ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **23**, Number 1, 2020, 55–68

Towards Knowledge Discovery in interlinked heterogeneous datasets of LOD cloud

Rajesh MAHULE , Ranjana VYAS , and Om Prakash VYAS

Indian Institute of Information Technology, Allahabad, India
E-mails: rmahule71@gmail.com, ranjanavyas@gmail.com, dropvyas@gmail.com

Abstract. In the last years, a huge volume of data was published on the web as *Linked Open Data* (LOD). Consuming and using this interlinked collection of heterogeneous data in classical data mining methods is a substantial challenge as it requires input in propositional *Feature Vector Table* (FVT) form. To overcome this consumption hurdle, this paper proposes a framework inspired by *Link Traversal Based Query Execution* (LTBQE) paradigm. The framework is designed to dynamically extract relevant features and build an FVT from a set of interlinked RDF datasets in a local environment. This article introduces a Content-Based similarity measure to evaluate generated FVT. Also, two representative data mining tasks are performed to evaluate the framework empirically which shows that the generated FVT assists in learning from heterogeneous LOD datasets. The evaluation work revealed some interesting patterns and also suggests an appropriate distance measure to handle dimensionality of the set-valued data attribute.

Key-words: Linked Open Data, Semantic Web, Data mining, Knowledge discovery, Feature vector generation.

1. Introduction

In the last decade *Linked Open Data* (LOD) cloud has evolved at a very fast rate [1]. The data available on the LOD cloud are structured, published using linked data principles, machine interpretable and covers various domains like media, publications, linguistic, government, geography, life sciences, social networking, etc., [2]. Most of the data on the LOD cloud is open and can be accessed using semantic web standards and SPARQL endpoints [3]. The huge volume of these linked data poses the LOD cloud as a knowledge base silo which can be harnessed for knowledge discovery and data mining. A number of machine learning methods are available to extract knowledge from these structured knowledge bases for classification purposes. The most popular approaches being *Inductive Logic Programming* (ILP) and graph based data mining. The

other method to learn from these structured data is to transform it into a feature vector using the process of propositionalization [4, 5]. The problem with these methods is the huge search space requiring the imposing of additional constraints [6]. The limit on crawl depth and length result in the discovery of substructures that are quite small and span only a few structural relations [7] and thus limits the finding of large discriminatory substructures. Also, classical machine learning methods need data to be represented in an attribute-value format, where the instances are represented using feature vectors of fixed length [8]. The methods to transform semantic web data into attribute-value format require the knowledge of RDF [9], RDFS [10], and SPARQL [11] etc. Also, extracting data from heterogeneous RDF datasets may result in new dimensions of knowledge. Therefore new methods of data extraction, reasoning or data representation is required to be explored. The method proposed in this paper is motivated by this need and follows the first line of research by providing a framework to extract features from the interlinked collection of LOD cloud datasets and build an FVT to be used in classical methods of data mining and knowledge discovery. Thus, the work proposed in this paper helps to bridge the gap between semantic web and the classical learning methods.

The remaining paper is organized as follows: section 2 outlines the related work, section 3 explains the feature vector generation process, section 4 discusses the experimental evaluation work and finally, section 5 concludes the paper.

2. Related Work

Most of the approaches available in the literature to extract data and perform knowledge discovery from LOD datasets use crawling based or SPARQL query-based method. The authors Grimes *et al.* in [12] have proposed and compared three approaches of instance extraction from the RDF dataset viz., (i) immediate neighborhood properties (ii) Concise bound Description [13] and (iii) Depth Limited Crawling. The approach used fixed depth and/or a fixed number of nodes for generating the feature vectors. The instances extracted from the dataset are used for clustering application on the semantic web data. The authors have found that in regard to the instance extraction methods, the clustering application depends upon the data being clustered.

The authors Kappara *et al.* in [14] have proposed a tool LiDDM, which facilitates users to build SPARQL queries and help in data extraction for data mining processes. Also, the tool provides options to include data from multiple data sources, data filtering, and data segmentation. Two use cases, using Dbpedia [1] , World-FactBook [2] and Linked Movie [3] datasets have been presented for the efficacy of this tool. The major drawback of the tool is the requirement of the preliminary knowledge of semantic web especially regarding terms and vocabularies used to formulate the SPARQL query. The authors Khan et al. in [15] have proposed a "Semweb plugin" for Rapid Miner [4] with RDF data pre-processing facilities. With the help of this plugin, users can specify SPARQL queries to select data and get it converted into an FVT.

The authors Cheng et al. in [16] have proposed an automated feature generation method in which the user specifies the "type" of features to be extracted, through a SPARQL query builder thus, making the task user dependent. A case study on the recommender system and tweet classification is used to evaluate the proposed approach.

