Graph Compression Strategies for Instance-Focused Semantic Mining

Xiaowei Jiang[1], Xiang Zhang[2], Feifei Gao[1], Chunan Pu[1], and Peng Wang[2]

[1] College of Software Engineering, Southeast University, Nanjing, China
{xiaowei,ffgao,chunan}@seu.edu.cn

[2] School of Computer Science and Engineering, Southeast University, Nanjing, China
{x.zhang,pwang}@seu.edu.cn

Abstract. Semantic mining is a research area that sprung up in the last decade. With the explosively growth of Linked Data, instance-focused Semantic Mining technologies now face the challenge of mining efficiency. In our observation, graph compression strategies can effectively reduce the redundant or dependent structures in Linked Data, thus can help to improve mining efficiency. In this paper, we first describe Typed Object Graph as a generic data model for instance-focused Semantic Mining; and then we propose two graph compression strategies for Linked Data: Equivalent Compression and Dependent Compression, each of which is demonstrated in specific mining scenarios. Experiments on real Linked Data show that graph compression strategies in Semantic Mining is feasible and effective for reducing the volume of Linked Data to improve mining efficiency.

Keywords: Semantic Mining, graph compression, linked data.

1 Introduction

Semantic Mining is a research area that sprung up in the last decade. It combines the Semantic Web with data mining, adapting various mining techniques to discover useful information in semantic data. In these years, Semantic Mining has undergone a transition from ontology-driven mining to instance-focused mining, from text mining to graph mining. In [1], for instance, Semantic Mining can be used to classify web documents based on text analysis. While introduced in [2], Semantic Mining techniques are used to discover frequent patterns and semantic associations, basing on an analysis on graph structure.

Lots of graph-based Semantic Mining algorithms have been proposed in various mining scenarios. Some typical scenarios include: analyzing semantic relationships between concepts or instances defined in ontologies, discovering patterns in RDF graphs, importance or popularity assessment of instances in RDF graphs, etc. Most of these algorithms have reasonable computational efficiency only on small datasets. However, as the explosive growth of Linked Data, these algorithms become more and more difficult to be adapted to large-scale Linked Data, which may consist of billion triples.

A possible solution is to divide large-scale Linked Data into suitable size of partitions prior to the process of mining, such as proposed in [3]. This approach is effective for large-scale mining, but is meanwhile complicated. The cost of integrating mining results in partitions may be high, and the approach is also theoretically prone to a loss of mining results. A more simple and intuitive approach is needed. In our observation, there are usually lots of repeated or inter-dependent structures in Linked Data. Given this characteristic, a structure-based compression can be performed on Linked Data prior to mining process.

In this paper, we propose a framework of two strategies for graph compression to reduce the volume of Linked Data. The first strategy is named Equivalent Compression, which reduces Linked Data by combining repeated structures; the second strategy is named Dependent Compression, which reduces Linked Data by contracting dependent structures. After compression, some graph structures in Linked Data will be combined or contracted into the inner structure of a special instance, which is called a "hypernode" in this paper, and the original graph will be consequently transformed into a relatively smaller "hypergraph". Our work is applicable to instance-focused Semantic Mining tasks, which usually discover instance-related information in Linked Data. Furthermore, different strategies of compression are applicable to different mining scenarios.

This paper is presented as following: a generic graph model for instance-focused Semantic Mining is firstly introduced in section 2. In section 3, a framework of two compression strategies will be described in detail. Two typical mining scenarios using graph compression are well-discussed in section 4 to convince the practicability of the strategies. Finally, experiments are conducted to make a quantitative analysis on how these compression strategies can reduce the volume of Linked Data to improve the efficiency of Semantic Mining.

2 Graph Model for Instance-Focused Semantic Mining

In the context of this paper, we refer to Semantic Mining as a mining on graph structures in Linked Data. Furthermore, an instance-focused Semantic Mining is a set of special mining tasks, which focus on discovering instance-related information or knowledge in Linked Data, not schema-related. For example, in [4], a Semantic Mining approach was put forward to find associations among semantic objects on the basis of a pattern-growth-based mining algorithm. As the growth of online Linked Data, instance-focused Semantic Mining has attracted lots of research interest.

