Random Forest Algorithm for Linked Data Using a Parallel Processing Environment

Dongkyu JEON(a), Member and Wooju KIM(b), Nonmember

SUMMARY In recent years, there has been a significant growth in the importance of data mining of graph-structured data due to this technology’s rapid increase in both scale and application areas. Many previous studies have investigated decision tree learning on Semantic Web-based linked data to uncover implicit knowledge. In the present paper, we suggest a new random forest algorithm for linked data to overcome the underlying limitations of the decision tree algorithm, such as local optimal decisions and generalization error. Moreover, we designed a parallel processing environment for random forest learning to manage large-size linked data and increase the efficiency of multiple tree generation. For this purpose, we modified the previous candidate feature searching method of the decision tree algorithm for linked data to reduce the feature searching space of random forest learning and developed feature selection methods that are adjusted to linked data. Using a distributed index-based search engine, we designed a parallel random forest learning system for linked data to generate random forests in parallel. Our proposed system enables users to simultaneously generate multiple decision trees from distributed stored linked data. To evaluate the performance of the proposed algorithm, we performed experiments to compare the classification accuracy when using the single decision tree algorithm. The experimental results revealed that our random forest algorithm is more accurate than the single decision tree algorithm.

key words: linked data, random forests, parallel processing, semantic Web

1. Introduction

Linked data [8] are the most common types of graph-structured data, which have grown to one of the largest types of publicly available collections of structured data. The application areas and domains of linked data have increased rapidly in recent years. Linked Open Data (LOD) is one of the most representative open datasets; LOD is composed of more than 295 datasets consisting of over 31 billion RDF triples, which are interlinked by approximately 504 million RDF links [13]. Information about users who use social networking and multimedia services, such as Facebook and YouTube, is also a type of large-size linked data. Along with the rapid growth of linked data, knowledge discovery in the linked data field is also becoming important to help encourage the future growth of linked data and their applications [17]. Various approaches have been taken to achieve implicit rule induction [5], [10], [12], link prediction [11], and pattern mining [8], [9] in the study of linked data [17].

The decision tree is one of the most popular data mining algorithms in the classification and rule induction mining fields. Many previous studies have investigated tree construction methods to make a tree with optimal classification accuracy and minimal size [6]. However, there are fundamental limitations in building a global optimal decision tree. First, to avoid losing generality, a decision tree should not be excessively grown to perfectly fit the training data. In addition, local-optimal decision nodes can occur in each tree node because the decision tree growth is based on a greedy algorithm. Some of the previous research on linked data have used decision tree algorithm to investigate graph structured data [5], [10], [12]. However, these studies also suffer from the same limitations noted above, i.e., over-fitting and local optimal decision nodes.

Random forest method is one approach to solve above limitations. The random forest method for linked data is a solution that avoids over-fitting and the occurrence of local optimal decision nodes. By generating multiple trees with randomly sampled training examples and features, random forests can increase the accuracy of the classification task in both training and test datasets [3], [6], [7]. Moreover, random forests can reduce the greedy effect of the decision tree learning procedure using attribute bagging [3].

There are two main algorithms for increasing the accuracy of random forests. The first algorithm is the feature selection method, which randomly selects a number of features among all features given for input attributes. Each tree is constructed based on the selected features, and thus independence and dissimilarity between trees can be increased through the feature selection method. This step is critical for random forests to achieve a higher level of accuracy than individual trees [7]. The second algorithm is the bootstrapping procedure, which involves training examples. For each decision tree learning procedure, a new training dataset is constructed from the original training examples. The accuracy of a tree increases if the tree is grown on the new training datasets with randomly selected features [3]. According to the use of randomly selected features and newly generated training examples, the generation of each decision tree is an independent procedure. Therefore, it is convenient to implement a parallel random forest process for efficient learning [7]. The parallel tree learning process can be used to efficiently reduce the learning time of multiple decision trees.

However, the conventional random forest algorithm cannot be applied to linked data. There exist two primary
obstacles to applying the feature selection, the bootstrapping method and the parallel-learning process to linked data. The first problem is that it is impossible to select the subset of features before beginning the tree construction procedure. In the case of linked data, the feature types are not determined prior to the learning decision tree, which means that they are dynamically changed according to each decision node of the tree [5], [10], [12]. The second problem is parallel processing of the learning decision tree, which is performed on a number of sampled training datasets generated by the bootstrapping procedure. In the case of the ordinary random forest algorithm, it simply separates the training datasets into different machines by a transaction of input data to learn decision trees simultaneously in a parallel processing environment. However, in the case of linked data, there is no specific standard to split linked data into distributed machines. Separating the linked data into several sub-networks related to training examples would result in redundant networks between distributed machines.

