I.J. Information Technology and Computer Science, **2018, 10, 62-68**
Published Online October 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2018.10.08

RDF Link Generation by Exploring Related Links on the Web of Data

Kumar Sharma
Department of Computer Science and Engineering, University of Kalyani, Kalyani, West Bengal, India.
E-mail: kumar.asom@gmail.com

Ujjal Marjit
Center for Information Resource Management (CIRM), University of Kalyani, Kalyani, West Bengal, India.
E-mail: marjitujjal@gmail.com

Utpal Biswas
Department of Computer Science and Engineering, University of Kalyani, Kalyani, West Bengal, India.
E-mail: utpal01in@yahoo.com

Received: 06 July 2018; Accepted: 12 August 2018; Published: 08 October 2018

***Abstract*—**Interlinking RDF resources is a vital aspect of the Semantic Web technology. It is the basis of Linked Data that provides interlinked datasets on the web. One of the principles of Linked Data is interlinking resources from different data sources on the web. Data interlinking is a critical and challenging problem that every Linked Data generation applications face. Various approaches have been evolved for resolving this problem, but, for more massive datasets, it becomes almost indefinite time while linking similar or related resources. Linking RDF resources is like the problem of entity matching, record matching or duplicate resource detection. More or less they attempt to point to the same problem, but the RDF link generation is the task of finding related resources on the web. In this article, we present an approach for generating RDF links using the similarity measure between two RDF resources and by exploring associated relationships of the matched resources. The idea is to find related resources and link them with an RDF resource that is being generated.

***Index Terms*—**Linked data, semantic web, link discovery, rdf, interlinking.

I. Introduction

Interlinking of RDF resources is the most common problem in a Linked Data generation activity [1]. It takes more time than needed and consumes more network resources when the dataset is too large. Before publishing RDF dataset based on Linked Data, it is relevant to identify related resources on the web and ingest links to those resources in the RDF dataset. In literature, this problem is sincerely attended by many researchers, and mostly they tried to find the sameness nature of the resources. However, if we recall the fourth principle of Linked Data [2] where it says, "*include links to other URIs, so that they can discover more things*" which implies that we can ingest not only similar resources but also the related resources that can entail more information. That is why in literature we find three kinds of RDF links: *relationship link*, *identity link*, and *vocabulary link*. The problem of interlinking resources is also known as co-reference resolution, identity uncertainty, record matching, instance identification [3], duplicate resource detection [4], entity co-reference [5], and record linkage [6]. The record linkage matches two records situated at two different datasets based on specific criteria. This is primarily used to establish the relationship between multiple RDF datasets. Such type of work is found in [7].

If we carefully look at the problem of identity matching of two resources, it reveals the idea of matching behaviors of the two resources. Like this RDF resources can have many behaviors that are described by their property-value pairs. RDF resources are instances of some classes in a domain ontology having *n* number of property-value pairs. From this, we derive that matching two resources require matching their domain ontologies and property-value pairs. If all these matches then we can say that two resources are identical or similar, this is also called the task of instance matching. However, for the relationship linking these requirements are not needed. Two resources can relate to each other even if they belong to different ontologies and share different behaviors. RDF is a flexible data model where resources can add new property-value pairs that belong to different vocabulary and ontology. Hence, such resources can break down the conditions of the instance-matching problem. But they can be matched using predefined rules with minimal behavioral matching. Identity and relationship linking issues are related, but they vary in similarity measure.

In this article, we present the problem of relationship linking. The source resources are matched with the target resources from the selected datasets. If the resources are matched using some property values, then they are

interlinked with the target resources as well as the linked datasets of the target resources. This activity is executed in a distributed environment during the generation of RDF resources from legacy datasets. This way, the task of link generation is achieved without involving large number pairwise resource comparisons, which helps in converting a considerable volume of legacy datasets into RDF while supporting information linking. The remaining sections of the article are organized as follows: in section 2 we discuss the related works, section 3 presents the research objective, and section 4 presents the proposed approach for link generation. Section 5 shows the experimental results as well as the performance analysis, and finally, section 6 concludes the article.