[1] http://dbpedia.org
[2] https://old.datahub.io/dataset/cia-world-factbook
[3] http://www.linkedmdb.org/
[4] http://www.rapidminer.com/

The authors Paulheim *et al.* in [17] have proposed an approach to generate features using a set of six SPARQL queries from the LOD cloud datasets. The features generated with these queries are either binary or numerical aggregates. Two of the six queries are related to type and data properties of the instances and the other four queries are concerned with other resources in the dataset connected to the instances. It also considers qualified relations (which generate aggregates on the relation and type of entity) along with the incoming and outgoing relations of the instances with other resources in the data set. The work proposed by Fanizzi et al. [18] use self-training strategy which iteratively uses labeled instances to predict a label for unlabeled instances. In their work the authors have compared several data mining algorithms like KNN, JRip, C 4.5, SVM, Naïve Bayes etc., to depict algorithm most suitable for mining linked data. The authors Tsoukalas et al. in [19] have proposed an API using network analysis algorithm that extracts data from the linked data cloud. The authors Mynarz et al. in their work [20] have combined SPARQL queries submitted by the users with SPARQL aggregates to generate features from the LOD dataset.

The authors Mencia *et al.* in [21] have proposed a method to transform linked data into the relational form using a tree traversal approach in order to apply traditional machine learning algorithms. The results generated from the extracted data are used for strategic decision making in procurement. However, the authors concluded that their solution is specific to the provided dataset and no general guidelines can be given. The authors Mathieu *et al.* in their work [22] have proposed an approach for interpreting the results of sequential mining process using additional data from the LOD cloud in a case study regarding student enrolment data in various courses. The benefit in enhancing additional information from the LOD cloud for data analytics and data mining is also discussed by the authors in their work. The authors Nolle et al. in [23] have proposed a plug-in for the Rapid miner tool to extract data from the LOD cloud by helping the user create SPARQL queries and transform the extracted data into the Rapid miner's internal data representation for further processing using conventional operators and thus to use the semantic web data in data mining applications. The work proposed by Paulheim *et al.* in [24] and Ritoski *et al.* in [25] is an extension to the work carried out by Paulheim *et al.* in [17]; the authors have proposed Rapid Miner Linked Open Data Extension – a tool that supports the user in all steps of performing data mining with data from the LOD cloud with attributes extracted using six SPARQL query feature generators as proposed by the authors Paulheim *et al.* in [17].

The authors Mahule *et al.* in their work [26] have proposed an approach to generate a feature vector table from multiple interlinked RDF datasets using LTBQE [27] and dynamic SPARQL queries. The approach initiates with the user provided seed URI of the class, the features for the instances of which are generated based on the neighboring properties up to a certain depth by the user. The approach traverse to the interlinked datasets using owl:sameAs[5] predicate relations to extract instance related features. However, only neighboring properties are considered for features in this approach.

The methods proposed by authors discussed above, do not follow a uniform way to extract instance related data. The methods used by authors in [12, 19, 21] follow graph based traversal method for instance extraction. The methods proposed by authors in [14–15, 20, 23] provide an interface to help users build SPARQL queries for data extraction. The methods proposed by authors in [16–18, 22, 24, 25], have SPARQL queries hard-wired in the system to extract data from the LOD cloud. These SPARQL queries needs to be build every time a new knowledge discovery task is to be performed to have feature data of entities. These queries only extract selected

[5] https://www.w3.org/TR/owl-ref/#sameAs-def

features already known or are specific to problem in hand missing some of the features which may be of more importance in knowledge discovery. Other issues with the methods discussed above are (i) does not follow link traversal mechanism to extract features within and across the interlinked datasets and thus do not take advantage of the very basic postulate of a "web of data" (ii) require the users to have the knowledge of RDF, RDF schema, linked data, SPARQL query language along with understanding of the structure of RDF dataset for data extraction (iii) does not address data heterogeneity problem for instances belonging to same class which may have multiple structure (iv) in the process of feature extraction metadata and schema related properties intertwined with instance data needs to be identified and filtered which otherwise may result in increase of dimension in the FVT and thus require extra processing.

In this paper, we propose a solution to overcome the above shortcomings. In particular, we address the problem of feature extraction from multiple datasets of the LOD cloud by providing the algorithms that cater the need of FVT generation for the identified instance entities belonging to interlinked RDF datasets. The algorithm performs on the fly filtering of metadata and schema related data (T-Box data). The method not only extracts neighborhood data properties as features but also extracts hypernymy [6] and *Concise Bound Description* (CBD) [13] data to enhance the features. The generated FVT may be used as input in the knowledge discovery process to have new insights from the heterogeneous collection of interlinked datasets of the LOD.

