RFSgIndex: Frequent Subgraph Index for Subgraph Matching in RDF Data

Khin Myat Kyu, Kay Thi Yar, Kyawt Kyawt San
University of Information Technology, Yangon, Myanmar
khinmyatkyu@uit.edu.mm, kaythiyar@uit.edu.mm, kyawtkyawtsan@uit.edu.mm

Abstract

Graphs are used to represent the relationships between objects in various domains such as bioinformatics, social networks, and Web exploration. With the rapid increase of RDF data, efficient graph data management technique is needed to store and retrieve these data. Indexing is an effective technique to reduce data searching space and retrieve them as fast as possible. In this paper, RDF frequent subgraph based approach (RFSgIndex) is proposed to find all the matches of a query graph (a SPARQL query) in a given single large graph (RDF data graph). Firstly, frequent sugraphs are extracted from the RDF data graph using Frequent Subgraph Mining algorithm. Given the query graph, the system finds all the possible occurrences of the query graph by using RFSgIndex. The proposed approach will reduce data searching space and speed up the query response time.

Keywords- RDF data, SPARQL query, graph indexing, frequent subgraph mining, subgraph isomorphism

1. Introduction

The Resource Description Framework (RDF) is a standard data model of the Semantic Web, and SPARQL was recommended by W3C as the standard query language to access RDF data. RDF is a flexible, schema-free and graph-structural data model. Today, the size of RDF data is very large. In order to process large-scale RDF data, many RDF systems have been proposed [3], i.e., Jena, Sesame, SW-store, RDF-3X, etc. They store RDF data in relational tables and process SPARQL queries using relational operators, such as scan and join operators. We call them as relation-based RDF stores, because they use the relational model.

The main problem of relation-based RDF stores is that they need too many join operations for processing SPARQL queries, especially for complex graph patterns. Thus, to address this problem, many techniques have been proposed, i.e., the clustered property table [1], vertical partitioning [1], multiple indexing [2]. These techniques have been focused on the optimized storage layout, indexing methods and efficient join processing. However, the graph-structural data model of the RDF and the graph pattern matching nature of SPARQL queries have significant challenges for efficient processing of SPARQL queries over large-scale RDF data.

In real life applications, we need not only to deal with large database graphs, but also to find all the matches of the query graph. For example, web data networks (social networks, and citation networks, etc.) are often much larger than the graphs used in previous indexing methods. A single RDF data graph may contain millions of vertices. A user may want to find all the occurrences of a particular pattern (subgraph), e.g., name of professors and lectures got their PhD/Master's degrees from universities which are located in London. In different occurrences of the pattern, the name of professors/ lecturers involved may be different since many universities may share the same location. As a result, all occurrences of a particular pattern need to be retrieved. Therefore, we want to solve the following problem for the necessity of real life research—how to find all the matches of a query graph in a large database graph, i.e., finding all subgraphs in the database graph that are isomorphic to the query graph.

Subgraph isomorphism test is believed to be an NP-complete problem and indexing the large graph is practically infeasible. In fact, indexing the large graph is already very difficult. According to our knowledge, many previous methods only apply to graphs of tens of vertices. In addition, most of biological networks, e.g., protein interaction networks, are of thousands of vertices. Unlike to the characteristics of the underlying datasets and the problem definition, algorithms developed for the database of multiple graphs setting may not be used to solve this single graph database problem. Thus, it is necessary to develop a novel solution. When the database is composed of a number of graphs, many of the predominant methods are frequent-substructure based. Researchers preprocess the database and adopt a filter-and-verification process to speed up the subgraph search. False positives are removed by a given pruning strategy. Then, a subgraph isomorphism algorithm is performed on each of the remaining candidates to obtain the final results. These methods have proven to be effective and efficient. However, since we have only a single database graph, the old definition of "frequency" cannot apply. The difficulty of defining "frequency" over a single database graph has been addressed by [8, 9]. Furthermore, since the database graph is much larger than the database graphs for which the frequent subgraph mining tools designed, those tools

may not work at all. The change of the problem setting requires us to define frequency from a new perspective, and at the same time segment the database graph in a meaningful way.