II. Related Works

RDF link generation is the process of determining resources having similar identities. This problem is also known as co-reference resolution, identity uncertainty, record matching, and instance identification [4]. It has been vastly researched in the field of relational databases and XML [8, 9, 10, 11].

In literature, there exist several efforts related to the task of link discovery. Most of them focus on finding semantic relation by exploiting the resources on the web of data. Silk [8], is a standard link discovery framework which discovers semantically related resources located at different linked data sources. It uses linkage rules and conditions to match the resources. Each resource must fulfill the linkage rules to develop the semantic relations between them. The resources are fetched using the SPARQL protocol. Rizopoulos propose an approach for the automatic discovery of semantic relation [12]. Two resources are compared using a bidirectional path. Each pair of the resource is compared two times dealing with the bidirectional similarity degrees. In [13], the authors provide an option for interlinking resources using the manual approach. With this approach, the entities from real-world datasets have to be entered by the user.

There are many other approaches which generally compare RDF resources with one another and finds the semantic relation by determining the similarity measure of the resources. With these approaches, it will take a considerable amount of time when the dataset is too large. Massive datasets require the scalable approach to avoid the time-consuming processing of data. Also, if we consider the link generation task during the initial stage of the Linked Data generation from the legacy data sources, then it will save time to process the huge datasets. Currently, there are minimal scalable approaches for the Linked Data generation from legacy data sources. Lim et al. [14] present a distributed method for converting data from the relational database to RDF using Hadoop and MapReduce framework. Their approach for converting legacy data to RDF is scalable, but they do not mention about the task for link generation. They use Hadoop and MapReduce framework with nine worker nodes, each node configured with Intel i5 Quad-Core 3.1GHz, 4GB RAM, and 8TB HDD. The input data is stored in MySQL database from DBT2 benchmark data.

Similarly, Vahdati et al. [15] present a distributed approach for converting research metadata from HBase, CSV and XML formats to RDF. They use the MapReduce paradigm for processing a large volume of the data in parallel over multiple nodes. The evaluation step shows that they use 12 worker nodes for data conversion from HBase to RDF, which took 20 billion HBase rows as input and produced 655 billion RDF triples in 17 minutes. They have considered the task for link generation for future work.

We propose a distributed solution for generating RDF data from legacy data sources in [16]. We plan to improve the process of link generation; however, the implementation has not yet finished comparing with other scalable approaches. Our solution for link generation does not require the pairwise resource comparison with every resource of the target dataset. It first performs the pairwise resource comparison with the limited resources. If the resources are matched, then it navigates to the target resources. When it navigates to the target resource, it extracts the other linked resources which are already generated by some other link generation framework. This overall process is performed on a distributed environment using Apache Spark.

III. Research Objective

The prime objective of this article is to discover semantic links for the RDF resources. The task of link generation is performed on a bibliographic dataset which is converted from MARC 21 Formats into RDF. The specific objectives are:

1. To develop an algorithm that can query over the web of data for a particular resource.
2. To develop an algorithm that can calculate the similarity measure between two RDF resources.
3. To discover the semantically related resources and ingest those links into the RDF resource.

In this article, we demonstrate the link generation task using bibliographic datasets. The bibliographic resources are generated from a data conversion framework [16], which converts MARC 21 Format for Bibliographic data into RDF format in a distributed environment. Here, we attempt to show the result of interlinking bibliographic entities with other entities stored into different datasets on the web and fulfill the requirement of Linked Data.

IV. RDF Link Generation

The solution for relationship linking starts with the count of minimal behavior parameter and matching properties. Considering the source and target datasets as *DS* and *DT* respectively, for a resource *r* in *DS*, query *DT* for *r*, and if any resource is found then compare *r* with matched resources in *DT*. The similarity measure at the

RDF statement level is used to analyze the resources. RDF statement level similarity measure ensures that the property of the resources in the target dataset agrees on the matching rule and the value is matched based on the value similarity measure. For each matched resource, explore the resource on the web, analyze the properties and look for the linked resources.

