Mining Link Patterns in Linked Data

Xiang Zhang[1], Cuifang Zhao[1], Peng Wang[1], and Fengbo Zhou[2]

[1] School of Computer Science and Engineering, Southeast Univesity, Nanjing, China
{x.zhang,cuifangzhao,pwang}@seu.edu.cn
[2] Focus Technology Co., Ltd., Nanjing, China
zhoufengbo@made-in-china.com

Abstract. As the explosive growth of online linked data, an emerging problem is what and how we can learn from these data. An important knowledge we can obtain is the link patterns among objects, which are helpful for characterizing, analyzing and understanding of linked data. In this paper, we present a novel approach of mining link patterns. A Typed Object Graph is proposed as the data model, and a gSpan-based algorithm is proposed for pattern mining. A type determination policy is introduced in cases of multi-types and a data clustering algorithm is proposed to improve scalability. Time performance and mining results are discussed by experiments.

Keywords: linked data, frequent link pattern, semantic web.

1 Introduction

As the rapid growth of semantic web in this decade, there is an exponential growth in the scale of online linked data. As indicated by W3C wiki of Linked Datasets[1], about 70 publishers open their linked data to public and lots of them contain a scale of billion triples. There is still enormous amount of linked data produced by social communities, companies, and even on the desktop of end-users. All these efforts are now producing an unprecedented huge web of data, bringing the semantic web from vision to practice. We have faced the problem of lack of semantic data, but now, the problem is what and how we can learn from these data.

One important thing to learn is the link patterns. Link patterns are consensus practices characterizing how different types of objects are typically interlinked. For example, in certain linked data, a *Researcher* may *focuses* on a *ResearchArea*, and *publishes* some *Papers* in *Proceedings* of a *Conference*. Besides, he *knows* some *Researchers* in the same *ResearchArea*. Link patterns describe widely-used relations among objects, which may or may not be defined explicitly in ontologies.

Link patterns are critical in several topics of research. First, they can be used to find useful semantic associations between objects by indicating what kind of object links are typical and thus significant [1]; Second, when accessing distributed RDF stores relying on different ontologies within different domains, link patterns are helpful in evaluating the potential contribution of each store to the processing of RDF queries,

[1] http://www.w3.org/wiki/DataSetRDFDumps

H. Gao et al. (Eds.): WAIM 2012, LNCS 7418, pp. 83–94, 2012.

because link patterns characterizes the content of RDF repositories [2]; finally, in producing concise and comprehensible summaries for human understanding of linked data, link patterns can play a role of indicating the most important part of the link data from the point of view of data providers[3].

Although usage mining in semantic web gains a lot of interest in these years [4, 5], mining link patterns is still a topic not well-discussed. This is because the lack of a formal definition of link pattern in linked data, and the complex graph structure of link patterns also limits the scalability and efficiency of mining. Our contributions in this paper are: first, we formally define link pattern by a notion of Typed Object Graph; second, we propose a pattern mining algorithm based on gSpan [6], a clustering algorithm is also proposed to improve the mining scalability and efficiency.

The rest of the paper is organized as following: Typed Object Graph and Link Pattern are defined in Section 2. We explain the policy of type determination in building Typed Object Graph in Section 3. A data clustering algorithm and a gSpan-based pattern mining approach are proposed in section 4. Experiments on time performance and mining results are discussed in Section 5.

2 Preliminary Concepts

Link patterns are frequent and typical styles of how different types of object are interlinked. In order to clearly define the notion of link patterns, we have to introduce a new data model, which embody object types as well as object links in the model. In this section, we name our new data as Typed Object Graph, and then we define Link Pattern based a notion of RDF2Pattern Homomorphism.

2.1 Typed Object Graph

Link patterns cannot be directly mined from RDF graphs. Object types are core elements in link patterns. In RDF graphs, object types are implicit and can only be determined by reasoning according to RDF semantics [7]. A novel graph model is needed for pattern mining, in which object types should be explicitly embodied.