The main difficulty of indexing a single graph with thousands vertices and edges lies in the fact that the subgraph isomorphism is an NP-complete problem. In the traditional database indexing research, the data set size is very large. Thus the goal is to optimize the disk access time. However, in the graph indexing problem setting, the raw graph is not very large, e.g., in the range of megabytes. The computation time to find all occurrences of a subgraph in a graph database is very long. There exist two extreme solutions: (1) Store and index all possible subgraphs of a graph. This is not practically feasible due to the exponential number of possible subgraphs for a graph of thousands edges and nodes, which may require terabyte of storage. (2) Only store the raw database graph. Since the size of the raw database graph is small, it can be easily fit in the main memory. However, the query (matching) time will be very long due to the NP-hard complexity. As a result, we need to identify a solution which lies somewhere between these two extremes, that only utilizes an index structure of a reasonable size and can provide efficient query time.

The remainder of this paper is organized as follows: related work is presented in Section 2. Section 3 defines the preliminary concepts. Section 4 present the proposed approach and describe the algorithm used to build the RFSgIndex. Expected results are presented in Section 5, and the final conclusions are drawn in Section 6.

2. Related Work

Graphs are used to model complex data objects in the real world, e.g., chemical compounds, biological networks, images, social networks and semantic web. Due to its wide usage, it is important to organize, access, and analyze graph data efficiently. As a result, graph database research has attracted a large amount of attention from the database and data mining communities, such as subgraph search in a database of multiple graphs [11, 12, 16, 17, 21], approximate subgraph matching [10, 13, 18], frequent subgraph mining [14, 15, 19, 22], and correlation subgraph query [20].

Among many graph-based applications, it is quite important to retrieve those database graphs containing the query graph efficiently. This is called a subgraph search problem and is closely related to work in this paper. To speed up the subgraph search, researchers preprocess the database and adopt a filter-and-verification framework. First, false positives are removed by a given pruning strategy. Then, a subgraph isomorphism algorithm is performed on each of the remaining candidates to obtain the final results.

Many pruning strategies have been proposed, which can be divided into two sub-categories. The first sub-category is the frequent discriminate substructure based filtering. The approaches in this sub-category apply data mining techniques to extract some discriminating substructures, then build inverted index for each feature.

Query graph q is denoted as a set of features, the pruning power of which is always dependent on the set of selected features. With the inverted indexes, we can find the complete set of candidates.

Many algorithms have been proposed to improve the effectiveness of the selected features, such as gIndex [16], TreePi [21], FG-Index [17] and Tree+δ [11]. In gIndex, the authors propose a discriminative ratio for features. Only frequent and discriminative subgraphs are chosen as indexed features.

In TreePi, due to the manipulation efficiency of trees, frequent and discriminate subtrees are chosen as feature set. The frequent subgraphs and edges are used as indexed features in FG-Index. In Tree+δ, the authors use frequent trees and a small number of discriminative subgraphs as indexed features.

The second sub-category is the path, vertex, and neighborhood substructures based filtering, in which no data mining based feature selection is necessary. There are several representative algorithms.

In GraphGrep [12], the authors propose to use all paths up to maxL length as index features. Similarly, GraphGrep also builds inverted index for each path. In Closure-Tree [10], a pseudo subgraph isomorphism test is performed by checking the existence of a semi-perfect match from vertices in the query graph to vertices in a data graph (or graph closure).

In TALE [18], an approximate matching method was proposed for complex query graphs based on neighborhood units. In [13], the authors introduced a pattern matching method based on a combination of techniques: use of neighborhood subgraphs and profiles, joint reduction of the search space, and optimization of the search order.

Since paths, vertices and neighborhood units are less discriminative than the frequent substructures, these algorithms may have less pruning power but better manipulation efficiency.

Most of these techniques only apply to a database of multiple small or medium sized graphs. These databases are large in the sense that they contain many graphs. Many of these methods care more about whether any database graph contains the query graph or not, instead of finding all the matches of the query graph in a given database graph.

3. Preliminaries

In this section, we present the formal data model of the RDF and SPARQL. We assume the existence of three

pairwise disjoint sets: a set of uniform resource identifiers (URIs) U, a set of literals L, and a set of variables VAR. We assume that blank nodes have their local URIs and treat them same as the resources. Variable symbols start with ‘‘?’’ to distinguish them from URIs and literals. An RDF data set is a collection of statements in the form of subject (s), predicate (p), object (o). A statement $t \in U \times U \times (U \cup L)$ (without variables) is called an RDF triple, and a statement $tp \in (U \cup VAR) \times U \times (U \cup L \cup VAR)$ (triple with variables) is called a triple pattern. It should be noted that in our model the joins that have predicate variables are not considered, because this join type is rarely used.