A. Target Dataset Selection

In this step, we search the target-dataset to find out the information related to the resource whose interlinking is to be done. For a given resource in a source dataset, the target dataset is queried via SPARQL query end-point. The target datasets have to be selected based on the nature of the data presented in the source dataset. For example, for a bibliographic data, the target datasets could be freebase[1], yago[2], DBpedia[3] etc. In each target dataset, we search for a given resource using resource's searchable properties. The matched resources are collected, and then those resources are again analyzed using similarity measure with the minimal behavioral match. The similarity measure gives the optimal result which can be determined whether a resource is related or not.

Algorithm 1: GenerateRDFLinks (Resource Rr): Given an RDF resource, returns the interlinked resource Rr'

```
Input: An RDF resource (rr)
Output: Modified resource (rr')
T ← a set of target datasets;
K ← ε; //visit limit
linkCounter ← 0;
for each t ∈ T do
    D_URI ←URILookUp(rr.getSearchTerms(), t.getAPIPath());
    for each r_URI in D_URI do
        tr ← getTargetResource(r_URI);
        sim ← sim_measure(tr, rr);
        match ← false;
        if sim == 1 then
            rr' ← rr.addPredicate(owl:sameAs, r_URI);
            linkCounter++;
            match ← true;
        else if sim >= θ then
            rr' ← rr.addPredicate(rdfs:seeAlso, r_URI);
            linkCounter++;
            match ← true;
        end if;
        if match == true
    rr'←rr.addPredicate(rdfs:seeAlso, Navigate (r_URI));
        end if
    if linkCounter >= K then;
        break;
    end if
    end for
end for
return rr';
```

The final matched resources are explored looking for the interlinked resources. We extract those resources that belong to a particular group of interlinking. Such a group is predefined which contains the properties such as *owl:sameAs*, *rdfs:seeAlso*, etc. We further navigate that resource and again extract the interlinked resources. Algorithm 1 & 2 demonstrate this process.

Algorithm 2: Navigate (Resource URI): Given an RDF resource URI, returns the linked resources

```
Input: An RDF resource URI (R_URI)
Output: List of linked resource URIs
P ← a set of linked properties;
K ← ε; //visit limit
linkCounter ← 0;
   R ← en empty list;
for each p ∈ P do
    R_URI ← SparqlQuery(R_URI, p.getURI(),?link);
       R.add(R_URI);
end for
return R;
```

Algorithm 1 takes an RDF resource as input and returns the resource by interlinking with the matched resources. It requires a set of target datasets to which the interlinking has to be made. At this moment, the datasets are manually searched and configured. Algorithm 1 visits the target datasets and fetches the resource URIs. With each resource in the visited dataset, the RDF resource is downloaded and matched with the given resource using resource similarity measure. The value of the similarity measure determines the match. For this, a threshold value is defined to determine whether the similarity measure is equal to or greater than the specified threshold, then the target resource is included under linkset. Further, the matched resource is navigated to find out the linked resources. Algorithm 2 shows this process. The linked resources are also included in the linkset.

B. Comparing RDF Resources

Since an RDF resource consists of a set of RDF triples, comparing two RDF resources is not a straightforward job. An RDF triple, in turn, consists of 3-components: Subject, Predicate, and Object. We consider the comparison process at three different levels: comparison at the resource level, statement level, and object (value) level. It also requires comparing each & every triple of the resources. Matching two resources requires a similarity metric to compare pairs of resources, RDF statement and their corresponding values (objects). We describe similarity metrics built for RDF data. In the following, we define the similarity function for comparing string data, statements, and resources.

C. String Similarity Measure

For matching string data, the cosine similarity metric is used. The similarity between two texts or strings is defined as the cosine angle between vector representations of the two strings. It is computed as follows:

$$Sim\_Str(A,B) = Cos(\theta) = \frac{A * B}{|A||B|} \qquad (1)$$

Where, Cos(θ) is the cosine similarity of two strings A and B. The cosine similarity metric has been used

[1] https://developers.google.com/freebase/
[2] https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/#c10444
[3] http://wiki.dbpedia.org/lookup/

because of its simplicity, efficient to evaluate, and always gives the value between 0 and 1. It is worth mentioning here that if the value belongs to number then no doubt they are easily matched, and if the value is date & time then their differences are calculated.