3. The Feature Vector Generation Process

To construct *feature vector table* (FVT) from a set of interlinked RDF dataset requires handling of two important issues: (i) recognizing instance entity in the RDF dataset and extracting features related to it and (ii) transformation of extracted features to build feature vector table. The proposed framework illustrated in Figure 1 shows the complete process of feature extraction starting from seed URI mapping, dynamic SPARQL query generation, traversal, features extraction & filtering, transformation, and FVT generation for the identified instance resource entities from the interlinked RDF graph datasets. The following sub-sections present the pre-requisites of the proposed work, components of the process and the underlying algorithms to carry out the process of the proposed framework.

3.1. Pre-requisite

The LOD cloud is a collection of interlinked RDF datasets with their respective SPARQL endpoints. Each dataset contains knowledge specific to a particular domain with statements in the form of triples (Subject, Predicate, and Object) where "Predicate" maps the relationship between the Subject and Object. The statements are constructed using languages RDF, RDFS, and OWL, where RDF (A-Box component) describes data and RDFS and OWL represents the terminology of the data set (T-Box component) [28]. The triples belonging to A-Box and T-Box is linked by particular classes (defined in T-Box and instantiated in A-Box), which can be extracted to get the structure of the dataset and the instance data to create the FVT. SPARQL queries are used to extract data from these RDF datasets. The SPARQL queries are generated dynamically and are used to traverse the RDF dataset to extract instance related data using an approach *Link Traversal Based Query Execution* (LTBQE) [27]. LTBQE allows discovering data by following links on

[6] https://en.wikipedia.org/wiki/Hyponymy_and_hypernymy

the web for the data on hand. The links followed point to entities within or across the interlinked datasets on the web which may further be traversed to point to other entities until the desired result or stopping criteria is reached. The discovered entities in the link are dereferenced for actual data values and are used to populate and build the FVT.

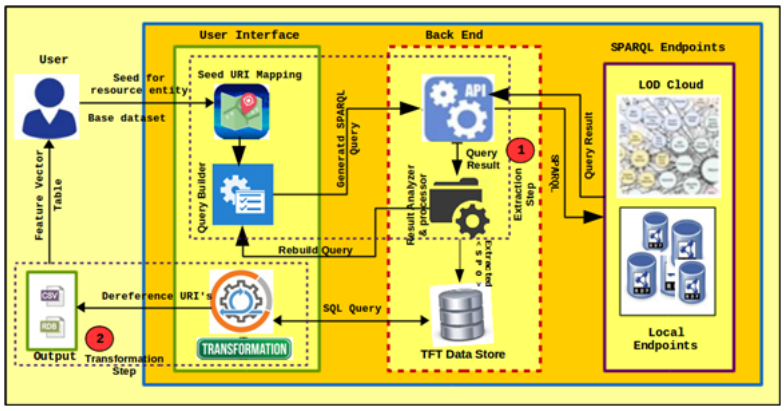

Fig. 1. Proposed framework to generate feature vector table from interlinked RDF datasets.

3.2. Instance Feature Extraction Process

The process to extract features from the interlinked RDF dataset initiates with the mapping of seed text to its corresponding URI in one of the identified RDF dataset (base dataset). The fetched seed URI is used by the Query Builder module of the framework to extract the "class" of seed instance entity which helps in generating instance entity list. For example, to get the list of all drug names from the DBpedia dataset, seed text "drug" is searched. <Subject> of the label match is traversed to get its class dbo:Drug which is identified as the seed URI. The SPARQL query SELECT ?s where {?s a dbo:Drug} is used which generates a list of all the entities of type drug. The entities in the fetched list of drugs are used as instance resources. The feature attributes for these entities are extracted up to a given depth to build a FVT. This FVT contain records related to the different drugs in the set of RDF datasets. The process to perform seed URI match and generate instance resource entity_list as performed by the Query Builder module of the framework is illustrated in Algorithm 1.

Algorithm 1: Generate_feature_set.

Input: Datasets D_1, D_2 ... D_n ., No. of hops N, Seed_Text
Output: Data table Triple_Feature_Table
1. DL $\leftarrow$ 0
2. Perform SPARQL Query to get the entity_list using the Seed_Text from D1 the base dataset
3. for each item S_i in the entity_list
4. function get_attribute(Dataset D_i, Subject S_i, N)
5. end for
6. **Return**

Algorithm 2: get_attribute.