Type information of instances is usually important for various mining tasks in instance-focused Semantic Mining. Proposed in [2], Typed Object Graph (TOG in short) is an appropriate graph model for generic-purposed Semantic Mining. TOG is derived from RDF graph by attaching type information to each instance. In the model, each instance has a unique identification and a unique type-attribute. For those instances that are defined to have multiple types, a set of rules are defined to determine their unique type-attribute according to the popularity, importance or universality of their types. A definition of TOG is defined in Definition 1. A real example of TOG is shown in Figure 1. In the example, Tim Berners-Lee is connected to other persons or places.

Definition 1. (Typed Object Graph): *Defining quintuple Q as* $\langle s, type(s), p, o, type(o)\rangle$ *where s, p, o represents the subject, predicate and object of an RDF triples. In Q, s and o should denote instances, which means their rdf:type should not be classes or properties. type(s) and type(o) are the unique types of s and o respectively. Typed Object Graph G is a directed and labeled graph formed by a set of quintuple* $G = Q_1, Q_2, \ldots, Q_n$.

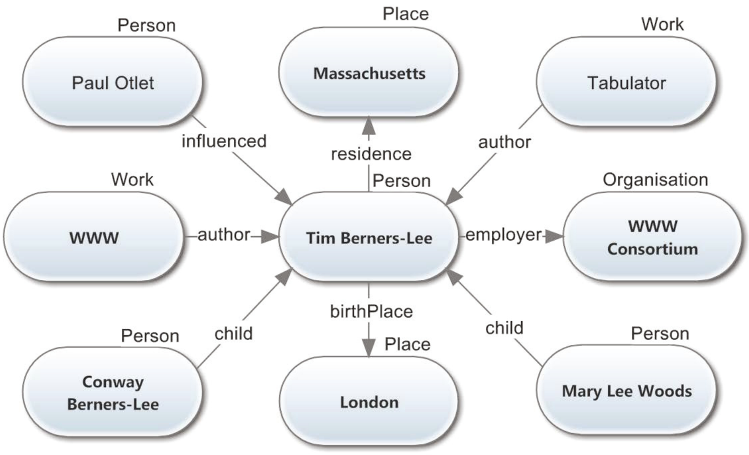

Fig. 1. An Example of TOG

3 Compression Strategies

The basic idea behind of our compression strategies is to use a single instance to represent a compressible graph structure, which connects a group of highly related instances in TOG. The single representative instance is named as "hypernode" in TOG, which can be an actual instance in TOG, or be a virtual instance standing for the compressed graph structure. The compressed instances are usually topologically close in TOG, and the compressed graph structures are usually repeated or inter-dependent structures. Our compression strategies make use of these structures, and reduce a TOG into a smaller graph.

In this section, a framework of two compression strategies will be discussed. The notion of "Family" of an instance is defined to describe the topological context of an instance in TOG. An Equivalent Compression strategy and a Dependent Compression strategy are separately proposed to reduce repeated structures and inter-dependent structures in TOG respectively.

3.1 The Family of an Instance

In Typed Object Graph, the topological context of an instance can be seen as its neighbors and its relations to its neighbors. The neighboring information about an instance is a the local structure around the instance, and usually characterizes its semantics. We use the notion "Family" to stand for the neighboring information.

Definition 2. (Family of an Instance): *The "family" of an instance family(i) is a set F=$\langle j, r\rangle|$ j is one of the direct neighbor of i, r is the relation from i to j or j to i.*

3.2 Equivalent Compression

In traditional theory of graph compression, one of the feasible approach is based on the similarity between two nodes in a graph. A set of highly similar nodes often leads to repeated and compressible graph structures. The similarity between nodes can be measured from different aspects, such as the node's label, type, degree, or shortest distance to their neighbors, and so on. Motivated by this idea, we use the notion of Family to evaluate the similarity between two instances. The fact that two instances have a very same Family will definitely indicates that they possess a very similar topological position in TOG, which means there are repeated structures in the graph. A virtual hypernode will be created to represent this repeated structure.

Equivalent Compression, namely, is to combine instances whose types are the same and who have an Equivalent Relationship in their Families.

