Extracting fuzzy summaries from nosql graph databases

Arnaud Castelltort, Anne Laurent

To cite this version:

Arnaud Castelltort, Anne Laurent. Extracting fuzzy summaries from nosql graph databases. FQAS: Flexible Query Answering Systems, Oct 2015, Cracow, Poland. pp.189-200, 10.1007/978-3-319-26154-6_15 . lirmm-01381078

HAL Id: lirmm-01381078
https://hal-lirmm.ccsd.cnrs.fr/lirmm-01381078v1

Submitted on 7 Oct 2019

Extracting Fuzzy Summaries from NoSQL Graph Databases

Arnaud Castelltort and Anne Laurent

University of Montpellier
{firstname.lastname@lirmm.fr}
http://www.lirmm.fr

Abstract. Linguistic summaries have been studied for many years and allow to sum up large volumes of data in a very intuitive manner. They have been studied over several types of data. However, few works have been led on graph databses. Graph databases are becoming popular tools and have recently gained significant recognition with the emergence of the so-called NoSQL graph databases. These databases allow users to handle huge volumes of data (e.g., scientific data, social networks). There are several ways to consider graph summaries. In this paper, we detail the specificities of NoSQL graph databases and we discuss how to summarize them by introducing several types of linguistic summaries, namely structure summaries, data structure summaries and fuzzy summaries. We present extraction methods that have been tested over synthetic and real database experimentations.

Keywords: Linguistic Summaries, Graph Databases, NoSQL, Fuzzy Graph Mining

1 Introduction

Representing data with graphs is now a proven practice. Graphs are used in many applications ranging from linguistics to chemistry and social networks. For instance, graphs allow one to intuitively illustrate the relationships between people, as well as those between people and the organizations they belong to.

Graphs are recognized to play an important role within the pattern recognition field [8]; indeed, they are a key technology for retrieving relevant information, such as in fraud detection [15] or social/biological interactions. Relevant information can be retrieved either via data mining methods or predefined queries [1].

Even if graphs are ubiquitous, their representation and use have taken many forms in the literature and in real applications [3, 14]. Theoretical works have attempted to formalize the representations and treatments of graphs More recently, they have been used in the semantic web framework, especially with ontologies. However, with the emergence of big data, the increasing volume and complexity of data and treatments, researchers have realized that robust database management systems are required. Additionally, classical relational database engines are not the solution, as shown in the performance comparisons done on this topic [7]. NoSQL is reputed for its suitability to handle big data [11]. NoSQL graph databases have thus been proposed as the best alternative for managing huge volumes of graph data and complex queries. There are several NoSQL engines; among them, Neo4j is one of the most popular.

In the NoSQL graph database model, the objects considered are nodes and relationships. Complex information on both nodes and relationships are managed as properties with $(key, value)$ pairs. In addition to properties, nodes and relationships may be labeled with types. For instance, Fig. 1 shows the relationships between people and the places they live in. The relationship types depict whether people *own* or *rent* their housing. The type of housing can be an apartment or a house: we define this value as the node type. Node information (e.g., age of people) and relationships (e.g., monthly rental fees) can be provided as node and relationship properties (e.g., $key = age$).