In this paper, we propose a new random forest algorithm for linked data that overcomes the problems encountered by the existing methods. In addition, we implement the parallel decision tree learning framework to increase the efficiency of the multiple decision tree learning process. Through our proposed random forest algorithm, we can more efficiently find global optimal solutions to the classification problem defined by the large amount of linked data.

The remainder of this paper is organized as follows. Section 2 presents the basic definitions of linked data used in this study, along with the base decision tree algorithm and its modification for linked data. Section 3 discusses the random forest algorithm for linked data. Section 4 presents the parallel random forest learning environment. Section 5 evaluates the system performance compared to that of the single decision tree algorithm. Section 6 presents the conclusions and the future directions for this study.

2. Preliminaries

2.1 Definition of Linked Data

This study assumes that the linked data are constructed based on the Semantic Web [2] technologies using the Resource Description Framework (RDF) [4] and Web Ontology Language (OWL) [15]. Therefore, the linked data are constructed based on URI-based resources [14] and various data type literal values. All of the linked data relationships are demonstrated by a set of triples, which represent a single directed connection between two nodes of linked data. Linked data \( L \) can be defined as \( L = \{ t_i | i = 1, 2, \ldots, n \} \), where \( n \) is the number of single connections in the entire linked data and \( t_i \) is a triple. Each triple \( t_i \) is defined as \( t_i = \{ \text{subject}, \text{predicate}, \text{object} \} \). \text{subject} and \text{predicate} are linked data resources, and \text{object} can be either a resource or literal value. Therefore, the meaning of this triple \( t_i \) is that \text{subject} is connected with \text{object} through \text{predicate}. The direction of \text{predicate} is determined based on its \text{domain} and \text{range} relationships.

2.2 Decision Tree Algorithm for Linked Data

In this study, the main goal of generating a decision tree on the Semantic Web based linked data is to classify instances based on the specific target property value. Therefore, the decision tree is constructed on a set of training instances whose \text{rdf:type} is the given target class, and a class label of each instance is its target property value.

Data structure of linked data differs from single table data (attribute, value) structure. Therefore, modified decision tree learning method required for linked data. For single table data, features are already defined when the decision tree learning procedure begins; therefore, each feature has a value in the training example. However, for linked data, possible features are not defined and hidden under the graph structure information, which is directly or indirectly related to the training examples. Previous studies on decision tree learning for linked data define features based on the combination of schema-level information of linked data and Description Logic (DL) [11] constructors [5], [10], [12]. They describe the diverse meaning of features using DL constructors, such as negation, intersection, and cardinality. During the decision tree learning procedure, required features are searched and defined for each decision node of the tree prior to decomposing it.

In some of the previous investigations, the features are defined by a random combination of the schema-level resources, and they are combined by randomly selected DL constructors [5], [12]. However, this stochastic base method does not guarantee that defined features are intimately related to the target class of the classification problem. Moreover, as the decision tree grows, the number of candidate features can be increased considerably by each tree node. In certain cases, the number of possible features can be infinite according to the linked data relationships. An example of a linked data schema is presented in Fig. 1: as shown in the figure, it is possible to define an infinite number of features by only using \text{hasMother} relationships, such as \text{mother of mother} (grandmother) or \text{mother of mother of mother}.
Therefore, this random combination method is not appropriate for use as the feature selection process of the random forest algorithm. It would be inefficient to select the set of features from a large set of randomly generated features at every decision node.

Unlike the stochastic method, the feature searching method of [10] incrementally increases the kinds of features from the target class to connected classes through links. The types of features of [10] employed in the proposed method are presented in Table 1. The decision tree learning procedure from reference [10] starts by searching candidate features based on the target class. After splitting the root node based on the selected feature, the algorithm repeats the candidate feature searching and decomposing tree node processes for each decision node until all training examples are completely classified. During the candidate feature searching process, if a parent node of the current decision node is classified by the object property-based feature, then the search space of the candidate features is expanded to connected classes through the object property used. Therefore, the types of features are dynamically diversified for each decision node from the target class to other connected classes during the decision tree growing procedure. For example, assume that the Person class in Fig. 1 is the target class of the decision tree learning procedure. If one of the decision nodes is split by the hasMother property-based feature, then the search space of its child node is expanded along with the hasMother property. Thus, both the training example’s own information and its mother’s information are used to define the candidate features.

