Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 4874 – 4884

doi:10.3906/elk-1412-28

Leveraging linked open data information extraction for data mining applications

Rajesh MAHULE*, Om Prakash VYAS
Department of Information Technology, Indian Institute of Information Technology, Allahabad, India

Received: 05.12.2014 • **Accepted/Published Online:** 18.09.2015 • **Final Version:** 06.12.2016

Abstract: The linked open data cloud, with a huge volume of data from heterogeneous and interlinked datasets, has turned the Web into a large data store. It has attracted the attention of both developers and researchers in the last few years, opening up new dimensions in machine learning and knowledge discovery. Information extraction procedures for these processes use different approaches, e.g., template-based, federated to multiple sources, fixed depth link traversal, etc. They are limited by problems in online access to datasets' SPARQL endpoints, such as servers being down for maintenance, bandwidth narrowing, limited numbers of access points to datasets in particular time slots, etc., which may result in imprecise and incomplete sets of feature vector generation, affecting the quality of knowledge discovered. The work presented here addresses the disadvantages of online data retrieval by proposing a simple and automatic way to extract features from the linked open data cloud using a linked traversal approach in a local environment with previously identified and known sets of interlinked RDF datasets. The user is given the flexibility to determine the depth of the neighboring properties to be traversed for information extraction to generate the feature vector, which can be used for machine learning and knowledge discovery. The experiment is performed locally with Virtuoso Triple Store for storage of datasets and an interface developed to build the feature vector. The evaluation is performed by comparing the obtained feature vector with gold standard instances annotated manually and with a case study for estimating the effects of demography in movie production for a country. The advantage of the proposed approach lies in overcoming problems with online access of data from the linked data cloud, RDF dataset integration in both local and web environments to build feature vectors for machine learning, and generating background knowledge from the linked data cloud.

Key words: Information extraction from linked open data, semantic Web data mining, mining linked data, linked open data

1. Introduction

The linked open data (LOD) cloud has attracted a lot of attention from both developers and the research community in recent years, with a huge volume of linked data produced following the linked open data principles [1] with consumption in different domains for varied application types, such as classifying events from Wikipedia, classifying Tweets, explaining statistics, as a source of additional information, etc. One of the main attractions of creating applications with the LOD datasets over traditional applications is in the nature of schema-flexible applications. The traditional applications generally are developed with fixed schema, which determines the way in which information is stored, presented, and manipulated. The main problem with fixed-schema applications is collecting information from multiple applications because of the applications being unaware of each other's schema. With the advent of the web of data (https://wed.fbk.eu), reflecting a vast number of distinct RDF schemas (www.w3.org/TR/rdf-schema/), applications with LOD can effectively

*Correspondence: rmahule71@gmail.com

present and manipulate information in any schema already available and/or create one as required. However, most of the research carried out in the area of Semantic Web and LOD is devoted to the underlying technologies, and there has been very little work and research in the direction of the development of user applications with LOD [2]. This work of information extraction can be seen as a step towards the development of small user applications using multiple RDF datasets developed in a local environment. However, the main motivation of this paper is to create feature vectors from multiple datasets of LOD for data mining applications.

This paper begins with the motivation and problem statement regarding the approaches to extracting information and building a feature vector table from the LOD cloud, which can be used with traditional machine learning algorithms. We continue by exploring related work in Section 3, the detailed method and algorithm in Section 4, and the experimental evaluation in Section 5, and finally Section 6 concludes the paper with our findings and a look at work for the future.

2. Motivation and problem statement

There have been approaches to developing applications and consuming the data from the LOD cloud in the field of data mining and machine learning, by extracting the data mostly in RDF formats and transforming it into feature vector form so that the already developed and available state-of-the-art machine learning algorithms can be directly applied to it. In a supervised environment, the process of selection of appropriate attributes and creation of propositional feature vectors [3] for machine learning algorithms requires an in-depth understanding of the schema of the available dataset, which can be easily determined for a single dataset. However, if the data are extracted from multiple interconnected datasets as available in the LOD, determining the schema is a tedious task and hence the creation of the feature vector.