Input:
D_i: Dataset;
S_i: Subject;
N: Depth level;
1. while (N >= 1)
2. {
3. Select and list all hypernymy features for S_i
4. Add generated S_i P_i O_i in Triple_Feature_Table
5. Select and list all neighboring properties of S_i using SPARQL query (within the restricted entity domain, if any) and store in list P
6. for each property P_i in P
7. if (P_i = = "owlSameAs")
8. fetch O_i for P_i using SPARQL query
9. for each dataset D_j in (D_1 ... D_n)), where $i \neq j$
10. if (ASK sparql query with O_i in datasets D_j = = True)
11. Add D_j to dataset_list DL_s
12. DL ← DL+1
13. Store subject S_i in S, store S_i = O_i
14. for each dataset in dataset_list DL_s
15. *function get_attribute(Dataset D_i , Subject S_i, N)*
16. DL ← DL – 1
17. end for
18. end if
19. end for
20. else
21. Get object value O_i for P_i
22. if O_i is a literal (*i.e., xsd:string, xsd:dateTime, xsd:decimal and xsd:integer)*
23. if (DL > = 1)
24. Write S, P_i, O_i in Triple_Feature_Table
25. else
26. Write S_i, P_i, O_i in Triple_Feature_Table
27. end if
28. if O_i is a URI
29. if (N > = 1) // Check depth level N (*i.e.* should be > = 1) required for more properties
30. Decrement N
31. S←S_i
32 S_i ← object value O_i
33. Goto step 5
34. else
35. Write S_i, P_i, O_i in Triple_Feature_Table
36. end if
37. If O_i is a blank-node
38. //Get properties of Blank-nodes *i.e.*, Bp_i's for O_i
39. S← S_i
40. Set S_i ← O_i
41. Get S_i, Bp_i, O_i and store S, P_i+Bp_j, O_j in Triple_Feature_Table
42. end if
43. end if
44. decrement N
45. end for
46. **Return**

The Query Builder unit works in tandem with the Result Analyzer and processor unit to get the query results, process the result for meta-data filtering, storing the extracted data into TFT data store and providing the intermediate result to query builder to generate further SPARQL queries. It dynamically builds SPARQL queries to (i) get hypernymy based features, (ii) get inbound, outbound and blank node properties (Concise Bound Description sub-graph) of the entities (except restricted properties) [13], (iii) look for owl:sameAs[5] properties in the extracted properties, (iv) check the availability of interlinked dataset in the local repository (when performed in a local triple store) and if available get again the CBD properties of the entity up to the desired depth, (v) store the entity URI, property URI, object value/URI in the TFT data store. The process is repeated to extract feature for each instance resource entity. The detail of the feature extraction process as performed by the Query Builder module of the framework is illustrated in Algorithm 2.

3.3. Transformation of Extracted Features to Build FVT

Algorithm 3: generate_FVT .

Input: Data table Triple_Feature_Table
Output: Database table T
1. Set Entity_Counter=0, Property_Counter=0
2. for each triple in the Triple_Feature_Table
3. Read triple data S_i P_i O_i
4. if S_i does not exist in Entity_list
5. Add S_i to Entity_list
6. Increment Entity_Counter
7. end if
8. if predicate P_i does not exist in Property_list
9. Add P_i to Property_List
10. Increment Property_Counter
11. end if
12. end for
13. Create Table T with row_size = Entity_Counter && column_size = Property_Counter with Column_name as P_i in the Property_list
14. for each Entity S_i in the Entity_list
15. Get predicates $P_i \ldots P_n$ and corresponding $O_i \ldots O_n$
16. Append Table T with the subject label, Object label/literal value for appropriate property URI on column concatenating existing column property value with a "-" symbol in between
17. end for
18. Dereference column header of Property URI's with label properties value
19. **Return**

The transformation of extracted features to build FVT is performed by the "Transformation" unit of the proposed framework. It uses the triples stored in the TFT data store to determine the database structure for FVT. The number of columns in the database is determined by the maximum number of possible "attributes" for an "instance" extracted from RDF datasets. The

<http URIs> of the properties are used for the attribute name of the columns in the created database. Instance entity (subject) related attribute (property), values (object) are populated in the corresponding column of the database using SQL insert query operation. Blank entries are inserted for entities not having particular attributes. Attributes having more than one object value are concatenated with the existing data value along with a "–" (hyphen) symbol resulting in "set-valued data". The details of the work performed by the "Transformation" unit to generate FVT from the TFT data store is illustrated in Algorithm 3. The <http URIs> in the populated FVT database is dereferenced for its label name in the final step of transformation. Finally, the data from the database is extracted and saved into a CSV file to be used for data mining applications.

4. Experimental Evaluation And Results

The authors conducted a set of comprehensive experiments for learning on heterogeneous knowledge using real-world datasets from the LOD cloud. This section describes the experimental set-up and explains the evaluation results.

The proposed framework is implemented using JDK 6.0 with virtuoso triple store[7] used to store RDF datasets. All experiments were conducted on a 2.66 GHZ Intel Core i7 processor, 12 GB memory and 1 TB HDD, running Ubuntu operating system.

The framework is proposed to be evaluated based on the following two questions: (i) Does it extract the instance data from LOD cloud datasets and transform it into FVT correctly? (ii) Does the framework help to achieve the intended objective of knowledge discovery using the generated FVT? To answer these questions, Firstly, a content-based similarity measure is designed which estimates the extent to which the instance related data extracted from the LOD cloud using the proposed framework overlaps with the instance data extracted manually following the same process. Secondly, two representative data mining case studies using features extracted from LOD cloud datasets is illustrated with the objective to provide an empirical evaluation of the proposed framework. Sections 4.1 and 4.2 below, respectively present the above two evaluation process and results.

