media
14. Directive, Nitrates: Danish nitrate action programme 2008–2015 regarding the nitrates directive; 91/676/eec. Technical report, Nitrates Directive (2012)
15. Open Geospatial Consortium: GeoSPARQL: A geographic query language for RDF data. W3C Recommendation (2014)
16. Pedersen, D., Riis, K., Pedersen, T.B.: Query optimization for OLAP-XML federations. In: DOLAP, pp. 57–64 (2002)
17. Vaisman, A., Zimányi, E.: Spatial data warehouses. In: Vaisman, A., Zimányi, E. (eds.) Data Warehouse Systems. Design and Implementation. DCSA, pp. 427–473. Springer, Heidelberg (2014)
18. Villazón-Terrazas, B., Vilches-Blázquez, L., Corcho, O., Gómez-Pérez, A.: Methodological guidelines for publishing government linked data. In: Wood, D. (ed.) Linking Government Data, pp. 27–49. Springer, New York (2011)
19. W3C.: Data Cube Implementations (2014). https://www.w3.org/2011/gld/wiki/Data_Cube_Implementations
20. Zhao, P., Li, X., Xin, D., Han, J.: Graph cube: on warehousing and OLAP multidimensional networks. In: SIGMOD, pp. 853–864 (2011)

Classification of News by Topic Using Location Data

Zolzaya Dashdorj[1(✉)], Muhammad Tahir Khan[2], Loris Bozzato[3], and SangKeun Lee[1]

[1] Korea University, 1, 5-ga, Anam-dong, Seongbuk-gu, Seoul, Republic of Korea
{zolzaya,yalphy}@korea.ac.kr

[2] Taiger Singapore, 3 Fusionopolis Place #04-56, Galaxis, Singapore, Singapore
tahir.khan@taiger.com

[3] Fondazione Bruno Kessler, Via Sommarive 18, 38123 Trento, Italy
bozzato@fbk.eu

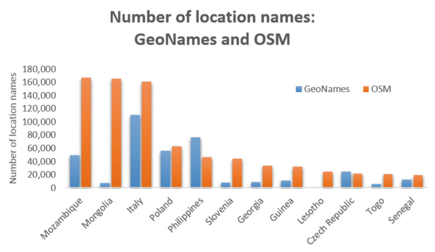
Abstract. In this work, we will consider news articles to determine geo-localization of their information and classify their topics on the basis of an available open data source: OpenStreetMap (OSM). We propose a knowledge-based conceptual and computational approach that disambiguates place names (i.e., geo-objects and regions) mentioned in news articles in terms of geographic coordinates. The geo-located news articles are analyzed to identify local topics: we found that the mentioned geo-objects are a good proxy to classify news topics.

1 Introduction

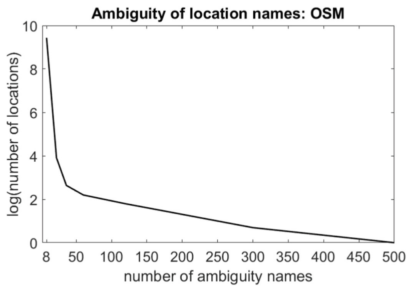
Enormous amount of information has been generated over web sources and social platforms by the activities of millions of people worldwide. The quantitative understanding of such information provides great impacts on the analysis of human behaviors and human dynamics on a macro-level [5,6,12]. However, in order to carry out the analysis on a micro level (i.e., provinces and districts), there is a need for geo-localization of information that describes different contexts of events (i.e., emergency vs non emergency) or human activities. Obtaining such useful and explanatory information from web sources and social platforms becomes an emerging interest in the areas of information retrieval, data mining and social network analysis [8].

However, the Natural Language Processing (NLP) successfully recognizes micro level entities (i.e., organization and location) from free text. But, it suffers from the ambiguity of place names mentioned in news in terms of geographic coordinates. To solve such ambiguity problem, previous studies [10,13] rely on geo-referenced texts in social platforms which include geo-coordinates in the meta data. More recent studies have considered to study local topics on social media [9] based on the area-specific term occurrence, estimating area specific scores of terms and occurrences using term frequency, as well as average and standard deviation of the longitude and latitude of raw geotagged information.

Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 305–314, 2016.
DOI: 10.1007/978-3-319-50112-3_23



(a) Place Names Over Countries in GeoNames vs OSM



(b) Ambiguous Place Names: OpenStreetMap

Fig. 1. Open spatial data-sources

However, geotagged information in media has not been studied much to determine local topics and no previous work has been done in the identification of local topics from non geo-referenced information on media. The fact that news topics can be extracted from the geo-localization of the article, which has not received an attention: still, researchers have studied topic modeling on news favorably using Latent Dirchlet Allocation (LDA) [2].

In this paper, we propose a news classification model based on location data included in the news by disambiguating the place names mentioned in the news, without an expensive estimation cost on geographical features. In order to deal with ambiguous place names mentioned in news in terms of geographic coordinates, we implement several heuristic disambiguation techniques using well known publicly available geographic gazetteers, namely Geo-names database (GNS) [1] and OpenStreetMap (OSM) [2]. However, GNS has been exploited in few studies [14]. But, it describes a less number of geo-names in some countries, while the OSM is well enriched with more geo-names including also geo-objects (i.e., organizations) that voluntarily collected with the official language of countries and other eight common languages. For example, the number of geo-names collected in GNS covering the country, Mongolia is around 7,017 in total which is almost 22 times smaller than the geo-objects we have collected from OSM. Figure 1(a) shows a comparative number of geo-names across different countries in public data-sources: GNS [3] vs OSM [4].

Thus, we use the OSM which is an open spatial data-source of rich information about geographical objects and features that localized over official administrative divisions of countries. Our research is concentrated on news articles which does not contain any geo-references data and to the best of our knowledge, this research is the first attempt on using OSM for news classification. The potential

[1] http://www.geonames.org.
[2] http://www.openstreetmap.org/.
[3] http://www.geonames.org/statistics.
[4] http://osmstats.neis-one.org/?item=countries.

application of this research is a location based recommender system in mobile computing and social networks.

2 Method Definition

We first build a knowledge base, called *Gazeto*, consisting of place names taxonomy referring to different coordinates, extracted from OSM. We define the term, *place name* in this research as a geographical coordinate associated to an organization or an administrative division. We apply conceptual and geo-computational approaches to web crawled news contents in order to identify the place names associated with the most likelihood that refer to their actual locations. Finally, we evaluate if news topic is predictable based on location features like regions, organizations, categories organization, by classifying the news into local topics as well as global topics. In this context, *local topics* are defined as topics which are identified from the news associated to location features. In the following sections, we will explain in detail each of the steps of our methods.

2.1 Place Names Knowledge Base

Using spatial open data in OSM, we collected the spatial features of 165,179 Points of Interest (POIs) over the entire country of Mongolia including of 378,491 properties (i.e., name, geo-object type, address) by using a spatial tool, OSM2PGSQL[5], and stored in geographical datastore - PostgreSQL[6]. In OSM, geo-objects are described in a pair of key and value e.g., *restaurant* is a sub type of the category *amenity*. The OSM dataset also contains a number of additional features like e.g. email, address, and website. We discarded these features and kept only primary features as described in OSM Wiki website[7] i.e., geo-objects along with their categories, such as sport shop, fast food restaurant and others. Using an *OSMonto* ontology [4], the primary features are refined to 125,398 geo-objects. We added a collection of hierarchical administrative divisions (e.g., countries, provinces, cities) in our KB. According to OSM, up to 8 administrative levels can be stored. The relations between geo-objects and administrative divisions are computed by PostGIS [8] functions allowing us to identify the administrative division of a given geo-object or the ascendant administrative division of a given division. We express a place name LN for an organization which is expressed by the pair of administrative division and geo-object name that formalized as: ⟨*Country, Province, District, Geo-Object*⟩. In total, we have collected 4,422 place names in our knowledge base. The ambiguity of place names referring to different coordinates is described in Fig. 1(b). Below 500 place names are ambiguous in terms of geo-references.

