



Using Graph Pattern Association Rules on Yago Knowledge Base

Wahyudi*, Masayu Leylia Khodra, Ary Setijadi Prihatmanto & Carmadi Machbub

School of Electrical Engineering and Informatics, Institut Teknologi Bandung
Jalan Ganesha 10, Bandung 40132, Indonesia
*E-mail: wahyudi23509023@student.itb.ac.id

Abstract. The use of graph pattern association rules (GPARs) on the Yago knowledge base is proposed. Extending association rules for itemsets, GPARS can help to discover regularities between entities in a knowledge base. A rule-generated graph pattern (RGGP) algorithm was used for extracting rules from the Yago knowledge base and a GPAR algorithm for creating the association rules. Our research resulted in 1114 association rules, with the value of standard confidence at 50.18% better than partial completeness assumption (PCA) confidence at 49.82%. Besides that the computation time for standard confidence was also better than for PCA confidence.

Keywords: *association rule; graph pattern; knowledge base; PCA confidence; standard confidence.*

1 Introduction

The Yago knowledge base [1] contains common sense knowledge. It is a collection of commonly known facts and information. Yago was built by extraction of data from Wikipedia and using the WordNet ontology. However, WordNet's ontology is limited, so Yago developed its own proprietary ontology. Yago evolved into Yago 2 [2] with the addition of data extracted from GeoNames so Yago 2 can describe spatial entities using spatial data such as longitude and latitude from GeoNames. Yago 2 was expanded into Yago 3 [3] in view of multilinguality; in previous versions Yago only extracted data from English Wikipedia, while in Yago 3 multilingual extraction from Wikipedia was done and grouping of entities was also done based on languages supported by Wikipedia.

Luis [4] developed the AMIE system to mine rules with incomplete facts using association rules [5]. AMIE uses the Yago KB and horn rules with data representation in the form of a relational database [6]. Luis explored the horn rules and tuples contained in the relations of each entity. Each entity is represented by a function and has a maximum functionality value of 1 and a minimum functionality value of 0. The function has an inverse function that

Received September 17th, 2018^{1st} Revision July 2nd, 2019, 2nd Revision August 8th, 2019, 3rd Revision August 30th, 2019, Accepted for publication September 4th, 2019.

Copyright © 2019 Published by ITB Journal Publisher, ISSN: 2337-5787, DOI: 10.5614/itbj.ict.res.appl.2019.13.2.6

also has a functionality value. For example: function $export(x,y)$ has inverse function $isExported(y,x)$. The functionality values of the function and the inverse function of each entity are compared and the largest value will be used by the system. The various connections between entities in the form of patterns were not discussed too much by Luis. The function and its inverse can be used on the relational database to find rules. Partial completeness assumption (PCA) confidence is used to generate or predict negative evidence, but these measurements need more processing time and more computational resources than standard confidence. In contrast, our research focused on diversified graph pattern association to generate rules.

Fan [7] proposed association rules utilizing graph patterns. The proposed method uses parallel computation and focuses on social graphs. It obtains potential customers using the diversified mining problem (DMP) technique and the entity identification problem (EIP) technique based on the support of each entity. This is different for knowledge bases, especially when determining association rules. Fan claimed PCA confidence does not perform better than Bayes factor-based confidence if facts are presented in graph patterns [8]. However, he used a different dataset under local closed world assumption (LCWA). Therefore we decided to use the graph pattern approach from Fan [7] and the mining model from Luis [6]. We wanted to investigate whether PCA confidence performs better than standard confidence when using graph patterns.

Our research used this combination of techniques on the Yago KB [9]. We used graph representation as data representation for connected data and for visualizing the graph database. The flexibility of the graph model allows us to add entities and their relationships without affecting or modifying existing data [10]. Some well-known apps like Facebook, Google, Wikipedia and IMDB as well as many other apps use graph representation as data representation.

We propose association rule mining of the Yago knowledge base. Finding these rules can serve several purposes, among others to predict new facts that are not in the dataset, to verify facts in the dataset when there are different facts from the rules, and to help understand the data better. More precisely, our contributions are: (1) we used graph pattern association rules (GPARs) on the Yago knowledge base; (2) we defined support and confidence for GPARs; (3) we experimentally verified the scalability and effectiveness of our algorithms for creating and mining rules.