Definition 1 (Link Quintuple): Given an RDF document *d*, and an Triple $t = \langle s,p,o \rangle$ in *d*, an Link Quintuple $q = \langle s,p,o,type(s,d),type(o,d) \rangle$ is an extension of *t*, where *s* and *o* must represent object nodes, *p* must be object property, *type*(*s,d*) and *type*(*o,d*) represent the *rdf*:*type* of *s* and *o* defined in *d* respectively.

Definition 2 (**Typed Object Graph**)**:** A Typed Object Graph g comprises a set of Link Quintuple: $\{q_1, q_2, \dots, q_n\}$, which is extended from an RDF Graph g' comprising a set of RDF Triple: $\{t_1, t_2, \dots, t_n\}$. And $\forall q_i \in g$ and $t_i \in g'$, q_i is extended from t_i.

For simplicity, multiple types are not allowed for an object in a link quintuple. We will introduce a type determine policy in Section 3.2. Shown in Figure 1, a fragment of RDF graph is extracted from a linked data of Semantic Web Dog Food[2], and the derived Typed Object Graph is shown aside (object URIs are omitted for simplicity).

[2] `http://data.semanticweb.org/`

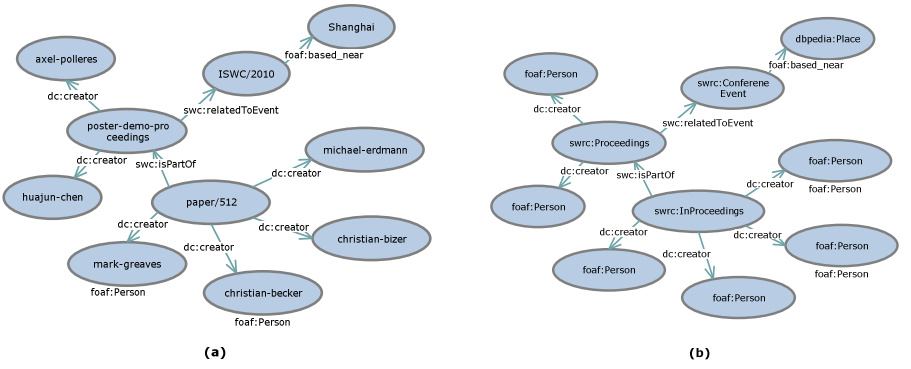

Fig. 1. (a) A fragment of an RDF Graph; (b) A derived Typed Object Graph

2.2 Link Pattern

Definition 3 (RDF2Pattern Homomorphism): Given a subgraph g of a Typed Object Graph derived from RDF document d and a graph p, An RDF2Pattern Homomorphism is an injective function f: $V(g) \rightarrow V(p)$, such that (1) $\forall u \in V(g), type(u,d) = f(\mathrm{u}) \in V(p)$, and (2) $\forall (u,v) \in E(g), (f(u), f(v)) \in E(p)$, in which $E(g)$ is the edge set of g.

Definition 4 (**Link Pattern**)**:** Given an RDF document dataset $D = \{d_1, d_2, \ldots d_n\}$, a Link Pattern p is a graph, satisfying that:

1) $p = \langle V,E \rangle$ is a directed graph, in which each vertex is an user-defined class and each edge is an object property.
2) There exists RDF2Pattern homomorphism in D, and $support(p) \geq min\text{-}sup$. $support(p)$ is the number of RDF documents, in which p is an RDF2Pattern homomorphism of derived Typed Object Graphs; *min-sup* is a predefined minimum support threshold.

3 Type Determination

Before mining link patterns, triples in linked data should be cleaned in advance in order to remove annotations, schema-level definitions, only object links are preserved. There is another issue we have to consider. In the definition of Typed Object Graph, an object must have a single type for the computational complexity of pattern mining. However practically, an object in linked data often be defined to multi-types. For example, in Semantic Web Dog Food, an accepted paper can be defined as an *swc*:*Paper* and meanwhile as a *swrc*:*InProceedings*. It should be eliminated before building Typed Object Graph, since multi-types will bring extra complexity in pattern mining, especially to subgraph homomorphism detection. Given $D = \{d_1, d_2, \ldots, d_n\}$ is the document set, and an object o defined in document d_{i} with multi-types $\{t_1, t_2, \ldots, t_m\}$, a set of heuristic rules are used to determine the single type of o:

Rule 1: If there is no other document in D defining the type of object o, $type(o, d_i)$ will be assigned with t_k, where $1 \leq k \leq m$ and type t_k has the largest set of instances in document d_i.