[5] http://wiki.openstreetmap.org/wiki/Osm2pgsql.
[6] http://www.postgresql.org/.
[7] http://wiki.openstreetmap.org/wiki/Map_Features.
[8] http://postgis.net/.

2.2 Place Names Disambiguation Methods

To choose a correct location L_j for the ambiguous candidate LN_i, we use heuristics in a similar fashion as [8,14]. However, the study was evaluated on macro-context (i.e., country, city and province). We propose the following heuristic techniques given news articles for place name disambiguation. The comparable baselines are NP and UC approaches.

– **NLP Pos Tagger (NP).** We adopt CRF Classifier—Stanford Named Entity Recognizer (NER) [7] for identifying the entities (i.e., organization and location) from given news articles. We train the sample data using one versus the rest method.
– **Unique Consistency (UC).** We examine if the candidate is non-ambiguous compared to the place names in our KB. Non ambiguous candidate refers to only one geographic coordinate. For entity extraction, we use the NLP package, *LingPipe NER* (Alias-i 2008) [1]: setting all matches phrase without case sensitive in the mapping. This approach will constitute our baseline estimation.
– **Sequence Consistency (SC).** The phrases in a paragraph are ordered and correlated to each other. Most place names are intuitively related to the near place names. We examine if the ascendant of the candidate LN_i is the same as the non ambiguous place name on the left side LN_{i-1} or the right side LN_{i+1} for the disambiguation.
– **Distance Consistency (DC).** The candidate with the minimum distance to non ambiguous place names will be chosen for disambiguation. If there is no non ambiguous place name defined in the news article, the candidate location is disambiguated based on the minimum distance to the central point of all the locations that associated to ambiguous candidates in the news.
– **Category Consistency (CC).** We verify the category of the candidate whether it is defined the same as the most probable category of non ambiguous place names in the news for the disambiguation. Otherwise, we check the similarity between the category of the candidate and the most probable category identified in terms of the parent category.
– **Neighborhood Category Consistency (NCC).** We check here up to 10 nearby geo-objects within the $radius = 5\,\mathrm{m}$ to the candidate if those share the same category as the most probable category of non-ambiguous place names in the news for the disambiguation. Otherwise, the similarity between the category of the nearby geo-objects to the candidate is calculated as the approach CC.

2.3 Classification of Local Topics from News

We classify topics over news based on OSMonto categorical taxonomies. First, we identify the correlation between geo-objects and news topics based on OSMOnto taxonomies, in order to estimate if the news topics are predictable based on such categorical taxonomies using learning algorithms. We then implement SVM multi-class classification using the one-against-all method. The news sets are pre-computed to generate the categorical probability distribution based on geo-object occurrence (P_{poi}) and categorical term occurrence (P_{news}). The correlation is estimated by Bhattacharyya coefficient[9] which is a correlation coefficient between the news category distribution P_{news} and categorical geo-object distribution P_{poi} over the entire news text: $BC(P_{poi}, P_{news}) = \sum_{i=1}^{n} \sqrt{P_{poi}(i) * P_{news}(i)}$, where BC is equal to 0 for a complete mismatch, otherwise 1 for a perfect match. The distributions, P_{poi}, P_{news} are labeled by k class denoted by k-mean. We construct k SVM models where k is the number of classes that describes categories organization and categorical terms based on the most geo-object occurrence and categorical term occurrence in each news, respectively. The vector features are the words in each news. The SVM for class k is constructed using the set of training examples and their desired outputs, (x_i, y_i). The mth SVM is trained with all of the examples in the mth class with positive labels, and all other examples with negative labels. The the decision function is: $x = argmax_{m=1,\ldots,k}((w^m)^T \phi(x) + b^m)$, where x is in the class which has the largest value of the decision function.

2.4 Classification of Global Topics from News

Given news articles where the place names are disambiguated, we obtain global topics comparing with the local topics based on the geo-location features using one of the most popular topic model, latent Dirichlet allocation (LDA) [2]. In this study, we use Fast LDA [11] to evaluate the topic modeling based on perplexity. The perplexity, used by convention in language modeling. The algorithm uses a substantially smaller number of variational parameters, with no dependency on the dimensionality of the dataset. By calculating the perplexity of a test set, we can evaluate the generalization ability of a model, a lower perplexity score indicates better generalization performance. For a dataset D_{test}, the perplexity is: $perplexity(D_{test}) = exp - \frac{\sum_d log(P(w_d))}{\sum_d N_d}$, where w_d are the words in test dataset that are not repeated, N_d is the whole words in test dataset.

3 Experimental Results and Evaluation

3.1 Data Pre-Processing

To the best of our knowledge, there is no ground-truth resource to evaluate our approach. We decided to build our own evaluation set to estimate the performance of our model, sampling 510 english news articles[10] crawled from a daily newspaper site, UB Post[11]. One may notice that the amount of dataset

[9] Wiki link: https://en.wikipedia.org/wiki/Bhattacharyya_distance.
[10] Dataset: https://www.dropbox.com/sh/sio0goqw2soaavd/AABZHaNdcNC3VAN1XbKgCATPa?dl=0.
[11] http://ubpost.mongolnews.mn/.

Table 1. Place names in the knowledge base

# All LN	# Ambiguous LN	# Non-ambiguos LN	# Top categories	# Total categories
9,099	952	3,232	56	721

is not very large: this is due to the cost constraints on geographic computation. The geo-objects which tagged in English language: *name*, $official\_name$ and $name : en$, are considered. Table 1 shows the total number of place names which populated in our KB including 21 provinces, 9 districts and the capital city. 11% of them is ambiguous referring to different coordinates, and the rest is non-ambiguous unique entities.

3.2 Disambiguation of Place Names

We identified a total of 1,847 locations in the set of sampled news. Table 2 shows the performance accuracy for the disambiguation methods. Precision, or Accuracy, is calculated as the number of correctly disambiguated locations divided by the number of locations identified in the news samples. Since we do not have a ground-truth resource to compare our approach, the baseline estimation is considered as an unique consistency (UC) approach. By combining all the methods, we out perform the baseline approach by 53 % more. We analyzed such correlation on the reduced categories which could make a more sense naturally about news topics, such as: *office*, *shop*, *amenity*, *cuisine*, *leisure*, *tourism*, *sport*, *historic*, *nature*, *religion*, *public transportation* and *other*. The correlation result is described in the table, where the (reduced) BC is relatively increased for all news. This indicates that the news topics are strongly correlated to the geo-objects mentioned in the text without considering locations.

3.3 Local Topics Identification

To avoid mis-disambiguation of place names in the news, we consider the place names which generated by the reliable ground truth (baseline) methods, UC and NP. This provides unique place names referred to one coordinate only. Given the news text, we estimate the statistical feasibility on news topics prediction based on geo-object types, vice versa. For example, a given news *"KFC is opened near central square in Mongolia as the nations first ever western fast food chain"*, the geo-object KFC is a fast food type of amenity and a business building which are described in OSMonto. Then, the topic of the news is associated to food and business. In fact, we can come up with an assumption that the local topics can be identified by the geo-objects described in the news text.

We pre-process the data by removing stop words in text D. We got totally 13,714 number of vocabularies in the text corpus. The BOW model is interpreted as a feature vector of the words given text D across each document. The BOW as vector for each news is labeled by the top categories in OSMonto when a geo-object is mentioned in the news text.