D. Similarity measure for RDF Statement

In general, two RDF statements are equal if their property-value pairs are similar. RDF statement measure is a measure of similarity between two RDF statements considering the similarity pairs of predicate and object. It is defined as follows.

Definition 1. *Given two RDF statements S1 and S2, a similarity function sim_stmt, and a similarity threshold* th_{stmt}*, then S1 and S2 are similar if sim_stmt(S1, S2)=* th_{stmt}.

For two RDF statements S1 and S2, the RDF statement similarity measure is calculated as:

$$Sim\_Stmt(S_1, S_2) = Eq(S_{1P}, S_{2P}) * Sim\_Obj(S_{1Obj}, S_{2Obj}) \quad (2)$$

Where Eq is a function telling whether two properties are the same or not. Sim_Obj is another similarity function which determines the similarity measure between two objects with the help of string similarity function.

E. Similarity measure for RDF Resources

As discussed, two RDF resources need comparison at the object level. RDF resource similarity is calculated using Jaccard similarity measure, which measures the similarity between finite sample sets and is defined as the cardinality of the intersection of sets divided by the cardinality of the union of the sample sets.

Definition 2. *Given two RDF resources R1 and R2, a similarity function sim_res, and a similarity threshold* th_{res}*, then these two resources are said to be similar if sim_res(R1, R2)=* th_{res}*. For this, two resources should have the same number of property-value pairs.*

Given two RDF resources R1 and R2, the resource similarity measure is calculated as:

$$Sim\_Res\left(R_1, R_2\right) = \frac{R_1 \cap R_2}{R_1 \cup R_2} \quad (3)$$

Where R1 and R2 contain finite sets of RDF statements.

V. Experiment

The experiment is performed on the bibliographic datasets that are being converted from MARC 21 Format for Bibliographic data into RDF. Java 1.8 and Apache Spark 2.2.0 have been used for converting data on Mac OS X 10.13.4 system having 16GB of RAM and Core i7 processor. First of all, we show the performance report of the proposed similarity measure. Then we provide the experimental result of link generation task which is performed using an RDF resource taken from a sample dataset[4]. This dataset is collected using the conversion framework as presented in [16]. Finally, we present the time analysis of the proposed approach by performing link generation task with larger datasets in a distributed environment.

A. Performance on Similarity Measure

The performance of the similarity measure is analyzed using the SPIMBENCH[5] training datasets. The goal of SPIMBENCH is to determine the similarity measure between two OWL instances. SPIMBENCH provides two sets of training datasets- one set contains ontology and instances (Tbox), and another set includes only instances (Abox). We have used only Abox datasets where instances are described using 22 classes, 31 DatatypeProperty, and 85 ObjectType properties. The task is to match the instances in the source dataset (Abox1) against the instances of the target dataset (Abox2). We thus tested the small set of training datasets[6] and produced a set of mappings (alignment) between the pairs of matching instances as shown in Table 1. Here we have shown only some part of the mappings. It shows that the adjustments are not yet accurate as expected. This is because some of the property-value pairs are slightly different as our approach matches for same property-value pairs using string similarity measure.

B. Querying Resources

The resources are searched using direct URI fetch on the target database. URI search is achieved by DBpedia lookup service. DBpedia look service provides DBpedia resource URIs by related keywords. However, this relation does not mean that the matched resources are semantically related all the time. It could only the label of the resource matches. That is why the resources have to be further measured using similarity values between the source resource and the target resource. The DBpedia lookup service provides two APIs: Keyword Search and Prefix Search. Here we follow keyword search which uses the given string to find the related resources. For example, the URL, as shown in Listing 1, fetches the associated resources for the string "Art for all."