Definition 1. (RDF graph). We define an RDF graph for the RDF database D as $G_D = (V_D, E_D, L_D)$, where V_D is a set of vertices corresponding to the subjects and objects of all triples in D ($V_D \subseteq (U \cup L)$), E_D is a set of directed edges corresponding to all triples that are from the subjects to the objects, and L_D is an edge-label mapping, $L_D : E_D \rightarrow P_D$, such that $t(s, p, o) \in D$, $L_D(s, o) = p$.

Definition 2. (Query graph) A query graph for a SPARQL query Q is defined as $G_Q = (V_Q , E_Q , L_Q)$, where V_Q is a set of vertices corresponding to the subjects and objects of all triple patterns in Q ($V_Q \subseteq (U \cup L \cup VAR)$), E_Q is a set of directed edges corresponding to all triples that are from the subjects to the objects, and L_Q is an edge-label mapping, $L_Q : E_Q \rightarrow P_D$, such that $tp(s, p, o) \in Q$, $L_Q (s, o) = p$.

Definition 3. (Frequent graph) A graph is assumed as a frequent graph if its support is larger than the minimum threshold defined by the user.

Definition 4. (Subgraph Isomorphism) Given two graphs, $g = (V, E, L)$ and $g' (V', E' , L')$, a subgraph isomorphism from g to g' is an injective function $f: V \rightarrow V'$, such that $\forall(u, v) \in E$, $(f(u), f(v)) \in E'$, $L(u) = L'(f(u))$, $L(v) = L' (f(v))$, and $L(u, v) = L'(f(u), f(v))$.

A graph g is called a subgraph of another graph g' (or g' is a supergraph of g), denoted as $g \subseteq g'$ (or $g' \supseteq g$), if there exists a subgraph isomorphism from g to g'.

4. Proposed Approach

In our proposed approach, there are two main phases: RFSgIndex construction and subgraph (SPARQL) query processing.

4.1. RFSgIndex Construction

We build RFSgIndex using the frequent subgraph mining algorithm, GRAMI, which was originally proposed for use in a single large graph. In this section, we present algorithms **FrequentSubgraphMining** and **SubgraphExtension** which are used to generate frequent subgraphs from the given RDF data graph.

Algorithm 1 FrequentSubgraphMining

Input: An RDF database D and minSup

Output: All subgraphs S such that sup (S) ≥ minSup

1. RFSgIndex ← ø
2. Let frequentEdges be the set of all frequent edges of D
3. for each e ∈ frequentEdges do
4. RFSgIndex ← RFSgIndex ∪ SubgraphExtension(e, D, minSup, frequentEdges)
5. remove e from frequentEdges
6. return RFSgIndex

Algorithm 2 SubgraphExtension (S, D, minSup, frequentEdges)

1. result ← S, candidateSet ← ø
2. for each e in frequentEdges and nodes of S
3. if e can be used to extend then
4. Let ext be the extension of S with e
5. if ext is not already generated then candidateSet ← candidateSet ∪ ext
6. for each c ∈ candidateSet do
7. if sup (c) ≥ minSup then
8. result ← result ∪ c
9. return result

FrequentSubgraphMining firstly identifies set of frequentEdges that contain all frequent edges (i.e., with support greater or equal to minSup). For each frequent edge, **SubgraphExtension** is executed. This algorithm takes as input a subgraph S and tries to extend it with the *e* of frequentEdges (Lines 2-5). All possible extensions are stored in candidateSet. The **SugraphExtension** (Line 5) algorithm checks all extensions which are already generated or not, which are extended from e. Then, Line 6-8 eliminates the members of candidateSet that less than minSup. We assume that their extensions are infrequent. Finally, **SubgraphExtension** is recursively executed to further extend the frequent subgraphs. In this way, all frequent subgraphs are extracted from the RDF data. After all the frequent subgraphs are obtained, the system store them into RDF-3X – open source relation-based RDF store.

Journal of Information Technology Convergence and Services, Vol. 2, No. 2, April 2012, p. 23.