Definition 3. (Equivalent Relationship): *Give a TOG $G = \langle V, E\rangle$ derived from an RDF graph, in which V is the set of nodes and E is the set of edges, there is an Equivalent Relationship between instance i_1 and instance i_2: Equivalent(i_1, i_2) iff 1) i_1,i_2 $\in V$; 2) type(i_1)=type(i_2) ; 3) family(i_1)=family(i_2).*

Definition 4. (Equivalent Compression): *Give a TOG G=$\langle V, E\rangle$, Equivalent Compression is a process of transforming G into another graph: ECG(G)=$\langle V', E'\rangle$ where V'=$V_{hyper} \cup V_{simple}$. V_{hyper} is a set of virtual hypernodes, in which each node represents a set of Equivalent instances; V_{simple} is a set of actual nodes in TOG, in which each node has no Equivalent Relationship to any other nodes in V_{simple}.*

As shown in Figure 2, Figure2(a) is a fragment of TOG and Figure2(b) is a corresponding compressed graph. In Figure2(a), both Tom and Mary are Persons, and they both know Jack and Kate. Therefore, Tom and Mary are considered to have Equivalent Relationship and can be combined into a virtual hypernode Tom&Mary. Hypernode is presented in thick line. A created hypernode represents multiple equivalent instances in the original TOG, thus can reduce repeated structure and improve the efficiency of Semantic Mining.

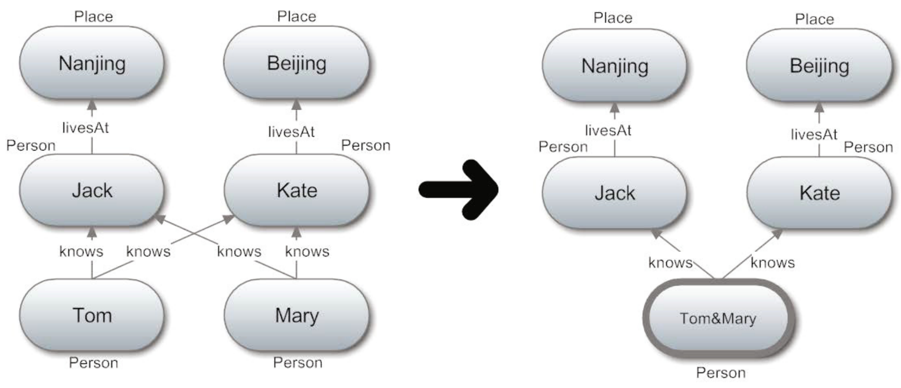

Fig. 2. (a)A subgraph of TOG (b)Corresponding ECG

3.3 Dependent Compression

Even in a dense TOG, there are still a lot of instances whose family has only one member, which means this kind of instances are not popularly referred by other instances, and their semantics are highly dependent on their only neighbor. They are terminal nodes hanging at the edge of a network formed by interlinked instances. In some Semantic Mining tasks, this kind of dependent structures can be compressed to improve the mining efficiency. The local structure of dependent instances can be contracted into the inner structure of its only neighbor. Different with Equivalent Compression, hypernodes are actual instances in Dependent Compression.

Definition 5. (Dependent Relationship): *Give the TOG G=$\langle V, E \rangle$, there is a Dependent Relationship between instance i_1 and i_2: Dependent(i_1,i_2) iff 1) i_1,$i_2 \in V$; 2) i_2 is the only instance in family(i_1) and there is only one edge connecting i_1 and i_2.*

Definition 6. (Dependent Compression): *Give a TOG G=$\langle V, E \rangle$. Dependent Compression is the process of transforming G into another graph: DCG(G)= $\langle V', E' \rangle$, where $V'=V_{hyper} \cup V_{simple}$ .V_{hyper} is a set of hypernodes. Each hypernode contains an actual instance with all instances that have Dependent Relationship to it. V_{simple} is a set of actual instances in TOG, in which there doesn't exist two instance i and j in V_{simple} that satisfy Dependent(i,j) or Dependent(j,i).*

In this strategy, if Dependent(i,j), instance i will be compressed into j, making j a hypernode. The original TOG is compressed in an iterative manner until no dependent instances can be compressed into other instances, and the compression ratio becomes steady.

As shown in Figure 3, the cardinality of family of each instance is first computed. And then, one-neighbor instances ("Tom" and "Mary") are compressed into their only neighbor, which forms the structure on the top of the figure.

The hypernode is presented in thick lines. For lack of space, the inner structures of hypernodes are not presented. The compressed graph can be iteratively compressed, until no Dependent Relationship exists, as the structure on bottom right of the figure, then the compression process comes to the end.

It can be proved that, after the iterative process, the inner structure of hypernodes is usually a tree structure without regards to the direction of quintuples in the tree. Each hypernode in DCG is an actual instance in TOG and is the root of the compressed tree.