In this study, we follow the decision tree learning procedure of [10] to reduce the meaningless candidate features. However, we modified the feature searching algorithm of [10] to perform on the instance level of linked data instead of on the schema level. In contrast to single table data, the types of possible features can be differentiated by each sampled training dataset that is generated by the bootstrapping procedure. For example, let us assume that there are two different training examples, $A = \{p_1, p_2\}$ and $B = \{p_3, p_4\}$, based on the linked data schema shown in Fig. 1. All $p$ ($p_1$-$p_{13}$) in Fig. 2 represent Person type instances connected through properties. As illustrated in Fig. 2, instances of $A$ have only hasMother and hasFather properties, whereas instances of $B$ have hasMother and hasSibling properties. In this case, if candidate features are searched on the instance level, then we are only required to search the sub-network of $A$ and $B$. However, the feature searching method on the schema level would be traversing the same networks in both instance sets, thus traversing some unnecessary networks for each set, because both $A$ and $B$ are defined using the same linked data schema. The instance level feature searching approach increases the efficiency of the random forest generation process by reducing the size of the candidate features for each decision tree. The difference in the traversing size for searching the candidate feature between the suggested method and previous approaches might be increased with regard to the complexity of connections in the linked data schema.

### 3. Random Forest Algorithm for Linked Data

In this section, we present the detail procedure of generating random forests for linked data based on the parallel processing environment.

#### 3.1 Overview of the Random Forest Learning Procedure

The pseudo-code of our proposed algorithm is presented in Algorithm 1, and the formal notations are listed with their definitions in Table 2. The basic process of the ordinary random forest learning algorithm includes three main procedures. First, training datasets and a subset of features are defined to initiate the learning decision tree. Next, a given number of decision trees are grown based on a pre-defined training dataset and features to classify the test datasets. Finally, the classification results of each decision tree are combined to determine the class label of the test examples.
Table 2  Definition of the notations.

| Notation | Definition |
|----------|------------|
| $D_i$    | $\{d_1, d_2, \ldots, d_k\}$ The set of decision trees |
| $d_i$    | A single decision tree |
| $N$      | The number of generated decision trees |
| $S = \{s_1, s_2, \ldots, s_n\}$ | The set of slave nodes |
| $s_j$    | A single slave node (distributed machine) |
| $M$      | The number of slave nodes |
| $K$      | The bagging size number of the training examples |
| $n_0$    | The number of decision trees required to learn on the first slave node $s_j$ |
| $n$      | The number of decision trees required to learn on the slave nodes |
| $t_c$    | Target class |
| $t_p$    | Target property |
| $x_0 = \{x_i, x_i \in \text{rdftype}(t_i)\}$ | The base training examples |
| $p_B$    | The feature selection rate |

In this paper, the classification problem is defined by the target class and target property of the linked data schema. The base set of training examples $x_0$ is searched based on the target class $t_c$. Training examples are the set of instances whose class type is a given target class $t_c$. For $x_0$, we used the bootstrap aggregating method [3] to generate a number of training datasets for each slave node $s_j$. For a given value $K$, the master node randomly samples $K$ numbers of instances from $x_0$. The random sampling of data is based on uniform probability. Each training example set has the same number of instances for each target property value. Algorithm 2 presents the pseudo-code of the bootstrap aggregating procedure used in Algorithm 1. The number of decision trees for each slave node is the quotient of $N$ divided by $M$. If there is a remainder, it is added to the first slave node.

A given number of decision trees are generated at each slave node according to the training examples and other input values.

In particular, we develop feature selection methods that are adjusted to the linked data-based decision tree algorithm. Details of the methods are provided in the following section.

3.2 Feature Selection Methods for Linked Data

As discussed in Sect. 2, for linked data, the features used for decision tree learning are not fixed. Therefore, the feature selection procedure cannot be performed before beginning the decision tree learning. Moreover, because the feature types vary for different decision nodes, it is not appropriate to choose the number of features to be selected. Algorithms 3 and 4 describe the modified feature selection methods employed in our study.