Table 2. Disambiguation and identification of place names on news sample

Method	# Count	Accuracy	Average *BC*	Reduced *BC*
NP	1,105 out of 4,231	26%	52%	52%
UC (Baseline)	837	45%	55%	69%
UC + SC	936	51%	53%	68%
UC + DC	1,356	73%	63%	83%
UC + DC + CC	1,382	75%	64%	83%
UC + DC + CC + NCC	1,813	98%	50%	89%

Table 3. SVM classifier

Estimation	Categories Organization - UC	Categories Organization - NP	Categorical Terms
Overall Accuracy	60.84	79.17	70.47
Macro-F1	50.0	100.0	100.0

In OSMonto, there are 385 category combinations in total: using these categories, we would like to classify the news topics. The categories are hierarchically organized into two-levels. For example, a *restaurant* is a sub-type of *amenity*, and a *supermarket* is a sub-type of *shop*. We sampled the dataset containing a geo-object found in articles in order to categorize the news. The result is reported on one against all validation (we chose 70 % for training and 30 % for testing; therefore we performed a random permutation before splitting the dataset for training and testing) in multi-class SVM classifier. The SVM classifier is denoted by a library, LIBSVM [3] which uses a linear kernel with penalty cost C=1. The news are localized based on the most likelihood administrative divisions or the most likelihood organizations. Table 3 shows the accuracy and macro-f1 of multi-class SVM classifier in average, over news categorized by the top categories organization which disambiguated by the baseline techniques UC and NP. Among the experiments, a classifier *Categories Organization - NP* performs well around 79.17 % of accuracy which considers news sets categorized by the top categories based on geo-object occurrence, where the place names have been disambiguated by NP method. This shows the geo-objects in news text as a proxy to predict news topics. But the overall accuracy and Macro-F1 are relatively small on a classifier *Categories Organization - UC* on the same categorized test sets, where we applied the UC method. This might be the impact of the dataset size that generated by UC method which generates a baseline set of news with non-ambiguous place names, around 45 % of geo-localized news. We also tested the classifier *Categorical Terms* on news categorized by the top categories based on the categorical word term occurrence. The result shows a similar accuracy with the classifier on the dataset of categories organization that indicates news topics are strongly correlated with the geo-objects mentioned in the news.

3.4 Global Topics Identification

In this study, we propose to classify news topics over the entire set incorporating with the Fast LDA to obtain lower dimensional feature representation for our subsequent classification task. The perplexity based evaluation is performed on test sets given a certain location features like organizations, categories organization and locations, respectively. In this paper, we took a sample as 10 % of the data for test purpose and used the rest for model training.

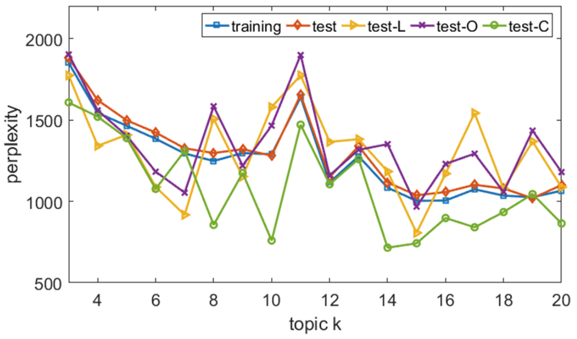


Fig. 2. Perplexity Over Entire and Geo-Located Dataset

Therefore we performed a random permutation before splitting the dataset for training and testing in the classifier. Hyper-parameter is set: use D_k/D to initialize *alpha* where D_k is the number of data points in class k and D is the total number of data points, $\epsilon = 0.01$, where k is the number of topics and V is the vocabulary size, ϵ is the laplacian smoothing parameter. The comparison of average perplexities over the training set, the test set and the classified test sets by location, organization and category organization (i.e., Oyu-tolgoi mining), is presented in Fig. 2. From the comparison, we observed similar perplexities in training and test set, but mildly higher perplexity on the test dataset. But the perplexities on the classified test sets are relatively lower that indicates a better performance. Thus, the global LDA classifier does not work well and it highlights that local topic identification is important to consider. For instance, we obtained a lowest perplexity on *topic-15*, where the model performs better, comparing the perplexities in training set and location feature classified test sets.

4 Conclusion and Discussion

We introduced a knowledge base geo-location recognition model based on OpenStreetMap as it allows us to extract the knowledge about geo-referenced place names in hierarchical administrative divisions. Using the knowledge base, our conceptual and computation methods can identify place names (i.e., organizations and administrative divisions) in free text that can be applied for any language. The place names disambiguation techniques perform reasonably well, around 98%. Therefore, we found that news topics are strongly correlated to the location based features mentioned in news and showed that geo-object is a good proxy to predict news topics, with around 79.17% accuracy (98% accuracy on reduced categories). However, news correlation to locations is highly dependent on the granularity of the city. Our approach could be largely affected by the distribution of geo-objects contributed to OSM, over different countries. In future works, we would like to use other gazetteers (i.e., GeoNames, Open Directory Project) for collecting a sufficient amount of location based features. For topic classification and rating task, we will analyze topics over time changes based on the features like topics importance and topics sentiment.

Acknowledgments. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (number 2015R1A2A1A10052665).

References

1. Alias-i. LingPipe 4.1.0 (2008). http://alias-i.com/lingpipe/
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
3. Chang, C.-C., Lin, C.-J.: LIBSVM: A library for support vector machines. ACM Trans. Intell. Syst. Technol. 2, 27:1–27:27 (2011). http://www.csie.ntu.edu.tw/~cjlin/libsvm
4. Codescu, M., Horsinka, G., Kutz, O., Mossakowski, T., Rau, R.: Osmonto - an ontology of openstreetmap tags. In: State of the map Europe (SOTM-EU) (2011)
5. Dashdorj, Z., Serafini, L., Antonelli, F., Larcher, R.: Semantic enrichment of mobile phone data records. In: MUM, p. 35. ACM (2013)
6. Dashdorj, Z., Sobolevsky, S.: Impact of the spatial context on human communication activity. In: Proceedings of the 2015 ACM UbiComp/ISWC Adjunct 2015, Osaka, Japan, 7–11 September 2015, pp. 1615–1622 (2015)
7. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics, pp. 363–370 (2005)
8. Inkpen, D., Liu, J., Farzindar, A., Kazemi, F., Ghazi, D.: Detecting and disambiguating locations mentioned in twitter messages. In: Gelbukh, A. (ed.) CICLing 2015. LNCS, vol. 9042, pp. 321–332. Springer, Heidelberg (2015). doi:10.1007/978-3-319-18117-2_24
9. Ishida, K.: Estimation of user location and local topics based on geo-tagged text data on social media. In: 2015 IIAI 4th International Congress on Advanced Applied Informatics (IIAI-AAI), pp. 14–17, July 2015
10. Kinsella, S., Murdock, V., O'Hare, N.: "I'm eating a sandwich in glasgow": Modeling locations with tweets. In: Proceedings of the 3rd International Workshop on Search and Mining User-generated Contents, SMUC 2011. ACM, New York (2011)

11. Shan, H., Banerjee, A.: Mixed-membership naive bayes models. Data Min. Knowl. Disc. **23**(1), 1–62 (2011)
12. Sobolevsky, S., Sitko, I., Grauwin, S., des Combes, R.T., Hawelka, B., Arias, J.M., Ratti, C.: Mining urban performance: Scale-independent classification of cities based on individual economic transactions. CoRR, abs/1405.4301 (2014)
13. Van Laere, O., Quinn, J., Schockaert, S., Dhoedt, B.: Spatially aware term selection for geotagging. IEEE Trans. Knowl. Data Eng. **26**(1), 221–234 (2014)
14. Volz, R., Kleb, J., Mueller, W.: Towards ontology-based disambiguation of geographical identifiers. In: Bouquet, P., Stoermer, H., Tummarello, G., Halpin, H. (eds.) Proceedings of the WWW 2007 Workshop I^3: Entity-Centric Approaches to Information and Knowledge Management on the Web, Canada, CEUR Workshop Proceedings (2007)