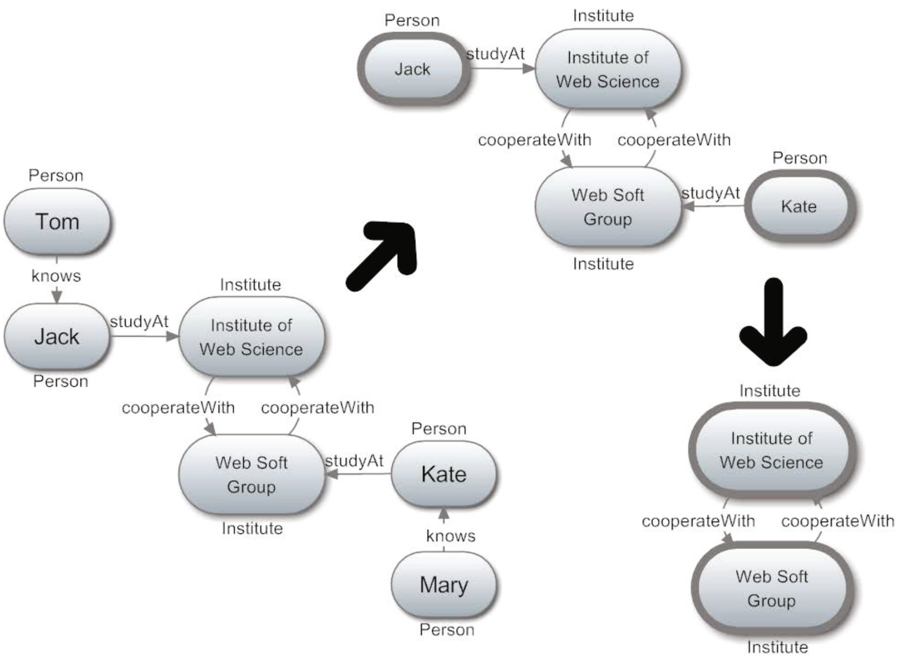

Fig. 3. An Example of Iterative Compression of TOG to Corresponding DCG

4 Application Scenarios

Our strategies can be applied to instance-focused Semantic Mining to achieve a better mining efficiency. In this section, two typical Semantic Mining tasks will be fully discussed as the application scenarios of our strategies. These scenarios are about the mining of semantic associations. Semantic associations are usually defined as a graph structure representing a group relationship among several instances as defined in [4], or a path structure representing a serial relationship between two instances as defined in [5,6]. Since the complexity of mining semantic associations are usually exponential, they will be inefficient especially in large-scale Linked Data. The mining process often consumes a large amount of time.

The applicability of each compression strategy depends on whether the mining task will make use of the inner structure of hypernodes. For example, since tree structures will be compressed in Dependent Compression, it will not be applicable to the mining tasks that rely on counting the subgraph frequency, such as discovering graph-structured semantic associations in the first application scenario. But Dependent Compression is applicable to discovering path-structured semantic associations, because in mining tasks on paths, the tree structure of hypernodes can be utilized to find shortest path between instances. In following scenarios, each scenario is discussed with an applicable compression strategy.

4.1 Mining Semantic Associations as Subgraphs

As defined in [4], frequently occurred subgraphs in Linked Data will be discovered to discover semantic associations as graphs connecting a set of instances. Generally, frequent pattern mining algorithms, such as gSpan [7] or Closegraph [8] will be used in this scenario. Subgraphs in Linked Data will be enumerated and for each subgraph, a "support" value, which defines the frequency that a subgraph appears in the Linked Data, will be computed to estimate whether the subgraph is frequent enough to represent a typical semantic association.

In the process of mining semantic associations, a minimum DFS code for each subgraph in Linked Data is defined to canonically identify a link pattern by a DFS traverse path. A rightmost extension is also defined to produce candidates based on mined link patterns. A minimal link patterns are first discovered (with 0-edges), and the mining process is called recursively to make a rightmost extension on mined link patterns so that their frequent descendants with more edges are found until their support is lower than a given min-sup or its DFS code is not minimum any more. All mined patterns comprise a lexicographic search tree. More details of gSpan and its expansion can be found in [7].

The counting of support value of subgraphs is significant in mining semantic associations, which affects the correctness and efficiency of mining. Equivalent Compression is applicable for this scenario because the support value of a subgraph is easy to compute after compression. In Equivalent Compression, the inner structure of a hypernode is a set of equivalent instances, rather than a tree structure. Therefore, it is not necessary to decompress hypernodes to count the support of subgraph. The only information needed to count the support value in ECG is the cardinality of compressed instances in hypernodes.