Algorithm 3 describes the feature selection method that selects a certain proportion of features from all possible candidate features of each decision node. Let $dn_i$ be a decision node of growing tree $d_i$ and $f_c$ be the set of all possible candidate features of $dn_i$. If the parent node of $dn_i$ is decomposed by the feature that is based on the object property $p_B$, then new features $f_{nc}$, searched from properties of instances connected to $x_i$ through $p_B$, are added to the previous candidate feature set $f_c$. Among the $f_c$, $B_p$ percent of the features are selected as the final feature set for $dn_i$.

In Algorithm 4, we select decision node features based on the selection of connected properties. According to the decision tree algorithm used in this study, the feature types vary based on the properties of the linked data schema. If some properties are randomly sampled in each decision node, then the search space of the decision tree is highly
4. Parallel Processing Environment for Linked Data

To learn the decision tree in distributed machines in parallel, each decision tree learning machine must be accessible to the entire set of linked data. In this study, linked data are stored in distributed machines using Elasticsearch\(^1\), which is the distributed searching and indexing engine. Through the Elasticsearch engine, any machine can access the entire dataset, which is stored in an Elasticsearch-based index installed in each separate machine and connected by a local area network. The elastic search index is composed of documents, each of which is defined by the set of fields and its values. In this paper, each document \(D\) is defined as \(D = \{\text{subject}, \text{predicate}, \text{object}, \text{object_type}\}\). All triples of linked data are transformed into a document. When all components of a triple constitute a resource, then each field of a document stores an URI value of resources. If the object type is not a resource, the object field stores a literal value, and its data type is stored in the \text{object_type} field. For example, as shown in Fig. 3, linked data are stored in the index as triples. According to the triple \(t_1 = [A, P1, B]\), resource \(A\) is connected to resource \(B\) through property \(P1\), and thus, the value of the \text{object_type} of indexed document \(d_1\) is a resource. Moreover, a triple \(t_3 = [B, P3, 22]\) means that resource \(B\) is connected to literal node \(22\) through \(P3\); then, triple \(t_3\) is indexed to document \(d_3\), whose object value is \(22\) and \text{object_type} is int.

Indexed documents that describe triple information are stored in the Elasticsearch index, which is separated into several shards. Each shard is allocated to a distributed machine; the shards are then grouped by Elasticsearch clusters. During the indexing process, generated documents are indexed and stored separately in each machine. For example, as shown in Fig. 3, \(d_1, d_2\) and \(d_3\) are stored in node 1, and \(d_4\) and \(d_5\) are stored in node 2. The decision tree learning module uses Elasticsearch to find training examples and candidate features from the indexed linked data. We designed a linked data search engine to traverse all of the linked data using a triple data query on the linked data index. The triple data query is composed of a subject, a predicate and an object query keyword. The search engine compares the query keyword and the value of the index for each matched index field. For example, if a search query \(q = \{\text{null}, P1, \text{null}\}\), which means “Search triples with any subject value, the value of the predicate field equal to \(P1\) and

\(^1\)http://elasticsearch.org
any object value”, is queried to the index shown in Fig. 3, then documents $d_1$ and $d_4$ are retrieved from the index.

The parallel processing architecture of our proposed system is presented in Fig. 4. The architecture of a system is separated into two parts: a master node and slave node. The master node is composed of the user interface and voting module. Through the user interface, users can run a random forest algorithm, confirm the generated decision trees and check the classification results, which are processed by the voting module. The voting module collects the classification results from each slave node and determines the classification value of the test data. The slave node consists of a decision tree learning module, a classifier and indices. Indices stored in each slave node are grouped by an Elasticsearch cluster. The decision tree module receives required information for tree learning from the master node and generates certain numbers of trees. The classification module classifies test examples based on decision trees that are generated from the same slave node. Both the master node and slave nodes have a TCP/IP server to communicate with one another. All information includes result trees, and the classification results are sent by JSON-formed messages.

5. Experiment

In this section, we describe the results of experiments performed to evaluate our random forest method. The main goals of the evaluation experiments were as follows:

- Evaluate the performance of decision tree generation on the parallel processing environment.
- Evaluate the accuracy of the proposed method compared to that of the single decision tree algorithm.

Both experiments are performed on a Java-based application, which is our own developed random forest learning system, to evaluate the performance of our proposed method. We connected three personal computers to compose a parallel processing environment, i.e., an i7 3200 MHz CPU with 16 Gb RAM, an i5 1800 MHz CPU with 8 Gb RAM and an i3 2300 MHz CPU with 8 Gb RAM.