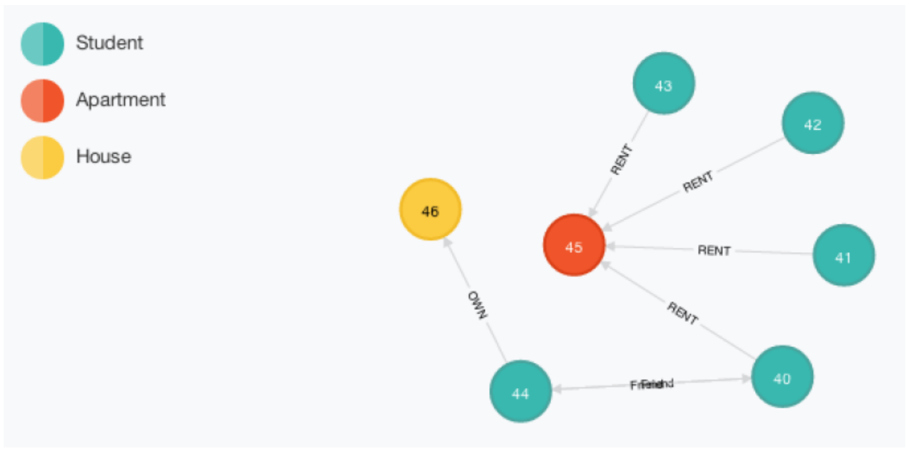


Fig. 1. Example of a Graph Database

As the amount and the complexity of information increase, the need for summaries increases as well. The literature is rich with propositions on linguistic summaries of relational databases [5]. Linguistic summaries are based on protoforms, the first one being Qy *are* P where Q stands for a fuzzy quantifier, y are the objects to be summarized and P is a possible value, such as in *Most students are young*. Linguistic summaries have been extended in many works, for instance handle time series [2]. However, these works cannot be easily applied to NoSQL graph databases. The summaries we aim to discover must indeed be transposable to linguistic summaries. Moreover such databases combine several criteria that have never been considered altogether: node and relationship types, complex information contained in the $(key, value)$ properties.

In this paper, we thus propose several types of linguistic summaries of NoSQL graph databases. The paper is organized as follows. Section 2 reports existing work on NoSQL graph databases and linguistic summarization. Section 3 introduces our definitions for linguistic summaries of NoSQL graph databases. In Section 4, we introduce the queries used to extract the summaries, by highlighting the power of NoSQL graph databases using the Cypher language. Section 5 concludes the paper and discusses perspectives for future work.

2 Background

Our work is strongly related to NoSQL graph databases and data summarization. We therefore review the basics of these two topics below.

2.1 Graphs

General Concepts. Graphs have been studied for a long time by mathematicians and computer scientists. A graph can be directed or not, labeled or not. It is defined as follows.

Def 1 (Graph) *A graph G is given by a pair (V, E) where V stands for a set of vertices and E stands for a set of edges with $E \subseteq (V \times V)$.*

Def 2 (Directed Graph) *A directed graph G is given by a pair (V, E) where V stands for a set of vertices and E stands for a set of edges with $E \subseteq \{V \times V\}$. That is E is a subset of all ordered permutations of V element pairs.*

When used in real world applications, graphs need to be provided with the capacity to label nodes and relations, thus leading to the so-called labeled graphs, or property graphs.

Def 3 (Labeled Oriented Graph) *A labeled oriented graph G, also known as oriented property graph, is given by a quadruplet (V, E, α, β) where V stands for a set of vertices and E stands for a set of edges with $E \subseteq \{V \times V\}$, α stands for the set of attributes defined over the nodes, and β the set of attributes defined over the relations.*

NoSQL graph databases [14] are based on these concepts, attributes and values over the attributes being stored thanks to the $(key, value)$ paradigm which is very common in NoSQL databases. Fig. 3 shows a graph and its structure in $(key, value)$ pairs.

Studies have shown that these technologies present good performances, much better than classical relational databases for representing and querying such large graph databases.

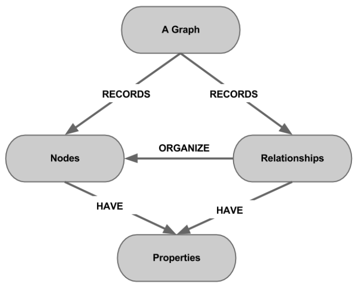


Fig. 2. Labeled Graph

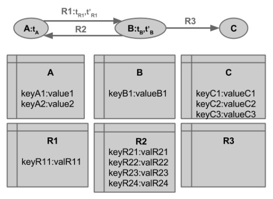


Fig. 3. Node and Relation Properties

Graph Summarization. Data Summarization has been extensively studied in the last decades to produce linguistic sentences, such as *Most of the students are young* [17]. These approaches are based on the so-called *protoforms* (*e.g.*, Qy *are* P) where Q is a fuzzy quantifier, y are the objects to summarize and P is a (fuzzy) predicate. They focus on relational data where source data are represented in the form of tuples defined over a schema. For instance, the tuples (John, 23, 45000), (Mary, 32, 60000) and (Bill, 38, 55000) are three tuples defined on the schema (Name, Age, Salary). The fuzzy quantifiers are defined over the $[0, 1]$ universe of proportions. We may for instance consider two quantifiers Few and $Most$ which membership functions are displayed by Fig. 4.