The rest of this paper is structured as follows. Section 2 discusses related works. Section 3 introduces the preliminaries and Section 4 presents our mining model. Section 5 discusses the implementation of the graph pattern association rules.

Section 6 presents our experiments, and Section 7 contains the conclusion and recommendations for future work.

2 Related Work

The Yago KB has the form of an RDF triple. RDF only has positive examples. It operates under open world assumption (OWA). This means that something not found in the KB is not necessarily assumed to be wrong but classified as unknown. This is a fundamental difference with database settings operating under closed world assumption (CWA). In CWA, facts that are not in the dataset cannot be assumed. For example, a KB contains the statement ‘John was born in Paris’. Then there is the question: ‘Was Alice also born in Paris?’ Under CWA we get ‘no’ as the answer, while under OWA we get ‘unknown’ as the answer. CWA eliminates the possibility that Alice was born in Paris, while OWA keeps open the possibility that Alice was born in Paris or not.

Association rules were introduced by Agrawal [5]. Association rules combine multiple items into antecedents and have one item as consequent. Two steps are executed to generate the association rules. Firstly, finding all itemsets that are present in at least $c\%$ of transactions. Secondly, finding association rules efficiently. Association rules have been well studied for discovering regularities between items in relational databases for promotional pricing and product placement. They have the traditional form of $X \Rightarrow Y$, where X and Y are disjoint itemsets.

Fanizzi [11] has attempted to mine rules from the semantic web using the inductive logic model (ILP). The goal was to find a hypothesis that included all positive examples in the absence of a negative example. This requires rules of various positive and negative examples to be investigated [12]. This is a problem in KBs because in KBs there are no negative examples. Another problem is that the ILP system cannot process large amounts of data while KBs contain a large amount of data.

Mining rules using ordinary techniques (inductive logic programming, logical rules) can only mine complete facts contained in a database. Incomplete facts cannot be used with this technique. Luis [13] used association rules under open world assumption (OWA) for KBs, introducing new thresholds for mining models called head coverage. This notion is used to filter rules based on the size of the head, replacing the count of support with an absolute number. Moreover, it uses a new notion of confidence measurement called partial completeness assumption (PCA) confidence. Our research applied this confidence for comparison with standard confidence [5].

The GPAR algorithm proposed by Fan [7] was used to create graph patterns for mining association rules in social media marketing and identifying potential customers under CWA using parallel computation. There are also existing algorithms for pattern mining graph databases. Large-scale mining techniques in a single graph have also been studied, notably top-k algorithms to reduce cost and scalable subgraph isomorphism algorithms adapted to generate pattern candidates [14].

Yago KB graph properties [15] can be seen as a set of facts, where each fact consists of two nodes that are connected by one edge (x,r,y) with x denoting node 1, r the relation (or edge), and y denoting node 2 of the fact. There are several equivalent alternative representations of facts. In this study, we borrowed the notation from Datalog and represent facts as $r(x, y)$. For example, we write $isLocatedIn(Bandung,Indonesia)$.

3 Mining Model

In this paper, we focus on Yago KB graph properties [15]. A graph property model is a graph consisting of nodes, edge, and properties [10]. We use properties such as entity properties. Each node has properties as depicted in Figure 1. It shows two nodes with the person label and a node with the book label. The two nodes are connected with the edge label *hasRead*. The person node has the property name and value *John Smith*, and the book node has two sets of properties, *title* and *author*. The title has the value *graph database* and the author has the value *Ian Robinson*.



Figure 1 Graph property model.

In this section, we will explain the mining model that will be generated. Most of the models we present were adopted from Luis [13] and Fan [7]. We adopted the approach of Luis for support, head coverage and confidence for graph patterns. We used a different approach than Fan's for graph patterns (see Section 4). The difference is explained as follows:

3.1 Support

The support of a rule quantifies the number of correct rules, i.e. the size of A. A rule's support is the frequency or number of itemsets in the data set. Support is calculated from the calculated number of itemsets compared to the total number of itemsets in the dataset. Support graph pattern P in a graph G , denoted by $supp(P, G)$, indicates the number of P s contained in G . Our approach uses support as the number of instantiations of a rule that appear in a KB. The support of graph pattern p is denoted by $supp(P(G))$. This is the number of nodes and edge pairs found in graph pattern $P(G)$. Support of rule R is denoted by $supp R$, i.e. the number of nodes and edge pairs present in R .

$$supp(P_{(G)}) = |P_{(G)}|, supp R = |R|$$

3.2 Head Coverage

As mentioned above, we use head coverage (HC) as the threshold for the strength of a rule (R), as in Eqs. (1) and (2). In association rules usually *min support* is used as the threshold for the strength of a rule. We used *min HC* = 0.01.

$$R: P(G) \Rightarrow r(u, w) \quad (1)$$

$$HC = \frac{supp(R)}{|r|} \quad (2)$$

where *HC* is head coverage, $supp(R)$ is the number of rules R , and $|r|$ is the size of the head in the dataset.