5.1 Linked Datasets for Experiments

We used three different linked datasets constructed using the Semantic Web technologies. The definitions of the classification problems for each dataset follow:

1. Family 1: This linked dataset describes the Forte problem [12], which consists of gender information and parent, consort, and sibling relations among people. The goal of the problem is to classify the uncle relationships. This classification problem has been used in many evaluation experiments of previous studies on decision tree algorithms for linked data [5], [12].

2. Family 2: This linked dataset was published by [16] and describes complex family relationships based on the publisher’s own family tree. For the experiment, we designed two classification problems, i.e., we added additional datatype properties and its values to the original data to describe if a person is an uncle or not and if a person is an aunt or not.

3. Media: This linked dataset describes media information such as films, TV shows and music videos. The schema is designed based on Freebase††. The instances of linked data are constructed based on the information from web portals††† in Korea. For the experiment, we separate the media instances into two groups; one group is composed of popular media instances and the other group is not. The popularity of the media is determined based on the preference rate of web users from the web portals.

The detailed specifications of each of the linked datasets are shown in Table 3.

5.2 Experiments on the Parallel Processing Performance

In this experiment, we evaluated the performance of the parallel decision tree learning environment. We estimated random forest learning time differences according to the number of computers used in our parallel system. We used Family 2 and Media datasets to perform the experiments. The comparison of learning time difference is estimated with respect to the number of decision trees and training examples.

The results of the estimated learning time measured as a function of the number of decision trees are shown in

---

††http://java.sun.com

†††http://www.freebase.com/

†††http://www.naver.com, http://daum.net
Fig. 5  Learning time with the number of trees.

Fig. 6  Learning time with the number of training examples.

Fig. 7  Experimental result of comparison of test datasets accuracies.

Table 4  p-values in significance test for comparison of accuracies.

| Family 1 | Family 2 (Uncle) | Family 2 (Aunt) | Media |
|----------|------------------|-----------------|-------|
| FS1      | .000             | .001            | .000  | .009 |
| FS2      | .000             | .016            | .000  | .018 |

5.3 Experiments on the Classification Accuracy

In this experiment, we compared our method of the parallel random forest algorithm to other single decision tree methods [10], where each tree was generated using randomly selected examples with all possible features. All generated decision trees were fully grown without any pruning. All datasets were separated into training data and testing data. The training data is 80% of the entire target instances. Through a bootstrapping procedure, new training and test examples are generated for each decision tree. The exact numbers of the sample sizes are shown in Table 3. We performed experiments with two different feature selection methods proposed by the present study, i.e., the feature selection method using Algorithm 3 (FS1) and the feature selection method using Algorithm 4 (FS2). The selection ratio for both FS1 and FS2 is 70% of all the candidate features and properties.

The experimental results in Fig. 7 show the accuracies of the single decision tree and random forests of our proposed methods. When the number of trees is small, the single decision tree may have better accuracy. However, when the number of trees increases to a large number, random forests are more accurate than a single decision tree in every classification problem. For the accuracy of the experimental results shown in Fig. 7, significance tests were performed by a one-sample t-test [18] to determine if the differences between the accuracy of the random forest and the single decision tree are statistically significant. The statistical software SPSS 21 is used for the implementation. The results of the significance test are shown in Table 4. In the table, the value of each cell is the p-value of the one-sample t-test. Under the 5% significance level, all p-values are smaller than 0.05, as in Table 4. Therefore, we conclude that the improved performance of our proposed method is significant.

In addition, we count the number of candidate features
for each experiment to compare the differences of feature search spaces between the proposed feature selection methods and the single decision tree method. The minimum, the maximum and the average numbers of candidate features are listed in Table 5. There are no significant differences between FS1 and FS2. However, both feature selection methods used fewer features than the single decision tree used. Random forests also have better accuracy with fewer candidate features.

### 6. Conclusion

In this paper, we have presented a new method to construct a random forest in a parallel processing environment in the context of linked data. To achieve our goal, we modified the candidate feature searching algorithm of decision tree learning and developed modified feature selection methods that are adjusted to the graph-structured data. We also designed a parallel processing environment for linked data to learn random forests in parallel using the distributed indexing and searching technology. Our proposed system enables users to simultaneously generate multiple decision trees from distributed stored linked data. This ability is significant because our proposed approach helps to avoid a local optimal solution and reduce generalization error, which frequently occur in the single decision tree algorithm. Parallel processing is also important to manage large-size linked data and achieve computational performance for learning decision trees.