When semantic associations are discovered using gSpan or CloseGraph, there is no difference when counting the support value of subgraphs containing only simple nodes after compression. But a modification on gSpan or CloseGraph is needed in the case of counting the support value of subgraphs containing hypernodes: the contribution of the occurrence of a hypernode to the support value is not 1, but the cardinality of the set of compressed instances (saying k), because the hypernode represents k equivalent instances, in which each instance contributes one occurrence to the support value.

4.2 Mining Semantic Associations as Paths

In some specific Semantic Web applications, such as semantic social network, researchers often focus on discovering relations between two persons, which are usually described as paths between two instances in RDF graph. The closeness of two instances is usually measured by the length of the path. A mining task to discover semantic associations between instances can be transformed into an problem of shortest path discovering.

Dependent Compression can help to improve the mining efficiency of this type of mining tasks. The improvement is based on the idea that shortest path between two nodes in a tree is easier to compute than in a graph. In a graph compressed by Dependent Compression, instances can be classified into three sets - simple instances, hypernodes as the roots of their inner tree structures and other compressed instances in the inner tree structure of hypernodes. S, R and I are used to denote each corresponding set of instances.

Naming the shortest path between instance i and j as $SP(i,j)$. $SP(i,j)$ can be computed in separate cases: 1) $SP(i,j)$ can be computed using the general all-pair-shortest-path algorithm when $i,j \in S \cup R$; 2) In the case that $i \in I$, $j \in S \cup R$, finds the root hypernode of i and names it as k, $k \in R$, then $SP(i,j) = SP(i,k)+SP(k,j)$; 3) In the case that $i,j \in I$, finds the root hypernodes of i and j and names it as k,l respectively, then $SP(i,j) = SP(i,k)+SP(k,l)+SP(l,j)$.

5 Experiment

To validate the effectiveness of two compression strategies, a set of experiments are conducted on five online Linked Data. In this section, we first describe the dataset, and then discuss the experiments and results.

5.1 DataSet

To validate the effectiveness of the compression strategies, five online Linked Data on different topics are selected as our dataset for evaluation. A summary of each dataset is given as following:

(i) **DBpedia**, which is a widely-used Linked Data on structured information extracted from Wikipedia.
(ii) **LinkedMDB**, which is a famous Linked Data on movies.
(iii) **SwetoDblp**, which is an ontology focused on bibliography data of publications from DBLP with additions that include affiliations, universities, and publishers.
(iv) **Jamendo**, which is a large Linked Data of Creative Commons licensed music, based in France. These datasets are diverse in topics, and are rather large in volume, which makes them difficult for Semantic Mining tasks.
(v) **John Peel sessions**, which is published by DBTune.org. It is a music-related repository on the Semantic Web, containing Linked Data of BBC John Peel sessions.

All the datasets have RDF dumps in their corresponding websites, and can be accessed through a portal of W3C DataSetRDFDumps[1] The statistical information of these datasets are given in Table 1.:

Table 1. The Statistical Information of Datasets

Dataset	#instancs	#quintuple	Ave.Degree
DBpedia	1664061	6014163	7.228
LinkedMDB	602796	1210921	4.0177
SwetoDblp	544678	627753	2.305
Jamendo	281468	373494	2.654
JohnPeel	71284	100403	1.408

5.2 Evaluation on Compression

The compression ratio CR is a conventional parameter to measure the performance of compression. It describes the relative volume of data after compression comparing to the volume of data before compression. In graph compression, CR can be defined either by compression ratio on number of edges or by compression ratio on number of nodes. In this paper, we define CR_q and CR_i in Equation 2-3 to represent the compression ratio on quintuples or instances in Linked Data respectively. A low compression ratio indicates that a large proportion of quintuples or instances in TOG can be compressed according to compression strategy.

$$CR_q = \frac{\#quintupleaftercompression}{\#quintuplebeforecompression} \tag{1}$$

$$CR_i = \frac{\#instanceaftercompression}{\#instancebeforecompression} \tag{2}$$

Both Equivalent and Dependent Compression Strategy are performed on each dataset, and the compression ratios are shown in Table 2 and Table 3 respectively. In the tables, $\#q$ and $\#i$ indicate the number of quintuples and instances in TOG. $-q$ and $-i$ indicate the number of compressed number of quintuples and instances in ECG and DCG.