3.3 Confidence, Standard Confidence and PCA Confidence

This is a measure to determine the strength of a rule. The value is between 0 and 1. A rule with high confidence is close to 1 and, vice versa, a rule with low confidence is close to 0. In this research, we used two types of confidence: standard confidence and PCA confidence.

Standard confidence (*conf*) is a measure of the ratio of the number of rules R compared to the facts we know in the form of graph pattern $P(G)$, as in Eq. (3) below:

$$conf(R_i) = \frac{|R_i|}{|P_{(G)i}|} \quad (3)$$

Standard confidence does not distinguish between facts that are not in the dataset and wrong facts in the dataset. In other words, standard confidence cannot distinguish between wrong facts and unknown facts. Since the

knowledge base has no negative facts, the partial completeness approach (PCA) was used, as proposed by Luis [4]. If $r(u, w) \in G$ for nodes u and w then:

$$\forall w' = r(u, w) \in G \cup \text{new true} \Rightarrow r(u, w') \in G$$

In other words, we assume that if graph G knows some attribute x of u , then we can see all attributes x of u . This assumption is converted to standard confidence, so it can be obtained with Eq. (4):

$$PCA\ conf(R_i) = \frac{|R_i|}{|P(G)_i \wedge r(u, w')|} \quad (4)$$

4 Graph Pattern Association Rules

In this section we will discuss the approach used in this research in detail. Graph pattern association rule $P(x, y)$ is defined as $B(x, y) \Rightarrow r(x, y)$, where $B(x, y)$ is a graph pattern in which x and y are two designated nodes and $r(x, y)$ is an edge labeled r from x to y on which the same search conditions as in B are imposed. We refer to B and r as the antecedent and consequent of P , respectively [7].

We use graph patterns to mine the association rules. We chose this option for the following reasons: we follow the Agrawal model by starting from 1 antecedent and increasing the number of antecedents to 2, 3 and 4. In the tool we use, we can automatically search for a graph pattern, but there is a problem with the edge direction: there is only one edge direction. This is certainly different from the actual data in the database, where the relationship can come from both sides, from the subject or the object. Figure 2 shows two graph patterns that have different directions in R2.

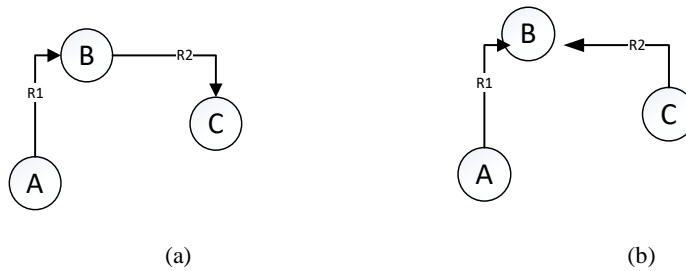


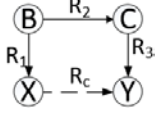
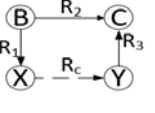
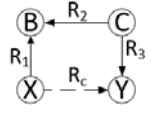
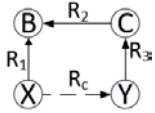
Figure 2 Graph pattern.

In Figure 2(a), R2 has the same direction as R1 but in Figure 2(b), R2 has the opposite direction from R1. This cannot be executed by one query; instead there should be a query for each graph pattern. In graph theory, the graph motif

technique is used, i.e. the isomorphism of two subgraphs is determined by the interaction pattern between node and edge. In the proposed method, we use the edge property (relation) as the motif for determining the isomorphism of subgraphs. Thus we use the graph pattern to get all possible directions from each of the existing entities. This is different from Fan [7], who used a pattern generator [14]. We use ten graph patterns, consisting of patterns that have one relation, two relations and, three relations, as shown in Table 1 below.