In addition, the tasks required for extraction of data in an integrated environment from multiple heterogeneous sources pose various challenges. Commonly, the query required to access data from multiple sources must be mapped to distributed data sources with query brokering, optimized execution, and result aggregation as in federated SPARQL queries, making the process expensive and time-consuming. In systems where the data sources are hardwired, it is necessary to rewrite portions of the application whenever there is a change in the data source. Moving all data to a data store gives flexibility in access, but at the cost of reimplementing and updating if the data sources are frequently changing.

However, there has been a handful of studies in this area of information extraction and creation of feature vectors from RDF datasets (https://dvcs.w3.org/hg/rdf/raw-file/default/rdf-dataset/index.html) available in the LOD cloud. Some of the approaches use template-based SPARQL queries [4] to generate the feature vector from single RDF datasets, and in some approaches, a federated SPARQL query is used to generate the feature vector from multiple RDF datasets of the LOD cloud [5]. Some approaches use immediate neighboring properties and the concise bound description approach for information extraction to be used to build the feature vector [6]. Most of the approaches used in the past follow a fixed depth and/or fixed number of nodes for instance extraction.

Looking into the challenges and the previous work done on information extraction [3–6], we have proposed an intermediate solution with a method to extract information from the LOD cloud using the linked traversal based query execution [7] approach in a local environment with preidentified and known sets of interlinked RDF datasets from the linked open data cloud; the user is given the flexibility to decide and set the depth (or the number of hops) of the neighboring properties to be traversed for information extraction to generate the feature vector for data mining applications.

To the best of our knowledge, no current system offers such flexibility for creating the feature vector with user involvement. The experiments are performed in a local environment with Virtuoso Triple Store (http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VOSTriple) to store RDF datasets in N-triples format, with programs in Java to interact, traverse, and extract information for building feature vectors. The approach can also be used for integrating datasets with SPARQL endpoints on the Web and for applications developed using multiple RDF datasets for background knowledge. Furthermore, different relational database tables and feature vectors created from the multiple LOD datasets can be put together and used for data mining with the help of this approach.

3. Related work

In contrast to the huge volume of data available for consumption in the LOD cloud, there has been only a little data consumed by data mining applications. Most of the data mining applications use the data from the LOD cloud as a background knowledge source, for example, to analyze the reasons why countries that lie in certain latitude ranges have a particular GDP value, or to analyze factors for the high/low corruption index for a country. In addition, there are applications that use the LOD cloud to enhance the dataset for a particular entity, e.g., a city, and hence add additional features to the entity to perform data analysis.

In one work, LiDDM [8], the authors applied techniques of data mining (clustering, classification, and association rules) to linked data by first extracting the data from individual LOD datasets using user-defined SPARQL queries, converting the result from multiple datasets into table form with JOIN operations, applying preprocessing techniques to format the data for mining, and then performing the data mining process. In [9], the authors proposed an approach to extract features from the structured Semantic Web knowledge bases for general learning tasks, using the ontological components that are used to organize the entities into class-subclass hierarchies with the help of the isA or type and subclass relationships. In [10], the author proposed to learn association rules from hybrid sources (RDBMS and ontologies) under the Open World Assumption (OWA). For this purpose, the definition of frequency (and thus of support and confidence) is changed so that unknown statements contribute with half of the weight of the true statements. In [4], FeGeLOD (feature generation from linked open data), a tool for preparing datasets for data mining and machine learning by generating additional features from LOD, was proposed, which uses the given input dataset, selects one of the attributes, and uses the LOD cloud to generate new features based on the 6 types of feature generators using simple filtering to select features.

