Path Index Based Keywords to SPARQL Query Transformation for Semantic Data Federations

Thilini Cooray, Gihan Wikramanayake

Abstract— **Semantic web is a highly emerging research domain. Enhancing the ability of keyword query processing on Semantic Web data provides a huge support for familiarizing the usefulness of Semantic Web to the general public. Most of the existing approaches focus on just user keyword matching to RDF graphs and output the connecting elements as results. Semantic Web consists of SPARQL query language which can process queries more accurately and efficiently than general keyword matching. There are only about a couple of approaches available for transforming keyword queries to SPARQL. They basically rely on real time graph traversals? for identifying sub-graphs which can connect user keywords. Those approaches are either limited to query processing on a single data store or a set of interlinked data sets. They have not focused on query processing on a federation of independent data sets which belongs to the same domain. This research proposes a Path Index based approach eliminating real time graph traversal for transforming keyword queries to SPARQL. We have introduced an ontology alignment based approach for keyword query transforming on a federation of RDF data stored using multiple heterogeneous vocabularies. Evaluation shows that the proposed approach have the ability to generate SPARQL queries which can provide highly relevant results for user keyword queries. The Path Index based query transformation approach has also achieved high efficiency compared to the existing approach.**

Keywords— **Semantic Web; Keyword query processing; SPARQL query generation; RDF Federations**

I. INTRODUCTION

Nwadays the World Wide Web (WWW) has become essential to everyone. People always tend to search the web to retrieve information about almost everything. Once a enters a query, the underlying query processors must be able to gather results from available sources. What user is interested is, receiving relevant answers for their questions, efficiently.

Ability to understand the meaning of user query is important to provide relevant results. Once the user requirement is understood, it should be presented in a way which underlying data sources can understand and process. The relevancy of results provided by the data source depends on both the completeness of data stored in the source and how well the user query is understood by the data source.

Manuscript received on 23 Nov 2015. Recommended by Prof. K. P. Hewagamage on 16 June 2016.
This paper is an extended version of the paper "An Approach for Transforming Keyword-Based Queries to SPARQL on RDF Data Source Federations" presented at the ICTer 2015 Conference.

Thilini Cooray holds a B.Sc. (Honours) in Computer Science from the University of Colombo School of Computing, Sri Lanka.(e-mail: thilinicooray.ucsc@gmail.com).
Prof. Gihan Wikramanayake is a Senior Lecturer at the University of Colombo School of Computing. (e-mail: gnw@ucsc.cmb.ac.lk).

WWW contains huge amount of details about variety of topics. Most of them are stored as web documents. Web documents are capable of preserving complete details about topics rather than compacting them to traditional databases where only the details matches with the database schema are stored while skipping others despite their necessity for the completeness of information. However, as the amount of web documents are extremely increasing, requirement for effective storage mechanisms and efficient searching mechanisms were highly demanded. This paved way to the emergence of the concept of transforming web documents to web data.

Semantic Web[1] was introduced as a method of storing web data in such a manner which is understandable to a computer. . Resource Description Framework[2] (RDF) was presented as the standard format for storing and exchanging data. RDF preserves the interconnections among data elements and use graph structures for storage. Using graph structures for data storing, is crucial for web data as they contain huge amounts of relationships that relational databases are incapable of maintaining. These relationships are essential when recognizing the relevancy of data for a user query. Recently, many researchers and academic institutes have taken the initiative of exposing their data to the Web in RDF format. SPARQL[3] is the query language for RDF data. It is capable of both representing information needs along with relationships among elements and dive in RDF sources to extract information considering those relationships.

Relaxed models such as keyword queries are convenient for general users to query data sources as they do not have to consider the underlying complexity such as data structures and schema when composing queries.

Many researches have been carried out in keyword query search over tree [1], [2], [3], [4] and graph [5], [6] structured data. Basic idea behind keyword search is to identify matching data elements for keywords from the underlying data source and retrieve substructures which connect all those identified elements.

Structured queries are capable of retrieving more relevant results efficiently and accurately compared to keyword queries. However, composing structured queries require expertise knowledge which is lacking among general users. SPARQL is capable of retrieving more relevant results from web data. Therefore, bridging the gap between user friendly keyword queries and SPARQL allows general users to retrieve highly relevant results without having knowledge about underlying complexities. Transforming keyword queries to SPARQL is still a novel topic which has not gained much attention among semantic research attention. However, it could be identified as one of the key points in familiarizing the importance of Semantic web to general public and sharing

[1] http://en.wikipedia.org/wiki/Semantic_Web
[2] http://www.w3.org/RDF
[3] http://www.w3.org/TR/rdf-SPARQL-query

its privileges with them for fulfilling their information needs efficiently while enhancing the relevancy of results.

The process of translating keyword queries to SPARQL can be decomposed to following steps. 1) Mapping user keywords to data elements. 2) Identifying sub-graphs which can connect mapped data elements. 3) Generating queries based on the relationships in the sub-graphs. Most of the available approaches exploit graph traversal in real time for identifying suitable sub-graphs [3], [4], [6], [8], [9]. Only limited set of functions are carried out as preprocessing. Most of these approaches provide approximate results because traversing RDF graphs with millions or billions of data is very expensive and highly time consuming. Hence there is a requirement to seek for approaches which can reduce graph traversal in query generating time.

There are many contributors of the WWW who provides information on related topics individually. For an example, DBLP and ACM contain academic publication data individually. None of them have entire publication data. In contrast, Google Scholar connects sources such as DBLP and ACM to provide more complete set of results for the public. Therefore, general public is attracted more towards Google scholar for their publication related information needs. Most RDF sources are also maintained as individual dumps. In order to provide more complete results with high accuracy, it is important to combine those together. RDF federations have been presented as a solution for this problem. Yet existing RDF federations only accept SPARQL queries. Seeking for approaches which can direct user queries to RDF federations will enhance completeness and accuracy of provided results. This research focuses on transforming keyword queries to SPARQL on a RDF federation in order to allow general users to access Semantic Web and fulfill their information needs.

Following are the main contributions of this research:

- **Proposing an approach to map user keywords to data elements resolving vocabulary level heterogeneity** - An available ontology alignment mechanism is utilized for resolving vocabulary level heterogeneity. Results of the alignment mechanism are combined with a keyword index to map source wise matching elements for user keywords. This mechanism is capable of returning a set of keyword matching elements for each data source in the federation.
- **Building a Path Index capturing full paths accurately** - Path Index is an existing concept which reduces the cost of real time graph traversing for keyword query processing. The existing logic intends to store full paths from vertices to sink nodes as a preprocessing task. However, the breadth-first search based algorithm presented is unable to filter only full paths, causing unnecessary graph traversals at real time. Therefore, this research has proposed a depth-first search based approach which is capable of accurately capturing full paths.
- **Utilizing the stored templates in the Path Index to generate SPARQL queries without graph traversing in real time** - Path Index was previously used only for keyword mapping. This research proposes a way which Path Index can be utilized for SPARQL query generation. The proposed approach is capable of generating queries which can be directly executed on SPARQL query engines. Results of generated queries exhibit high precision and recall values.