Monitoring and Automating Factories Using Semantic Models

Niklas Petersen[1,2](✉), Michael Galkin[1,2,3], Christoph Lange[1,2], Steffen Lohmann[2], and Sören Auer[1,2]

[1] University of Bonn, Bonn, Germany
{petersen,galkin,langec,auer}@cs.uni-bonn.de
[2] Fraunhofer IAIS, Sankt Augustin, Germany
steffen.lohmann@iais.fraunhofer.de
[3] ITMO University, Saint Petersburg, Russia

Abstract. Keeping factories running at any time is a critical task for every manufacturing enterprise. Optimizing the flows of goods and services inside and between factories is a challenge that attracts much attention in research and business. The idea to fully describe a factory in a digital form to improve decision making is called a virtual factory. While promising virtual factory frameworks have been proposed, their semantic models lack depth and suffer from limited expressiveness. We propose an enhanced semantic model of a factory, which enables views spanning from the high level of supply chains to the low level of machines on the shop floor. The model includes a mapping to relational production databases to support federated queries on different legacy systems in use. We evaluate the model in a production line use case, demonstrating that it can be used for typical factory tasks, such as assembly line identification or machine availability checks.

1 Introduction

The Industry 4.0 vision [2] aims at digitizing engineering, production and manufacturing with the goal of (i) a seamless integration of devices, sensors, machines as well as software and IT systems, (ii) increased flexibility thanks to pushing more intelligence from centralized planning systems to the edge, (iii) increased efficiency due to automated data exchange and analysis within the value chain. Currently, much information is isolated within different applications, which prevents efficient access for real-time analytics [8]. The ultimate goal of Industry 4.0 (and related initiatives with different names in different regions, such as Industrie du Futur in France or Industrial Internet in the US) is the creation of a *Smart Factory* [6].

A Smart Factory is defined as a factory that supports people and machines in performing their tasks by providing context-aware information. For instance, the location of information about orders, products, machines, the available work force and the overall factory are rarely available in a unified database and format. The related idea of a *Virtual Factory* [14] proposes a framework that links all

© Springer International Publishing AG 2016
Y.-F. Li et al. (Eds.): JIST 2016, LNCS 10055, pp. 315–330, 2016.
DOI: 10.1007/978-3-319-50112-3_24

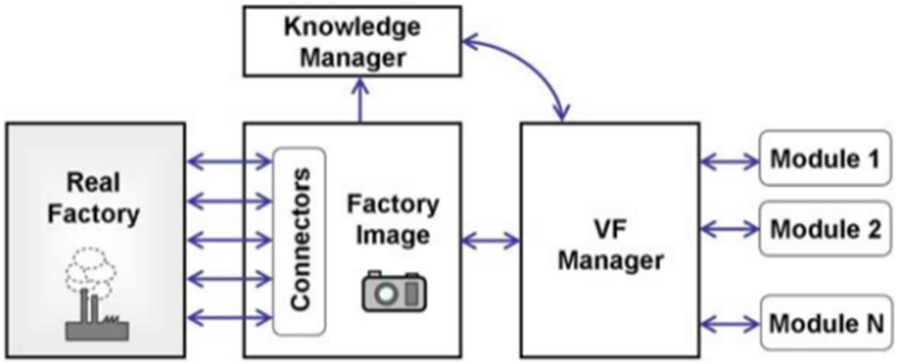


Fig. 1. Virtual factory framework as proposed by [12]

this information together, providing a mirror of the real factory (see Fig. 1) and thus paving the way towards more innovative factory prototyping, assembly line optimization, product design and mass customization [12].

In order to realize such a virtual factory, a number of interoperability challenges need to be solved. These include the identification of relevant data and information, their representation, unified access and interlinking. Of particular importance in this regard is the support of different views on the data (logistics and supply chain, manufacturing, quality control, etc.), the support of different levels of granularity of information representation (operational, strategic, etc.) as well as the access and integration of various data models and structures as used by existing systems and applications (XML, relational, enterprise models, etc.).

An integrated approach that provides a holistic view on an enterprise and its assets (such as factories) has not yet emerged. To fill this gap, we develop the notion of a *Semantic Factory*, employing semantic knowledge representation formalisms and technologies. The rationale is to employ a network of ontologies and vocabularies as a semantic fabric to represent, interlink and integrate the heterogeneous information being distributed in a variety of systems and information sources (e.g. manufacturing execution, quality management and enterprise resource planning systems or sensors, etc.). For creating the ontology, in a first step, we represent the static assets of an enterprise, such as its factories, assembly lines, workforce, etc. We then integrate dynamic information including business processes, shift plans, orders, etc. Finally, we map our model to production databases that contain the respective data for the dynamically changing concepts.

The mappings are performed in a minimally-invasive way, equipping existing systems of record with semantic interfaces (e.g. by using the W3C R2RML standard for mapping relational data to RDF). As a result, information and data in a factory can be integrated in a pay-as-you-go fashion, where mappings as well as the network of Semantic Factory ontologies evolve as required by specific use cases and application scenarios. Examples of use cases are (1) energy management, where the energy consumption is allocated to specific machines, work orders or customers, and (2) tool management, with the goal to minimize the time needed to equip machines with the required tools. In addition to introducing the Semantic Factory model, we demonstrate how decisions within an enterprise can be based on data currently hidden in different legacy systems. This includes the performance of supply chains, the detection of suitable assembly lines and the analysis of assets on a map.

The rest of the paper is structured as follows: We provide motivating examples and derive requirements for an integrated virtual factory representation in Sect. 2. In Sect. 3, we give an overview of related work. The overall architecture and factory ontology as its core model are presented in Sect. 4. In Sect. 5, we describe the implementation. We evaluate the performance of our approach in Sect. 6. The paper is concluded in Sect. 7 with an outlook on future work.

2 Motivating Examples and Requirements

A key motivation of our Semantic Factory model is to establish a holistic and integrated view on an enterprise in order to reduce the overall complexity and improve decision making. This includes the workforce, business processes, machines, shift plans, supply chains, etc. While a lot of this information is already captured by different IT systems, it is rarely accessible in a combined way without investing significant manual effort. Thus, the goal of this work is to make all data that is currently stored in various systems available in a unified model to support users with different roles in decision making.

2.1 Motivating Examples

An example is a factory planner who requires diverse information about order plans, workforce availability and machine maintenance dates. Another example is a machinist who needs to know which tools are to be mounted into which machine, where these tools are located, where the material is stored and what quality control standards are required during the production process. A controller, on the other hand, wants to keep track of the productivity of a factory and get an overview of certain Key Performance Indicators (KPIs). These comprise, in particular, information on the production time and effort required by each machine for each product, such as employee effort and energy consumption data.

To provide all those stakeholders with the information needed to perform their tasks and optimize decision making, we aim at semantically describing as many assets of a factory as possible, taking into account information from different manufacturing systems.

2.2 Requirements

We elicited the following requirements in the context of a research project for a global manufacturing company. The company's objective to gain a better picture of its assets (e.g. machines and factories) led us to develop an ontology that serves as the core element in the overall architecture.

From descriptions of the assets provided by the company and from interviewing domain experts, we gained an overview on typical tasks, processes and problems of each stakeholder. The interviews took place at the company site in multiple meetings, where the company's current IT infrastructure was described in detail. That way, we gathered requirements for the Semantic Factory model step by step:

Semantic Multi-modality. The types of data found in a factory context are diverse. Hence, the representation of various information, including attribute trees, relational, sensor, tabular, graph and entity data, must be supported.