Rule 2: If there is more than one document defining the type of o in D, $type(o, d_i)$ will be assigned with t_k, where $1 \leq k \leq m$, and comparing to other types, o is defined to t_k most frequently, or saying, with the highest documents frequency.

Rule 3: For a triple $\langle s, p, o \rangle$ in document d, if the type of s or o is not defined explicitly, and the domain or range of p is defined explicitly, then we have: $type(s, d) = domain(p)$ and $type(o, d) = range(p)$.

In Rule 1, the single type of an object is assigned to a local dominated type; while in Rule 2, the single type is assigned to a global dominated type. If the dominance of each type is not obvious, the type determination becomes random.

There is still a case we have to consider, in which the type of an object may be not defined explicitly. In this case, Rule 3 will be applicable, and the domain and range definitions of an ObjectProperty are utilized to determine an objects' type.

Here we only try to determine the type of objects by exploring the domain and range of ObjectProperty. More powerful reasoning ability could be utilized for type determination, such as described in [8]. Here we use a rather lightweight reasoning to make the pattern mining scalable.

4 Mining Link Patterns

Among some frequent pattern mining algorithms, we select gSpan algorithm for mining link patterns in linked data. It is efficient in mining frequent graph patterns in massive data. Before pattern mining, we first cluster related Typed Object Graphs into groups for the sake of scalability. And then pattern mining is performed on each group separately in a divide-and-conquer manner.

4.1 Graph Clustering

Given a whole set of a linked data, the occurrences of a link pattern are often localized in some RDF documents of the same topic, while not in documents of other topics. For example, in Falcons dataset[3], we can find link patterns linking persons and organizations from some FOAF documents, or patterns linking genes markers and chromosomes from other bio2rdf documents (originated from bio2rdf[4]). If we can cluster RDF documents according to the link patterns being concerned, we can perform pattern mining in a divide-and-conquer way.

There have been several works on RDF clustering or classification, such as the RDF metadata clustering [9], instance clustering [10], clustering for snippets generation [11], and ontology classification [12] [13]. Here we introduce a clustering

[3] http://ws.nju.edu.cn/falcons
[4] http://bio2rdf.org/

approach for Typed Object Graphs. The intuition is: if two Typed Object Graphs share a common link pattern, they are affinitive and should be clustered together.

Definition 5 (**Direct Connections between Typed Object Graphs**)**:** Given two Typed Object Graphs: $g = \{q_1, q_2, \ldots, q_n\}$ derived from RDF document d, and $g' = \{q'_1, q'_2, \ldots, q'_m\}$ derived from RDF document d'. g and g' are connected, denoted as $connected(g, g')$, iff $\exists q_i = \langle s_i, p_i, o_i, type(s_i, d), type(o_i, d)\rangle \in g$ and $q'_j = \langle s'_j, p'_j, o'_j, type(s'_j, d'), type(o'_j, d')\rangle \in g'$, satisfying (1) $p_i = p'_j$; (2) $type(s_i, d) = type(s'_j, d')$; (3) $type(o_i, d) = type(o'_j, d')$.

In Definition 5, two Typed Object Graphs are directly connected if and only if they share at least one common link pattern. Given a set of Typed Object Graph $G = \{g_1, g_2, \ldots g_n\}$, it can be divided into a set of disjoint connected clusters through a testing of connectivity. It is easy to prove that we can use each cluster as a standalone dataset for pattern mining (although the support is still calculated globally), because different clusters don't share common link patterns. An example from Falcons dataset is shown in Figure 2, in which five fragments of Typed Object Graphs are clustered into two groups.