[6] F. Abiri, M. Kahani, and F. Zarinkalam, "An Entity Based RDF Indexing Schema using Hadoop and HBase", 4th International Conference on Computer and Knowledge Engineering (ICCKE), Oct 2014, pp. 68-73.

[7] K. Kim, B. Moon, and H. J. Kim, "RG-index: An RDF graph index for efficient SPARQL query processing", Expert Systems with Applications, Vol. 41, August 2014, pp. 4596 – 4607.

[8] B. Bringmann, S. Nijssen, "What Is Frequent in a Single Graph?", Advances in Knowledge Discovery and Data Mining, 2008, pp. 858-863.

[9] M. Fiedler, C. Borgelt, "Subgraph Support in a Single Large Graph", IEEE 7th International Conference on Data Mining (ICDM) Workshops, Oct 2007, pp. 399-404.

[10] H. He, A.K. Singh, "Closure-Tree: an index structure for graph queries", Proceedings of the 22nd International Conference on Data Engineering (ICDE), April 2006, p. 38.

[11] P. Zhao, J.X. Yu, and P.S. Yu, "Graph indexing: tree + delta <= graph", Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB), Sep 2007, pp. 938-949.

[12] R. Giugno, D. Shasha, "GraphGrep: A Fast and Universal Method for Querying Graphs", Proceedings of 16th International Conference on Pattern Recognition (ICPR), Vol. 2, 2002, pp. 112-115.

[13] H. He, A.K. Singh, "Graphs-at-a-time: Query Language and Access Methods for Graph Databases", Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, June 2008, pp. 405-418.

[14] M. Kuramochi, G. Karypis, "Frequent subgraph discovery", Proceedings IEEE International Conference on Data Mining (ICDM 2001), 2001, pp. 313-320.

[15] X.Yan, J. Han, "gSpan: graph-based substructure pattern mining", Proceedings IEEE International Conference on Data Mining (ICDM 2003), 2003, pp. 721-724.

[16] X. Yan, P.S. Yu, and J. Han, "Graph indexing, a frequent structure-based approach", Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data, June 2004, pp. 335-346.

[17] J. Cheng, Y. Ke, W. Ng, and A. Lu, "FG-Index: Towards verification-free query processing on graph databases", Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, 11 June 2007, pp. 857-872.

[18] Y. Tian, J.M. Patel, "TALE: A Tool for Approximate Large Graph Matching", IEEE 24th International Conference on Data Engineering, 7 April 2008, pp. 963-972.

[19] M. Koyuturk, A. Grama, and W.Szpankowski. "An efficient algorithm for detecting frequent subgraphs in biological networks", Bioinformatics, Vol. 20, 4 August 2004, pp. i200-i207.

[20] Y. Ke, J. Cheng, and W. Ng, "Correlation search in graph databases", Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 12 August 2007, pp. 390-399.

[21] S. Zhang, M. Hu, and J. Yang. "Treepi: A novel graph indexing method", IEEE 23rd International Conference on Data Engineering (ICDE 2007), 15 April 2007, pp. 966-975.

[22] A. Inokuchi, T. Washio, and H. Motoda, "An apriori-based algorithm for mining frequent substructures from graph data", Principles of Data Mining and Knowledge Discovery, 2000, pp. 13-23.

[23] Y. Guo, Z. Pan, and J. Heflin, "LUBM: A benchmark for OWL knowledge base systems", Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 3, 31 Oct 2005, pp. 158-182.

[24] J. Hoffart, F.M. Suchanek, K. Berberich, and G. Weikum, "YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia", Artificial Intelligence, 2012, pp. 28-61.

[25] M. Schmidt, T. Hornung, G. Lausen, and C. Pinkel, "SP2Bench: A SPARQL performance benchmark", Proceedings of the 25th International Conference on Data Engineering (ICDE), 29 March 2009, pp. 222-233.

[26] M. Elseidy, E. Abdelhamid, S. Skiadopoulos, and P. Kalnis, "Grami: Frequent subgraph and pattern mining in a single large graph", Proceedings of the VLDB Endowment, Vol. 7, No. 7, 1 March 2014, pp.517-528.