4.1. Content-based Similarity Measure

This method of evaluation performs a comparison of FVT generated through the proposed framework FVT^p with FVT^m generated using two human annotators. The comparison is performed by a content-based similarity measure which is expressed as:

$$ContentSimilarity(FVT^m,\ FVT^p) = 1 - contentDistance(FVT^m,\ FVT^p) \quad (1)$$

where *contentDistance* counts the number of rows by the extent to which it contains the exact match of the contents. It is expressed as:

[7] https://virtuoso.openlinksw.com/

$$ContentDistance(FVT^m,\ FVT^p) = \sum_{i=1}^{m} \frac{1}{m} \times rowDistance(row^m,\ row^p) \quad (2)$$

where *m* is the number of rows of FVT^m, the $rowDistance$ counts the number of cells by the extent to which it contains the exact match of the cell contents. It is expressed as:

$$rowDistance(row^m,\ row^p) = \sum_{i=1}^{n} \frac{1}{n} \times cellDistance(cell^m,\ cell^p) \quad (3)$$

where n is the number of cells in the row. Here, $cellDistance$ is based on the content similarity of individual cells in that particular row, *i.e.*, the content in each cell of the row exactly matches the corresponding cell of the row in the manual FVT. Thus, $cellDistance = 1$, means that the two values are different, whereas a 0 $cellDistance$ value means the two cell values are syntactically equal.

From the equations (1), (2) and (3) it is concluded that: if the value of equation (3) is 0, i.e., the two rows in comparison always contain the same values, thus the value of equation (2) is 0, *i.e.*, FVT^m and FVT^p contain the same content and so the value of equation (1) is equal to 1 stating the two FVT as similar. On the contrary, if the value of (3) is 1, *i.e.*, the two rows of FVT^m and FVT^p contain different values, then the value of (2) is 1 saying, FVT^m and FVT^p contain different content and hence the value of (1) will be 0, *i.e.*, the two FVT's does not match.

To evaluate the proposed framework using the Content-based similarity measure four inter-related & interlinked datasets from the LOD cloud viz., Sider[8] , Diseasome[9] , Dailymed[10] and Drugbank[11] are downloaded as RDF dumps in n-triple format and loaded into a local virtuoso triple store[7]. The SPARQL endpoints of the datasets are configured locally for feature and data extraction. The selection of these datasets for the evaluation process is done as many of the entities in these datasets are connected to the same resources in the interlinked datasets with owl:sameAs[5] predicate link. These links are used to traverse the interlinked dataset to extract additional features. The information contained in the respective datasets along with detailed statistics is shown in Table 1.

In this experiment, FVT for the marketed drugs is built upon the proposed framework and manually by two human annotators using "Gruff" [12] (an RDF dataset visualization tool). The content-based similarity measure index is computed by comparing the two generated FVT's which returns a value of 1, indicating an exact match of the generated FVT's and thus demonstrates its purpose. However, in the evaluation process, a few feature attributes were dropped for inconsistent data values.

[8] https://datahub.io/dataset/bio2rdf-sider
[9] https://datahub.io/dataset/fu-berlin-diseasome
[10] https://datahub.io/dataset/fu-berlin-dailymed
[11] https://datahub.io/dataset/bio2rdf-drugbank
[12] http://franz.com/agraph/gruff/

Table 1. Detailed statistics and information contained in inter-linked datasets

Datasets	Entities	Triples	Links	Distinct Outgoing Links	Distinct Outgoing Dataset	Data Contents
Sider	2674	193249	20294	2	7	information regarding marketed drugs and their side effects
Diseasesome	8152	91182	31750	6	10	collection of disorders and disease genes along with the genetic origin of the disease
Dailymed	3600	164276	39635	3	8	chemical structures of drug compounds, therapeutic purposes, indications, and usages, contradictions, warnings,precautions, adverse reactions, over-dosage etc.
Drugbank	19696	766920	56958	14	18	drugs, their chemical and pharmaceutical datawith sequence, structure and pathway information

4.2. Case Studies

Most of the work discussed in the literature use dataset's SPARQL endpoints available online or implemented in a local environment and thus are not reproducible for the load point and latency of the network. Also, online data extraction from datasets in LOD cloud has other problems like server down for maintenance, bandwidth narrowing, limited access to the dataset at different timeslots etc., which needs to be taken into account to perform a comparison of the proposed framework with other methods. The non-availability of benchmark datasets is another problem in performing evaluation based on the comparison. Thus, in this paper, two representative data mining case studies are discussed with the objective to provide an empirical evaluation of the proposed framework. Sections 4.2.1 and 4.2.2 below discuss the two case studies.