In [11], the authors proposed an approach for converting a dataset from LOD to relational format so that it can be used with the traditional machine learning approaches to generate results to support strategic decision-making regarding procurement. In [12], an approach was presented using additional information from the LOD cloud to interpret the results of a sequential pattern mining process with an example case of data about students' enrollment in course modules. The authors discussed the advantage of exploring and adding additional information from the LOD for data mining and learning analytics. In [13], the authors proposed a machine learning approach to mine information from LOD using a self-training strategy. A number of algorithms like C 4.5, JRip, SVM, K-NN, and NB have been used to determine the best algorithm to mine linked data. In [14], the authors proposed an API that can be used to extract information from LOD based on the use of a network analysis algorithm. Additionally, a plug-in has been provided in the Rapidminer tool so that data from LOD can be extracted to it for data mining applications [15].

Most of the studies available in the literature, and some of those mentioned above, performed information

extraction either for converting it into a feature vector for machine learning or to add additional information from the LOD cloud from single and/or multiple datasets and apply data mining algorithms to it. The approach used in almost all of the work related to information extraction uses SPARQL or federated SPARQL queries that are hardwired in the application, the result of which is then transformed into a feature matrix. Our approach is different from these approaches, as it dynamically generates the SPARQL queries and traverses the preidentified set of datasets for information extraction and builds the feature vector that can be used for data mining applications, hence helping users with a little knowledge of the Semantic Web to explore the datasets and build applications.

4. Method and algorithm for building feature vector table

A general pipeline of typical machine learning with RDF data from the LOD cloud is shown in Figure 1 [16], with each of the steps represented as a black box, allowing for the discretion of the researcher to have some standardized method for it. From the pipeline, it is evident that there are 2 methods in which machine learning can be facilitated with data from the LOD cloud: graph-based learning and feature-based learning. It is presumed that the first method is more attractive and in the long term more opportunistic, as it exploits graph-based learning from LOD cloud data, as RDF data span a graph of resources with edges representing predicates. However, graph-based learning has its own disadvantages. In [17], the authors stated that it is important to notice that "in LOD no two different nodes in an RDF graph have the same URI unless the corresponding concept of each URI is considered. Thus, any graph mining would be restricted to concept mining and not data mining". The graph-based learning tasks mostly consist of unstructured classification, where the inputs are represented with fixed-length feature vectors, and the output is mainly a set of discrete labels. In contrast, in machine translation, both the inputs and outputs contain strings of words with variable length, resulting in varying possible output with unlimited numbers. Hence, we have opted to explore the second approach, i.e. by transforming the linked data into a tabular form that can be further propositionalized for creating features from LOD, with the advantage that the already developed and available state-of-the-art machine learning algorithms can be directly applied to it. Our work is confined to performing the instance extraction and feature extraction steps as in Figure 1 to create a feature vector that can be used for feature-based learning.

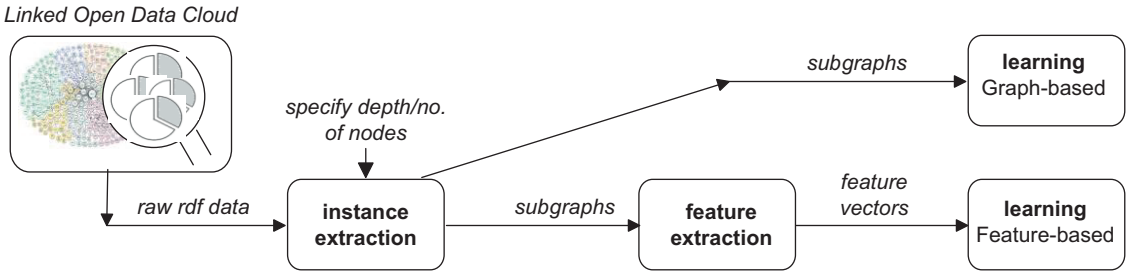

Figure 1. Overview of machine learning pipeline for RDF data from the LOD cloud.