[4]https://github.com/kumarsharma/LegacyData2LinkedData/blob/master/RdfXmldata/rdf_bibo_1.rdf
[5]https://project-hobbit.eu/challenges/om2018/om2018-tasks/
[6]http://users.ics.forth.gr/~jsaveta/.index.php?dir=OAEI_IM_SPIMBENCH_2017_5

Table 1. Similarity measure between two datasets

Entity 1	Entity 2	Expected Alignment	Achieved Alignment
http://www.bbc.co.uk/things/1#id	http://www.bbc.co.uk/things/1#id	1.0	0.64
http://www.bbc.co.uk/things/2#id	http://www.bbc.co.uk/things/1496558551#id	1.0	0.65
http://www.bbc.co.uk/things/6#id	http://www.bbc.co.uk/things/822880662#id	1.0	0.68
http://www.bbc.co.uk/things/7#id	http://www.bbc.co.uk/things/10992588#id	1.0	0.83
http://www.bbc.co.uk/things/13#id	http://www.bbc.co.uk/things/13#id	1.0	0.73
http://www.bbc.co.uk/things/328#id	http://www.bbc.co.uk/things/328#id	1.0	0.82
http://www.bbc.co.uk/things/202#id	http://www.bbc.co.uk/things/815017456#id	1.0	0.89
http://www.bbc.co.uk/things/127#id	http://www.bbc.co.uk/things/553524636#id	1.0	0.89
http://www.bbc.co.uk/things/200#id	http://www.bbc.co.uk/things/2109779712#id	1.0	0.90
http://www.bbc.co.uk/things/158#id	http://www.bbc.co.uk/things/1151399101#id	1.0	0.90
http://www.bbc.co.uk/things/74#id	http://www.bbc.co.uk/things/1636544670#id	1.0	0.70
http://www.bbc.co.uk/things/36#id	http://www.bbc.co.uk/things/1949877939#id	1.0	0.90

Listing 1. URL to fetch the related resource from DBpedia

```
http://lookup.dbpedia.org/api/search.asmx/KeywordSearch?QueryClass=&QueryString=Art_for_all
```

The result is processed and parsed to get the related resource URIs. For each matched resource URI, the RDF resource is fetched, and they are compared using the similarity measure function. The SPARQL query to fetch the resource using its URI is given in Listing 2.

Listing 2. SPARQL query to fetch an RDF resource

```
SELECT ?predicate ?object
WHERE{
<resource_URI> ?predicate ?object
}
```

The above query fetches the corresponding resource in RDF/XML format.

Listing 3. An RDF Resource

```
- <rdf:Description rdf:about="http://localhost:8080/BibliographicLinkedData/BibResources/Art_for_all.">
    <rdagroup1elements:publishersName>F.A. Waugh</rdagroup1elements:publishersName>
    <foaf:birthday>1869-1943.</foaf:birthday>
    <rdagroup1elements:placeOfPublication>Amherst Mass.</rdagroup1elements:placeOfPublication>
    <marcont:dateTimeOfLatestTransaction>20020606090541.3</marcont:dateTimeOfLatestTransaction>
    <foaf:name>(Frank Albert)</foaf:name>
  - <marcont:hasNote>
      Reprinted from: School and society V. 15 no. 382 April 22 1922.
    </marcont:hasNote>
    <marcont:hasCoverage>Study and teaching</marcont:hasCoverage>
    <DC:extent>10 p.</DC:extent>
    <VCARD:Region>United States.</VCARD:Region>
    <marcont:hasRecordLength>595</marcont:hasRecordLength>
    <DC:type>Language material</DC:type>
    <rdagroup1elements:dateOfPublication>1922</rdagroup1elements:dateOfPublication>
    <marcont:systemControlNumber>ocm78722478</marcont:systemControlNumber>
    <marcont:hasEncodingScheme>UCS/Unicode</marcont:hasEncodingScheme>
    <VCARD:Locality>Landscape</VCARD:Locality>
    <VCARD:Locality>Nature (Aesthetics)</VCARD:Locality>
    <rdagroup1elements:keyTitle>Art for all.</rdagroup1elements:keyTitle>
    <foaf:name>Waugh F. A.</foaf:name>
    <marcont:hasRecordStatus>New</marcont:hasRecordStatus>
    <marcont:hasControlNumber>000000360-3</marcont:hasControlNumber>
    <rdagroup1elements:dimensions>23 cm.</rdagroup1elements:dimensions>
</rdf:Description>
```

As shown in Listing 3, the resource's key-title is "Art for all." Hence the search string will be "Art for all." We perform related resource finding using DBpedia's URI lookup operation. Here the minimum behavior match is "*keyTitle*" and its value. The actual URL to call the API is shown in Listing 4.