4.2.1. Case Study 1: To Estimate the Effect of Demography on Movie Production

The first case study is concerned with exploiting the potential of enormous data on LOD cloud for knowledge discovery. Looking into the richness and heterogeneity that the datasets provide with interlinks between them four RDF datasets viz., LMDB[4], World-Fact-Book[2], Dbpedia[1], and Geonames[13] are identified.Using film which maps to URI for the film as http://data.linkedmdb.org/resource/film as the seed-entity, features are extracted from the four datasets in a local environment and FVT is build using the program developed on the proposed framework. In the process, the value of N (the depth to be traversed) was set to 2 (two) which fetched attributes like movies, actors, genre, year of production, director, producer, countries, GDP per capita, GDP composition from agriculture, GDP composition from industry, population, median age, government, latitude, longitude, etc. From the generated instances 1000 tuples are selected and filtered

[13]https://datahub.io/dataset/geonames-semantic-web

for interesting features for the experiment. Segmentation of LMDB data for movie production country-wise as per census period (in years) is performed and a total number of movies produced in that particular period is consolidated. The segmented data is sent as input to the Apriori algorithm [29] and it results in two association rules which are proved to be very accurate. The rules are:

- Movie production is high, when the population is above 58147733 and median age more than 38, with a confidence of 1;
- Movie production is low when the population is in the range of 58147733 and 190010647, and median age below 38, with a confidence of 1.

The above results show that the proposed framework is helping us in providing FVT from interlinked datasets in LOD. It helped in revealing interesting patterns which can become a powerful resource for making decisions on capitalizing in movie production. This proves the utility of the proposed framework and the associated algorithms in knowledge discovery from heterogeneous data sources.

4.2.2. Case Study 2: Evaluate Distance Measures to Pre-process Set-Valued Data Attribute in FVT

In the transformation step of FVT generation, discussed in algorithm 3 of section 3.3 it is discussed that the multiple object value belonging to the same attribute are concatenated and stored with a hyphen (–) symbol in-between. These multiple object value for an attribute is called set-valued data [30]. To extract knowledge from FVTs containing set-valued attribute, dimensionality reduction task needs to be performed. The dimensionality reduction task reduces the $n \times n$ feature of the set-valued attribute to $n \times k$ where $k << n$. This dimensionality reduction task involves the use of distance measure in conjunction with the *fastmap algorithm* [31]. Determining the most appropriate distance measure for the purpose above is again a problem and requires an evaluation of the available distance measures [32].

In the study, eight distance measures (D_1, D_2 ... D_8) viz., Set Overlap, Tanimoto [33], Average Linking [34], RIBL [35], Single Linkage [36], Complete Linkage [37], Sum of Minimum Distances and Hausdorff distance measure[14] are evaluated for its use in dimensionality reduction task. For the evaluation process RDF dump of the pre-classified LMDB dataset, stored and configured in a local environment, is used to extract movie-related features and build FVT using the proposed algorithms. In the FVT, 1350 movie instance are selected for the experiment. The FVT contained attributes viz., *movie_id, movie_release_date, country, cinematographer, director, writer, actors, producer, movie_length, genre, language, movie_rating*, etc. of which the attributes *cinematographer, director, writer, actors, produce*r, and *genre* are set-valued features. The pre-classified attribute *movie_rating* is removed from the FVT. The distance measures D_i in conjunction with the *fastmap method* [31] is used to replace $n \times n$ set-valued attributes to their respective reduced $n \times k$ attributes in the FVT. The resulting feature vector table (FVT_i) is clustered for *moving_rating* followed by calculating micro-precision value (PV_i) in matching the clusters with the pre-classified class. The process is repeated with the remaining distance measures ($D_j \ldots D_8$) thus, generating micro-precision values ($PV_j \ldots PV_8$). The distance measure

[14] https://en.wikipedia.org/wiki/Hausdorff_distance

resulting in highest micro-precision value is designated as the most appropriate one. The micro-precision values generated using different distance measures in conjunction with the *fastmap method* in this experiment is illustrated in Table 2.

Table 2. Micro-precision values using different distance measures

Distance Measures	Average Linkage	Complete Linkage	Single Linkage	Haus-dorff	RIBL	Tani-mato	Set Overlap	Sum of Minimum distances
Micro Precision	0.49666	0.25	0.21	0.5933	0.5433	0.42	0.48	0.43

In the evaluation process, it is seen that *Hausdorff* distance measure score the highest micro-precision value and thus is most appropriate to be used in conjunction with the *fastmap method* to reduce the dimension of set-valued attributes in FVT. However, from the evaluation point of view, the significance of this study is the generation of FVT from the LOD dataset using the proposed framework and its use in classical learning methods.