The LOD cloud consists of a large number of datasets from various domains, with each dataset consisting of resource description framework (RDF) statements; the datasets are no longer separated neatly into instances. Here, each of the instances is represented by a resource in an RDF graph, which by itself contains no information. The neighborhood around the resource gives the description of the instances and a full subgraph is extracted to a given depth to create the feature vector. There are also interlinks to other datasets in the cloud using *owl:sameAs* links (http://www.w3.org/TR/owl-ref/), which may give additional descriptions about the resource.

In this work, we propose an algorithm for instance extraction with a predefined and known group of interlinked datasets from the LOD cloud with flexibility given to the user to choose the depth for extracting the description of a resource. Additionally, the descriptions of the resource can be generated with the *owl:sameAs* predicate links between the predefined datasets by again choosing the depth for extracting the description (here we have fixed it to 2, but it can be made variable by adding 1 more parameter). The algorithm is based on the link traversal method for crawling up to a particular depth and uses a dynamic SPARQL query (http://www.w3.org/TR/rdf-sparql-query/) to extract the description for the resources; it uses temporary files to store lists and other large data as intermediate storage, lowering the memory requirements for the system. The feature vector table is stored as a flat file in CSV format. Figure 2 [18] shows a short part of the LMDB RDF graph along with the converted CSV file for illustration. The first row of the CSV file shows the column header. The features with a hyphen between them represent multiple values (set valued data) for that particular attribute.

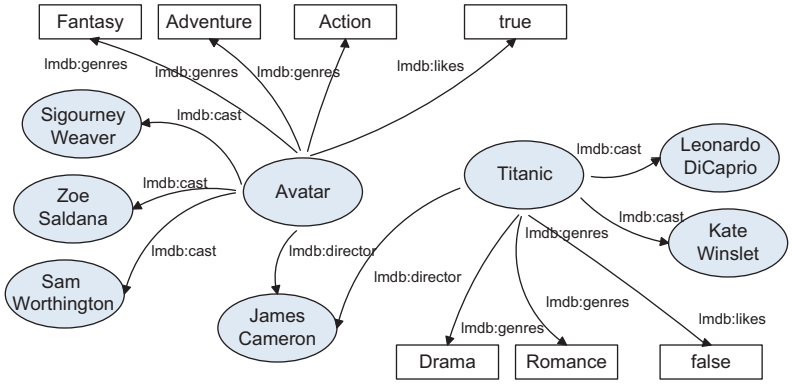

Figure 2. Short part of RDF dataset and its converted CSV file.

To understand the workings of our approach and the underlying algorithms, let there be a set of n datasets (D_1, D_2 ... D_n) in N-triples format stored locally, with D_1 being the initial dataset containing the resource to extract the descriptions in its immediate neighborhood within a restricted entity domain [6] (if any), which is to be converted into a feature vector table, T. The number of hops required to be traversed, N, is set (it may also be taken as input from the user). The process for integration and transformation on the fly is depicted in the folllowing algorithms.

In Algorithm 1, the first dataset is taken and the instance items are selected by executing a query, for example to get the instance list of drug names from the DBpedia [19] dataset (say, dataset D_1). A SPARQL query like *SELECT ?s WHERE { ?s a dbo:Drug. }* can be executed, the results of which serve as the entities in the entity list. This entity list is then used in Algorithm 2 to get the neighboring properties and object values and put them into the feature vector table T for the corresponding instance iteratively. If an instance already has an object value for an attribute in the table and the same attribute is found in the neighbor while traversing in the other hop, the object value is concatenated with the previous existing object value with a hyphen between them. This cell contributes to the set valued data [18] for the attribute and requires further preprocessing and is not part of the present study in this paper. During the extraction of properties, if the *owl:sameAs* link is found to be the neighboring predicate, function get_sameas_neighbours( ) as in Algorithm 3 is invoked, which looks for the *owl:sameAs* properties object value among the interlinked set of datasets available locally; if found, the algorithm with the get_attribute( ) function is invoked with that dataset to get neighboring properties, and the process executes to form the feature table T.