The paper is organized as follows: Section 2 discusses related work; Section 3 gives an overview of the methodology, Section 4, 5 and 6 gives an in-depth explanation about each component of the suggested approach; finally Section 7 provides experimental evaluations and in Section 8, conclusions and future work are presented.

II. Related Work

Even though there is no existing approach for transforming keyword queries to SPARQL on RDF federations, research has taken place in addressing each step required for keyword to SPARQL transformation on RDF federations. They are as follows; *resolving vocabulary level heterogeneity*, *mapping user keywords to data source elements* and *identifying suitable sub-graphs connecting keyword elements*.

Heterogeneity resolution is a core function of federations according to Sheth and Larson [16]. There are different levels of heterogeneity such as vocabulary level heterogeneity and data level. However, vocabulary level heterogeneity must be resolved to process keyword based queries in a federation. SPARQL query rewriting is one approach for resolving vocabulary level heterogeneity [17], [18], [19]. The input query must be written in SPARQL using a specific vocabulary for applying this solution. This cannot be applied when input query is in keyword.

Ontology alignment is another method proposed for heterogeneity resolution among RDF data sources. Concept level, property level and instance level are the ontology alignment types according to Gunaratna et al. [20]. BLOOMS [13], Aroma [21] and RiMOM [14] are some concept level alignment approaches. Gunaratna et al. [20] have mentioned that very less amount of research have been focused on property level alignments. Alignment API [12] can be identified as a proper tool for property level alignments from different vocabularies. Since this research aims at heterogeneity resolution for a federation, both concept level and property level vocabulary resolutions are required.

Indexing is the common approach adopted by almost the entire keyword query processing approaches for matching data elements with keywords. However types of indices they have used are different. SearchWeb [8], Bidirectional Search [6] keeps indices for both vertices and edges. They consider the possibility of a user keyword occurs at an edge as well as at vertices. BLINKS [5] believes that keywords can only occur in vertices of the graph so index only vertices. Path Index [7] only store sink nodes in their index arguing that keywords only reside in sink nodes. When storing details, most approaches index only the label of the graph element. However SearchWeb [8] and Hermes [15] store some additional information along with the label. Those details are used for efficient sub-graph identification on that approach.

Many approaches have been suggested for identifying suitable sub-graphs which can connect keyword elements. Basic tree search algorithms such as Breadth First Search [22] and Depth First Search [23] were first applied for substructure identification in tree structured data. Then several other algorithms [24], [25] were proposed by modifying those basic concepts. As RDF data sets mostly have a graph structure, graph exploration approaches were

proposed such as Backward Search [9] and Bidirectional algorithm [6]. SearchWeb [8] has suggested a summarized graph which has reduced the size of graph which needs to be traversed at real time for finding suitable sub-graphs.

Real time graph traversing is highly time consuming. M-KS [26] uses a matrix to store the keyword relationships to eliminate graph traversal at real time. They only focus on binary relationships. G-KS [27] proposes a keyword relationship graph to find a suitable sub-graph to resolve the weakness of M-KS. Tran et al. [10] suggest a graph based model to sub-graph identification. Cappellari et al. [7] suggest a path index based approach, which stores edge sequences from source nodes to sink nodes of RDF graphs. As they have totally eliminated graph traversing at real time, efficiency of this approach is really high. However they require more storage space than other approaches.

Among keyword searching approaches, only three methods have been proposed for transforming keyword queries to SPARQL. SearchWeb [8] and Hermes [15] have proposed a method for converting keyword queries to SPARQL by identifying suitable sub-graph and converting it to conjunctive queries. They have only proposed that approach either to a single data source or a set of linked data sources. They have not considered the federation scenario. Unger et al. [28] have suggested a linguistic analysis based approach for transforming natural language queries to SPARQL. They have ignored the capabilities of Semantic Web in their solution.

III. Methodology

We propose an approach for transforming keyword queries to SPARQL on RDF data source federations within a set of defined limitations. The main objective of our research is to examine the feasibility of proposed approach for keyword to SPARQL transformation on federations as no existing approach has addressed this issue. We do not focus on examining the generalizability and scalability of this approach at this initial stage.

We use academic publication data for explaining and evaluating this approach at the initial phase even though this proposed approach can be used domain independently. Author name, published year and publication title are the initial set of keyword fields we are using for generating keyword queries. These fields were selected as those are the fields which are highly queried regarding academic publications even in digital libraries such as ACM and DBLP. We have defined a specified format of queries to this approach. All conditions of the user query should be represented according to the format <field name>:<field value>. Comma should be used if multiple conditions are presented. Field which needs results should be indicated using a question mark (?). For an example, accepted keyword query for “What are the publications of James published in 1995?” is “publication:?,author:James,year:1995”.

RDF data are stored in graph structures. Therefore cycles can occur. Inverse relationships can be identified as a main reason for causing cycles in RDF graphs. This research only aims at resolving cycles caused by inverse relationships on RDF graphs.

Heterogeneity resolving is a main focus in federations. Vocabulary level heterogeneity and instance level heterogeneity are the main levels of heterogeneity in RDF federations. Approaches for dealing with those heterogeneous situations are needed to be included in the architecture. Resolving vocabulary level heterogeneity in a federation is essential for retrieving complete results, because different vocabularies usually use different terminologies for similar concepts. Identifying the similarity among vocabularies paves the way for extracting relevant results from heterogeneous data sources. When consider instance level heterogeneity, it contains a separate level of complexity. Different data sets may store same value in different formats. For an example, an author name "A.Bernstein" may have stored in one data set as "A.Bernstein", while another data set as "Arnold Berstein". Same name can reside in several attribute fields as well. For an example, name "Levenshtein" can b e a name of an author as well as a part of an article titled such as "Levenshtein Distance". Heterogeneity caused by same entity identified by different names are not going to be resolved in this approach. Because the original format which the data are stored at each data store is required for generating SPARQL queries and retrieve answers. The ambiguity of same literal reside under different fields (author, title etc.) are resolved by introducing specific keyword fields to the user.

Architecture of our proposed approach is depicted in Fig. 1. It contains three main components namely query validator which validates the user inserted query, keyword to attribute mapper which identifies the elements in user query, their relationships and how they can be related to existing data elements in the federation. Final component is query converter where the SPARQL queries are generated based on the keyword queries entered by user. Each of these components is discussed in upcoming sections.