Table 1 Graph patterns used in this research.

Graph Pattern	Association Rule	Graph Pattern Association Rule
	$X - [R_1] \rightarrow Y \Rightarrow X - [R_c] \rightarrow Y$	$Y \leftarrow [R_c] - X - [R_1] \rightarrow Y$
	$X - [R_1] \rightarrow B - [R_2] \rightarrow Y \Rightarrow X - [R_c] \rightarrow Y$	$Y \leftarrow [R_c] - X - [R_1] \rightarrow B - [R_2] \rightarrow Y$
	$X - [R_1] \rightarrow B \leftarrow [R_2] - Y \Rightarrow X - [R_c] \rightarrow Y$	$Y \leftarrow [R_c] - X - [R_1] \rightarrow B \leftarrow [R_2] - Y$
	$X \leftarrow [R_1] - B - [R_2] \rightarrow Y \Rightarrow X - [R_c] \rightarrow Y$	$Y \leftarrow [R_c] - X \leftarrow [R_1] - B - [R_2] \rightarrow Y$
	$X - [R_1] \rightarrow B - [R_2] \rightarrow C - [R_3] \rightarrow Y \Rightarrow X - [R_c] \rightarrow Y$	$Y \leftarrow [R_c] - X - [R_1] \rightarrow B - [R_2] \rightarrow C - [R_3] \rightarrow Y$
	$X - [R_1] \rightarrow B - [R_2] \rightarrow C \leftarrow [R_3] - Y \Rightarrow X - [R_c] \rightarrow Y$	$Y \leftarrow [R_c] - X - [R_1] \rightarrow B - [R_2] \rightarrow C \leftarrow [R_3] - Y$

Graph Pattern	Association Rule	Graph Pattern Association Rule
	$X \leftarrow [R_1] - B - [R_2] \rightarrow C - [R_3] \rightarrow Y \Rightarrow X - [R_c] \rightarrow Y$	$Y \leftarrow [R_c] - X \leftarrow [R_1] - B - [R_2] \rightarrow C - [R_3] \rightarrow Y$
	$X \leftarrow [R_1] - B - [R_2] \rightarrow C \leftarrow [R_3] - Y \Rightarrow X - [R_c] \rightarrow Y$	$Y \leftarrow [R_c] - X \leftarrow [R_1] - B - [R_2] \rightarrow C \leftarrow [R_3] - Y$
	$X - [R_1] \rightarrow B \leftarrow [R_2] - C - [R_3] \rightarrow Y \Rightarrow X - [R_c] \rightarrow Y$	$Y \leftarrow [R_c] - X - [R_1] \rightarrow B \leftarrow [R_2] - C - [R_3] \rightarrow Y$
	$X - [R_1] \rightarrow B \leftarrow [R_2] - C \leftarrow [R_3] - Y \Rightarrow X - [R_c] \rightarrow Y$	$Y \leftarrow [R_c] - X - [R_1] \rightarrow B \leftarrow [R_2] - C \leftarrow [R_3] - Y$

We developed the algorithm shown in Figure 3 to generate rules from graph patterns. This algorithm is called the Rule-Generated Graph Pattern (RGGP) algorithm. It has as input a knowledge base and graph patterns. In the first step, we sort the ten graph patterns in an array and select them one by one with the matchPattern function. The expected results are rule body (r_b), rule head (r_h) and the count of rules (c) generated.

Algorithm 1 Rule-generated graph pattern (RGGP)

Input : Knowledge Bases (K), Graph Pattern G_P
Output: Collection of rules($collection$) $\{ \langle R_b, R_h \rangle, c \}$

- 1: $G_p = [P_1, P_2 \dots P_n]$
 - 2: Set $collection = \{ \}$
 - 3: **for each** $P_i \in G_P$ **do**
 - 4: list $\langle R_b, R_h \rangle = \mathbf{matchPattern}(P_i, K)$
 - 5: $c = \mathbf{count}(\langle R_b, R_h \rangle)$
 - 6: $collection.add\{\langle R_b, R_h \rangle, c\}$
 - 7: **return** $collection$
-

Figure 3 Rule-generated graph pattern (RGGP) algorithm.

