Efficiency and Precision Trade-Offs in Graph Summary Algorithms

Stéphane Campinas Renaud Delbru Giovanni Tummarello

Digital Enterprise Research Institute
National University of Ireland, Galway
{firstname}.{lastname}@deri.org

June 1, 2013

Abstract

In many applications, it is convenient to substitute a large data graph with a smaller homomorphic graph. This paper investigates approaches for summarising massive data graphs. In general, massive data graphs are processed using a shared-nothing infrastructure such as MapReduce. However, accurate graph summarization algorithms are sub-optimal for this kind of environment as they require multiple iterations over the data graph. We investigate approximate graph summarization algorithms that are efficient to compute in a shared-nothing infrastructure. We define a quality assessment model of a summary with regards to a gold standard summary. We evaluate over several datasets the trade-offs between efficiency and precision. With regards to an application, experiments highlight the need to trade-off the precision and volume of a graph summary with the complexity of a summarization technique.

1 Introduction

With the advent of the Semantic Web, there is a considerable amount of graph-structured data on the Web from various domains, e.g., life science, e-commerce, e-health, statistics, geography, e-social, ... The Linked Data[1] movement promotes the inter-linking of different kinds of data. This creates a massive graph of knowledge, best depicted by the LOD Cloud[2]. However, the sheer size of a graph prevent applications from its effective exploitation. Indeed, in the scenario of data exploration, a user who seeks a global understanding of the data graph would need to visit each node, see what nodes are connected and how. In the scenario of graph pattern recommendation, e.g., in a SPARQL query [2], one needs to dig into the data graph to know if a given pattern exists, or whether the pattern leads to interesting data or not. In these scenario and many others, a summary of the graph is used instead of the data graph itself, because of its reduced size and the descriptive statistics it can provide.

[1] Linked Data: `http://linkeddata.org/`
[2] LOD Cloud: `http://lod-cloud.net/`

In order to deal with such large data graphs, there is a significant research on the concept of *graph summarization*, i.e., the substitution of the data graph with a homomorphic graph. This other graph, that we call the *summary*, contains ideally less nodes and edges with regards to the data graph. Because it is homomorphic to the data graph, applications can use a summary instead of the original data graph. For example, [7, 28, 35] propose a graph summarization approach for optimizing query processing. The authors in [15, 34] leverages a summary for data exploration. A summary is used in [2, 10] to guide users writing graph queries. In practice, voluminous graphs are generally processed using a shared-nothing infrastructure (e.g., MapReduce). However, current graph summarization approaches are too complex to be applied in such a distributed fashion.

In this paper, we investigate several approximate graph summarisation algorithms that are efficient to compute on a shared-nothing infrastructure such as Hadoop. To that end, we introduce a precision model for evaluating the accuracy of graph summary algorithms compared to a gold standard summary. This precision model takes into account two aspects of the graph summary: the structural summary and the schema summary. We then analyse the trade-offs between the efficiency and the precision of the graph summarisation algorithms, over 15 real-world datasets of various size and complexity. The experimental results show that it is possible to approximate quite accurately the gold standard graph summary but with a much lower space and time complexity. The experimental results also provide initial evidences about the applicability of the algorithms in different context, e.g., schema summarisation or structure summarisation.

This paper is structured as follows. We first describe in the Section 2 the data model of a graph summary. Then, we present in the Section 3 several graph summarization algorithms. In the Section 4, we introduce the precision model for a graph summary. In the Section 5, we perform an evaluation using this precision model, and show the trade-offs between precision and efficiency of an algorithm. We present in the Section 6 works related to the graph summarization.

2 Data Model

We introduce in this section a generic graph model for semi-structured data. Then, we define the graph summary, and how it is created from a data graph.