Multi-dimensionality. Information along several dimensions must be represented and captured, such as:

- *Business processes:* Temporal views on diverse business activities are required to judge the success of an enterprise.
- *Spatial hierarchies:* The exploration of assets from a geographical perspective must be possible to increase the findability of said assets and related information.
- *Lifecycle:* Product and business lifecycles must be represented to support strategy management and business innovation.

Multi-granularity. Views on different levels of detail must be provided:

- *Components:* Instant access to sensor and component data must be enabled to support possible intervention measures.
- *Factories:* To decrease the complexity of factories, master and operational data needs to be accessible in a singular view.
- *Organization:* A big picture of all business units is needed, including their hierarchies and responsibilities.

Traceability and Integration. Data and information is currently spread across various systems, such as manufacturing execution systems, quality assurance systems, enterprise resource planning systems, etc. It is important to integrate all relevant information from these systems, while maintaining the systems' record-keeping character. When integrating information from these systems, the provenance of the data must be preserved, and changes to information in the source systems must be reflected in the integrated views, wherever possible in real time.

3 Related Work

There are two categories of related work: (i) existing frameworks that aim to describe factories as completely as possible, and (ii) existing ontologies representing assembly lines.

Terkaj et al. [13] propose a Virtual Factory Data Model represented as an OWL ontology based on the Industry Foundation classes (IFC)[1] standard. The purpose of their ontology is to describe business processes that involve machines requiring specific resources. However, the advantage of modeling each concept twice, once as a class whose instances represent real occurrences (e.g. `IfcProduct`, `IfcProcess`), and then as a class whose instances are supposed to describe generic objects and types (e.g. `IfcTypeProduct` "describes a generic object type that can be related to a geometric or spatial context", `IfcTypeProcess` "describes a generic process type to transform an input into output") is not clearly justified. A significant part of the ontology employs such a logical duplication which is misleading for non-experts. Furthermore, the rationale of proposing property classes such as `VffProcessProperties` to "characterize processes" instead of using object or data properties is not described.

Chen et al. [4] propose a multi-agent framework to monitor and control dynamic production floors. The ontology, serialized in XML, is optimized for the communication between different agents. It describes Radio-Frequency Identification (RFID) tags [16] attached to factory objects and addresses requirements specific to a bike manufacturing use case. Although RFID sensors are an important component of Industry 4.0, the purely XML-based ontology without logical formalisms behind, as they are provided by RDF(S) and OWL, lacks semantics and does not allow for universal and convenient querying.

Büscher et al. [3] introduce the Virtual Production Intelligence platform based on the Condition Based Factory Planning (CBFP) approach. The authors developed an OWL-based CBFP ontology advocating "the decoupling of domain business logic and the technical implementation of a planning system" [3]. The ontology is relatively small, consisting of only five classes. As it is not available online, we consider the CBFP ontology rather abstract and superficial. Detailed evaluations and experiments are not provided, making it hard to assess the practical contribution of the work.

Kim et al. [7] propose an OWL ontology and an information sharing framework to allow collaborative assembly design. The heart of the ontology is the assembly line and its direct environment. The ontology defines assemblies and constraints leveraging capabilities of SWRL and OWL. However, the lack of a published online version prevented us from reusing it. Nevertheless, the conceptual design influenced the one of our ontology, i.e., several concepts in the classes hierarchy and a few properties have been recreated.

[1] http://www.buildingsmart-tech.org/specifications/ifc-overview.

Ameri et al. [1] propose the Digital Manufacturing Market (DMM), a semantic web-based framework for agile supply chain deployment. DMM employs the Manufacturing Service Description Language (MSDL) at a semantic level. MSDL is an upper-level ontology expressed in OWL DL. Description Logic is extensively used to characterize supply and demand entities on several levels, such as the supplier, shop, machine and process levels. However, the granularity and ramification (especially for an upper-level ontology) impose restrictions on the usability, i.e., only a domain expert would have enough expertise to create a working model with accompanying queries. Furthermore, the ontology is again not available online, which prevented us from performing a thorough semantic analysis and considering an adaptation of concepts.

Zuehlke [17] introduces the SmartFactory initiative, which comprises best practices from the technical, architectural, planning, security and human dimensions. The initiative is envisioned to define and elaborate on the concept of *factory-of-things* as a vision of future manufacturing. Although semantic services involving ontologies and knowledge bases are claimed to be a part of the concept, the author does neither provide any examples nor references of such ontologies. Therefore, the presented concept is rather an implementation roadmap than a technical contribution.

4 Semantic Factory Architecture and Ontology

Based on the requirements presented in Sect. 2, we designed an architecture for a Semantic Factory application (see Fig. 2). Its core is a factory ontology, which describes real world objects, such as employees, machines and factories, locations of assets and their relations with each other. Operational data, such as information about work orders, machine sensor and process data, is also covered by the factory ontology; in practice, this data is dynamically mapped from the respective databases to the ontology.

The ontology is made available through an RDF triple store. Different applications can execute queries on the data, which is expressed as RDF or available in relational databases. Finally, for geospatial data, an external map provider service is used for drawing, among others, factories on a world map.

The following subsections explain how we developed the factory ontology following the methodology proposed by Uschold et al. [15]. We first defined the purpose and scope of the ontology; then, we captured step-by-step the domain knowledge, conceptualized and formalized the ontology and aligned it with existing ontologies. Finally, we evaluated the ontology by measuring the performance of certain queries for different sized datasets (see Sect. 6).

4.1 Purpose and Scope

The purpose of the ontology is to provide a holistic view of an enterprise. This is realized by implementing the requirements specified in Sect. 2. The intended

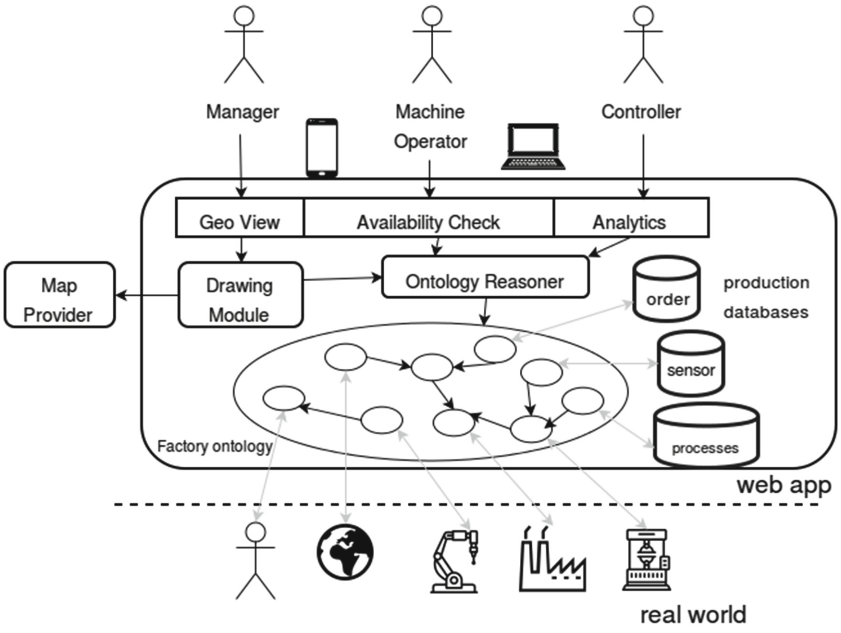


Fig. 2. Semantic Factory architecture

users are different stakeholders of an enterprise, such as managers, machine operators and controllers. Each of them needs different information to effectively and efficiently perform the corresponding tasks and duties. Thus, the ontology enables viewing the factory from different perspectives to support each class of stakeholders in their decision making.