Listing 4. URL to request related resources

```
http://lookup.dbpedia.org/api/search.asmx/KeywordSearch?QueryClass=&QueryString=Art%20for%20all
```

The result is processed and parsed to get the related resource URIs. Now, for each related resource URI, the actual RDF resource is fetched and listed under target resource list. For each target resource, the given resource is compared using the similarity measure function. The resultant resource with linked properties (*rdfs:seeAlso*) is shown in Listing 4, and the full listing is available here[7]. In the resultant linked dataset, it is to be noted that we have linked to other resources from multiple sources. Initially, we queried into DBpedia dataset and after that using DBpedia resource the three different datasets such as Yago, Freebase and Wikidata are visited and fetched the linked resources.

Listing 5. RDF Resource with linked properties

```
- <rdf:Description rdf:about="http://localhost:8080/BibliographicLinkedData/BibResources/Art_for_all.">
    <rdagroup1elements:publishersName>F.A. Waugh</rdagroup1elements:publishersName>
    <foaf:birthday>1869-1943.</foaf:birthday>
    <rdfs:seeAlso rdf:resource="http://wikidata.dbpedia.org/resource/Q4127439"/>
    <marcont:dateTimeOfLatestTransaction>20020606090541.3</marcont:dateTimeOfLatestTransaction>
    <rdagroup1elements:placeOfPublication>Amherst Mass.</rdagroup1elements:placeOfPublication>
    <foaf:name>(Frank Albert)</foaf:name>
  - <marcont:hasNote>
      Reprinted from: School and society V. 15 no. 382 April 22 1922.
    </marcont:hasNote>
    <DC:extent>10 p.</DC:extent>
    <marcont:hasCoverage>Study and teaching</marcont:hasCoverage>
    <rdfs:seeAlso rdf:resource="http://de.dbpedia.org/resource/Allrussische_Industrie-_und_Handwerksausstellung_1896"/>
    <rdfs:seeAlso rdf:resource="http://dbpedia.org/resource/Art_for_All_Foundation"/>
    <VCARD:Region>United States.</VCARD:Region>
    <marcont:hasRecordLength>595</marcont:hasRecordLength>
    <DC:type>Language material</DC:type>
    <marcont:hasEncodingScheme>UCS/Unicode</marcont:hasEncodingScheme>
    <marcont:systemControlNumber>ocm78722478</marcont:systemControlNumber>
    <rdagroup1elements:dateOfPublication>1922</rdagroup1elements:dateOfPublication>
    <VCARD:Locality>Landscape</VCARD:Locality>
    <VCARD:Locality>Nature (Aesthetics)</VCARD:Locality>
    <rdagroup1elements:keyTitle>Art for all.</rdagroup1elements:keyTitle>
    <foaf:name>Waugh F. A.</foaf:name>
    <rdfs:seeAlso rdf:resource="http://yago-knowledge.org/resource/All-Russia_Exhibition_1896"/>
    <marcont:hasRecordStatus>New</marcont:hasRecordStatus>
    <rdfs:seeAlso rdf:resource="http://rdf.freebase.com/ns/m.0gk_fs"/>
    <rdfs:seeAlso rdf:resource="http://www.wikidata.org/entity/Q4127439"/>
    <marcont:hasControlNumber>000000360-3</marcont:hasControlNumber>
    <rdagroup1elements:dimensions>23 cm.</rdagroup1elements:dimensions>
</rdf:Description>
```

[7] https://raw.githubusercontent.com/kumarsharma/LegacyData2LinkedData/master/RdfXmlLinkedData/rdf_bibo_1_linked.rdf

C. Performance

The performance of the proposed approach is tested on the distributed environment having three distributed nodes on Mac OS X environment. For evaluation, we use Harvard Library Bibliographic Datasets from Harvard Library. Table 2 shows the input & output dataset size and the total time to convert the input dataset into RDF using the proposed distributed approach. This conversion is without link generation. It shows that the data conversion without link generation is faster as compared to a non-distributed approach as shown in Table 3.