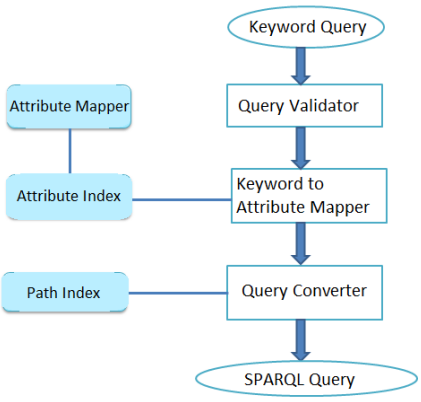

Fig. 1 Architecture of the proposed approach

IV. Query Validator

This component is used for validating the format of input keyword query. Keyword queries adhere to a defined format are only focused in this methodology. Keyword queries of proposed format are only expected by next steps in this approach as well. Therefore, validating the input query format is essential. A simple regular expression based lexical

analyzer and a parser is used for this purpose. Once the user query is inserted, it is directed to the lexical analyzer for tokenizing and removing unnecessary white spaces. If any error occurred during this stage, the query will be rejected and an error is displayed. If the query was accepted by the lexical analyzer, a token steam is sent to the lexical parser. A regular expression which defines the accepted format of user query is included in the lexical parser. If the input query matches the grammar rules, it is a valid query. Invalid queries are notified as errors. Valid queries are stored as key-value pairs and sent to the next component.

V. Keyword to Attribute Mapper

Once the Query Validator sent a set of key - value pairs based on the input user query, Keywords to Attribute Mapper have several tasks to complete. They are as follows: identifying the fields which user query consists of, retrieving matching data source elements for user keywords from the federation, retrieving vocabulary dependent terms for user query's variable fields from heterogeneous vocabularies in the federation, clustering identified matching element sequences for each data source based on the vocabulary they belong to and finally ranking those clusters based on their capability to answer the user query.

There are several sub tasks aligned with the above list of tasks. In order to identify matching elements for user keywords, this research proposes maintaining pre-processed data as practiced by several other existing solutions [1], [6], [8], [10], [11]. With the pre-processed data, searching can be done faster. Two types of pre-processed data are proposed for this phase namely Attribute Mapper and Attribute Index.

A. Attribute Mapper for Heterogeneity Resolution in the Federation

Vocabulary level heterogeneity is a common characteristic of RDF data source federations. Even in the domain of academic publications, fields similar in meaning are represented in different labels. For an example, ACM RDF data source identifies publication title as "title" and SWE DBLP identifies publication title as "label". However, in order to identify that all those different labels means the same entity, a mapping is required among those vocabularies.

Ontology alignment is a highly researched field which can also be utilized for resolving vocabulary level heterogeneity only. On the other hand, WordNet[4] is a lexical database created for English. It has categorized different English words based on their similarities. Also, it keeps details about the origin of words. For example, if we consider the word, "author", we can retrieve that the parent class of "author" is "person". Also if we input "author" and "editor", we can retrieve the common word "person" which can be used to identify both of them. Hyponym and hypernym relations indicated in WordNet serve this purpose. These are two options for resolving vocabulary level heterogeneity. When considering the terms in vocabularies, it was identified that some terms which are defined as concepts on some vocabularies are defined as properties on other vocabularies in the federation. Therefore, both concept level and property level alignments are required in this approach. Alignment API [12] was selected for this task as it has capability of property and concept alignment. Its accuracy is better than other approaches [13], [14] and it also has the capability of integrating WordNet which provides added advantage.

Alignment API [12] is only capable of aligning two vocabularies at a time. Therefore, this research uses a semi-automated approach to resolve vocabulary level heterogeneity among all the vocabularies in the federation and construct the Attribute Mapper. First, the required concepts for the specified scope are manually identified from a single vocabulary. For an example, if DBLP is considered, "label" is the predicate used to identify publication title, "author" is the predicate used for indicating author list of a particular publication etc. Then all other vocabularies in the federation are aligned with DBLP source using Alignment API. For an example align vocabulary of ACM, vocabulary of CiteSeer with DBLP each at a time. Once all the output alignment details are received, only the alignment of all vocabularies gets started. All the entities (classes, attributes etc.) aligned with previously identified concepts of DBLP are clustered together along with their data source details. This helps to extract different terms used by heterogeneous sources to identify same entity in the federation.

B. Attribute Index for Mapping Keywords with Data Elements in the Federation

Matching data elements for user keywords must be identified as the first step of keyword to SPARQL conversion. Commonly used keyword index approach [5], [15] is decided to use for this phase with several modifications.

Definition 1: A *keyword index* is a keyword to element map which returns a set of matching elements to a keyword.

RDF is a graph structured data store. Data vocabulary elements are represented by vertices and edges of a graph.

Definition 2: A *RDF graph g* is a tuple (V, L, E) where

- V is a finite set of vertices as the union $V_E \cup V_V$ with entity vertices V_E and value vertices V_V
- L is a finite set of edge labels as the union $L_R \cup L_A$ with relation labels L_R and attribute labels L_A
- E is a finite set of edges of the form $e(v_1,v_2)$ with $v_1,v_2 \in V$ and $e \in E$. Following types of edges can be defined:
  - $e \in L_A$ (attribute edge) if and only if $v_1 \in V_E$ and $v_2 \in V_V$
  - $e \in L_R$ (relation edge) if and only if $v1,_{v2} \in V_E$
  - type is a special relation which indicates the membership of an entity to a particular class

Most of the available approaches have indexed both V and L in their keyword index, arguing that keywords can occur in V and L both. But Cappellari et al. [7] mentions that keywords can mostly reside on sinks. Sink is a node in RDF graph which does not have any outgoing edges from it. They have introduced a term called source to identify vertices which have no incoming edges. So they have only indexed sink vertices of a RDF graph. We also have decided to follow this approach and generate the index only for sink vertices. We are adopting the data structure used by Tran et al. [8] to

[4] http://wordnet.princeton.edu/

store additional details about indexed elements. Additional details are the set of adjacent edges directed from one-edge distant vertices to this element, set of primary edges which are the adjacent edges to a source which this element belongs to, type of the source V_E to which this sink V_V belongs to and data source identifier. This additional information is required efficiently identifying suitable sub-graphs for query generation. Apache Lucene[5] document index was utilized for building our Attribute Index.

In most data sources, sink elements do not reside in one edge distance from its source vertex. For an example if we consider a publication, it can have many authors. In such cases, a collection approach is required to store multiple authors. So the adjacent edge to the element may not be the adjacent edge to the source. Adjacent edge to the source is the one which defines the relationship not the adjacent edge to the sink. Situations where these two are different have not been addressed by previous approaches. Therefore, this research proposes a primary edge to be stored as well as the adjacent edge. Primary edge is the adjacent edge of the source to which element connects. There can be either no other edges between primary edge and adjacent edge on the path or both adjacent and primary edges can be same or several edges can occur between primary and adjacent edge. But they are not subject to store in the data structure.

Proposed approach uses same keyword index to store all sink values from all the data sources in the federation. If same element resides in two or more datasets, several data structures for each source are created and stored under the same key. Only the sink element labels get indexed while all the information resides in data structure gets mapped to indexed element. Therefore, it takes a less amount of space to store this keyword index than other methods' indexes who index all the vertex data.