2.1 Data Graph

The Semantic Web is at its base a collection of structured data sources that are exposed on the Web through various forms such as HTML tables, Deep Web databases, XML documents, documents embedding semantic markups, e.g., Microformats, Microdata, RDF, RDFa. Since each data source might have its own defined schema, ranging from loosely to strictly defined, the data structure does not follow strict rules as in a database. Even within a given data source, the schema might not be fixed and may change as the information grows. The information structure evolves over time and new entities can require new attributes. We therefore consider the Semantic Web as being *semi-structured* [1].

We represent the knowledge in such heterogeneous environments via labelled directed graphs, that we shall refer to as *data graphs*.

Definition 2.1 (Data Graph) *Let V be a set of nodes and A a set of labelled edges. Let $\mathscr{L}$ be a set of labels composed of a set of node labels $\mathscr{L}^V$, and of a set of edge labels $\mathscr{L}^A$. A* data graph *G is a tuple $G = \langle V, A, l_v \rangle$ where $l_v : V \mapsto \mathscr{L}^V$ is a node labelling function. The set of labelled edges is defined as $A \subseteq \{(x, \alpha, y) | x \in V, \alpha \in \mathscr{L}^A, y \in V\}$. We abbreviate an edge with $x \xrightarrow{\alpha} y$.*

Throughout the paper, we will use the following vocabulary to describe graphs in general.

Path We call a *path* a sequence of edges label $\alpha_1.\cdots.\alpha_n$. A path exists in the data graph G if there is a sequence of nodes $\{v_1, \cdots, v_{n+1}\} \subseteq V$ connected with such labelled edges. The Figure 1a depicts a graph of people and their relations with various places. The path *author.lives* exists in that graph, as it connects the nodes $\{N_0, N_2, N_3\}$.

Type Edge Among labelled edges, we refer as *type edges* those that define a type to which the node belongs to, e.g., the edge *type* in Figure 1a assigns the type *Person* to the node N_1. We denote the type edge with τ. We define $\mathscr{L}^T$ as the set of types occurring in G. Let $types(x)$ be the set of node labels related to the node x via a type edge; that is $types(x \in V) = \{l_v(c) | \forall (x, \tau, c) \in A\}$. We note that any set of edges may be defined as a type edge. It is assumed for simplicity that all type edges are renamed with a same label, i.e., *type*.

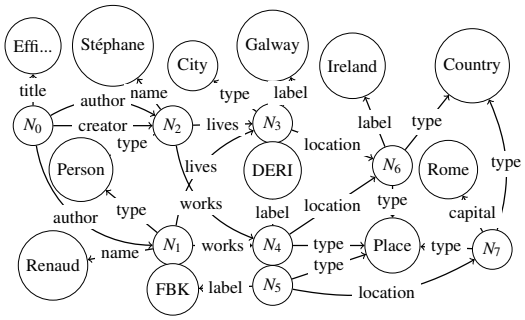
Attribute We refer as *attribute* the label of an edge, e.g., in the Figure 1a, the edge that connects N_0 to N_1 is labelled *author*. We define as $attributes(x)$ the set of attributes associated with a node x in G, i.e., $attributes(x \in V) = \{\alpha \in \mathscr{L}^A : (x, \alpha, y) \in A\}$.

2.2 Data Graph Summary

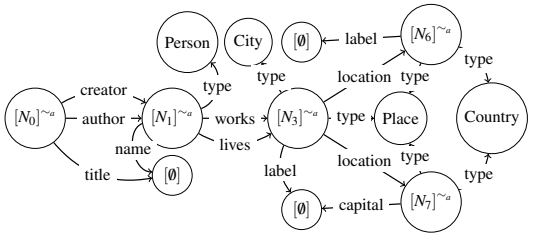
We call a graph summary [2], or *summary*, a graph $G^{\sim}$ that is homomorphic to the data graph G. The author in [32] first proposed the use of a summary in order to reduce the size of the data graph. A summary $G^{\sim}$ is constructed via an equivalence relation $\sim$ on the data graph G, where a node represents an equivalence class on nodes of G.