Table 2. Size of input data, output data and total time taken for distributed data conversion

Dataset	Input Size (GB)	Output Size (GB)	Triples Count (Billion)	Time Taken (Min)
Dataset 1	1.16	0.29	27.42	1.13
Dataset 2	1.02	0.34	22.16	1.09
Dataset 3	0.98	0.33	20.36	1.03
Dataset 4	0.917	0.32	21.46	1.08
Dataset 5	0.916	0.34	21.95	1.04

Table 3. Size of input data, output data and total time taken for non-distributed data conversion

MARC 21 Dataset	Input Size (GB)	Output Size (GB)	Triples Count (Billion)	Time Taken (Min)
Dataset 1	1.16	3.33	27.42	6.41
Dataset 2	1.02	3.40	22.16	6.10
Dataset 3	0.98	3.19	20.36	5.58
Dataset 4	0.917	3.30	21.46	5.61

Table 4 shows the result of comparison with the other two distributed approaches for converting legacy data to RDF. We have used up to three number of distributed nodes, but with an increased number of nodes the performance will always be improved.

Table 4. Comparison result

Approach	No. of Nodes	No. of Triples (Billion)	Time Taken (Min)
Lim et al. [14]	9	116	3.1
Vahdati et al. [15]	12	655	17
Sharma et al. [16]	3	174	11

Notice that the average time for generating RDF triples without link generation is always faster on a distributed approach. But whenever the task of link generation is involved the performance is degraded even with the distributed approach. This is mainly because of network operations. As shown in Table 5, the time taken to perform link generation on the above datasets using the proposed approach is more than 15 hours using three distributed nodes. Hence it is observed that the task of link generation is a time-consuming job because of network fetch operations.

Table 5. Performance analysis on link generation

Dataset	Triples Count (Billion)	Links Count (Billion)	Time Taken (hour)
Dataset 1	27.42	1.33	18.80
Dataset 2	22.16	1.07	22.40
Dataset 3	20.36	0.86	16.15
Dataset 4	21.46	1.60	16.22
Dataset 5	21.95	1.11	12.90

VI. Conclusion

In this article, we present a link generation framework for generating related links of RDF resources in a distributed environment. The proposed approach uses the idea of exploring related resources and extracting the linked resources on the web. The related resource URIs are queried using DBpedia APIs, and the corresponding RDF resources are extracted using SPARQL queries. Furthermore, we proposed an algorithm for comparing RDF resources using resource similarity measure. The proposed approach for link generation is useful when RDF resources required to be linked with related resources that belong to other data sources on the web. The proposed work needs further improvement on the performance when larger datasets are used as well as the comparative analysis with different approaches which will be the future scope of the work.

References

[1] S. Dietze, H. Q. Yu, D. Giordano, E. Kaldoudi, N. Dovrolis, and D. Taibi, "Linked Education: Interlinking Educational Resources and the Web of Data," *In Proceedings of the 27th Annual ACM Symposium on Applied Computing*, ACM, 2012, pp. 366-371.

[2] C. Bizer, T. Heath, and T. Berners-Lee, "Linked Data: Principles and State of the Art," *In World Wide Web Conference*, 2008, pp. 1-40.

[3] A. Myłka, A. Myłka, B. Kryza, and J. Kitowski, "Integration of Heterogeneous Data Sources in an Ontological Knowledge Base," *Computing and Informatics*, 31 (1), 2014, pp. 189-223..

[4] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate Record Detection: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, 19 (1), 2007, pp. 1-16.

[5] E. Ioannou, O. Papapetrou, D. Skoutas, and W. Nejdl, "Efficient Semantic-Aware Detection of Near Duplicate Resources," *In Extended Semantic Web Conference*, Springer, Berlin, Heidelberg, 2010, pp. 136-150.

[6] D. Song, and J. Heflin, "Domain-Independent Entity Coreference in RDF Graphs," *In Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, ACM, 2010, pp. 1821-1824.

[7] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov, "Discovering and Maintaining Links on the Web of Data," *In International Semantic Web Conference*, Springer, Berlin, Heidelberg, 2009, pp. 650-665.