Finally, rules are generated from the RGGP algorithm. The next step is to determine the rules to be used in the association rules. The algorithm used is the graph-pattern association rules (GPAR) algorithm shown in Figure 4. This algorithm has as input the collection rules that were generated by the RGGP algorithm. The first step of the algorithm is to process the rules one by one, from the first rule to the last, after the values of head coverage, support, standard confidence, and PCA confidence have been set to zero. Each rule counts the number of body rules and head rules, after which the value of head coverage is counted. Rules that qualify are rules that have minimum head coverage ≥ 0.01 . Support, standard confidence and PCA confidence are calculated for each rule.

Algorithm 2 Graph-pattern association rule

Input : Collection of rules(*collection*) $\{ \langle R_b, R_h \rangle, c \}$

Output:Confidence output (*out*) $\{ \langle rbody, rhead \rangle, suppr, minHC, stdConf, pcaConf \}$

```

1: Set out = {}
2: Set minHC, stdConf, pcaConf = 0
3: for each  $\{ \langle R_b, R_h \rangle, c \} \in collection$  do
4:   rhead = collection.get.R_h
5:   rbody = collection.get.R_b
6:   suppr = collection.get.c
7:   ch = count(rhead)
8:    $minHC = \frac{suppr}{ch}$ 
9:   if  $minHC \geq 0.01$  then
10:    cb = count(rbody)
11:    for each  $\langle rbody, rhead \rangle$  do
12:       $name(par_1, par_2) = getNameParameter(rhead)$ 
13:       $r_{pca} = addRel\ name(par_1, z)$ 
14:      removeRel rhead
15:       $c_{pca} = count(r_{pca})$ 
16:       $stdConf = \frac{suppr}{cb}$ 
17:       $pcaConf = \frac{suppr}{c_{pca}}$ 
18:      conf.add( $\langle rbody, rhead \rangle, suppr, minHC, stdConf, pcaConf$ )
19: return out

```

Figure 4 Graph-pattern association rule (GPAR) algorithm.

5 Experiment

Using the graph properties of the Yago KB, we conducted an experiment to generate collection rules and association rules. The experiment used the 20

types of graph patterns shown in Table 1. It used standard confidence as well as PCA confidence, because we wanted to investigate whether PCA confidence performs better than standard confidence when using graph patterns.

5.1 Experimental Setup

We used Neo4j for visualizing the graph database and graph processing. Our dataset, Yago, had 600 K nodes of more than 50 different types, and 980 K edges of 30 types, such as *isPoliticianOf*, *isLeaderOf*, etc. All experiments ran on a laptop with 8 GB of RAM and four physical CPUs (Intel core i3 at 1.7 GHz).

We tested the Neo4J web server. We compared query execution to generate nodes with Neo4j running time queries. Our research used indexation on nodes in the dataset with node name (nn) properties to increase the query execution time. The index speeds up the search for data based on certain properties (in this research nn properties). This test displays graph visualization, so it requires additional time compared to graph processing. For query execution that produces 10-100 nodes requires 2-5 seconds running time. The full results are in Table 2

Table 2 Comparison of Neo4J execution time.

Running time (s)	Nodes Generated
< 1	< 10
2 – 5	10 – 100
- 12	101– 500
13 - 20	500 – 800
>20	> 800

5.2 Standard Confidence vs PCA Confidence

This experiment generated 1114 rules that met head coverage ≥ 0.01 . For each rule its confidence was calculated using standard confidence and PCA confidence. From the result of the experiment 559 rules had standard confidence better than PCA confidence (50.18%), whereas 555 rules (49.82%) had PCA confidence better than standard confidence, as shown in Figures 5(a) and 5(b). Table 3 shows the 3 rules that had the best standard confidence vs PCA confidence.

We conducted tests on the top 3 standard confidence and PCA confidence rules. We used the QAGPAR tools [16] and tested 50 times for each rule. The results can be seen in Table 2. Figure 5(d) is a comparison of the performance

precision of each rule. The results obtained show that the performance of PCA confidence was better than that of standard confidence for the top 3 rules.