Algorithm 1. Generating feature set.

Input: Datasets $D_1, D_2 \ \ldots \ D_n$., *No. of hops N*

Output: *Data table T*

1. *Perform SPARQL Query and list the entity items with the entity type from* D_1 *the base dataset*
2. **foreach** *item in the entity list*
3. *function get_attribute(Dataset* D_i *, Subject* S_i, N)
4. **endfor**
5. **return**

Algorithm 2. Function get˙attribute(*Dataset* D_i, *Subject* S_i, *N*).

1. **while** (N >= 1)
2. **{**
3. *Select and list all neighboring properties of Si using SPARQL query (within the restricted entity domain, if any)*
4. **foreach** *of the properties generated in step 3 above, check the property value*
5. **if** *property value is owl:sameAs*
6. *get˙sameas˙neighbours(*O_{sa}, D_i*)*
7. **else** *check the object value*
8. **if** *the object value is literal (xsd:string, xsd:dateTime, xsd:decimal, and xsd:integer)*
9. *dereference predicate to get predicate name*
10. **if** *predicate name already exists as an attribute in table T*
11. **if** *object value already exists for the corresponding row entity*
12. *concatenate the current value with the new value with a hyphen in between*
13. **else**
14. *put the object value for the corresponding entity row entry and property name as attribute name (extract entity name and property name by looking at the label property using SPARQL query) in the table*
15. **else**
16. *create an additional attribute column in table T, put the object value for the corresponding entity row entry and property name as attribute name (extract entity name and property name by looking at the label property using SPARQL query) in the table*
17. **endif**
18. **else**
19. **if** *the object value is a URI*
20. $S_i \leftarrow O_i$
21. *decrement N*
22. **endif**
23. }
24. **if** *object property value for the* S_i *is a URI*
25. *dereference URI for both predicate value and object value*
26. *Go To step 10*
27. **endif**
28. **return**

Algorithm 3. Function get_sameas_neighbours(O_{sa}, D_i).

1. **foreach** D_X *except* D_i *in Datasets* D_1, D_2 ... D_n
2. *perform ASK SPARQL query to check the existence of object URI*
3. **if** *result of ASK Query == true*
4. *set active dataset* D_x, $N = 2$ *(assuming the neighboring properties up to only two hops are to be considered for owl:sameAs predicate links)*
5. $S_x \leftarrow$ *the object URI*
6. *get_attribute(Dataset* D_x, *Subject* S_x, N)
7. **endif**
8. **endfor**
9. **return**

Here the intermediate SPARQL queries are built and handled dynamically. The main drawback of the approach above is blank cells in the table for many of the instances, as all the attributes may not be present as predicated for the corresponding instance in the RDF graph, which is being optimized and will be presented in our future work. However, the aspect that promises to be positive and notable is the applicability of familiar tools and methodologies for machine learning, business analytics, knowledge discovery, and other purposes using the feature vector table.

5. Experimental evaluation of the proposed methodology

The evaluation part of the work is based on empirical case studies only. The first reason for this is that the other approaches mentioned in the literature use live queries to retrieve the data from the SPARQL endpoint of the datasets, which means the experiments are not reproducible for the load point and latency of the network over time. Secondly, the structure of the underlying RDF datasets queried in the previous works may change over time and hence may return different results for different instances of querying the SPARQL endpoint. Additionally, there may be problems with consistent online access to the SPARQL endpoint of the dataset, like the server being down for maintenance, bandwidth narrowing, a limited number of access points to particular datasets in a particular time slot, etc., which need to be taken into consideration if a comparison with existing systems is to be made, which is practically infeasible. Hence, comparison with existing approaches is difficult.