A possible future research direction is to extend our focus to simply represented graph structured data. There are larger structured datasets available than the datasets considered here, but they do not have detailed graphical definitions, such as the directions of edges or the exact labeling of nodes and edges. Although this study focuses on Semantic Web technology-based linked data, the proposed random forests generating procedure, except for the decision tree learning process, can be extended to any type of structured data. Therefore, future research will investigate the development of proper features and selection methods for simplified graph structured data. In addition, the parallel processing framework should be modified for other types of linked data.

### References

[1] F. Baader, I. Horrocks, and U. Sattler, “Description logics as ontologies languages for the semantic web,” in Mechanizing Mathematical Reasoning, pp.228–248, Springer Berlin Heidelberg, 2005.

[2] T. Berners-Lee, J. Hendler, and O. Lassila, “The semantic web,” Scientific American, vol.284, no.5, pp.28–37, 2001.

[3] L. Breiman, “Random forests,” Machine Learning, vol.45, no.1, pp.5–32, 2001.

[4] D. Brückley and R.V. Guha, “RDF vocabulary description language 1.0: RDF schema,” 2004.

[5] N. Fanizzi, C. d’Amato, and F. Esposito, “Induction of concepts in web ontologies through terminological decision trees,” in Machine Learning and Knowledge Discovery in Databases, pp.442–457, Springer Berlin Heidelberg, 2010.

[6] T.K. Ho, “Random decision forests,” Proc. Third International Conf. on Document Analysis and Recognition, pp.278–282, Montreal, Que, Aug. 1995.

[7] T.K. Ho, “The random subspace method for constructing decision forests,” IEEE Trans. Pattern Anal. Mach. Intell., vol.20, no.8, pp.832–844, 1998.

[8] J. Hoxha, M. Maleshkova, and P. Korevaar, “Knowledge Discovery meets Linked APIs,” Proc. Workshops of 10th International Conf. ESWC 2013, pp.56–65, Montpellier, France, May 2013.

[9] A. Inokuchi, T. Washio, and H. Motoda, “Complete mining of frequent patterns from graphs: Mining graph data,” Machine Learning, vol.50, no.3, pp.321–354, 2003.

[10] D. Jeon and W. Kim, “Development of semantic decision tree,” Data Mining and Intelligent Information Technology Applications (ICMIA), 2011 3rd International Conf. on, IEEE, Macau, Oct. 2011.

[11] L. Munasinghe and R. Ichise, “Link prediction in social networks using information flow via active links,” IEICE Trans. Inf. & Syst., vol.E96-D, no.7, pp.1495–1502, July 2013.

[12] J. Lehmann and P. Hitzler, “Concept learning in description logics using refinement operators,” Machine Learning, vol.78, no.1-2, pp.203–250, 2010.

[13] LINKINGOPENDATA, “SweoIG/TaskForces/CommunityProjects/LinkingOpenData,” http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData, accessed May 23, 2014.

[14] L. Masinter, T. Berners-Lee, and R.T. Fielding, “Uniform resource identifier (URI): Generic syntax,” 2005.

[15] D.L. McGuinness and F.V. Harmelen, “OWL web ontology language overview,” W3C recommendation, 2004.

[16] Ontologies, “The University of Manchester, School of Computer Science, BHIG, Robert Stevens,” http://www.cs.man.ac.uk/~stevens/ menupages/ontologies.php, accessed May 23, 2014.

[17] A. Rettinger, U. Lösch, V. Tresp, C. d’Amato, and N. Fanizzi, “Mining the semantic web,” Data Mining and Knowledge Discovery, vol.24, no.3, pp.613–662, Feb. 2012.

[18] H. Michael, ed., Encyclopedia of Mathematics, Springer, 2001.
Dongkyu Jeon is a Ph.D. candidate of Information and Industrial Engineering at Yonsei University in Korea. He received an M.S. degree in Information and Industrial Engineering from Yonsei University, Korea in 2010. His primary research interests are data mining, Semantic Web, linked data mining and semantic information retrieval.

Wooju Kim is a Professor of Information and Industrial Engineering at Yonsei University in Korea. He received a B.B.A. degree from Yonsei University in 1987 and a Ph.D. in Management Science from KAIST in 1994. His current research areas are decision support systems in Semantic Web environments, Semantic Web mining, knowledge management and intelligent web services.