Fig. 2 shows a sample data graph fragment of DBLP. "James Peter" is a source as it does not have any outgoing edges. Paper-Peter95 is its source. Source does not have any incoming edges. The data structure which returns from the Attribute Index for the term "James Peter" is [James Peter (node label), ns#1(adjacent edge), author (primary edge), Book Chapter (type of source), DBLP (data source id)].

Once Attribute Mapper and Attribute index are ready, real time processing begins. Using the key-value pairs received from query validator, this component identifies what are the variable fields of the keyword query. Those variables are sent to Attribute Mapper and retrieve vocabulary dependent terms for them. Condition values of the keyword query are sent to Attribute Index and receive matches.

[5] http://lucene.apache.org/

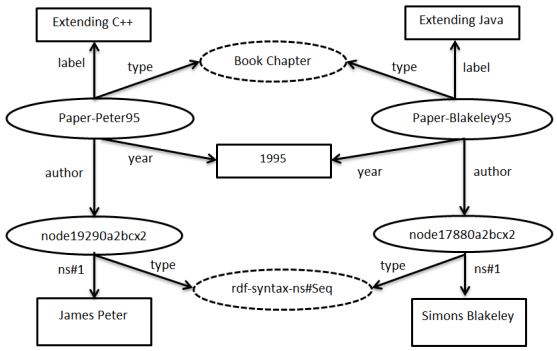

Fig. 2. Data graph fragment of DBLP

Primary edges of those index results are matched with attribute mapper to make sure we receive the values with the required attribute we want. Now cluster those results based on the data source identifier. Then we employ a data source ranking approach to decide which source is most capable of answering user query. In a non-distributed environment, it is most advisable to generate single query, process and send its results to users without keeping user waits until all queries for the entire federation are generated because users need efficiency. If a data source has matching elements for all the user keywords while another source only have matching elements for some of the user keywords, the former data source gets a high priority when query converting. Those ranked data sources are then sent to Query Converter component.

Following table shows sample tuples of DBLP and ACM for the query "publication:?,author:James,year:1995". Variable field of the query is "publication". That was sent to attribute mapper and "label" and "has-title" was retrieved for DBLP and ACM respectively. When consider condition values, DBLP has matching elements for both "James" and "1995" while ACM has matching elements only for "James". Therefore it is clearly understood that there is a more possibility of retrieving accurate results from DBLP them ACM for this query. Therefore DBLP gets higher priority.

TABLE I
OUTPUT TUPLES FROM KEYWORD TO ATTRIBUTE MAPPER

Source	Keyword	Primary Edge	Adjacent Edge	Source Type
DBLP	James Peter	author	ns#1	Book Chapter
	1995	year	year	Book Chapter
	Variable field : publication , vocabulary dependent value : label			

ACM	James McClelland	has-author	full-name	Article Reference
	Variable field : publication , vocabulary dependent value : has-title			

VI. Query Converter

This component is used to identify the most suitable sub-graph which can connect identified keyword elements from the previous component and generate SPARQL queries based on identified sub-graphs. SPARQL queries are generated by identifying the format (template) of the sub-graphs which consist of the answers for the query. Vertices of the target sub graph were found by Keyword to Attribute Mapper. However, edges which connect those vertices were not provided. Hence Query converter first has to seek for a method for identifying relationships among sent elements by previous component. Many approaches [6], [8], [9] try to find suitable sub-graphs by graph traversing at real time, which is highly time consuming. A path store [7] is utilized in this research for sub-graph identification because it has totally eliminated real time graph traversal.

Objective of Path Index is to keep records of how vertices (classes, property values and literals of RDF graphs) are connected to each other prior to actual query processing starts. Paths are defined as the route from given element to another in RDF graph. Efficiency of query processing improves by utilizing those pre-processed data. Therefore, the cost required for real time graph traversal reduces heavily. Those path data are stored in a relational database.

Sources and Sinks are the main elements required for defining paths. Full path is defined by a route from a source to a sink. Template of a path is retrieved by replacing vertices of a path by wild cards. Templates indicate the relationship among vertices. When considering the SPARQL query generation context, those templates are the components which we need to find for generating queries. When considering a SPARQL query, intermediate nodes of a path is always indicated by variables. Therefore, we can easily generate queries by utilizing suitable templates.

Database schema presented in Path Index mainly consists of four main relations; Node, Template, Path and PathNode. Path Index assumes that user queries targets only sinks [7]. Therefore, only data about sinks are stored in Node table. Path table keeps data about all the unique paths from sources to sinks. Each tuple consists of path is, template id, length of the path and id of the final node of the path. PathNode table keeps track of which node resides in which position of a path. Index Organized Tables concept is used for indexing these tables.

Database schema presented in Path Index was adopted for this research due to its simplicity and usefulness for generating queries. Cappellari et al. [7] has presented a breadth first search based approach for exploring the RDF graph when populating tables with data in the database. This approach stores intermediate paths as well as full paths in Path table. When utilizing Path Index for SPARQL query generation, it was decided that storing full-paths are only required. Since the main intention of using this index is to identify relationships among keyword elements, sources can be considered as most promising connecting elements which can reach many other elements quickly once matching elements are found from sinks. Therefore sources can behave as local connecting points when we are trying to find the best sub-graph to connect keyword elements.

A. Full-path Identification

Cappellari et al. [7] have presented a breadth-first search based algorithm for capturing full-paths in a given RDF dataset. However that algorithm stores full paths as well as partial paths which requires huge unnecessary space. Therefore a depth-first search based algorithm was proposes in this methodology which can identify full-paths accurately avoiding unnecessary space wastage caused by the original algorithm. Proposed algorithm explores the RDF graph in Depth First manner. Sources available for each dataset are identified and depth first traversal starts from each source. It searches the entire RDF graph until it meets all the sinks which can be reached by the current source. Algorithm locally creates an n-ary tree considering current source as the root. All the triples whose subject is root are considered as the branches of the tree. If the object of each triple has become a subject in another triple, those branches grow accordingly. Sink nodes never become a subject of a triple so occur as leaves of the tree. Proposed algorithm for generating path tree for each source is shown below.

Algorithm 1 DFS based graph exploration for full-path capturing

Input : RDF triple dataset *DATA*, path tree *TREE*, parent node *P*, matching triple set *TRIPLES*

```
for each triple t in TRIPLES do
    add t to node P on TREE
    var newtriples = get triples from
    DATA (subject = t.object)
    if (newtriples.count > 0) then
        DFSbasedgraphtraversal (DATA,
    TREE,t.object,newtriples)
    end if
end for
```

Once a source-tree is generated, a method is required to identify all the full paths in the tree because those are needed to be stored in Path Index. Since tree nodes have only a single parent, full-paths can be captured by recursively traversing from each leaf to root. Proposed algorithm is shown below.