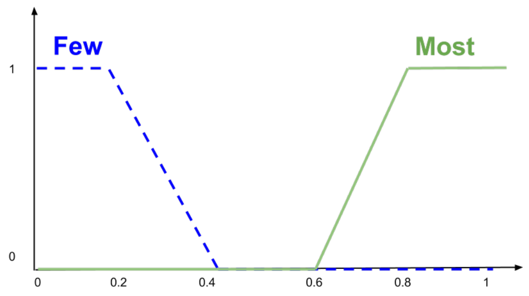


Fig. 4. Example of Fuzzy Quantifier Membership Functions

The quality of linguistic summaries can be assessed by many measures, the seminal one being T, the *degree of truth* that can be simply computed with a σ-count:

$$T(Qy's\ are\ P) = \mu_Q\left(\frac{1}{n}\sum_{i=1}^{n}\mu_P(y_i)\right)$$

Where n is the number of objects (y_i) that are summarized, and μ_P , and μ_Q are the membership functions of the summarizer and quantifier, respectively.

There are various ways to examine summaries: Researchers have focused on the design of protoforms, quality measures, efficient algorithms, etc [5].

We will not recall here all the literature on fuzzy linguistic summaries it has been amply presented in previous works. We will focus instead on the subject of graph data. In this framework, two main characteristics have to be highlighted. First, graph databases are not provided with a strict and given schema such as relational data. In fact, they are close to semi-structured data. Second, graph databases focus on relationships.

Summarizing graph data has been considered for many years, aiming for instance at compressing such data with the use of supernodes as shown by Fig.5 from [13].

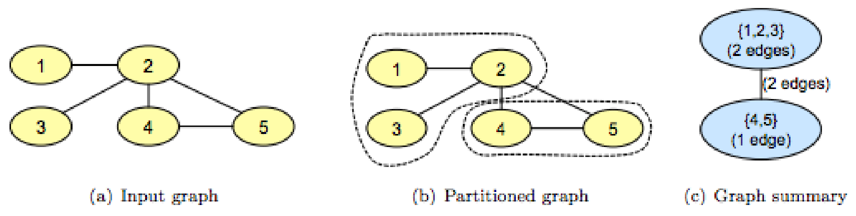


Fig. 5. Graph Summarization using Supernodes [13]

To some extent, graph summarization can be assimilated to graph mining. Graph (and tree) mining is seen as the problem of extracting frequent patterns (subgraphs/subtrees) from a large graph. It is often presented as an extension of the so-called itemset mining methods. Such methods have been successfully applied to large graphs by considering efficient approaches [9, 12, 18].

Several works in the literature have focused on schema extraction in the context of semi-structured graph data, i.e., XML data. The schema extraction problem consists in identifying a schema S from a given set of XML data documents D, such that S captures the structural information of the documents in D in the most minimal way. The schema extraction process is also referred to as schema inference [4]. The underlying structure of a given collection of XML documents can be described using Document Type Definitions (DTD), XML Schema, or via a more general representation such as tree or a graph. The structure extraction techniques in the literature aim to infer three kinds of representations: tree or graph summaries, DTD or XML Schema.

2.2 NoSQL Graph Databases

NoSQL graph databases [14] are based on graph concepts with the following additional points:

– Nodes and relationships are labeled with types;
– Properties are defined over the nodes and relationships stored according to the (key, value) paradigm, which is very common in NoSQL databases.

It should be noted that types are distinguished from properties, as in NoSQL engines such as Neo4j. These types appear in Fig. 1 as colors for nodes (e.g., Student, House) and as labels for relationships (e.g., Owns).

Generalizing the definition of labeled directed graphs, we propose a formal definition of NoSQL graph databases hereafter:

Def 4 (NoSQL Graph Database) *A NoSQL graph database G is given by a tuple $(V, E, \theta, \tau, \alpha, \beta)$ where*

– *α stands for the set of node properties defined by (key:value) pairs;*
– *β stands for the set of edge properties defined by (key:value) pairs;*
– *θ stands for the set of node types;*
– *τ stands for the set of edge types;*
– *V stands for a set of vertices with $\forall v \in V, v = (id_v, t_v, \kappa_v)$ s.t. $t_v \subseteq \theta$ stands for the types of v, $\kappa_v \subseteq \alpha$ stands for the properties of v and id_v is the vertice identifier;*
– *E stands for a set of edges with $\forall e \in E, e = (id_e, (v_e^1, v_e^2), t_e, \lambda_e)$ s.t. $(v_e^1, v_e^2) \in \{V \times V\}$, $t_e \subseteq \tau$ stands for the types of e, $\lambda_e \subseteq \beta$ stands for the properties of e and id_e is the edge identifier.*

The set of properties of a node v is denoted by α_v, the set of types is denoted by θ_v. The set of properties of a relation e is denoted by β_e, the set of types is denoted by τ_e.