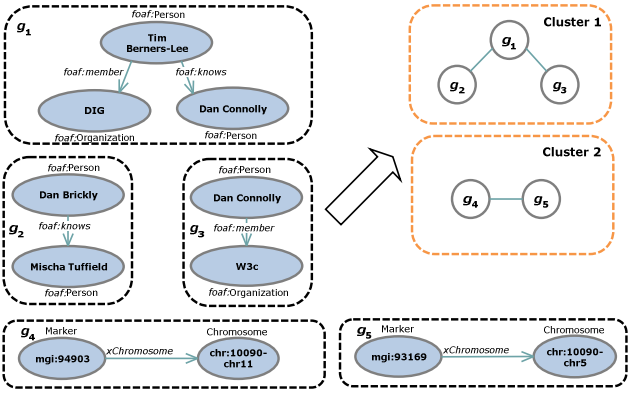

Fig. 2. Four Typed Object Graphs in Falcons are clustered into groups

4.2 gSpan-Based Pattern Mining

Our pattern mining approach follows the idea of pattern-growth-based frequent graph pattern mining algorithms, in which gSpan is a typical and efficient one. Basically, a candidate frequent pattern is produced by extending a mined frequent pattern by adding a new edge.

The kernel ideas in gSpan are the *minimum DFS code* and the *rightmost extension*. The *minimum DFS code* is introduced to canonically identify a pattern by a DFS traverse path; the *rightmost extension* is used producing candidates based on mined patterns. Both ideas can reduce the generation of duplicated candidates. In gSpan, the minimal frequent patterns are firstly discovered (with 0-edges), and gSpan is called recursively to make a

rightmost extension on mined patterns so that their frequent descendants (with 1-edges, 2edges and so on) are found until their support is lower than *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 [14].

The original gSpan algorithm is designed for undirected and simple graphs. However in our scenario, Typed Object Graphs are directed and non-simple graphs. Self-loops (edges join vertexes to themselves) and multiple edges (multiple edges connecting two same vertexes) should be taken into consideration. According to suggestions proposed in CloseGraph [15], we modify gSpan algorithm, especially the DFS coding, to make it adaptable to Typed Object Graphs.

Given an link quintuple $q = \langle s, p, o, type(s,d), type(o,d) \rangle$ in a Typed Object Graph, it is represented in our implementation by a 6-tuple: $\langle i, j, l(i), l(i,j), l(j), \kappa \rangle$, in which i and j are DFS subscripts of vertex s and o; $l(i)$ and $l(j)$ are labeling functions of i and j, which equals to $type(s,d)$ and $type(o,d)$ respectively; $l(i,j)$ is the edge label, which equals to p; and κ represents the direction of the edge between i and j, in which $\kappa = 0$ represents $i \rightarrow j$, and $\kappa = 1$ represents $i \leftarrow j$.

There are two parameters to control the mining results in our implementation. One is *min-sup*, which indicates the minimum threshold of supports that a discovered link pattern can be seen as "frequent"; the other is *max-edge*, which limits the size of link patterns that can be mined in our algorithm.

We will illustrate the mining process through an example, which is shown in Figure 3. Three fragments of Typed Object Graphs are extracted from Semantic Web Dog Food, in which object URIs are omitted. For conciseness, shown in Table 1, we assign with each node and each edge a new label according to their lexicographical order. According to the DFS coding rules in gSpan, the minimum DFS code of each graph is:

g_1: $\langle 0,1,C_2,p_1,C_4,1\rangle \langle 1,2,C_4,p_4,C_5,1\rangle \langle 2,1,C_5,p_5,C_4,1\rangle$
g_2: $\langle 0,1,C_2,p_1,C_4,1\rangle \langle 1,0,C_4,p_2,C_2,0\rangle \langle 1,2,C_4,p_4,C_5,1\rangle \langle 2,1,C_5,p_5,C_4,1\rangle$
g_3: $\langle 0,1,C_1,p_3,C_4,1\rangle \langle 1,2,C_4,p_1,C_2,0\rangle \langle 2,1,C_2,p_2,C_4,1\rangle \langle 1,3,C_4,p_4,C_5,1\rangle$
$\langle 3,1,C_5,p_5,C_4,1\rangle \langle 3,4,C_5,p_6,C_3,0\rangle$

Figure 3 shows the rightmost extension from 1-edge to 3-edge candidate link patterns. The *min-sup* is set to 3. Each candidate in dotted-line should be pruned, because their DFS code is not minimum, which indicates that at least one isomorphic candidate has already been mined. Other candidates with $min\_sup \geq 3$ will be qualified as link patterns in the final result.