Algorithm 2 Complete algorithm for Path Index building

Input : RDF triple dataset *DATA*, path tree *TREE*, source list *SOURCES*

```
for each source s in SOURCES do
    var triples = get triples from DATA (subject = s)
    var TREE = generate tree (root = s)
    Algorithm 1 (DATA,TREE,s,triples)
    var pathlist
    var leaves = TREE.leaves
    for each leaf l in leaves do
        var leafnode = leaf
        var path
        while (leafnode ≠ TREE.root) do
            add leafnode to path
            leafnode = leafnode.parent
        end while
        add path to pathlist
    end for
    store pathlist in PathIndex
end for
```

B. Resolving Cycles Caused by Inverse Properties

Inverse relationships are a main reason for occurring cycles in RDF graphs. Suppose there is a triple is a RDF graph whose subject is *S*, predicate is *P,* object is *O*. If there exists another triple in the same RDF graph whose subject is *O*, predicate is *P'* and object is *S*, *P* and *P'* has an inverse relationship.

Both triples connected by an inverse relationship contains same amount of information. Therefore removing one does not cause any information loss in the RDF graph. A popularity based approach is used for filtering the most appropriate triple. Popularity is measured by the number of unique predicates each subject has. Higher the number of unique predicates, higher the amount of information it has access to. Proposed algorithm for inverse property based cycle resolution is shown in Algorithm 3.

Algorithm 3 Resolution for inverse property based cycles in graphs

Input : RDF triple dataset *DATA*
Output : Cycle resolved dataset *DATA*
Initialisation : Inverse statement list *inverselist*

```
1.  for each triple t in DATA do
2.    if (inverselist.notcontain(t)) then
3.      var inversetriples = get triples
      (subject=t.object,object=t.subject)
4.      for each inverse i in inversetriples do
5.        var subjectpopularity = get unique
      predicate count (i.subject)
6.        var objectpopularity = get unique
      predicate count (i.object)
7.        if (subjectpopularity > objectpopularity)
      then
8.          add t to inverselist
9.          remove t from DATA
10.       else
11.         add i to inverselist
12.         remove i from DATA
13.       end if
14.     end for
15.   end if
16. end for
```

C. Path Index Based Sub-graph Identification and SPARQL Query Generation

Once Path Index generated, SPARQL query generation should be done in real time. Once mappings are retrieved, Path Index is queried to retrieve paths whose final node is the value of the mapping received from Attribute Index. There can be several sub-graphs in a data source which can connect those keyword elements. But they all use the same schema (vocabulary) Consider a situation in DBLP where 3 sub-graphs exist which connects author James, year 1995 with 3 publications. Those publications become answers as publication was the variable in user query. If all the vertices of each sub-graph are replaced with wild cards, the result graph is totally similar. That graph is the sub-graph which is needed to traverse to get answers. That is the sub-graph which we should convert to SPARQL syntax for generating the query. This sub-graph with wildcards is known as template graph. . For an example, template of full-path "PaperPerter95-year-1995" is "#-year-#". When converting a sub-graph to SPARQL syntax, we have to replace the vertices by variables.

Several different templates can be received as matching paths when same keywords repeat under different concepts in the vocabulary. For an example, consider person "James". He can be a program committee member of one conference in 1995 while being an author of publications. Since templates were extracted only considering full-paths whose sink nodes are keywords without focusing on their relationship with variable field, templates matching for both scenarios will occur. If results were generated for both these template graphs, overall relevancy of results will become low. Therefore a filtering process is required. Additional information stored in index documents comes to use at this situation. Filtering process considers templates which matches with primary and adjacent edges keyword elements and ignore others. If there are several results on this approach, shortest template will be selected as the suitable sub-graph as lesser the length of the path, faster the query processor can reach it and output results. Tran et al. [8] has also mentioned path length as a common matrix used by many graph traversal approaches to rank selected sub-graphs. Lesser the path length, there is a high probability that it would reduce the overall size of the sub-graph it resides in.

Next this proposed methodology looks for extracting suitable templates for variable fields in the user query. Shortest template which contains the vocabulary specific properties of the variable field is selected as the template. A sub-graph which can connect all the keywords is required for generating a SPARQL query. Now paths for each field have been retrieved, finding connecting elements is required to generate a sub-graph using these paths. Details stored in Path Index are used for finding connecting elements. Sources of the data sets were identified while building Path Index. Sources are operating as centre nodes to connect all the property values of the source together. Sources mostly represent the main focus of the vocabulary. For example, if DBLP and ACM vocabularies are considered, publication is their main focus. All the attributes related to publication are defined as properties. Therefore sources can be identified as a potential element for connecting the paths for generating a possible sub-graph. SPARQL queries are generated based on this argument. Following is a sample query generated for the example keyword query "publication:?, author:James, year:1995"..

```
SELECT ?z
WHERE {
    ?x type Book Chapter .
    ?x label ?z .
    ?x year 1995.
    ?x author ?y.  ?y ns#1 "James Peter". }
```

If there are several matching elements for a single keyword (Ex : Many different people with "James" as a part of their name) or many different sub-graphs match for the query, FILTER option of SPARQL is used in generated queries. Following is an example is a scenario where there are multiple authors named "James" available.

```
SELECT ?z
WHERE {
    ?x type Book Chapter .
    ?x label ?z .
    ?x year 1995.
    ?x author ?y.
    ?y n s # 1 ? a .
     F I L T E R ( r e g e x ( s t r ( ? a ) , " J a m e s " ) ) .
 }
```

VII. RESULTS

The Proposed approach was implemented using Java with the support of Apache Jena Semantic Web library and Oracle 11g DBMS. Once a keyword query is inserted, it identifies the data sources in the federation which can answer the keyword query and transforms input query to vocabulary dependent SPARQL queries to match with the data sources in the federation.

Evaluation setup is as follows. A federation of 3 publication data sets were created. DBLP[6] RDF dataset with 37446 triples, ACM[7] RDF dataset with 63980 triples and Semantic Web Dog Food[8] (SDF) dataset with 37105 triples were used as test data. Portions from DBLP and ACM data sets were used because SDF data set was not large enough as them and publication details between 1986 and 2005 were only assessed due to the availability of data on all 3 data sets. These three sources were selected based on following aspects.

Publishers of these data sets have exploited three different ontologies (SWEtoDBLP, aktors and SWRC ontologies respectively) for publishing these data. Therefore schema level heterogeneity is clearly showcased among the selected data sources. RDF data are stored in graph structures. Therefore problems in graph data handling also arise when dealing with RDF data. Cycles are one such problem. Here we selected two data sets with cycles. ACM data set has cycles caused by the subject of a triple has become its own object. SWRC ontology is an integrated ontology of several ontologies. Therefore it has inverse relations. These inverse properties have introduced cycles in SDF data set. These data sets are used to experiment the proposed approach's ability to deal with common cyclic scenarios of RDF data. DBLP data set does not consist of cycles. However, specialty with this data set is that it consists of blank nodes. Blank nodes are anonymous nodes in a RDF graph. These can be used to group sub-properties of an instance. Likewise, data sets which exhibit different characteristics which covers most of the common characteristics of RDF graphs are used for the experiment in order to show the generalizability of this approach for RDF federations.