Definition 2.2 (Summary) *Let $\sim$ be an equivalence relation and $G^{\sim} = \langle V^{\sim}, A^{\sim}, l_v^{\sim} \rangle$ be the $\sim$-summary of G, where $l_v^{\sim}$ be a $\sim$-equivalence class labelling function. The nodes $V^{\sim}$ are the $\sim$-equivalence classes on nodes of G such that $V^{\sim} = \{[x]^{\sim} | x \in V\}$, with $[x]^{\sim} = \{y \in V | y \sim x\}$. For every edge $x \xrightarrow{\alpha} y$ of G, there is an edge $[x]^{\sim} \xrightarrow{\alpha} [y]^{\sim}$ in $G^{\sim}$, i.e., $A^{\sim} \subseteq \{(x, \alpha, y) | x \in V^{\sim}, \alpha \in \mathscr{L}^A, y \in V^{\sim}\}$.*

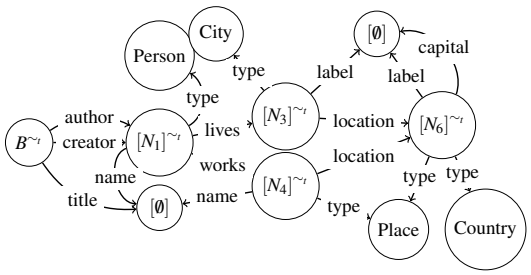
For example, let $\sim_t = \{(x, y) | types(x) = types(y)\}$ be a relation where two nodes are equivalent if both share the same set of types. The Figure 1c depicts the $\sim_t$-summary of the graph in the Figure 1a. The $\sim_t$-equivalence class $[N_1]^{\sim_t}$ contains the set $\{N_1, N_2\}$, since both are associated with *Person* via the



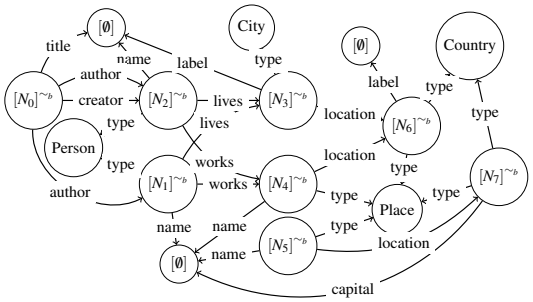
(a) A simple graph. The attribute *type* denotes a type edge.



(b) $\sim_a$-summary of the graph G in Figure 1a.



(c) $\sim_t$-summary of the graph G in Figure 1a.



(d) $\sim_b$-summary of the graph G in Figure 1a.

Figure 1: A data graph and three possible summaries. $[x]^{\sim}$ is the $\sim$-equivalence class of a node $x \in G$, $[\emptyset]$ is the sink equivalence class, and $B^{\sim}$ is the blank equivalence.

type edge. We note that the summary $G^{\sim}$ being a graph, the previously defined vocabulary still holds for a summary. For example, we have for the node $[N_1]^{\sim_a}$ in the Figure 1b $types([N_1]^{\sim_a}) = \{Person\}$ and $attributes([N_1]^{\sim_a}) = \{works, lives, name\}$.

In a data graph, the knowledge represented by a node is exprimed by outgoing edges. Therefore, sink nodes do not have any usable features for defining an equivalence class. We define the *sink equivalence class* as the set of sink nodes, e.g., the node labelled *Ireland* in the Figure 1a is part of this set. We remark that in the RDF data model, *literals* are part of the sink equivalence class.

Definition 2.3 (Sink Equivalence Class) *The set of sink nodes in G is in $G^{\sim}$ represented by the sink equivalence class $[\emptyset]$, i.e., $[\emptyset] = \{x \in V | \nexists (x, \alpha, y) \in A\}$.*

Depending on the $\sim$-equivalence, there can be nodes that do not have the required features to be assigned to a $\sim$-equivalence class. We define the *blank $\sim$-equivalence class* as the set of such nodes of G, e.g., the node N_0 in the Figure 1a is assigned to the node $B_t^{\sim}$ in the Figure 1c.