Table 1. Relabeling Nodes and Edges

Node URI	Label	Edge URI	Label
foaf:Document	C_1	dc:creator	p_1
foaf:Person	C_2	foaf:maker	p_2
swrc:ConferenceEvent	C_3	ical:url	p_3
swrc:InProceedings	C_4	swc:hasPart	p_4
swrc:Proceedings	C_5	swc:isPartOf	p_5
		swc:relatedToEvent	p_6

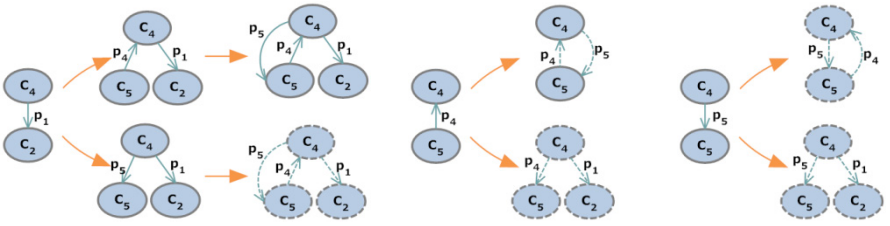

Fig. 3. Rightmost extension from 1-edge to 3-edge candidates

5 Experiments

We evaluate our approach on two sets of linked data. The first is Semantic Web Dog Food, which has a highly unified topic, and the other is a random selected subset of Falcons dataset, which is a collection of online linked data with various topics. In our evaluation, we mainly discuss the time performance and the number of discovered link patterns under the impact of various *min-sup* and *max-edge*. Our algorithm is implemented in C++ and experiments are performed on a 3GHZ Intel Core2 Duo PC with 4G main memory, running on Windows 7.

5.1 Experiment on Semantic Web Dog Food

Semantic Web Dog Food is a well-known and widely-used linked data for scholars. In the dataset, detailed information is provided on accepted papers, people who attended, and other things that have to do with the main conferences and workshops in the area of Semantic Web research.

This linked data is small in size and it describes limited types of objects. The main feature of it is that objects in this dataset are densely linked, and RDF documents in this dataset are highly unified in style: each describes a same topic using a same link pattern. It is very typical for those centrally generated, domain-specific linked data, such as each topic of dbpedia, GO annotations, etc.

Table 2 shows the statistics of Semantic Web Dog Food. Since the dataset has a highly unified topic, all derived Typed Object Graphs are connected, thus they are clustered into a single group.

Table 2. Statistics of Semantic Web Dog Food

RDF Document	Triple	Object
47	166,083	16,281
Quintuple	**Typed Object Graph**	**Cluster**
54,540	47	1

The time performance of the mining process and the number of discovered link patterns are shown in Figure 5. We fix *min-sup* to 3. As *max-edge* increases, time consumed in mining process and discovered link patterns increase dramatically in an exponential way. This is caused by the nature of the data. Objects in Semantic Web Dog Foods are densely connected. Within each iteration of rightmost extension, too many candidates are generated, which leads to a huge lexicographical search tree.

In Figure 5, we perform an evaluation on the impact of changing *min-sup*. Shown in Figure 8, with the increase of *min-sup*, time consumption and number of discovered link patterns both keep declining. This result indicates that a higher threshold of supports will lead to a loss of frequent patterns, but meanwhile it improves the time efficiency of mining.

Besides, experiments shows that link patterns can be huge. Figure 6 shows a very large link patterns discovered in this dataset with $support = 5$. This pattern describes a typical scenario when two researchers are co-authors of two papers, and the papers are both included in proceedings of a conference and are both accessible via an online document.