Experiments were conducted on a machine with AMD V120 processor of 2.2 GHz and 4GB RAM. A test keyword query set of 10 queries were created by considering all the possible combinations of the three keyword fields (author, year, publication) we selected for academic publication data. Table II shows test query set.

TABLE III
TEST QUERY SET

	Keyword query
Q1	publication : consistency , author : ?
Q2	publication : distributed , year : ?
Q3	author : sylvia , year : ?
Q4	author : andrew , publication : ?
Q5	year : 1986 , author : ?
Q6	year : 1988 , publication : ?
Q7	publication : multimedia , author : daniel , year : ?
Q8	publication : concurrent , year : 1990 , author : ?
Q9	author : david , year : 2004 , publication : ?
Q10	publication : protocol , author : ? , year : ?

A. *Quality Evaluation of the Federation*

The first evaluation criterion was to evaluate whether our proposed approach actually carries out its intended task of correctly translating user keyword queries to SPARQL. Measurements were taken by considering the relevancy of the retrieved results by executing the generated queries. Since main intention of this approach is to give general users more relevant and accurate results using the privileges of Semantic web, quality of results were measured using following measurements. Quality of the generated SPARQL queries were evaluated by measuring precision, recall and F-measure of the results received for above mentioned test query set. F-measure is a balanced measurement used to capture the balance between precision and recall of each result set. This measure was used as some results can be high in precision but low in recall and vice versa. F-measure gives a balanced score in such situations. Gold standard results were obtained by running SQL queries on Path Index of the data sources and manual evaluation on the raw RDF data sets. Fig. 3 shows the results graphically.

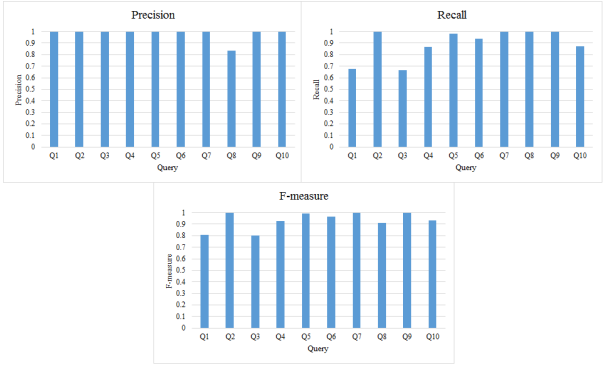

Fig. 3. Quality evaluation of the federation

[6] http://lsdis.cs.uga.edu/semdis/swetodblp/
[7] http://datahub.io/dataset/rkb-explorer-acm
[8] http://data.semanticweb.org/

Overall precision has an average of 0.98 and overall recall has an average of 0.9. Based on the precision results, proposed approach is capable of generating queries which can give more accurate results. However, a loss of recall was detected compared to precision. This was caused by the decision made about finding the connecting elements of sub-graphs. The source node was selected as a connecting element and its type was decided by the type which has maximum matching from the attribute index. Sometimes this type did not match with the actual source of the paths which were retrieved from the Path Index. It caused loss of results. Another point was that the capability of a data source in producing results were decided for a user query if that source have matching elements for all the keywords in the query. However there are situations which keyword element reside in the RDF graph, but there is no a sub-graph which can actually connect them all. In such situations generating a query is wastage of effort.

Then recall values were compared of each data source with the recall of the federation. This evaluation was done to identify whether there is a significant impact on the result by processing the query over a federation rather than on an individual data source. One of the objectives of transforming a keyword query to SPARQL on a federation was to give a more complete result set to the user. If a single source is capable of giving the same result set, there is no requirement of a federation. Recall is the measurement which indicates the contribution of each source over gold standard result set. Fig. 4 shows that federation gives more complete results over individual sources. Therefore it can be shown that federation approach is capable of producing more complete results than any of the individual sources.

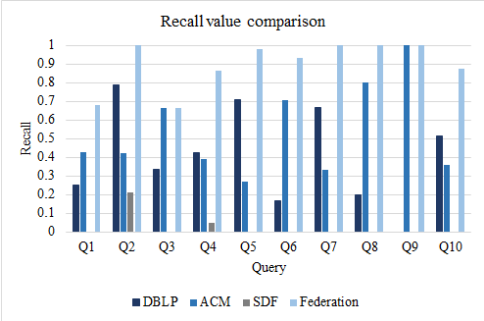

Fig. 4. Recall comparison of the federation

B. *Performance Comparison for the Proposed Path Index Based Approach*

Performance evaluation was carried out after receiving satisfying results for the quality of the proposed approach. Prior to this research, all the presented approaches for this task [8], [28] have either used online graph traversal approach or natural language processing support for query translation. Path index [7] was first suggested for keyword searching. Its functionality was exploited for generating SPARQL queries from keyword queries. An index based approach was adopted related to this method used by Tran et al. [8] Both these approaches and SearchWeb have utilized a keyword index for mapping keywords with data source elements. However proposed method used a path index based approach for identifying substructures which can connect keyword elements while they followed a real time graph traversal mechanism.

The graph exploration and top-k query calculation approach presented in [8], performs better compared to other available methods for finding sub-graphs such as backward search [9], bidirectional search [6], breadth first search and depth first search. Tran et al. [8] has shown in their evaluation that query generating time has achieved a comparable decrement by using their graph exploration mechanism. Natural language based approach suggested in [28], is totally deviated from the proposed approach. The intention of using a path index based approach for query generation was to experiment whether it can achieve a performance gain in query generation time by pushing graph traversing to the pre-processing stage. Therefore, query generation time of proposed approach against SearchWeb approach presented by Tran et al. [8] was evaluated.

Time taken to identify the most suitable query substructure which can be used for query generation as our matrix of performance was measured. Once it is identified, either proposed approach or their approach could have been used for translating the substructure to match with SPARQL query syntax. Time taken by SearchWeb approach to generate top-1 query was only considered since proposed approach only output a single query per data source. Query generation time only for ACM data source was compared at this section as SearchWeb method has only suggested for query generating for a single data source. In the next section, the performance comparison when it comes to federation scenario will be demonstrated. Fig. 5 shows the performance comparison.

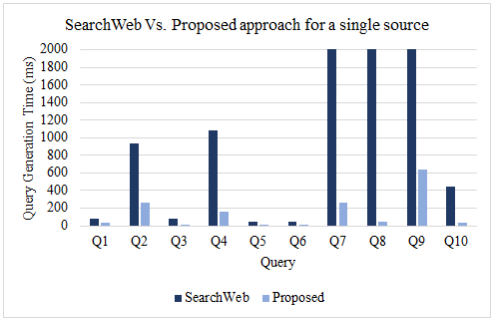

Fig. 5. SearchWeb Vs. Proposed approach for a single source

Figure illustrates that proposed approach has a significant performance gain compared with SearchWeb suggested by Tran et al. [8] for the tested query set. Since search web has already been outperformed other related methods, we only evaluated against SearchWeb. Main reason behind the high query time for search performance is its real time graph traversing. Even though SearchWeb doesn't do a full graph traversal in query generation time, it uses graph traversal. First, they create a graph summary by extracting the class vertices and entity vertices from the original RDF graph. This summary graph behaves as the schema. Once the matching elements are retrieved using the keyword index, SearchWeb embeds those matching elements to the summary graph while exploiting the "adjacent edge" property in the retrieved index records. Therefore, more the keyword matching elements, bigger the summary graph will be.