Definition 2.4 (Blank $\sim$-Equivalence Class) *The set of nodes in G for which there is no $\sim$-equivalent class is equal to the blank $\sim$-equivalence class $B^{\sim}$, i.e., $B^{\sim} = \{x \in V | \forall y \in V, x \not\sim y\}$.*

A set of summaries over a same data graph can be ordered using the relation $\sqsubseteq$ on the summaries [6]. Intuitively, imagine two equivalences $\sim_1$ and $\sim_2$, with $\sim_2$ having an additional constraint when

compared to $\sim_1$. Because of this constraint, $\sim_2$ produces equivalence classes on G that are included in those of $\sim_1$.

Definition 2.5 ($\sqsubseteq$ Ordering) *Let G be a data graph and let $\sim_1$ and $\sim_n$ be two equivalences on G. There is a partial order $\sqsubseteq$ on the $\sim_1$ and $\sim_2$ summaries, noted $\sim_1 \sqsubseteq \sim_2$, if and only if:* $\forall [x]^{\sim_1} \in V^{\sim_1}, \exists [x]^{\sim_2} \in V^{\sim_2}, [x]^{\sim_1} \subseteq [x]^{\sim_2}$.

For instance, we have $\sim_b \sqsubseteq \sim_t$ and $\sim_b \sqsubseteq \sim_a$ for the data graph in Figure 1a. However, there is no ordering relation between the $\sim_t$ and $\sim_a$ summaries, i.e., we have neither $\sim_t \not\sqsubseteq \sim_a$ because of the node $[N_6]^{\sim_t}$, nor $\sim_a \not\sqsubseteq \sim_t$ because of the node $[N_3]^{\sim_a}$.

3 Data Graph Summary Algorithms

In this section, we introduce several definitions of an equivalence relation for creating a data graph summary, which we order by the computation complexity.

$\sim_{st}$-equivalence We define $\sim_{st}$ the equivalence relation that relates two nodes if they share at least a same type. Formally:

$$\sim_{st} = \left\{(x,y) \in V^2 | \exists (x,\tau,c) \in A \wedge \exists (y,\tau,c) \in A\right\}$$

The nodes N_4, N_5, N_6, and N_7 in the Figure 1a are $\sim_{st}$-equivalent because they are associated with the class *Place*.

$\sim_t$-equivalence We define $\sim_t$ the relation for which two nodes are equivalent if they share the same set of types; formally:

$$\sim_t = \left\{(x,y) \in V^2 | types(x) = types(y)\right\}$$

The nodes N_6 and N_7 in the Figure 1a are $\sim_t$-equivalent because they are associated with the types *Place* and *Country*. The graph in Figure 1c is the $\sim_t$-summary of the data graph in Figure 1a.

$\sim_a$-equivalence Two nodes are equivalent as per $\sim_a$ if they share the same set of attributes; that is the equivalence relation:

$$\sim_a = \left\{(x,y) \in V^2 | \forall (x,\alpha_x,t_x) \in A, \forall (y,\alpha_y,t_y) \in A, \alpha_x = \alpha_y\right\}$$

The nodes N_3, N_4, and N_5 are $\sim_a$-equivalent because they share the same set of attributes, i.e., $\{label, location\}$.

$\sim_{at}$-equivalence We define $\sim_{at}$ the relation for which two nodes are equivalent if they share the same set of types and attributes; that is the equivalence relation:

$$\sim_{at} = \left\{(x,y) \in V^2 | x \sim_a y \wedge x \sim_c y\right\}$$

The nodes N_1 and N_2 are $\sim_{at}$-equivalent because they are associated with the types, i.e., *Person*, and they have the same attributes, i.e., $\{lives, works, name\}$.