4.2 Capturing Domain Knowledge

We captured the domain knowledge in three ways:

1. The company provided us with descriptive material of the domain, including maps of factories, descriptions of machines and work orders, process information, sensor data and tool knowledge. The types of input material ranged from formatted and unformatted text documents to spreadsheets and SQL dumps.
2. A live demonstration of a particular machine execution was given, including a discussion of further contextual information which was missing in the material. In subsequent meetings, open questions were clarified and concrete use cases of the ontology were discussed.
3. We reviewed relevant existing ontologies with the intention to build upon available conceptualizations and formalizations of domain knowledge.

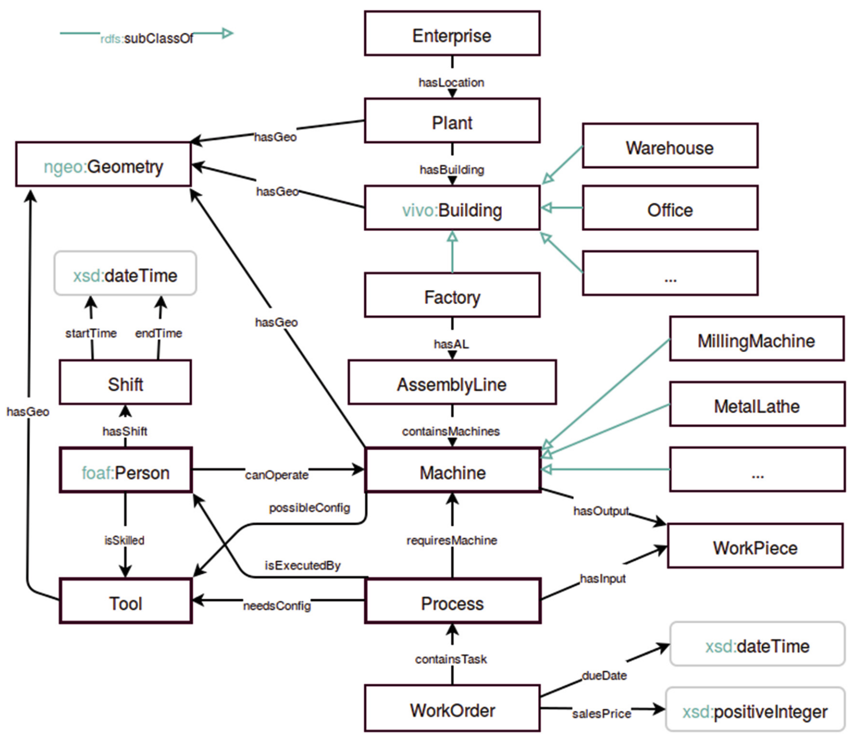


Fig. 3. Core concepts of the factory ontology

4.3 Conceptualizing and Formalizing

The resulting ontology comprises 86 classes, 73 object properties and 142 datatype properties. Since it has been designed to support an industrial project with sensitive business logic descriptions, not the entire ontology could be made publicly available. However, the part of the ontology that can be published is accessible via a permanent URL[2]. In the following, we describe the ontology, starting from the high-level organizational layer to the low-level machine and sensor layers. The core concepts of the ontology are depicted in Fig. 3.

In any concrete scenario, the class `Enterprise` is instantiated to represent the organization to which all further resources belong. Possible instances may be “Volkswagen”, “General Electric” or “Samsung”. Inter-organizational supply chains could involve multiple enterprises. Different production locations of an enterprise are described using the class `Plant`. A plant can comprise one or many `Buildings`, such as an office building, a factory building or a warehouse building. Typically, we can assume that each building serves a single function, though other configurations can also be represented, as OWL ontologies support

[2] https://w3id.org/i40/smo/.

multi-membership. Therefore, the subclasses of `Building` are not defined to be disjoint in the ontology.

Each `Factory` building may have one or multiple `AssemblyLines`. An assembly line usually consists of a sequence of multiple machines. The classes `MillingMachine` and `MetalLathe` are examples of subclasses of the abstract class `Machine`. Machines can be configured to use certain `Tools` to produce specific work pieces. As an example, a milling machine may be equipped with different lathes or end mills of varying granularity. `WorkPieces` represent everything which is an output of a machine. Once a work piece reaches its final stage of production, it becomes a product and is ready for shipping. Plants, factories and machines may have a representation of their geographical location (`ngeo:Geometry`[3]), which can, for instance, be used by front-end applications to display them on a map. A clear description of the entire capital of an enterprise supports the controllers and managers to keep track of utilized and unutilized assets.

As a next step, we describe the part of the ontology that represents the everyday operation of a factory. Employees are instances of the class `foaf:Person`. Each employee is qualified to operate specific machines (`canOperate`) and skilled to use certain tools (`isSkilled`). The class `WorkOrder` describes orders driven by customers. Each such order contains one or more `Processes`, which need to be executed to fully complete the order. Typical processes may be the configuration of a machine, the execution of a machine, quality control, etc. Each order has a due date and a sales price. The properties `requiresMachine`, `hasInput`, `needsConfig`, `isExecutedBy` are used together with the `Process` class. For example, they define which `Machine` is required for that process together with the needed configuration (tools assembled) and the input `Material`. All these details are required by the machine operator in the event of reconfiguring the machine for a specific order.

4.4 Aligning with Existing Ontologies

The ontology includes concepts and properties from well-known ontologies. The *Semantic Sensor Ontology*[4] provides us with a rich description of sensors, their measurements, devices and related concepts. The workforce and employees are described based on definitions by the *Friend of a friend* (FOAF)[5] ontology. Coordinates of factories and machines are based on the latitude and longitude definitions of the *W3C Geo Vocabulary*[6]. Finally, we reused geometrical concepts, such as the representation of polygons, from the *NeoGeo Geometry Ontology*[7].

[3] Prefixes are defined according to http://prefix.cc.
[4] https://www.w3.org/2005/Incubator/ssn/ssnx/ssn.
[5] http://xmlns.com/foaf/spec/.
[6] https://www.w3.org/2003/01/geo/wgs84_pos.
[7] http://geovocab.org/geometry.html.

5 Implementation and Application

We developed a software system that implements the presented Semantic Factory architecture and ontology and applied it to industry data. In this section, we first describe the front-end and back-end implementation. Then, we illustrate its usage by presenting various SPARQL queries that retrieve information for different production management tasks.

5.1 Front-End

The front-end is realized as a web application to facilitate access from different devices. The decision is motivated by the diversity of IT systems, platforms and devices usually deployed in a factory. The application uses the web framework *AngularJS*[8], which follows the model-view-controller design pattern to separate logic from representation. We created multiple views to address the collected requirements:

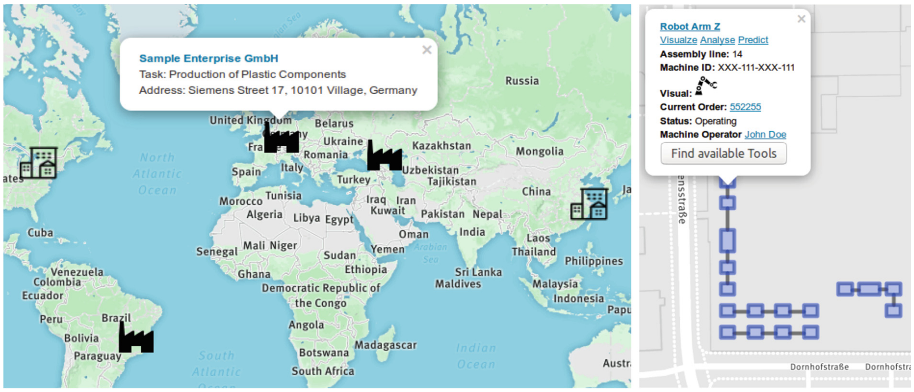
The *map view* (Fig. 4a) projects all instances (e.g. buildings, machines) with a geographical representation on a map by making use of the map provider *MapBox API*[9]. This API offers map tiles based on the open geographical database *OpenStreetMap*[10]. The projection itself is realized using the *leaflet.js*[11] JavaScript library. Coordinates in the factory ontology are represented using the *NeoGeo Geometry Ontology*[12] concepts and properties translated into `leaflet` geographical objects to be drawn on the map.