The effectiveness of generating a feature vector using our approach is discussed in this section. We empirically investigated our approach with comparison of the generated feature vector with those of the gold standards in an application developed to generate knowledge from the generated feature vector. These experiments were carried out mainly to confirm the scalability and performance of our approach. The experimental results verify the approach in that the feature vector generated from LOD is beneficial to machine learning and knowledge discovery. In the following subsections, we describe the learning tasks and the gold-standard comparison approach.

5.1. Comparison with the gold standard

The experiments were carried out with the generation of the feature vector table with mainly small RDF datasets downloaded as RDF dumps in n-triple format available on the Web. To evaluate the proposed method, we generated the feature vector from datasets available in the LOD, namely: (i) Sider—contains information about marketed drugs and their side effects; (ii) Diseasome—has a huge collection of disorders and disease genes along

with the genetic origin of diseases; (iii) Dailymed—contains chemical structures of drug compounds, therapeutic purposes, indications and usage, contraindications, warnings, precautions, adverse reactions, overdosage, and patient counseling; (iv) Drugbank—contains information about drugs including chemical and pharmaceutical data with sequence, structure, and pathway information. The Table below shows the details of the datasets used in the experiment.

Table. Details of the datasets used in the experiment.

Dataset	Size	Triples	Entities
Sider	16 MB	91,569	2674
Diseasome	12 MB	72,463	8152
Dailymed	15 MB	164,276	3600
Drugbank	98 MB	517,150	19,696

We chose these datasets because entities within these datasets are mostly linked together through *owl:sameAs* links, which can be traversed to extract the feature vector. Regarding the SPARQL endpoints, we created 4 endpoint services for each dataset dump loaded onto Virtuoso Triple Store installed on a local Linux machine; however, the same can be performed with the respective SPARQL endpoints available on the Web as LOD. The classical extract–transform–load (ETL) process as used in the data warehousing approach is followed, with the difference being that it follows the link traversal for selecting the relevant fragment of an instance for extraction. The program to extract data from the interlinked datasets and the creation of the feature table with the intermediate data structure required was performed in the Java programming language with Netbeans as the programming environment on an Intel I-7 machine with 12 GB RAM.

The evaluation of our methodology is performed in terms of how well it can support the creation of instances to form the feature vector table. Two human annotators selected valid instances starting with the Drugbank dataset and, as per our approach, traversed the set of predetermined datasets, constructing a gold standard with 25 instances including each term candidate with the tree traversal approach. In particular, these annotators did not have access to the results of the proposed algorithms when creating their feature vector table. We compared the automatically generated feature table for the instances in the gold-standard manually constructed feature table. The system achieved a precision of 80% and a recall of 68% in the task of instance extraction with the appropriate attributes and object values. However, the results are slightly low because of a problem with URI disambiguation and dereferencing, which is not discussed in detail in this paper but is available in the literature.

5.2. Case study: estimating the effect of demography in movie production

To estimate the effect of demography in movie production, 3 interlinked datasets available as RDF dumps from the LOD cloud, the World Factbook dataset, Dbpedia dataset, and Linked Movies Data Base (LMDB) dataset, are used. The World Factbook dataset provides information on the history, people, government, economy, population, geography, and other important data for all countries in the world. LMBD provides information on the country, actors, genre, year of production, director, producer, etc. for movies. The Dbpedia dataset gives structured information from Wikipedia info boxes and serves as a hub with interlinks with a huge number of datasets; it is used here as it has links to both the World Factbook dataset and LMBD.

All of the datasets are loaded into the local repository and the user is asked to select the number of hops to traverse in the dataset. The Dbpedia dataset is selected as the first dataset, which is the base dataset, and is traversed for type:country (as in Algorithm 1 in Section 4). The features from the base dataset are collected,

and during collection it is found that the *owl:same_as* property link of country exists (Algorithm 2 in Section 4), which is looked for in the other 2 datasets, the World Factbook dataset and LMDB (Algorithm 3 in Section 4). The feature vectors are generated, which are then given to the user to filter the feature vector for the required features using a user interface with all the feature attributes in the feature vector in a list; the user is prompted to make selections from the list. A selection is made from the feature vector table with different attributes like median age, population of the country, etc., of the World Factbook dataset and the details of the movie produced for each country, etc.; association rule mining is performed with it using WEKA API in our application after converting the selected features to the format suitable for data mining with user intervention through a user interface.