$\sim_{ioa}$-equivalence We define $\sim_{ioa}$ the relation for which two nodes are equivalent if they share the same set of incoming and outgoing attributes; that is the equivalence relation:

$$\sim_{ioa} = \{(x,y) \in V^2 | \forall (s_x, \alpha_x, x) \in A, \forall (s_y, \alpha_y, y) \in A, \alpha_x = \alpha_y \\ \wedge x \sim_a y\}$$

All three nodes N_3, N_4, and N_5 are $\sim_a$-equivalent. Under the $\sim_{ioa}$ equivalence however, each is assigned to a different equivalence class, since all three have different incoming attributes set, i.e., $\{lives\}$, $\{works\}$, and $\emptyset$, respectively.

$\sim_{ioat}$-equivalence We define $\sim_{ioat}$ the equivalence where nodes share the same set of incoming and outgoing attributes, as well as the same set of types; that is the equivalence relation:

$$\sim_{ioat} = \left\{(x,y) \in V^2 | x \sim_{ioa} y \wedge x \sim_c y\right\}$$

The nodes N_1 and N_2 are $\sim_{at}$-equivalent, but are not according to the relation $\sim_{ioat}$, because the node N_1 and not N_2 has the incoming attribute *creator*.

For these relations, the complexity is equal to $O(|A|)$, since we need to visit each edge in order to assign the node at the tail to an equivalence class. For the algorithms $\sim_{ioa}$ and $\sim_{ioat}$, the incoming attributes set is computed by simply reversing the edges direction. Except for the $\sim_{st}$-equivalence, these equivalences assign a node to at most one equivalence class.

$\sim_b$-equivalence The forward and backward bisimulation [13], that we refer by *bisimulation* in the rest of the paper and is abbreviated with $\sim_b$, is an equivalence where two nodes are equivalent if they have the same set of outgoing and incoming paths. The definition of $\sim_b$ relies on the backward bisimulation $\sim_{bb}$, and on the forward bisimulation $\sim_{fb}$, which ensure the equivalence on the incoming and outgoing paths, respectively. Formally, that is $x \sim_b y$ if and only if:

$$\sim_{bb} = \begin{cases} 1. & x \sim_{at} y \\ 2. & \forall (s_x, \alpha_x, x) \in A, \exists (s_y, \alpha_x, y) \in A, s_x \sim_{bb} s_y \\ 3. & \forall (s_y, \alpha_y, y) \in A, \exists (s_x, \alpha_y, x) \in A, s_x \sim_{bb} s_y \end{cases}$$

$$\sim_{fb} = \begin{cases} 1. & x \sim_{at} y \\ 2. & \forall (x, \alpha_x, t_x) \in A, \exists (y, \alpha_x, t_y) \in A, t_x \sim_{fb} t_y \\ 3. & \forall (y, \alpha_y, t_y) \in A, \exists (x, \alpha_y, t_x) \in A, t_x \sim_{fb} t_y \end{cases}$$

$$\sim_b = \{(x,y) \in V^2 | x \sim_{bb} y \wedge x \sim_{fb} y\}$$

The $\sim_b$-summary depicted on the Figure 1d assigns a unique equivalence class to each node of the graph in Figure 1a. The sole difference of the summary with the data graph in this example is the abstraction from instance information via the sink equivalence class, e.g., the node *Ireland* is represented by the node $\emptyset^{\sim_b}$.