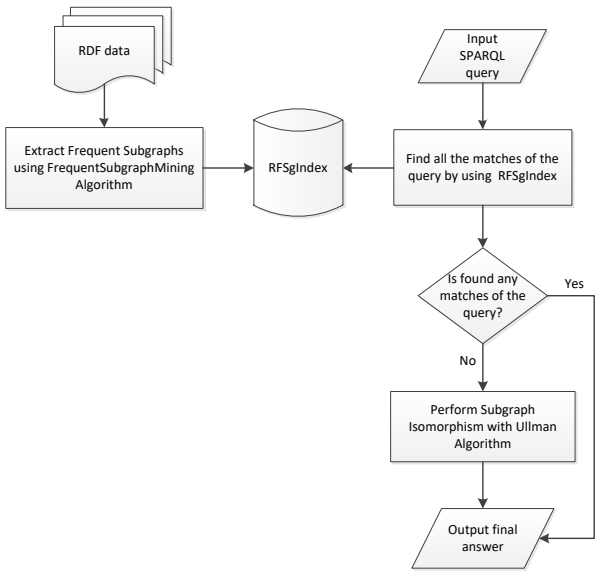

Figure 1. System Flow of Proposed System

4.2. SPARQL query processing

When a SPARQL query is given to the query processor, the query processor finds all the possible matches of the query by using RFSgIndex. If any matches of the query are found in the RFSgIndex, the exact answers of the query are returned. If not found, the query processor have to perform subgraph isomorphism to obtain the final results. We use Ullman algorithm, state-of-the-art subgraph isomorphism algorithm, to find the subgraphs that are isomorphic to the query graph. Since an infrequent graph means the graph occurs in only a small number of graphs in the database, the time of subgraph isomorphism tests will be small.

5. Expected Result and Evaluation

Finding all occurrences of a subgraph in a graph database is a time consuming as the size of the database is very large. At that case, we need to reduce data search space to get efficient query processing. So, this paper proposes the RDF frequent subgraph index (RFSgIndex) for subgraph matching in the RDF data. If the query that is input to the system is found in the RFSgIndex, the proposed approach will reduce data searching space and query execution time. But, even if the query is not found in the RDFSgIndex, the time for processing subgraph isomorphism of the query graph is too small because infrequent graphs are rare in the RDF data and will not be queried frequently. So, it does not degrade the overall performance for most queries.

A comprehensive performance study will be conducted using three datasets: Lehigh University Benchmark (LUBM) [23], Yet Another Great Ontology 2 (YAGO2) [24], and SPARQL Performance Benchmark (SP2B) [25]. LUBM is a benchmark dataset whose domain is the university, YAGO2 is a knowledgebase derived from Wikipedia, WordNet, and GeoNames, and SP2B is a benchmark that simulates the DBLP scenario.

The proposed approach could improve the query processing time by indexing only frequent subgrpahs in the RDF data. In our ongoing research, the proposed approach will be verified in terms of index construction time, size of indexes, and query execution time over the three datasets.

6. Conclusion and Future Work

There are various challenges and issues in graph data management. Many different approaches have advantages and disadvantages. Some are focused on storage layout, indexing methods and efficient join processing. In this paper, frequent subgraph indexing (RFSgIndex) is proposed for efficient subgraph matching in RDF data. Indexing only frequent subgraph can get better query processing than indexing all possible subgraph patterns. The proposed method will provide smaller search space and faster query processing.

We will adopt the DFScode canonical form as in gSpan [15] to check extensions which are extended from *e* are already generated or not (Line 5 of SubgraphExtension algorithm). This will be considered in our future work.

7. References

[1] M.T. Ozsu, "A survey of RDF data management systems", Frontiers of Computer Science, Vol. 10, No. 3, June 2016, pp. 418-432.

[2] S. Sakr, G. AI-Naymat. "Graph indexing and querying: a review", International Journal of Web Information Systems, Vol. 6, No. 2, June 2010, pp. 101-120.

[3] Y. Luo, F. Picalausa, G.H. Fletcher, J. Hidders, and S. Vansummeren, "Storing and indexing massive RDF datasets", In Semantic Search over the Web, Springer Berlin Heidelberg, 2012, pp. 31-60.

[4] X. Wang, S. Wang, P. Du, and Z. Feng, "CHex: An Efficient RDF Storage and Indexing Scheme for Column-Oriented Databases", International Journal of Modern Education and Computer Science, Vol. 3, No. 3, June 2011, p. 55.

[5] K. Lakshmi, T. Meyyappan, "A comparative study of frequent subgraph mining algorithms", International