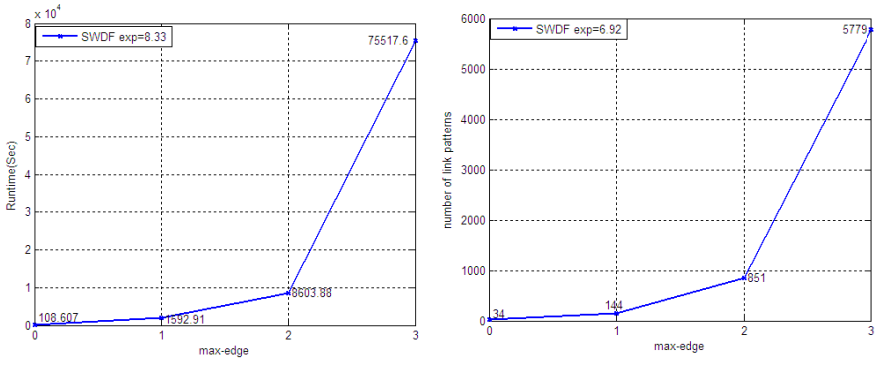

Fig. 4. Time Performance and number of link patterns with various max-edge (SWDF)

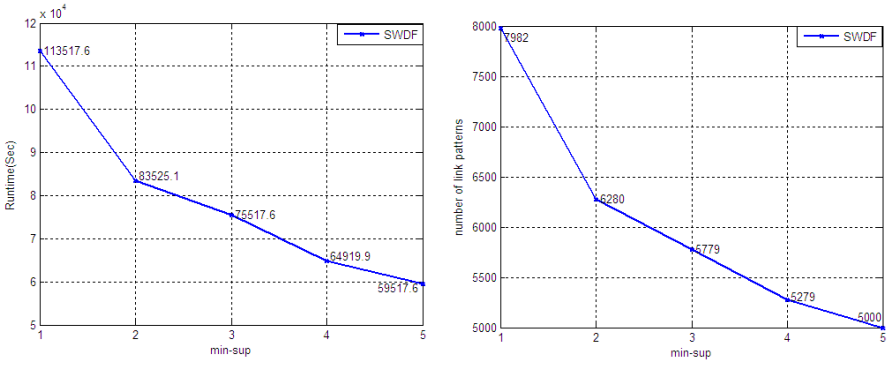

Fig. 5. Time Performance and number of link patterns with various min-sup (SWDF)

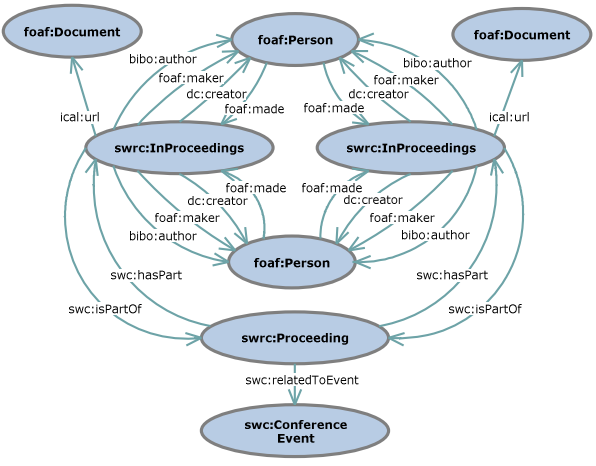

Fig. 6. A huge pattern discovered in Semantic Web Dog Food with 23 edges

5.2 Experiment on Falcons Subset

Falcons is a semantic web search engine. It collects online linked data, and provides searching service of objects, ontologies and RDF documents. The full dataset of Falcons contains over 2 billion triples. We randomly select a subset of Falcons, and the statistics in shown in Table 3.

The selected subset is diverse in topics. The 394 derived Typed Object Graphs are clustered into 8 groups. This dataset is similar to those domain-independent or crawled linked data, such as Billion Triple Challenge Dataset, etc.

Figure 7 shows the time performance and number of discovered link patterns with various *max-edge* and a fixed *min-sup* = 3:

Table 3. Statistics of Falcons

RDF Document	Triple	Object
394	481,213	87,135
Quintuple	**Typed Object Graph**	**Cluster**
96,103	394	8

We can see for this dataset, the increasing rate of both time consumption and discovered link patterns are smaller than the ones of Semantic Web Dog Food. In Falcons, objects are not linked so densely, and link patterns are not shared frequently among documents. The nature of sparsely-linked objects results in a relatively smaller lexicographical search tree, and fewer candidate patterns. The nature also leads to a comparatively more remarkable decrease rate when increasing *min-sup*, as shown in Figure 8.