[8] A. T. Bayrak, A. Tuğrul, A. İ. Yılmaz, K. B. Yılmaz, R. Düzağaç, V. Bilimi, and O. T. Yıldız, "Near Duplicate Detection in Relational Databases," *In 2018 26th Signal Processing and Communications Applications Conference (SIU)*, IEEE, 2018.

[9] P. Achimugu, A. Soriyan, O. Oluwagbemi, and A. Ajayi, "Record Linkage System in a Complex Relational Database-MINPHIS Example," *Studies in Health Technology and Informatics*, 160 (Pt 2), 2010, pp. 1127-1130.

[10] M. Weis, and F. Naumann, "Detecting Duplicate Objects in XML Documents," *In Proceedings of the 2004 International Workshop on Information Quality in Information Systems*, ACM, 2004, pp. 10-19.

[11] M. Weis, and F. Naumann, "Detecting Duplicates in Complex XML Data," *In Data Engineering*, 2006. *ICDE'06. Proceedings of the 22nd International Conference on*, IEEE, 2006, pp. 109-109..

[12] N. Rizopoulos, "Automatic Discovery of Semantic Relationships Between Schema Elements," *In ICEIS*, (1), 2004, pp. 3-8.

[13] A. K. Joshi, P. Hitzler, and G. Dong, "LinkGen: Multipurpose Linked Data Generator," *In International Semantic Web Conference*, Springer, Cham, 2016, pp. 113-121.

[14] K. B. Lim, H. G. Jun, and H. J. Kim, "Semantics Preserving MapReduce Process for RDB to RDF Transformation," *International Journal of Metadata, Semantics and Ontologies*, 10 (4), 2015, pp. 229-239.

[15] S. Vahdati, F. Karim, J. Y. Huang, and C. Lange, "Mapping Large Scale Research Metadata to Linked Data: a Performance Comparison of HBase, CSV and XML," *In Research Conference on Metadata and Semantics Research*, Springer, Cham, 2015, pp. 261-273.

[16] K. Sharma, U. Marjit, and U. Biswas, "MAchine Readable Cataloging to MAchine Understandable Data with Distributed Big Data Management," *Journal of Library Metadata*, 18 (1), 2018, pp. 13-29.

Authors' Profiles

Kumar Sharma: Mr. Kumar Sharma holds bachelor & master degree in Computer Application. Presently, he is pursuing Ph.D. degree in the Department of Computer Science & Engineering, University of Kalyani, West Bengal, India. His research interests include Semantic Web, Ontology, Web Technologies and Big Data. He also has vast experience in mobile (iOS) application development in the field of Education, Point of Sale, and utility applications.

Dr. Ujjal Marjit: Dr. Ujjal Marjit is the System-in-Charge at the C.I.R.M.(Centre for Information Resource Management), University of Kalyani. He obtained his M.C.A. degree from Jadavpur University, India in 2000. His vast areas of research interest reside in Web Service, Semantic Web, Semantic Web Service, Ontology, Knowledge Management, e-Governance as well as Software Agents etc. More than 50 papers are published in the several reputed national and international conferences and journals.

Dr. Utpal Biswas: Dr. Utpal Biswas received his B.E, M.E and Ph.D. degrees in Computer Science and Engineering from Jadavpur University, India in 1993, 2001 and 2008 respectively. He served as a faculty member in NIT, Durgapur, India in the Department of Computer Science and Engineering from 1994 to 2001. Currently, he is working as a Professor in the Department of Computer Science and Engineering, University of Kalyani, West Bengal, India. He is a co-author of about 120+ research articles in different journals, book chapters, and conferences. His research interests include Optical Communications, Ad-hoc and Mobile Communications, Sensor Networks, Semantic Web Services, E-governance, etc.

How to cite this paper: Kumar Sharma, Ujjal Marjit, Utpal Biswas "RDF Link Generation by Exploring Related Links on the Web of Data", International Journal of Information Technology and Computer Science(IJITCS), Vol.10, No.10, pp.62-68, 2018. DOI: 10.5815/ijitcs.2018.10.08