Keyword elements matching for "distributed" in Q2 is around 50 and "andrew" in Q4 is around 30. Once those 30 vertices are added to the summary graph, it gets bigger. SearchWeb traverse through all the possible starting from the shortest once generated the augmented summary graph by adding those matching elements. This is done to retrieve elements which can connect all keyword elements. Once a connecting element is found, SearchWeb considers all the path combinations among keyword elements even they have same adjacent edge to get the shortest path. Bigger the graph, higher the number of combinations will be. This leads to high query generation time.

Q7, Q8 and Q9 have two conditional keywords in the query. Therefore augmented graph of SearchWeb becomes bigger. Number of possible paths among elements also increases drastically. That is the reason for the huge query generating time.

In comparison, proposed approach shows a huge performance gain as there is no real time graph traversal in it. No matter how much matching elements are output from the keyword index, if they have same template, all of them are considered as a single element from template's point of view.

C. Performance Comparison for the Federation

Query generation times for the entire federation. Proposed approach is capable of generating SPARQL queries to all the sources in the federation in one go. However SearchWeb can only generate a SPARQL query for one source at a time. Therefore queries to all the sources in the federation were repeatedly generated and got the total time for the comparison. A huge performance gain was retrieved by this proposed approach this time as well. This showed that real time graph traversal for SPARQL query generation and keyword searching is highly inefficient when dealing with huge data sources. Proposed approach is the first research which utilized Path Index [7] for SPARQL query generation. Satisfying results were obtained on its performance. Fig. 6 shows performance comparison for the federation.

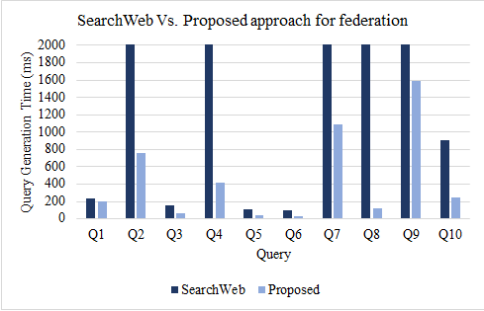

Fig. 6. Performance comparison for the federation

D. Proposed Federation Approach Vs. Digital Libraries

The main reason for emerging Semantic Web concept over Web of Data is ability of Semantic web to extract the meaning and relationships of data elements and exploiting them forgiving more meaningful and relevant results for user queries.

Once after comparing the performance of the proposed approach, this section focused on evaluating this aspect by running the same set of user queries on both digital libraries and proposed semantic web approach. Three digital libraries were selected for this evaluation. DBLP, ACM and Semantic Web Dog Food data dumps were used for sample federation in this research. Therefore DBLP digital library[9], ACM digital library [10] and Semantic web [11] online data store were selected as test digital libraries for this evaluation criterion. Google Scholar[4] was not used for this evaluation as it consists of publications from many sources, not only from above three.

Among these three digital libraries, DBLP and ACM only have advanced search capabilities. Semantic Web Dog Food website doesn't have an advanced search capability. Therefore, Semantic Web digital library was excluded from evaluation. When consider the query executable ability of digital libraries compared to proposed semantic web federated approach, proposed approach was capable of executing all the test queries. This means the proposed approach was capable of executing any combination of target query fields (publication title, author name, published year). In contrast, digital libraries were mostly capable of providing direct answers when only question needs publication titles as results.

For an example, consider Q4 and Q6. They ask for the paper titles authored by author named "Andrew" and paper titles published in year 1988. In Q9 user requests titles of publications authored by "David" in 2004. DBLP and ACM were able to answer those three queries correctly but Semantic web search was not advanced enough to directly provide answers to match with those conditions.

When it comes to queries like Q1, Q2, Q3, Q5, Q7, Q8 and Q10, those queries does not request publication titles. For an example, Q1 wants all the author names publishes papers titles with "consistency". Q3 wants a list of years which author "Sylvia" has publications. Digital libraries were not capable of providing either an author name list or year list as answers for these queries. They provided a list of publications with word "consistency" for Q1. By manually extracting only their author names could be retrieved. Similarly for Q3, a publication list was received which were authored by "Sylvia". Their years needed to be manually extracted. Years were easier to be captured in DBLP while author list could be easily received with filtering in ACM. In comparison, proposed semantic web approach was capable of providing direct answers for all those queries as well. Fig. 7 shows comparison among F-measures of DBLP and ACM digital libraries with proposed federated approach.

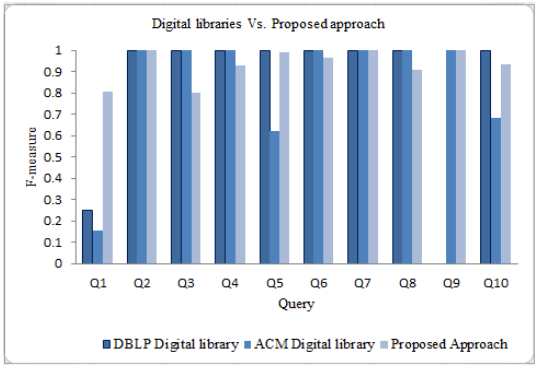

Fig. 7 Quality comparison of Digital libraries vs. Proposed Approach

[9] http://dblp.uni-trier.de/
[10] http://dl.acm.org/
[11] http://data.semanticweb.org/

This shows that our Semantic Web based federation approach is capable of giving highly relevant results for user queries than existing keyword matching digital libraries.

Google Scholar[12] follows a non- semantic web approach for combining all those results. However it also cannot answer queries like Q1, Q2 and Q5 directly. It also only focuses on filtering publications. If this proposed approach can be applied on all the publication sources, It could have performed better in answering all the publication related queries no matter whether it is about an author, publication, year or venue. This clearly exhibits the usefulness of semantic web and level of accuracy which can be gained from executing SPARQL queries over keyword based queries.

In queries like Q1, Q2 and Q5, user needs a list of authors or years. Semantic web approach was capable of giving that relevant information to user. However digital libraries were unable to output the relevant results. Users have to manually filter in order to retrieve the relevant result set. Therefore in relevancy of results, proposed semantic web approach stays in a high level compared to digital libraries including Google Scholar.