Further information, such as the person currently operating a machine, which order is executed or the status of a machine, is provided in the *machine view* that can be opened from the map view (see Fig. 4b) or independently by the machine operator. Besides static information, the pop-up contains also links to operational views and services. For example, the machine operator can follow the order link to retrieve additional information of that order. Furthermore, based on the tools required for the next machine operation, the “Find available tools” functionality points to the respective geographical location of the tools in the factory. The links “Visualize”, “Analyze” and “Predict” point to external pages that provide additional graphical content about the machine, machine usage indicators and prediction dates when machine parts are worn out and need to be replaced.

5.2 Back-End

The back-end consists of a Python web server[13] that supports REST API calls from the front-end. Each request triggers the generation of SPARQL queries

[8] https://angularjs.org.
[9] https://www.mapbox.com/developers/.
[10] http://www.openstreetmap.org.
[11] http://www.leafletjs.com.
[12] http://geovocab.org/geometry.html.
[13] http://flask.pocoo.org.


Sample Enterprise GmbH
Task: Production of Plastic Components
Address: Siemens Street 17, 10101 Village, Germany

Robot Arm Z
Visualze Analyse Predict
Assembly line: 14
Machine ID: XXX-111-XXX-111
Visual:
Current Order: 552255
Status: Operating
Machine Operator John Doe
Find available Tools

(a) Map View (b) Machine View

Fig. 4. Implementation of the Semantic Factory (geographic views).

executed either on the factory ontology or the production databases. To provide access to the ontology, the Python library *rdflib*[14] is used.

Access to the relational production databases is realized using the `D2RQ`[15] system. `D2RQ` provides a generator for creating an RDF mapping file for the database tables and columns, thus preserving the schema of the relational database. Using the mapping file, incoming SPARQL queries are translated ad-hoc into SQL queries and are executed on the respective database. Thus, `D2RQ` acts as a gateway between the web server and relational databases.

Once the data is obtained, it is returned in JSON data format[16] and processed by the respective front-end controller.

5.3 Factory Queries

In the following, we provide a set of SPARQL queries that demonstrate the usage of the factory ontology.

Order Feasibility Check. Listing 1.1 shows a query that determines if certain machines in the factory are free to use or already scheduled for other production plans. Each factory work order contains a list of tasks to be completed by a different machine. Thus, each needed machine is checked for its availability. Only if all machines are available, the query returns a positive answer such that the work order can be started. The query always checks the current state of the factory.

[14] https://github.com/RDFLib/rdflib.
[15] http://d2rq.org.
[16] http://json.org.

```
ASK
{
        # get tasks
        ?order a :WorkOrder .
        ?order :requiredMachines ?machineList .
        ?machineList rdfs:member ?machine.

        # check if the needed machines are free
        EXISTS { ?machine :isFree false } .
}
```

Listing 1.1. Order feasibility check

Retrieve Geographical Coordinates. Listing 1.2 shows a query to retrieve the machines, their names and coordinates. The outline of a machine is conceived as a polygon, represented as an `rdf:List` of geographical points, each with latitude and longitude, which is linked to the machine using the `ngeo:posList` datatype property. This information is returned to the front-end to be projected on the world map.

```
SELECT ?machine ?label
            (GROUP_CONCAT( ?lat ; separator=";") AS ?lats)
            (GROUP_CONCAT(?long ; separator=";") AS ?longs)
WHERE {
        ?machine rdfs:label ?label .
        ?machine ngeo:posList/rdf:rest*/rdf:first ?point .
        ?point geo:lat ?lat .
        ?point geo:long ?long .
} GROUP BY ?machine ?label
```

Listing 1.2. Retrieve geographical coordinates of machines

Suitable Assembly Lines. Listing 1.3 shows a query to find suitable assembly lines with regard to their sequence. Assembly lines that contain more machines than required but fulfill the correct order are still considered suitable. Thus, certain stations may be skipped within an assembly line.

Suppose, for example, that the machines `2` and `4` are required for an order. Suitable assembly lines include those having the following sequences of machines: `2,4` or `1,2,3,4,5`. Sequences such as `4,2` or `4,3,2` are considered non-suitable.

The query itself works as follows: First, assembly line candidates are filtered (`MINUS`) based on whether they contain the required machines in the work order. Second, of those assembly lines, the position of the needed machines is calculated. This is achieved by preparing the needed order (`?reqSequence`) and then retrieving the position of each machine in that order (`?machineSeq`). Third, these sequences are concatenated into strings and, finally, it is checked by a regular expression if the sequence is increasing.[17]

```
SELECT ?assemblyLine {
  ?assemblyLine :machineList ?lists .

  # filter all assembly lines with the wrong order
  FILTER REGEX(?seq,"^0*1*2*3*4*5*6*7*8*9*$")

  # concatenate order of the lists into a string
  {SELECT ?lists (GROUP_CONCAT(?machineSeq; separator="")
                              AS ?seq)

   #Machine Sequence, _sorted_ by required order Sequence
  {SELECT ?lists ?machineSeq {
        {SELECT ?lists ?machineInstance ?machines
        (STRAFTER(STR(?memberProp), "_") AS ?machineSeq)
               {?lists rdfs:member ?machineInstance .
                ?lists ?memberProp ?machineInstance .
                ?machineInstance a ?machines . }}

   # get required Sequence of the Work Order
   {SELECT ?machines (STRAFTER(STR(?prop), "_")
                             AS ?requiredSeq) {
            :sampleOrder :requiredMachines ?orderList .
            ?orderList   ?prop             ?machines .
            FILTER (STRSTARTS(STR(?prop), STR(rdf:_)))}
            ORDER BY ?requiredSeq}

    # Identify Assembly Lines Candidates
  {SELECT ?lists ?assembly
         {?assembly d:machineList ?lists.}}
   MINUS
  {SELECT ?lists {
           ?lists       a                   rdf:Seq .
           ?workOrder   :requiredMachines  ?neededMachineList
                .
           ?neededMachineList  rdfs:member ?machineType .
    FILTER NOT EXISTS{?lists (rdfs:member/a) ?machineType
        .}}}
   } ORDER BY ?lists ?requiredSeq
} GROUP BY ?lists }}
```

Listing 1.3. Find suitable assembly lines

[17] The query is limited to sequences of up to 9 machines but may be extended.

6 Evaluation

We evaluated our factory ontology and software application by testing the performance of the SPARQL queries introduced in Subsect. 5.3. These queries were chosen due to their representativeness in an industrial setting. For that, we prepared multiple datasets, consisting of 10 K, 100 K, 1 M, 2 M and 5 M triples. The datasets contain generated test data based on our factory ontology, such as order information, workforce details, assembly lines, etc.

The queries were executed using the *ARQ* SPARQL processor version 2.13.0[18]. The machine we used for the experiment contains 8 GB of RAM, 256 GB SSD and an Intel i7-3537U CPU with 2.00 GHz.

Figure 5 depicts the results of the performance evaluation. While the growth for the "Retrieve machine coordinates" query of Listing 1.2 is linear, a response time of 25 s in larger datasets is not satisfactory for a front-end application. Thus, large datasets should be split to keep the execution time end-user friendly.