In order to compute the $\sim_b$-summary, the algorithm proposed in [30] is generally used. It offers a $O(|A| \times log(|V|))$ complexity for the computation for the $\sim_{fb}$-equivalence. We note the algorithm starts from an existing partitioning of the data graph, i.e., here a partitioning by the $\sim_{at}$-equivalence. In addition, the algorithm of the computation of $\sim_{fb}$ requires iterations over the data graph, the number of iterations equal to the data graph diameter in the worst case. However, iterative algorithms are sub-optimal for shared-nothing infrastructures, e.g., MapReduce. The reason is the data graph needs to be read and written at each iteration, creating an important IO load. The computation of the $\sim_b$-summary is achieved by applying over the $\sim_{fb}$ graph a slightly modified version of the [30] algorithm, in order to account for the incoming edges in the $\sim_{bb}$ definition. We remark the complexity of previous algorithms are significantly less than the complexity of $\sim_b$. Thus, it permits the previous algorithms to scale to large and heterogeneous data graphs using shared-nothing infrastructures.

4 Summary Quality

The quality of a summary depends on the equivalence relation used. In this section, we discuss the quality of a summary in terms of volume of the graph, and of precision of the summarized knowledge.

4.1 Summary Volume

It is desirable for a summary to be smaller than the data graph it is created from. One reason is the purpose of a summary is to be used instead of the data graph for increased performance. A summary is smaller if its size $|A^{\sim}|$ and its order $|V^{\sim}|$ are inferior to the size $|A|$ and the order $|V|$ of the data graph. We call the *volume* of a graph the sum of its size and order. Apart from $\sim_{st}$, the presented equivalence relations create a summary which is always smaller or equals to the data graph, since they assign a single equivalence class for each node of the data graph.

The $\sim_{st}$ equivalence however may assigns several equivalence classes to a node x, i.e., in the case where $|types(x)| > 1$. The order of a $\sim_{st}$-summary is at most equal to $|V^{\sim_{st}}| = |\mathscr{L}^C|$. However, in the worst case, its size can equal $|A^{\sim_{st}}| = |\mathscr{L}^C|^d$, with d being the diameter of the data graph. In the worst case, every node of the data graph can be associated with all the types, i.e., $\forall x \in V, types(x) = \mathscr{L}^C$. Therefore, for each path in G, we need to create an edge to each type for every node of the path.

The $\sim_a$-summary is sensible to heterogeneous data graph. Indeed, there can be a node in a $\sim_a$-summary for each element of the attribute powerset, i.e., $|V^{\sim_a}| = 2^{|\mathscr{L}^A|}$. Such an order of the summary can be expected for any equivalence based on the set of attributes, i.e., the equivalences $\sim_{at}$, $\sim_{ioa}$, $\sim_{ioat}$, and $\sim_b$.

We note that reducing the order and size of a summary comes with a price, that of reduced precision of the summary, which will be discussed next.

4.2 Summary Error

The confidence one may put on knowledge deduced from a summary has a more or less severe impact depending on the application. Also, the severity of an error depends on what information about the data graph is affected by the error. In this section, we introduce a model for measuring the precision of a summary.

4.2.1 Error Model

The data graph summary is a summary of the structure of the data graph. Therefore, errors in the summary boil down to the presence of incorrect edges. A path, or a combination of paths, may exists in the summary $G^{\sim}$, but not in the data graph G. This precision model accounts for the paths that exist in $G^{\sim}$, and not in G.

As per the recursive definition of $\sim_b$-equivalence, a node in the $\sim_b$-summary contains nodes of G that share the same incoming and outgoing paths. Thus, the $\sim_b$-summary is the most precise summary for a data graph with regards to the structure, i.e., all paths in $G^{\sim_b}$ do exist in G. According to the $\sqsubseteq$ relation, a $\sim_b$-summary is smaller than any of the other presented $\sim$ summaries. Thus, we have nodes of $G^{\sim}$ that contain several nodes of $G^{\sim_b}$. From the $\sim_b$-summary perspective, this creates non-existing edges between nodes of $G^{\sim_b}$, since those edges can be induced from $G^{\sim}$. We define $Err^{\sim}$ as the set of such induced edges.