VIII. DISCUSSIONS

A path index based keywords to SPARQL query transformation mechanism which aims at RDF data source federations is presented in this paper. A keyword index along with ontology alignment based vocabulary level heterogeneity resolution approach was suggested to identify matching elements for user keywords. A Path Index based approach was used for identifying suitable sub-graphs for connecting keyword elements. Real time graph traversal is one of the drawbacks of existing keyword query processing approaches as they extremely effect the performance as RDF data sources contains thousands to billions of nodes. This approach has totally eliminated real time graph traversal for query generation by generating an index for graph data in the pre-processing stage. It showed a significant performance gain over existing query generation approaches. Promising level of results were achieved from the quality evaluation of this approach and it was proved that federations are capable of giving more complete results for a user query than just querying from a single data source. This research also emphasized that Semantic Web related keyword query processing approaches can give more relevant results for user queries than traditional keyword matching.

This research can be further extended by including capabilities for handling more relaxed format queries on this approach. An approach which can accurately decide the connecting elements for the extracted paths from the Path Index in order to generate sub-graphs can be used to further enhance the accuracy of this approach. A mechanism which can decide whether there is actually a sub-graph existing for a given set of keywords before generating the SPARQL query is needed to be merged with this approach in the future. Also this research can be further extended to generate SPARQL queries for Linked Data sources by exploiting the Path Index. Map-reduce based mechanisms can be used to enhance this approach to function on distributed environments which will improve the scalability of this approach.

[12] https://scholar.google.com/

ACKNOWLEDGMENT

We would like to thank Sarasi Lalithasena, PhD candidate of Kno.e.sis Research Center, Wright State University, USA, for her expert advice and encouragement throughout this research.

REFERENCES

[1] Hristidis L. G. V. and Papakonstantinou Y. (2003). Efficient ir-style keyword search over relational databases. *VLDB*, pp. 850–861.

[2] Hwang V. H. H. and Papakonstantinou Y. (2006). Objectrank: a system for authority based search on database. *SIGMOD Conference*, pp. 796–798.

[3] Liu W. M. F., Yu C. T., and Chowdhury A. (2006). Effective keyword search in relational databases. *SIGMOD Conference*, pp. 563–574.

[4] Kimelfeld B. and Sagiv Y. (2006). Finding and approximating top-k answers in keyword proximity search. *PODS*, pp. 173–182.

[5] Wang H. He, H., Yang J., and Yu P.S. (2007). BLINKS : Ranked Keyword Searches on Graphs. *Proceedings of the 2007 SIGMOD International Conference on Management of Data*, ACM.

[6] Kacholia S. C. S. S. R. D. V., Pandit S., and Karambelkar H., (2005). Bidirectional expansion for keyword search on graph databases. *VLDB*, pp. 505–516.

[7] Cappellari P., De Virgilio R., Maccioni A., and Roantree M. (2011). A Path-Oriented RDF Index for Keyword Search Query Processing. *DEXA*, pp. 366–380.

[8] Tran S. R. T., Wang H., and Cimiano P. (2009). Top-k exploration of query candidates for efficient keyword search on graph-shaped (rdf) data. *ICDE,IEEE.*

[9] Bhalotia C. N. S. C. G., Hulgeri A., and Sudarshan S. (2002). Keyword searching and browsing in databases using banks. *ICDE*, pp. 431–440.

[10] Tran T. and Zhang L. (2013). Keyword Query Routing. *IEEE Transactions on Knowledge and Data Engineering*, 1(2).

[11] Freitas A., Curry E., Oliveira J.G., and O'Riain S. (2011). a Distributional Structured Semantic Space for Querying Rdf Graph Data. *International Journal of SemanticComputing*, 05: 433–462.

[12] David F. S. J., Euzenat J., and dos Santos C. (2011). The Alignment API 4.0. *Semantic web journal*, 2(1): 3–10.

[13] Jain A. K. P., Hitzler P., and Yeh P. (2010). Ontology alignment for linked open data. *The Semantic Web ISWC 2010*, pp. 402–417, Springer Berlin Heidelberg.

[14] Li Y. J., Tang J. and Luo Q. (2009). RiMOM: A dynamic multistrategy ontology alignment framework. *Knowledge and Data Engineering, IEEE Transactions*, 21: 1218–1232.

[15] Tran T., Wang H., and Haase P. (2009). Hermes : Data Web search on a pay-as-you-go integration infrastructure. *Web Semantics: Science, Services and Agents on theWorld Wide Web*, 7(3):189–203.

[16] Sheth A.P. and Larson J.A. (1990). Federated Database Systems for Managing Distributed , Heterogeneous , and Autonomous Databases. *ACM Computing Surveys (CSUR)*, 22(3): 183–236.

[17] Makris K., Gioldasis N., and Bikakis N. (2010). Ontology Mapping and SPARQL Rewriting for Querying Federated RDF Data Sources (Short Paper). *On the Move toMeaningful Internet Systems, OTM 2010*, pp. 1108–1117, Springer Berlin Heidelberg.

[18] Correndo G., Salvadores M., Millard I., Glaser H., and Shadbolt N. (2010). SPARQL Query Rewriting for Implementing Data Integration over Linked Data. *Proceedings of the 2010 EDBT/ICDT Workshops*, p. 4, ACM.

[19] Makris K., Bikakis N., Gioldasis N., and Christodoulakis S. (2012). SPARQL-RW : Transparent Query Access over Mapped RDF Data Sources. *Proceedings of the 15th International Conference on Extending Database Technology*, no. c, pp. 610–613, ACM.

[20] Gunaratna S. K. and Sheth A. (2014). Alignment and dataset identification of linked data in Semantic Web. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 4: 139–151.

[21] David F. J. and Briand H. (2006). Matching directories and OWL ontologies with AROMA. *Proceedings of the 15 th ACM international conference on Information and knowledge management*, pp. 830–831, ACM.

[22] Wikipedia, Breadth-first search," 2002. [online], http://en.wikipedia.org/wiki/Breadth-first_search (Accessed: 21 June 2014).

[23] Wikipedia, "Depth-first search," 2002. [online], http://en.wikipedia.org/wiki/Depth-first_search (Accessed: 21 June 2014).

[24] Florescu D. K. D. and Manolescu I. (2000). Integrating keyword search into xml query processing. *Computer Networks*, 33(1-6): 119–135.

[25] Guo C. B. L., Shao F. and Shanmugasundaram J. (2003). Xrank: Ranked keyword search over xml documents. *SIGMOD Conference*, pp. 16–27.

[26] Yu B., Li G., Sollins K.R., and Tung A.K.H. (2007). Effective Keyword-based Selection of Relational Databases. *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pp. 139–150, ACM.

[27] Vu Q.H., Ooi B.C., Papadias D., and Tung A.K.H. (2008). A Graph Method for Keyword based Selection of the top-K Databases. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 915–926, ACM.

[28] Unger C., Bühmann L., Lehmann J., Ngonga Ngomo A.C., Gerber D., and Cimiano P. (2012). Template-based Question Answering over RDF Data. *Proceedings of the 21st international conference on World Wide Web*, vol. ACM, pp. 639–648.