5. Conclusion

In this paper, a framework and the required algorithms have been proposed to extract data mining features from a set of interlinked RDF datasets from the LOD cloud in a local environment. The features are used to build an FVT that helps in the extraction of knowledge from the heterogeneous data sources. The steps performed in the framework are seed URI matching, dynamic SPARQL query generation, feature extraction & filtering, transformation and FVT generation. The framework filters meta-data and schema related predicates and thus also help in the pre-processing step of data mining. The framework can be tuned to extract features up to the required depth. In the experiments, it is found that the dimensionality of the features raises exponentially with the increase in depth of traversal which leads to an increase in the missing attribute values in the FVT. Thus, a threshold depth level is required to be set depending upon the learning algorithm and problem of study in hand. The features extracted from cross-domain datasets are noisy, inconsistent and/or are less significant thus it requires specific processing. The evaluation works discussed in this study demonstrates the utility of the proposed framework and the associated algorithms in generating FVT from heterogeneous data sources for the data mining process. It presents a significant contribution by introducing a content-based similarity metric for tabular data comparison. In regard to the utility of the FVT built using the proposed framework in data mining applications, one of the evaluation work helped in revealing some interesting patterns which can become a powerful resource for making decisions on capitalizing in movie production. The other evaluation work suggests an appropriate distance measure to handle dimensionality of set-valued data attribute in the generated FVT.

In the future, it is proposed to extend this framework to include on the fly feature selection method in order to have only significant features in the resulting FVT. The future work can also be explored to look for same entity link in non-interlinked datasets thus help in discovering new dimensions in knowledge.

References

[1] C. BIZER, T. HEATH, and Berners-LEE, *Linked Data - The Story So Far*, Int. J. Semant. Web Inf. Syst. **5**(3), pp. 1–22, 2009.

[2] M. SCHMACHTENBERG, C. BIZER, and H. PAULHEIM, *Adoption of the Linked Data Best Practices in Different Topical Domains*, in Proceedings of International Semantic Web Conference, 2014.

[3] L. M. CAMPBELL and S. MACNEILL, *The Semantic Web, Linked and Open Data-A briefing paper,* in JISC CETIS-The Center for Educational Technology and Interoperability Standards (JISCCETIS), Bolton, UK, 2010.

[4] M.-A. KROGEL, S. RAWLES, Z. FILIP, P. A. FLACH, N. LAVRĂ, and S. WROBEL, *Comparative Evaluation of Approaches to Propositionalization*, in Proceedings of the 13th International Conference on Inductive Logic Programming (ILP 2003), LNCS **2835**, pp. 197–214, 2003.

[5] N. LAVRAC and P. A. FLACH, *An extended transformation approach to Inductive Logic Programming*, ACM Trans. Comput. Log. **2**(4), pp. 458–494, 2001.

[6] C. NATTEE, S. SINTHUPINYO, and M. NUMAO, *Inductive Logic Programming for Structure-Activity Relationship*, in The 2005 Symposium on Applications and the Internet Workshops (SAINT-W05), pp. 332–335, 2005.

[7] A. INOKUCHI and T. WASHIO, *Complete Mining of Frequent Patterns from Graphs*, Mach. Learn. **50**(3), pp. 321–354, 2002.

[8] E. FRANK, M. HALL, G. HOLMES, R. KIRKBY, B. PFAHRINGER, and I. H. WITTEN, *WEKA - A Machine Learning Workbench for Data Mining*, in Data Mining and Knowledge Discovery Handbook, O. Maimon and L. Rokach, Eds. Springer, pp. 1269–1277, 2009.

[9] G. KLYNE and J. J. CARROLL, *Resource Description Framework (RDF): Concepts and Abstract Syntax*, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210.

[10] D. BRICKLEY, R.V. GUHA, *RDF Schema 1.1 W3C Recommendation*, Retrieved June 11, 2018, from http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[11] E. PRUD'HOMMEAUX, A. SEABORNE, *SPARQL query language for RDF. W3C Recommendation*, Retrieved June 11, 2018, from http://www.w3.org/TR/rdf-sparql-query/.

[12] G. A. GRIMNES, P. EDWARDS, and A. PREECE, *Instance based Clustering of Semantic Web Resources*, in Proceedings of the ESWC-2008, S. Bechhofer, Ed. Springer, pp. 303–317, 2008.

[13] P. STICLER, *CBD Concise Bounded Description*, online, 2005.

[14] V. N. P. KAPPARA, I. RYUTARO, and O. P. VYAS, *LiDDM: A Data Mining System for Linked Data*, in Workshop on Linked Data on the Web (LDOW2011), Hyderabad, India, 2011.

[15] M. A. KHAN, G. A. GRIMNES, and A. DENGEL, *Two pre-processing operators for improved learning from Semantic Web data*, in First Rapid Miner Community Meeting and Conference (RCOMM-2010), Dortmund, Germany, 2010.