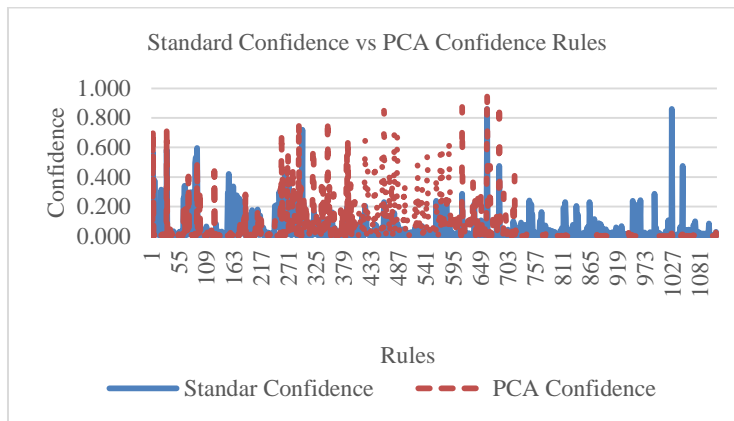
Table 3 Top 3 Rules for Standard Confidence vs PCA Confidence

(a) Top 3 standard confidence rules

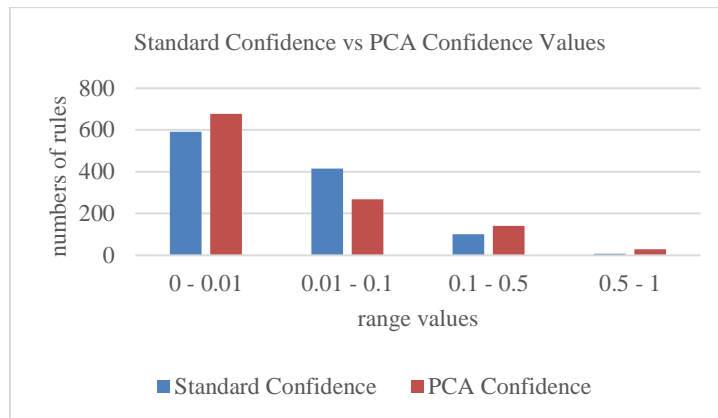
Graph Pattern	Standard Confidence	PCA Confidence	Precision
	0.86	1.00	74%
	0.59	0.74	52%
	0.53	0.02	28%

(b) Top 3 PCA confidence rules

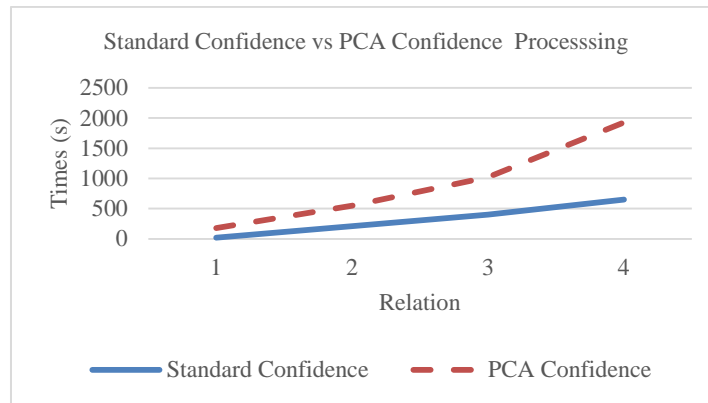
Graph Pattern	Standard Confidence	PCA Confidence	Precision
	0.86	1.00	74%
	0.59	0.74	54%
	0.59	0.74	52%



(a) Confidence values of each rule

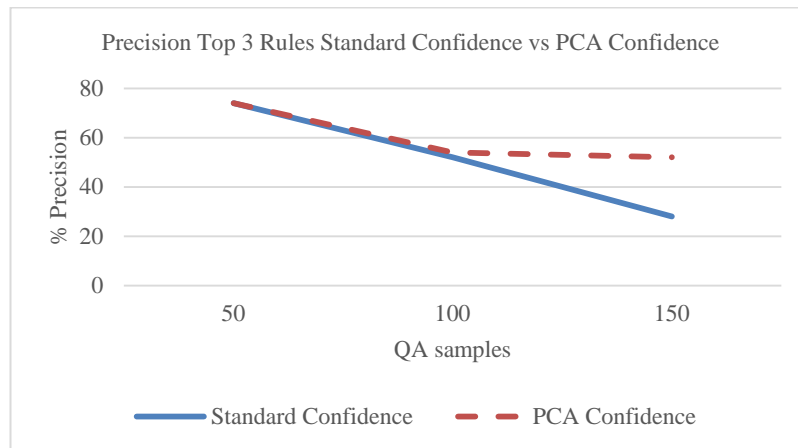


(b) Range of confidence values



(c) Processing time by relation

Figure 5 Evaluation of confidence measures.