Fig. 3 shows a graph and its structure in $(key, value)$ pairs. On this figure, we have:

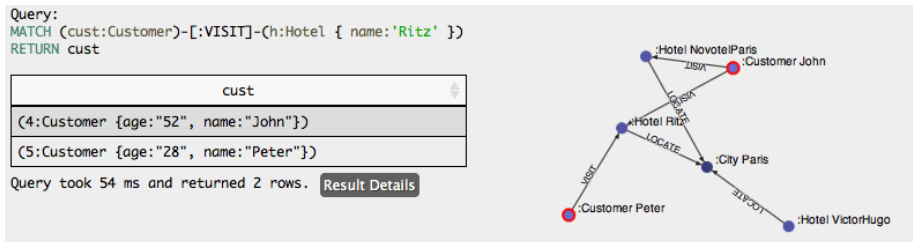
– $\alpha = \{(keyA1, valueA1), \ldots\}$
– $\beta = \{(keyR11, valueR11), \ldots\}$
– $V = \{(A, \{t_A\}, \{(keyA1, valueA1), (keyA2, valueA2)\}), (B, \{t_B, t'_B\}, \{(keyB1, valueB1)\}), (C, \emptyset, \{(keyC1, valueC1), (keyC2, valueC2), (keyC3, valueC3)\})$
– $E = \{(R1, (A, B), \{t_{R1}, t'_{R1}\}, \{(keyR11, valueR11)\}), \ldots\}$

There exist several NoSQL graph database engines (OrientDB, Neo4J, HyperGraphDB, etc.) [3]. Neo4J is considered the best perfomer [16]. All NoSQL graph databases require developers and users to use graph concepts to query data. Queries are called traversals, refering to the action of visiting elements, i.e. nodes and relationships. There are three main ways to traverse a graph:

– Programmatically: using an API;
– By functional traversal: using a traversal based on a sequence of functions applied to a graph;
– By declarative traversal: explicitly expressing what we want to do and not how we want to do it. The database engine then defines the best way to achieve this goal.

In this paper, we focus on declarative queries over a NoSQL graph database. The Neo4j language is called Cypher.

For instance, in Fig. 6, a query to return the customers who have visited the “Ritz hotel is displayed. Those customers are both displayed in the list and circled in red in the graph.



```
Query:
MATCH (cust:Customer)-[:VISIT]-(h:Hotel { name:'Ritz' })
RETURN cust
```

cust
(4:Customer {age:"52", name:"John"})
(5:Customer {age:"28", name:"Peter"})

Query took 54 ms and returned 2 rows.

Fig. 6. Displaying the Result of a Cypher Query

Cypher clauses are similar to SQL ones. It is based on a “ASCII art” way of writing graph elements. For example, directed relations are written using the -[]->symbol. Types and labels are written after a semi-column (:).

More specifically, queries in Cypher have the following syntax[1]:

Listing 1.1. Query example on a Graph

```
[START]
[MATCH]
[OPTIONAL MATCH WHERE]
[WITH [ORDER BY] [SKIP] [LIMIT]]
RETURN [ORDER BY] [SKIP] [LIMIT]
```

As shown above, Cypher is comprised of several distinct clauses which are listed below in Listing 1.2.

Listing 1.2. Cypher Clauses

```
START: Starting points in the graph, obtained via index lookups or by element IDs.
MATCH: The graph pattern to match, bound to the starting points in START.
WHERE: Filtering criteria.
RETURN: What to return.
CREATE: Creates nodes and relationships.
DELETE: Removes nodes, relationships and properties.
SET: Set values to properties.
FOREACH: Performs updating actions once per element in a list.
WITH: Divides a query into multiple, distinct parts.
```

These operations can even be extended to fuzzy queries [6]. In this work, they are used for computing the linguistic summaries introduced below.