The process followed for generating the patterns is as depicted in Figure 3 below.

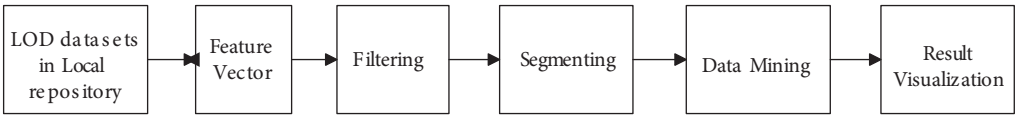

Figure 3. Process flow for data mining.

We found some interesting rules with the population and the median age in regards to their effect on the production of movies, etc. Through this experiment, we found that the generated feature with our approach plays an effective role in knowledge generation and machine learning.

6. Conclusion and future work

This paper can be seen as a step towards generating instances from the LOD cloud with a predetermined and known set of RDF datasets in a local environment that can be used to transform the linked data into a tabular form, which can be further propositionalized for creating features from LOD, thus bridging the gap between machine learning and Semantic Web technologies. The method is advantageous as it helps overcome some of the problems in online access to SPARQL endpoints of the dataset, such as the server being down for maintenance, bandwidth narrowing, limited number of accesses to a particular dataset, etc., as all ETL activities are performed locally. We have empirically tested the algorithms proposed in this work with the datasets as stated in the previous section and also with many other small interlinked datasets, and we have generated a feature vector table with a handful of instances and a good number of attributes to ensure that this methodology can be used for a number of related linked datasets in a local environment. During the evaluation, we have found that with an increase in either the size and/or the number of datasets, the performance of the system degrades. The main reason for this is the limited resources of a single machine system in terms of memory and processing capabilities, which can be encountered when running the application in a distributed environment. The other advantage of this approach is that it can be applied to integrating datasets for applications developed using multiple interlinked RDF datasets available on the Web, and also for applications requiring background data from the LOD cloud.

One of the important issues in this approach is the selection of the datasets and the selection of the base dataset. In this work, we have found that if the datasets so selected are not interlinked or if very few interlinks exist between the datasets, then the resulting feature vector is not very populated. In addition, there is serious information loss in the process of transformation from the linked data to tabular form as the datasets are downgraded to tabular format from the RDF form. Other problems, like the flooding of blank cells in the table for some instances that do not have the predicate attributes in the RDF graph, were found, but the aspect

that promises to be positive and notable is the application of familiar tools and methodologies for business analytics and knowledge discovery, and their ease of use in Semantic Web data.

In this work, we have also found that the feature vectors generated are generally of very large dimensions and may have inconcise, irrelevant, and unpropositionalized attributes, which is a further step in the process of knowledge discovery and needs to be considered as preprocessing steps in knowledge discovery and machine learning for feature vectors with semantic Web data.

References

[1] Bizer C, Heath T, Berners-Lee T. Linked data: the story so far. International Journal on Semantic Web and Information Systems 2009; 5: 1-22.

[2] Karger DR. The semantic web and end users: what's wrong and how to fix it. IEEE Internet Comput 2014; 18: 64-70.

[3] Ristoski P, Paulheim H. A comparison of propositionalization strategies for creating features from linked open data. In: CEUR Workshop Proceedings LD4KD 2014: Proceedings of the 1st Workshop on Linked Data for Knowledge Discovery Co-Located with European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases; 19 September 2014; Nancy, France. pp. 1-11.

[4] Paulheim H, Fürnkranz J. Unsupervised generation of data mining features from linked open data. In: WIMS '12: Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics; 6–8 June 2012; Craiova, Romania. New York, NY, USA: ACM. pp. 1-31.