From the results in Table 2, it is observable that $-q$ is normally higher than $-i$, because a compression of a set of equivalent instances usually leads to a compression of a larger set of equivalent quintuples in the families of these instances. In our observation, the graph structures in DBpedia, LinkedMDB and JohnPeel are non-redundant, which means few repeated structures can be discovered by Equivalent Compression. On these datasets, our strategies have limited performance, with both CR_q and CR_i are over or almost over 90%. Especially for JohnPeel, only one pair of equivalent instances can be found. However, Equivalent Compression have sound performance on the other two datasets. For SwetoDblp,

[1] W3C DataSet RDF Dumps: `http://www.w3.org/wiki/DataSetRDFDumps`.

Table 2. Compression Ratio of Equivalent Compression

Dataset	Before Compression		After Compression		$-q$	$-i$	CR_q	CR_i
	$\#q$	$\#i$	$\#q$	$\#i$				
DBpedia	6,014,163	1,664,061	5,877,935	1,555,424	136,228	108,637	0.977	0.935
LinkedMDB	1,210,921	602,796	1,097,710	537,653	113,211	65,143	0.907	0.892
SwetoDblp	627,753	544,678	146,730	64,038	481,023	480,640	0.234	0.118
Jamendo	373,494	281,468	306,328	214,568	67,166	66,900	0.820	0.762
JohnPeel	100,403	71,284	100,401	71,283	2	1	1.000	1.000

Table 3. Compression Ratio of Dependent Compression

Dataset	Before Compression		After Compression		$-q$	$-i$	CR_q	CR_i
	$\#q$	$\#i$	$\#q$	$\#i$				
DBpedia	6,014,163	1,664,061	5,724,074	1,373,972	290,089	290,089	0.952	0.826
LinkedMDB	1,210,921	602,796	1,068,716	460,591	142,205	142,205	0.883	0.764
SwetoDblp	627,753	544,678	115,372	32,297	512,381	512,381	0.184	0.059
Jamendo	373,494	281,468	150,289	58,263	223,205	223,205	0.402	0.207
JohnPeel	100,403	71,284	59,787	30,668	40,616	40,616	0.595	0.430

surprisingly almost 90% of instances with 80% of quintuples are equivalent and can be compressed.

Shown in Table 3, Dependent Compression have a better compression ratio comparing to Equivalent Compression. All the ratios, except CR_q on DBpedia, are less than 90%. Dependent Compression also has the best performance on SwetoDblp, in which CR_i can even reach to 6%. That indicates SwetoDblp consists of too many repeated and meanwhile dependent structures. Another observable fact is that the $-q$ is just the same with $-i$ for each dataset after Dependent Compression. This is because a compression of a dependent instance always leads to a simultaneous compression of the quintuple connecting it to its only neighbor. In other words, the number of instances and the number of quintuples in the inner structures of hypernodes in DCG are always the same.

6 Related Work

The goal of graph compression for graph mining is to reduce the volume of graph and to achieve a reasonable mining efficiency when the graph is large. Some approaches have been proposed for graph compression. In [12], Chen proposed a graph mining algorithm via randomized graph summaries. For each graph in the graph set, a summary is built and the shrunk graph is used for mining, which decreases the embedding enumeration cost. However, after a graph summarization, the algorithm may suffers from a loss of patterns. Thus randomized summaries are generated and the mining process is repeated for multiple rounds for a minimum loss of patterns. The works presented in [13] and [14] are closely related to ours. These paper independently proposed to construct graph summaries of unweighted graphs by grouping similar nodes and edges to supernodes and superedges. The difference between their works and ours lies in two aspects: first, their works are not working on Linked Data, type information is undefined for the nodes in their graph models; second, their works considered the compression of repeated structures, but not considered compression of dependent structures.

Besides these works, some other solutions on graph compression have been proposed. Toivonen proposed in [15] a solution to compress graph by node and edge mergers. Toivonen also introduce another solution in [16], which is also known as graph simplification. Nodes and edges in weighted graphs are grouped to supernodes and superedges. The supernodes and superedges are selected to minimize approximation errors and meanwhile maximize the amount of compression. Both of the works provides approach to approximate compression, and thus are both lossy. They are quite different with our work, which works on non-weighted graph and is lossless in the process of compression and decompression.