[1] http://docs.neo4j.org/refcard/2.0/
http://docs.neo4j.org/chunked/milestone/cypher-query-lang.html

3 Building Fuzzy NoSQL Graph Summaries

Below, we define the protoforms for summing up NoSQL graph databases.

3.1 Structure Summaries

Structure summaries are meant to retrieve the structure of the graph embedded in element types, which could somehow be associated with relational database schema. Such summaries could thus be associated with with schema mining in the literature.

Def 5 (Structure Summary) *Let $G = (V, E, \theta, \alpha, \beta)$ be a NoSQL graph database. A structure summary S of G is defined as $S = (a -[r]-> b, Q)$ where $a, b \in \theta$ (node types), $r \in \tau$ (relation type) and Q is a fuzzy quantifier.*

The structure summary can be expressed in a linguistic form as follows: In G, Q of the $a\ r\ b$

Example 1 *In the toy example, $(Student\text{–}[rent]\text{–>}apartment, Most)$, expressed as* "Most of the students rent an apartment*", is a structure summary.*

3.2 DataStructure Summaries

Data structure summaries are meant to refine structure summaries. They allow one to differentiate cases when schema depend on the value of properties. For instance, depending on their salary, employees may rent apartments instead of houses.

Def 6 (Data Structure Summary) *Let $G = (V, E, \theta, \tau, \alpha, \beta)$ be a temporal NoSQL graph database. A Data Structure Summary S is defined as $S = (a.X -[r.Z]-> b.Y, Q)$ with $a, b \in \theta$ (node types), $r \in \tau$ (relation type), $X, Y \subseteq \alpha$ (node properties), $Z \subseteq \beta$ (relation properties) and Q a fuzzy quantifier.*

Example 2 *In the toy example, $(Student(Age : 28) -[rent(fees : 1200)]-> apartment, Few)$ is a data structure summary.*

Such summaries are extended in order to allow fuzzy linguistic labels in the refinement. Indeed, it would be both difficult and useless to define summaries on single values such as "the age is 28, as in fuzzy data mining for fuzzy association rule mining. Using fuzzy linguistic labels makes it possible to retrieve fuzzy linguistic summaries where young students and low rental fees are considered.

Def 7 (Fuzzy Data Structure Summary) *Let $G = (V, E, \theta, \tau, \alpha, \beta)$ be a NoSQL graph database. Let F_α and F_β be sets of fuzzy properties. A Fuzzy Data Structure Summary S is defined as $S = (a.X -[r.Z]-> b.Y, Q)$ with $a, b \in \theta$ (node types), $r \in \tau$ (relation type), $X, Y \subseteq \alpha \bigcup \mathcal{F}_\alpha$ (node properties and node fuzzy properties), $Z \subseteq \beta \bigcup F_\beta$ (relation properties and relation fuzzy properties) and Q a fuzzy quantifier.*

Example 3 *In the toy example,* $(Student(Age : young) -[rent(fees : low)]\text{–>} apartment, Most)$ *is a fuzzy structure summary.*

For all these summaries, it is important to discuss the way to assess them. For this purpose, we propose to rely on the extension of the degree of truth.

3.3 Degree of Truth

In our framework, the degree of truth determines the extent to which the relationship appearing in the summary is truthful regarding the fuzzy quantifier. For instance, if the summary mentions that *most of the students rent an apartment,* then the degree of truth describes to which extent a high proportion of students rent an apartment.

There are two ways of calculating this degree. Indeed, it may consist in computing the proportion of students who rent an apartment over the whole student population, or the proportion of students who rent an apartment over the number of students who rent their housing. These two definitions are provided below.

Def 8 (Degree of Truth of NoSQL Graph Summaries) *Given a graph database* G *and a summary* $S = a -[r]\text{–>} b, Q$*, the degrees of Truth of* S *in* G *are defined as:*

$$Truth_1(S) = \mu_Q\left(\frac{count(distinct(S))}{count(distinct(a))}\right)$$

$$Truth_2(S) = \mu_Q\left(\frac{count(distinct(S))}{count(distinct(a -[r]\text{–>} (?)))}\right)$$

The second type of degree of truth is also called the *diversity of target source* denoted by D_T later on in this paper.