[5] Jones J, Kuhn W, Keßler C, Scheider S. Making the web of data available via web feature services. In: Connecting a Digital Europe through Location and Place: International AGILE 2014 Conference; 13–16 June 2014; Castellon, Spain. Cham, Switzerland: Springer International Publishing. pp. 341-361.

[6] Grimnes G, Edwards P, Preece A. Instance based clustering of semantic web resources. **Lect Notes Comp Sci** 2008; 5021: 303-317.

[7] Hartig O, Bizer C, Freytag JC. Executing SPARQL queries over the web of linked data. In: Proceedings of the 8th International Semantic Web Conference; 2009. Berlin, Germany: Springer. pp. 293-309.

[8] Kappara PVN, Ichise R, Vyas OP. LiDDM: A data mining system for linked data. In: Workshop on Linked Data on the Web; 28–29 March 2011; Hyderabad, India.

[9] Cheng W, Kasneci G, Graepel T, Stern D, Herbrich R. Automated feature generation from structured knowledge. In: 20th ACM Conference on Information and Knowledge Management; 24–28 October 2011; Glasgow, UK. New York, NY, USA: ACM. pp. 1395-1404.

[10] Amato C, Bryl V, Serafini L. Data-driven logical reasoning. In: International Workshop on Uncertainty Reasoning for the Semantic Web; 2012. pp. 51-62.

[11] Menica EL, Holthausen S, Schulz A, Janssen F. Using data mining on linked open data for analyzing e-procurement information: a machine learning approach to the linked data mining challenge. In: Proceedings of the International Workshop on Data Mining on Linked Data with Linked Data Mining Challenge collocated with the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases; 23–27 September, 2013; Prague, Czech Republic.

[12] Mathieu A, Jay N. Interpreting data mining results with linked data for learning analytics: motivation, case study and directions. In: Proceedings of the Third International Conference on Learning Analytics and Knowledge. New York, NY, USA: ACM, 2013. pp. 155-164.

[13] Fanizzi N, d'Amato C, Esposito F. Mining linked open data through semi supervised learning methods based on self training. In: Proceedings of the 2012 IEEE Sixth International Conference on Semantic Computing; 19–21 September 2012; Palermo, Italy. Piscataway, NJ, USA: IEEE. pp. 277-284.

[14] Tsoukalas C, Dervos D, Gil JM, Francisco J, Montes JFA. TheMa: an API for mining linked datasets. In: 16th Panhellenic Conference on Informatics; 5–7 October 2012; Piraeus, Greece. Piscataway, NJ, USA: IEEE. pp. 449-453.

[15] Nolle A, Nemirovski G, Sicilia A, Pleguezuelos J. An approach for accessing linked open data for data mining purposes. In: Proceedings of RapidMiner Community Meeting and Conference; 27–30 August 2013; Porto, Portugal.

[16] Bloem P, Vries G. Machine learning on linked data: a position paper. In: Linked Data for Knowledge Discovery; 15–19 September 2014; Nancy, France. Heidelberg, Germany: Springer. p. 69.

[17] Abedjan Z, Naumann F. Context and target configurations for mining RDF data. In: Proceedings of the 1st International Workshop on Search and Mining Entity-Relationship Data; 24–28 October 2011; Glasgow, UK. New York, NY, USA: ACM. pp. 23-24.

[18] Khan MA, Grimnes GA, Dengel A. Two pre-processing operators for improved learning from semantic web data. In: First RapidMiner Community Meeting and Conference; 13–16 September 2010; Dortmund, Germany.

[19] Lehmann J, Isele R, Jakob M, Jentzsch A, Kontokostas D, Mendes PN, Hellmann S, Morsey M, Kleef PV, Auer S et al. DBpedia: a large-scale multilingual knowledge base extracted from Wikipedia. Semantic Web Journal 2014: 1-5.