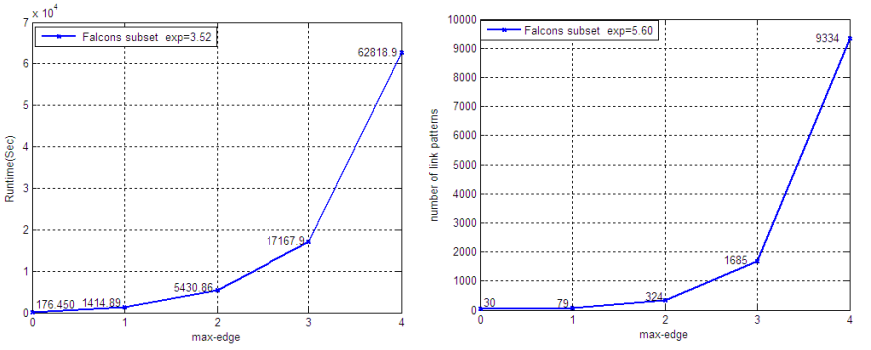

Fig. 7. Time Performance and number of link patterns with various max-edge (Falcons)

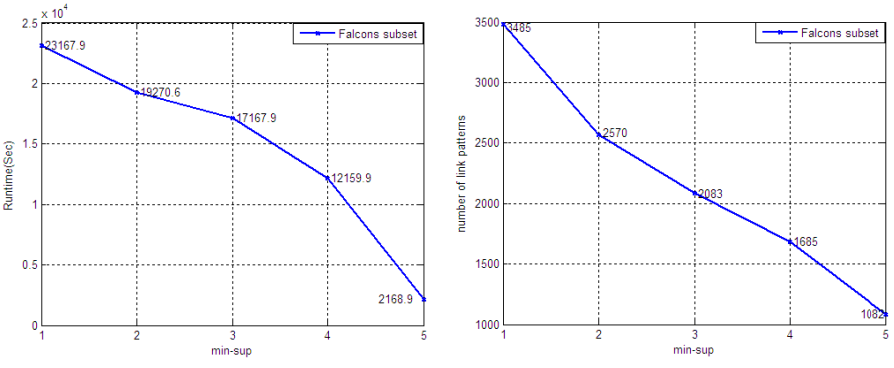

Fig. 8. Time Performance and number of link patterns with various min-sup (Falcons)

6 Related Works

Finding frequent patterns has been studied for years and has become a focused theme in data mining research. Algorithms of frequent graph pattern mining can be roughly classified into three categories: greedy algorithms, Inductive-Logic-Programming-based algorithms, and graph-theory-based algorithm. The last category can be further classified into Apriori-based [16] and Pattern-Growth Approach. The Apriori-based approach has to use the breadth-first search strategy because of its level-wise candidate generation, and thus consumes too much memory and produces too many candidate patterns. gSpan and CloseGraph [15] are two typical Pattern-Growth algorithms, both employing the depth-first search strategy.

Basse et al. has proposed in [2] a DFS-based approach for extracting frequent patterns in triple store, which is much closed to our approach. They improve the canonical representation of RDF graphs based on DFS code proposed by gSpan, and then provide a join operator to significantly reduce the number of frequent graph patterns generated from the analysis of the content of triple stores. The differences between our work and theirs lie in that: their motivation of pattern mining is to

describe the schematic characteristic of the content of RDF bases, to benefit queries involving distributed RDF bases maintained in different servers, while our motivation is to extract typical link patterns for a general purpose of retrieving, understanding and creating linked data; besides, our definitions of data model, as well as the type determination policy, data clustering approach, are fully novel.