7 Conclusion and Future Work

The research on Semantic Mining is now facing the challenge of contradiction between the growing volume of Linked Data and the complexity of mining algorithms. The approach of graph compression can effectively improve the efficiency of instance-focused Semantic Mining by simplifying the graph structure of large-scale Linked Data. In this paper, a set of two strategies are proposed for graph compression on a generic graph model for instance-focused Semantic Mining. Repeated and Dependent structures in TOG are compressed by Equivalent and Dependent Compression respectively. Two typical scenarios are discussed to illustrate the applicability of each strategy. Experiments on real datasets show that our approach is feasible to reducing repeated and dependent structures in TOG, and practically improves mining efficiency in typical application scenarios.

The combination of graph compression and graph partitioning will be considered in our future work. Indicated by experimental results, graph compression strategies have limited performance on very densely-connected TOG. For mining on this kind of TOG, graph partitioning will help to divide TOG into minable blocks. The combination of graph compression and partitioning is expected to provide a complete solution to large-scale Semantic Mining.

Acknowledgements. The work is supported by the NSFC under Grant 61003055, 61003156, and by NSF of Jiangsu Province under Grant BK2011335. We would like to thank Lei Wu and Yu Guo for their valuable suggestions and work on related experiments.

References

1. Svatopluk, F., Ivan, J.: Semantic Mining of Web Documents. In: Proceedings of International Conference on Computer Systems and Technologies, pp. 21–26 (2005)
2. Zhang, X., Zhao, C., Wang, P., Zhou, F.: Mining Link Patterns in Linked Data. In: Gao, H., Lim, L., Wang, W., Li, C., Chen, L. (eds.) WAIM 2012. LNCS, vol. 7418, pp. 83–94. Springer, Heidelberg (2012)
3. Zhao, C.F., Zhang, X., Wang, P.: A Label-based Partitioning Strategy for Mining Link Patterns. In: Proceedings of 7th International Conference on Knowledge, Information and Creativity Support Systems, pp. 203–206 (2012)
4. Jiang, X.W., Zhang, X., Gui, W., Gao, F.F., Wang, P., Zhou, F.B.: Summarizing Semantic Associations Based on Focused Association Graph. In: Proceedings of the 8th International Comference, pp. 564–576 (2012)
5. Anyanwu, K., Sheth, A.: p-Queries: Enabling Querying for Semantic Associations on the Semantic Web. In: Proceedings of the 12th International World Wide Web Conference, pp. 690–699 (2003)
6. Sheth, A., Aleman-Meza, B., Arpina, I.B., et al.: Semantic Association Identification and Knowledge Discovery for National Security Applications. Journal of Database Management 16(1), 33–53 (2005)
7. Yan, X., Han, J.W.: gSpan: Graph-based Substructure Pattern Mining. In: Proceedings of the IEEE International Conference on Data Mining, pp. 721–724 (2002)
8. Yan, X., Han, J.W.: CloseGraph: Mining Closed Frequent Graph Patterns. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 286–295 (2003)
9. Hage, P., Harary, F.: Eccentricity and Centrality in Networks. Social Networks 17, 57–63 (1995)
10. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Technical Report, Stanford University (1998)
11. Kleinberg, J.: Authoritative Sources in a Hyperlinked Environment. In: Proceedings of the 9th ACM SIAM Symposium on Discrete Algorithms, pp. 668–677 (1998)
12. Chen, C., Lin, C.X., Fredrikson, M., Christodorescu, M., Yan, X.F., Han, J.W.: Mining Graph Patterns Efficiently via Randomized Summaries. In: Proceedings of the 35th International Conference on Very Large Data Bases, pp. 742–753 (2009)
13. Navlakha, S., Rastogi, R., Shrivastava, N.: Graph Summarization with Bounded Error. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 419–432 (2008)
14. Tian, Y., Hankins, R., Patel, J.: Efficient Aggregation for Graph Summarization. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 567–580 (2008)
15. Toivonen, H., Zhou, F., Hartikainen, A., Hinkka, A.: Network Compression by Node and Edge Mergers. In: Berthold, M.R. (ed.) Bisociative Knowledge Discovery. LNCS, vol. 7250, pp. 199–217. Springer, Heidelberg (2012)
16. Toivonen, H., Zhou, F., Hartikainen, A., Hinkka, A.: Compression of Weighted Graphs. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 965–973 (2011)