[16] W. CHENG, G. KASNECI, T. GRAEPEL, D. STERN, and R. HERBRICH, *Automated Feature Generation from Structured Knowledge*, in 20th ACM Conference on Information and Knowledge Management (CIKM 2011), Glasgow, United Kingdom, 2011.

[17] H. PAULHEIM and J. FURNKRANZ, *Unsupervised Generation of Data Mining Features from Linked Open Data*, in International Conference on Web Intelligence, Mining, and Semantics (WIMS'12), Craiova, Romania, 2012.

[18] N. FANIZZI, C. D'AMATO, and F. ESPOSITO, *Mining Linked Open Data through Semi-supervised Learning Methods based on Self-training*, in Sixth International Conference on Semantic Computing (ICSC), 2012.

[19] C. TSOUKALAS, D. DERVOS, J. MARTINEZ-GIL, and J. F. ALDANA-MONTES, *TheMa: An API for Mining Linked Datasets*, in 16th Panhellenic Conference on Informatics (PCI), pp. 449–453, 2012.

[20] J. MYNARZ and V. SVATEK, *Towards a Benchmark for LOD-Enhanced Knowledge Discovery from Structured Data*, in Second International Workshop on Knowledge Discovery and Data Mining Meets Linked Open Data, pp. 41–48, 2013.

[21] E. L. MENCIA, S. HOLTHAUSEN, A. SCHULZ, and F. JANSSEN, *Using Data Mining on Linked Open Data for Analyzing E-Procurement Information*, in 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 (ECMLPKDD), 2013.

[22] M. D'AQUIN and N. JAY, *Interpreting Data Mining Results with Linked Data for Learning Analytics: Motivation, Case Study and Directions*, in Third International Conference on Learning Analytics and Knowledge, 2013.

[23] A. NOLLE and G. NEMIROVSKI, *An Approach for Accessing Linked Open Data for Data Mining Purposes*, in RapidMiner Community Meeting and Conference (RCOMM 2013), 2013.

[24] H. PAULHEIM, P. RISTOSKI, E. MITICHKIN, and C. BIZER, *Data Mining with Background Knowledge from the Web*, in RapidMiner World, Boston, USA, 2014.

[25] P. RISTOSKI, C. BIZER, and H. PAULHEIM, *Mining the Web of Linked Data with RapidMiner*, Web Semant. Sci. Serv. Agents World Wide Web **35**, pp. 142–151, 2015.

[26] R. MAHULE and O. P. VYAS, *Leveraging linked open data information extraction for data mining applications*, in Turkish Journal of Electrical Engineering and Computer Sciences, **24**(6), pp. 4874–4884, 2016.

[27] O. HARTIG, C. BIZER, and J. FREYTAG, *Executing SPARQL Queries over the Web of Linked Data*, in The International Semantic Web Conference, pp. 293–309, 2009.

[28] V. HAARSLEV and R. MOLLER, *Racer: A Core Inference Engine for the Semantic Web*, in 2nd International Workshop on Evaluation of Ontology-based Tools (EON 2003), Sanibel Island, Flourida, 2003.

[29] R. AGRAWAL and R. SRIKANT, *Fast Algorithms for Mining Association Rules*, in 20th International Conference on Very Large Databases, pp. 487–499, 1994.

[30] M. A. KHAN, *Handling set-valued features when learning from Semantic Web data*, in Dissertation, German Research Center for Artificial Intelligence Knowledge Management Group, Technical University of Kaiserslautern, Germany, 2010.

[31] C. FALOUTSOS and K. LIN, *FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets*, in ACM SIGMOD Conference, pp. 163–174, 1995.

[32] R. MAHULE, A. GARG, N. KUMAR, and O. P. VYAS, *Performance evaluation of distance measures for Preprocessing of set-valued data in feature vector generated from LOD datasets*, in 2015 IEEE UP Section Conference on Electrical Computer and Electronics, UPCON 2015, 2016.

[33] T. T. TANIMATO, *Tanimato Distance Measure*. IBM Internal Report 1957, 1957.

[34] A. KALOUSIS, A. WOZNICA, and M. HILARIO, *A unifying framework for relational distance-based learning founded on relational algebra*. University of Geneva, Switzerland, 2005.

[35] S. DZEROSKI, *Multi-Relational Data Mining: An Introduction*, ACM SIGKDD Explor. **5**(1), pp. 1–16, 2003.

[36] K. FLOREK, J. LUKASZIWICZ, J. PERKAL, H. STEINHAUS, and S. ZUBRZYCKI, *Sur la liaison et la division des points dun ensemble fini.pdf*, Colloq. Math. **2**(3-4), pp. 282–285, 1951.

[37] R. R. SOKAL and C. D. MICHENER, *A Statistical Method for Evaluating Systematic Relationships*, Univ. Kansas Sci. Bull. **38**, pp. 1409–1438, 1958.