7 Conclusion and Future Works

With the explosive growth of online linked data, usage mining on linked data is becoming crucial. In this paper, we present an approach for mining link patterns that describe how different types of objects are frequently interlinked. A Typed Object Graph is defined as the data model, and a gSpan-based algorithm is proposed to discover link patterns. In our algorithm, a policy of type determination is introduced to handle cases of multi-types and an algorithm of data clustering is proposed to improve the scalability. Experiments are performed on two dataset. Time performance and mining results are fully discussed with various parameters.

In our future work, we will explore and compare different pattern mining approaches, to improve the time efficiency of mining. And we will also study the approach of reducing candidate patterns by means of semantic filtering, which will fully utilize the semantics in linked data.

Acknowledgements. The work is supported by the NSFC under Grant 61003055, 61003165, and by NSF of Jiangsu Province under Grant BK2009136, BK2011335. We would like to thank Huaping Chen for his valuable suggestions and their work on related experiments.

References

1. Sheth, A., Aleman-Meza, B., Arpinar, B., et al.: Semantic Association Identification and Knowledge Discovery for National Security Applications. Journal of Database Management 16(1), 33–53 (2005)
2. Basse, A., Gandon, F., Mirbel, I., et al.: DFS-based Frequent Graph Pattern Extraction to Characterize the Content of RDF Triple Stores. In: Proceedings of the WebSci1 2010: Extending the Frontiers of Society Online (2010)
3. Thor, A., Anderson, P., Raschid, L., Navlakha, S., Saha, B., Khuller, S., Zhang, X.-N.: Link Prediction for Annotation Graphs Using Graph Summarization. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 714–729. Springer, Heidelberg (2011)
4. Dai, H., Mobasher, B.: Integrating Semantic Knowledge with Web Usage Mining for Personalization. In: Web Mining: Applications and Techniques, pp. 273–306 (2004)
5. Xu, X., Cong, G., Ooi, B.C., et al.: Semantic Mining and Analysis of Gene Expression Data. In: Proceedings of the 30th International Conference on Very Large Data Bases, pp. 1261–1264 (2004)
6. Yan, X., Han, J.W.: gSpan: Graph-based Substructure Pattern Mining. In: Proceedings of the 2002 IEEE International Conference on Data Mining, pp. 721–724 (2002)
7. Hayes, P.: RDF Semantics. W3C Recommendation (February 10, 2004), `http://www.w3.org/TR/rdf-mt/`

8. Cheng, G., Qu, Y.: Integrating Lightweight Reasoning into Class-Based Query Refinement for Object Search. In: Domingue, J., Anutariya, C. (eds.) ASWC 2008. LNCS, vol. 5367, pp. 449–463. Springer, Heidelberg (2008)
9. Maedche, A., Zacharias, V.: Clustering Ontology-Based Metadata in the Semantic Web. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002. LNCS (LNAI), vol. 2431, pp. 348–360. Springer, Heidelberg (2002)
10. Grimnes, G.A., Edwards, P., Preece, A.D.: Instance Based Clustering of Semantic Web Resources. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 303–317. Springer, Heidelberg (2008)
11. Penin, T., Wang, H., Tran, T., Yu, Y.: Snippet Generation for Semantic Web Search Engines. In: Domingue, J., Anutariya, C. (eds.) ASWC 2008. LNCS, vol. 5367, pp. 493–507. Springer, Heidelberg (2008)
12. Patel, C., Supekar, K., Lee, Y., Park, E.K.: OntoKhoj: A Semantic Web Portal for Ontology Searching, Ranking and Classification. In: Proceedings of 5th ACM International Workshop on Web Information and Data Management, pp. 58–61 (2003)
13. Seidenberg, J., Rector, A.: Web Ontology Segmentation: Analysis, Classification and Use. In: Proceedings of 15th International Word Wide Web Conference, pp. 13–22 (2006)
14. Han, J.W., Kamber, M.: Data Mining Concepts and Techniques, 2nd edn. Elsevier Inc. (2006)
15. Yan, X., Han, J.W.: CloseGraph: Mining Closed Frequent Graph Patterns. In: Proceedings of the 9th ACM SIGKDD Internal Conference on Knowledge Discovery and Data Mining, pp. 285–295 (2003)
16. Inokuchi, A., Washio, T., Motoda, H.: An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 13–23. Springer, Heidelberg (2000)