Example 4 *In the toy example from Fig. 1, the degree of truth of the summary "*$S = Student -[rent]\text{–>} apartment, Most$*" is given by the membership degree to the fuzzy quantifier*[2] *of the ratio between the number of times a relationship appears between a Student and an Apartment (s)he rents over the number of relations of type Rents starting from a Student node. In this example, we thus have* $Truth_2(S) = \mu_{Most}\left(\frac{4}{5}\right) = \mu_{Most}(0.8)$

The ratio appearing in the definition of the degree of truth can be compared to the $confidence$ in association rule mining which acts as a conditional probability.

[2] We do not mention here the detailed membership function of the *Most* quantifier which can be defined in a very classical manner as done in the literature of fuzzy quantifiers and fuzzy summaries.

4 Extracting the Summaries

Below we detail how to extract the above-defined summaries from NoSQL graph databases and some first results.

4.1 Queries

Most of the treatments we propose can be performed effectively by defining queries over the NoSQL graph database, so as to obtain a more declarative than procedural way to extract summaries. This property is based on the fact that NoSQL graph databases provide powerful pattern-matching features, as intented in inductive relational databases [10].

The structure summaries can be retrieved by considering Cypher queries such as:

```
MATCH (a)-[r]->(m)
RETURN DISTINCT labels(a), type(r), labels(m), count(r)
```

In the above query, all the structures are retrieved together, along with the number of times they appear.

The query from Listing 1.3 extracts the summaries from a graph and calculates the degree of truth for every summary.

Listing 1.3. Retrieving Structure Summaries

```
MATCH (a)-[r]->(b)
WITH DISTINCT labels(a) AS labelsA, type(r) AS typeR, labels(b) AS ↩
    labelsC, toFloat(count(*)) AS countS
MATCH (a1)-[r2]->(m)
WHERE labels(a1)= labelsA AND type(r2)= typeR
WITH DISTINCT labelsA, typeR, labelsC, countS, labels(a1) AS labelsA1,↩
    type(r2) AS typeR2, count(*) AS count2
RETURN labelsA, typeR, labelsC,
       tofloat(countS)/ count2 AS Truth,
       MuMost(countS/count2) as TruthMost,
       MuFew(countS/count2) as TruthFew,
       MuVeryFew(countS/count2) as TruthVeryFew
```

Where μ_{Most}, μ_{Few} and $\mu_{VeryFew}$ respectively stand for the membership function of the fuzzy subsets $Most$, Few and $VeryFew$ defined on the $[0,1]$ universe which are implemented as the $MuMost$, $MuFew$ and $MuVeryFew$ functions in Cypher.

4.2 Experimental Results

Experiments have been run on synthetic and real databases with a Java implementation run on an Intel Core i5 2.4 Ghz. The Movie real database deals with Directors, Movies and Actors[3]. It contains about 12,000 movies and 50,000 actors. Below is an example of the results from the synthetic database:

[3] http://neo4j.com/developer/example-data

Summary	$TruthMost$	$TruthFew$	$TruthVeryFew$
Student-Rents-Apartment	$\mu_{Most}(4/5)$	$\mu_{Few}(4/5)$	$\mu_{VeryFew}(4/5)$
Student-Friend-Student	$\mu_{Most}(2/5)$	$\mu_{Few}(2/5)$	$\mu_{VeryFew}(2/5)$
Student-Owns-House	$\mu_{Most}(1/5)$	$\mu_{Few}(1/5)$	$\mu_{VeryFew}(1/5)$

The Listing 1.4 displays some examples of fuzzy data structure summaries.

Listing 1.4. Fuzzy Data Structure Summaries Extracted from the Movie Database

```
DS12:(Person(old)-[DIRECTED]->Movie,Most); Truth(DS12)=1.0
DS16:(Person(young)-[DIRECTED]->Movie,VeryFew); Truth(DS16)=0.21
DS17:(Person(middleAge)-[DIRECTED]->Movie,VeryFew); Truth(DS17)=0.47
DS18:(Person(old)-[DIRECTED]->Movie,VeryFew); Truth(DS18)=0.0
```

5 Conclusion and Perspectives

In this paper, we have presented an approach to summarize NoSQL graph databases in the form of linguistic summaries. These databases are quite specific with respect to existing work, as they combine several difficulties: a focus on relationships, node and relationship types management, management of complex (key:value) pair information management, etc.