(d) Precision test on 3 top rules

Figure 5 Continued. Evaluation of confidence measures.

We evaluated the scalability of standard confidence vs PCA confidence using used relation = 1 and found that different relations had impact. We found that there were differences in the generated number of nodes and edges for standard confidence vs PCA confidence in the same relation. This caused the processing time for PCA confidence to be higher than that for standard confidence, as shown in Figure 5(c).

6 Conclusion and Future Works

In this paper, graph pattern association rules (GPARs) were proposed for itemsets in syntax and semantics to support confidence metrics and graph properties for mining association rules from the Yago knowledge base. Our confidence metrics used standard confidence and PCA confidence. The experimental result indicated that standard confidence performed slightly better than PCA confidence. We obtained an average value for PCA confidence that was lower than that of standard confidence for the graph patterns. Therefore, further research is needed to determine the appropriate confidence for graph patterns.

Reference

- [1] Suchanek, F.M., Kasneci, G. & Weikum, G., *YAGO: A Large Ontology from Wikipedia and WordNet*, *Web Semantics*, **6**(3), pp. 203-217, 2008.
- [2] Hoffart, J. & Suchanek, F.M., *YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia*, pp. 3161-3165, 2013.

- [3] Mahdisoltani, F., Biega, J. & Suchanek, F., *YAGO3: A Knowledge Base from Multilingual Wikipedias*, in 7th Biennial Conference on Innovative Data Systems Research (CIDR 2015), pp. 177-185, 2015.
- [4] Galárraga, L., Teflioudi, C., Hose, K. & Suchanek, F.M., *Amie: Association Rule Mining under Incomplete Evidence in Ontological Knowledge Bases*, In: WWW, pp. 413-422, 2013.
- [5] Agrawal, R., Imieliński, T. & Swami, A., *Mining Association Rules between Sets of Items in Large Databases*, ACM SIGMOD Record, **22**(2), pp. 207-216, 1993.
- [6] Galárraga, L., Teflioudi, C., Hose, K. & Suchanek, F.M., *Fast Rule Mining in Ontological Knowledge Bases with AMIE+*, The VLDB Journal, **24**(6), pp. 707-730, 2015.
- [7] Fan, W., Wang, X., Wu, Y. & Xu, J., *Association Rules with Graph Patterns*, Proceedings of the VLDB Endowment, **8**(12), pp. 1502-1513, 2015.
- [8] Fan, W. & Hu, C., *Big Graph Analyses: From Queries to Dependencies and Association Rules*, Data Science and Engineering, **2**(1), pp. 36-55, 2017.
- [9] Suchanek, F.M., Kasneci, G. & Weikum, G., *YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia*, in Proceedings of the 16th International Conference on World Wide Web – WWW '07, 2007.
- [10] Robinson, I., Webber, J. & Eifrem, E., *Graph Databases*, Second Edition. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA, United States, 95472, 2015.
- [11] Fanizzi, N. & Esposito, F., *Inductive Learning for the Semantic Web: What Does It Buy?*, **1**, pp. 1-5, 2009.
- [12] Muggleton, S., *Inductive Logic Programming: Issues, Results and the Challenge of Learning Language in Logic*, Artificial Intelligence, **114**(1-2), pp. 283-296, 1999.
- [13] Galárraga, L., Preda, N. & Suchanek, F.M. *Mining Rules to Align Knowledge Bases*, in Proceedings of the 2013 Workshop on Automated Knowledge Base Construction- AKBC '13, pp. 43-48, 2013.
- [14] Yan, X. & Han, J., *gSpan: Graph-based Substructure Pattern Mining*, in IEEE International Conference on Data Mining, Proceedings, **1**(d), pp. 721-724, 2002.
- [15] Wahyudi, Khodra, M.L., Prihatmanto, A.S. & Machbub, C., *Knowledge-based Graph Compression using Graph Property on Yago*, in 2017 3rd International Conference on Science in Information Technology (ICSITech), pp. 127-131, 2017.
- [16] Wahyudi, Khodra, M.L., Prihatmanto, A.S. & Machbub, C., *A Question Answering System Using Graph-Pattern Association Rules (QAGPAR) on YAGO Knowledge Base*, in International Conference on Information Technology Systems and Innovation (ICITSI), pp. 536-541, 2018.