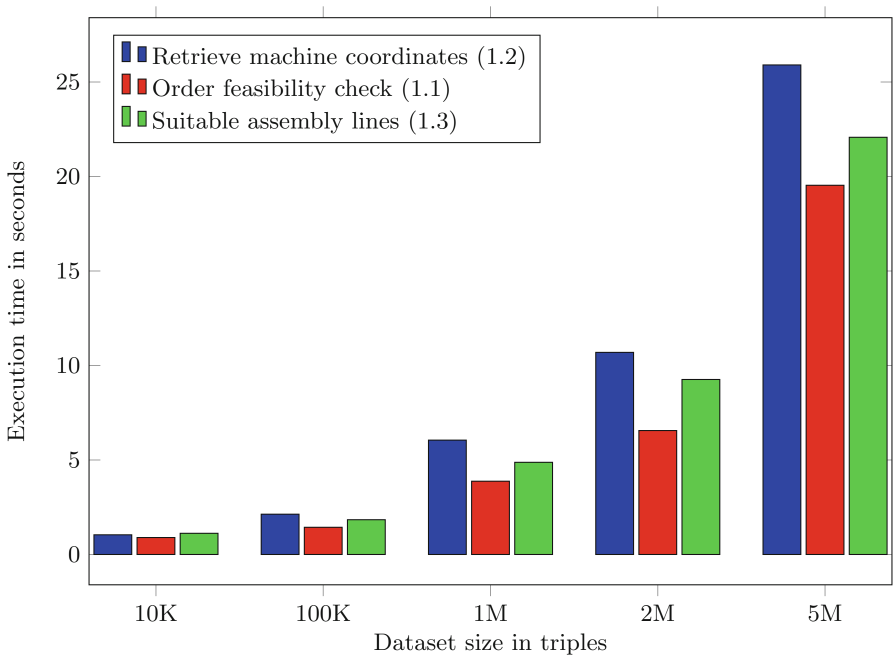


Fig. 5. Query execution performance

Similarly, as in the previous query, the execution time for the "Order Feasibility Check" query of Listing 1.1 grows linearly. While the overall performance

[18] https://jena.apache.org/documentation/query/.

is slightly better, one should nevertheless keep machine status data in an isolated dataset.

Finally, for the large "Suitable Assembly Lines" query of Listing 1.3, it is rather surprising that the execution time is quite similar to the short previous queries. As before, with a linear growth, the performance evolution becomes predictable and it is recommended to split instance data depending on certain services.

Overall, ontology-centered web applications are feasible with a satisfactory performance. As a common rule of thumb, the acceptable response time for complex operations is less than 10 s in order to keep the user's attention on the task [9]. Thus, certain crucial instance data should be kept in different triple stores to stay below that threshold.

However, the `d2rq` system for accessing relational databases reveals performance issues. First experiments using `ontop`[19] to query databases seem to be a promising alternative.

7 Conclusion and Future Work

The use of data-centric approaches in engineering, manufacturing and production is currently a widely discussed topic (cf. Industry 4.0, smart manufacturing or cyber-physical systems initiatives). The complexity of data integration in general is perceived to be one of the major bottlenecks of the field. A key issue in engineering, manufacturing and production is to be able to efficiently and effectively manage factories.

This paper described an ontology and an integration infrastructure to obtain a holistic view of the status of a factory from different perspectives. We see the work presented in this paper as a first step towards establishing an ontology-based integration approach for manufacturing, which is centered around a common information model, but at the same time supports the management of data in a decentralized manner in the existing systems of record. The integration follows a loosely-coupled architecture, where the decentralized data sources are mapped on demand to the factory ontology. The factory ontology is not supposed to be a fixed, monolithic schema, but rather a flexible, evolving and interlinked knowledge fabric. For this purpose, we have developed the collaborative vocabulary development methodology and support environment VoCol [5].

We see a number of directions for future work. In particular, the Semantic Factory approach could be expanded from single factories to an integration approach covering the entire enterprise as well as supply networks (e.g. based on the SCOR model [10]) involving a large number of organizations. Another promising direction of future work is the exploitation of the integrated data for advanced analytics and forecasting [11].

Acknowledgments. This work has been supported by the German Federal Ministry of Education and Research (BMBF) in the context of the projects LUCID (grant no. 01IS14019C), SDI-X (no. 01IS15035C) and Industrial Data Space (no. 01IS15054).

[19] http://ontop.inf.unibz.it.

References

1. Ameri, F., Patil, L.: Digital manufacturing market: a semantic web-based framework for agile supply chain deployment. J. Intell. Manuf. **23**(5), 1817–1832 (2012)
2. Brettel, M., Friederichsen, N., Keller, M., Rosenberg, M.: How virtualization, decentralization and network building change the manufacturing landscape: an industry 4.0 perspective. Int. J. Mech. Ind. Sci. Eng. **8**(1), 37–44 (2014)
3. Büscher, C., Voet, H., Krunke, M., Burggräf, P., Meisen, T., Jeschke, S.: Semantic information modelling for factory planning projects. Procedia CIRP **41**, 478–483 (2016)
4. Chen, R.S., Tu, M.A.: Development of an agent-based system for manufacturing control and coordination with ontology and rfid technology. Expert Syst. Appl. **36**(4), 7581–7593 (2009)
5. 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. In: 20th International Conference on Knowledge Engineering and Knowledge Management. Springer Verlag (2016, in print)
6. Hermann, M., Pentek, T., Otto, B.: Design principles for industrie 4.0 scenarios: a literature review. Technische Universität Dortmund, Dortmund (2015)
7. Kim, K.Y., Manley, D.G., Yang, H.: Ontology-based assembly design and information sharing for collaborative product development. Comput. Aided Des. **38**(12), 1233–1250 (2006)
8. Newman, D., Gall, N., Lapkin, A.: Gartner defines enterprise information architecture. Gartner Group (2008)
9. Nielsen, J.: Response times: The 3 important limits. Usability Engineering (1993)
10. Petersen, N., Grangel-González, I., Coskun, G., Auer, S., Frommhold, M., Tramp, S., Lefrançois, M., Zimmermann, A.: SCORVoc: vocabulary-based information integration and exchange in supply networks. In: 10th International Conference on Semantic Computing (ICSC 2016), pp. 132–139. IEEE (2016)
11. Petersen, N., Lange, C., Auer, S., Frommhold, M., Tramp, S.: Towards federated, semantics-based supply chain analytics. In: Abramowicz, W., Alt, R., Franczyk, B. (eds.) BIS 2016. LNBIP, vol. 255, pp. 436–447. Springer, Heidelberg (2016). doi:10.1007/978-3-319-39426-8_34
12. Sacco, M., Pedrazzoli, P., Terkaj, W.: VFF: virtual factory framework. In: Proceedings of 16th International Conference on Concurrent Enterprising (ICE 2010), pp. 21–23. IEEE (2010)
13. Terkaj, W., Urgo, M.: Virtual factory data model to support performance evaluation of production systems. In: Proceedings of the Workshop on Ontology and Semantic Web for Manufacturing (OSEMA 2012). CEUR-WS, vol. 886, pp. 24–27 (2012)
14. Upton, D.: The real virtual factory. Harvard Bus. Rev. **74**(4), 123–133 (1996)
15. Uschold, M., Gruninger, M., et al.: Ontologies: principles, methods and applications. Knowl. Eng. Rev. **11**(2), 93–136 (1996)
16. Want, R.: An introduction to RFID technology. IEEE Pervasive Comput. **5**(1), 25–33 (2006)
17. Zuehlke, D.: SmartFactory–towards a factory-of-things. Annu. Rev. Control **34**(1), 129–138 (2010)

Author Index