Several types of linguistic summaries are proposed. They can be easily extracted using existing declarative query languages, making it possible to deploy them on every commercial tool. Experiments have been made on Neo4j using the Cypher query language, and have demonstrating the interest of our proposal.

Our work opens several perspectives. First, we plan to run more experiments to test scalability. The propositions could also be extended to several graphs. Moreover, we might try and integrate other types of summaries for managing time and space information. Another perspective is to consider extended linguistic summaries containing several relationships.

References

1. Aggarwal, C.C., Wang, H. (eds.): Managing and Mining Graph Data, Advances in Database Systems, vol. 40. Springer (2010)
2. Almeida, R.J., Lesot, M., Bouchon-Meunier, B., Kaymak, U., Moyse, G.: Linguistic summaries of categorical time series for septic shock patient data. In: FUZZ-IEEE 2013, IEEE International Conference on Fuzzy Systems, Hyderabad, India, 7-10 July, 2013, Proceedings. pp. 1–8. IEEE (2013), http://dx.doi.org/10.1109/FUZZ-IEEE.2013.6622581
3. Angles, R., Gutiérrez, C.: Survey of graph database models. ACM Comput. Surv. 40(1) (2008)
4. Bex, G.J., Neven, F., Vansummeren, S.: Inferring XML schema definitions from XML data. In: Koch, C., Gehrke, J., Garofalakis, M.N., Srivastava, D., Aberer, K., Deshpande, A., Florescu, D., Chan, C.Y., Ganti, V., Kanne, C., Klas, W., Neuhold, E.J. (eds.) Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007. pp. 998–1009. ACM (2007)

5. Bouchon-Meunier, B., Moyse, G.: Fuzzy linguistic summaries: where are we, where can we go ? In: IEEE Conf. on Computational Intelligence for Financial Engineering & Economics (CIFEr). pp. 317–324. CIFEr 2012, IEEE (2012)
6. Castelltort, A., Laurent, A.: Fuzzy queries over nosql graph databases: Perspectives for extending the cypher language. In: International Conference on Processing and Management of Uncertainty in Knowledge-Based Systems. Springer (2014)
7. Cattell, R.: Scalable SQL and NoSQL data stores. SIGMOD Record 39(4), 12–27 (2010)
8. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty Years Of Graph Matching In Pattern Recognition. International Journal of Pattern Recognition and Artificial Intelligence (2004)
9. Cook, D.J., Holder, L.B.: Mining Graph Data. John Wiley & Sons (2006)
10. De Raedt, L.: A perspective on inductive databases. SIGKDD Explor. Newsl. 4(2), 69–77 (Dec 2002)
11. Han, J., Haihong, E., Le, G., Du, J.: Survey on nosql database. In: Proc. of the 6th International Conference on Pervasive Computing and Applications (ICPCA). pp. 363–366 (2011)
12. Kuramochi, M., Karypis, G.: Frequent subgraph discovery. In: Proceedings of the 2001 IEEE International Conference on Data Mining. pp. 313–320. ICDM '01, IEEE Computer Society, Washington, DC, USA (2001), http://dl.acm.org/citation.cfm?id=645496.658027
13. LeFevre, K., Terzi, E.: Grass: Graph structure summarization. In: Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA. pp. 454–465. SIAM (2010)
14. Robinson, I., Webber, J., Eifrem, E.: Graph Databases. O'Reilly (2013)
15. Sadowski, G., Rathle, P.: Fraud detection: Discovering connections with graph databases. In: White Paper - Neo Technology - Graphs are Everywhere (2014)
16. ThoughtWorks: Technology advisory board (May 2013), http://thoughtworks.fileburst.com/assets/technology-radar-may-2013.pdf
17. Yager, R.R.: A new approach to the summarization of data. Information Sciences 28(1), 69 – 86 (1982)
18. Yan, X., Han, J.: gspan: Graph-based substructure pattern mining. In: Proceedings of the 2002 IEEE International Conference on Data Mining. pp. 721–. ICDM '02, IEEE Computer Society, Washington, DC, USA (2002), http://dl.acm.org/citation.cfm?id=844380.844811