Definition 4.1 (Summary Error $Err^{\sim}$) *Let $\sim$ be an equivalence relation on G. The set $Err^{\sim}$ is the set of non-existing edges between nodes of $G^{\sim_b}$ that can be deduced from $G^{\sim}$.*

$$Err^{\sim} = \{(u,\alpha,v) \in V^{\sim_b} \times \mathscr{L}^A \times V^{\sim_b} : \exists (x,y) \in u \times v, ([x]^{\sim}, \alpha, [y]^{\sim}) \in A^{\sim} \wedge ([x]^{\sim_b}, \alpha, [y]^{\sim_b}) \notin A^{\sim_b}\}$$

We note the $Err^{\sim}$ is equal to all possible combinations of nodes pair on G, keeping only the ones that exists in $G^{\sim}$ but not in $G^{\sim_b}$. The Figure 2 depicts the $\sim_t$-summary in Figure 1c from the perspective of the $\sim_b$-summary in Figure 1d. Sink equivalence classes are omitted for clarity. $G^{\sim_b}$ is depicted with solid lines, and $G^{\sim_t}$ with dashed lines. Edges from the set $Err^{\sim_t}$ are represented with red dotted arrows. As per the $\sqsubseteq$ relation, the nodes $[N_1]^{\sim_b}$ and $[N_2]^{\sim_b}$ of $G^{\sim_b}$ are included into the node $[N_1]^{\sim_t}$ of $G^{\sim_t}$. Similarly, the nodes $[N_4]^{\sim_b}$ and $[N_5]^{\sim_b}$ are within the node $[N_4]^{\sim_t}$. Since the edge $[N_1]^{\sim_t} \overset{works}{\rightarrow} [N_4]^{\sim_t}$ exists in $G^{\sim_t}$, this creates non-existing edges *works* in $G^{\sim_b}$ from the nodes $[N_1]^{\sim_b}$ and $[N_2]^{\sim_b}$ to $[N_5]^{\sim_b}$. These edges are part of the $Err^{\sim_t}$ set. Doing as such for each node of $G^{\sim_t}$, this causes the nodes $[N_1]^{\sim_b}$, $[N_5]^{\sim_b}$, and $[N_6]^{\sim_b}$ to be connected. The non-existing path $works.location.capital$ can then be deduced from the $\sim_t$-summary.

From the perspective of $G^{\sim_b}$, a $\sim$-summary creates additional edges. We call $\mathscr{G}^{\sim}$ the *virtual* $\sim_b$-summary that is expanded with these additional edges.

Definition 4.2 (Virtual $\mathscr{G}^{\sim}$ Summary) *Let $\sim$ be an equivalence relation and $\sim_b$ the bisimulation relation on a data graph G. We define $\mathscr{G}^{\sim}$ as the $\sim_b$-summary graph, with in addition the edges from the $Err^{\sim}$ set.*

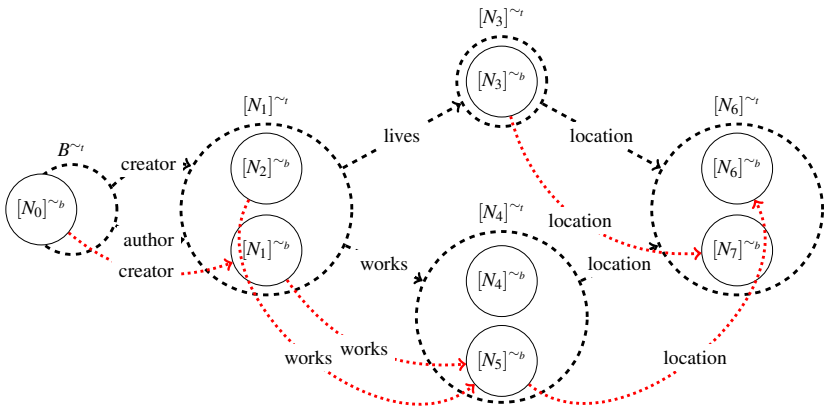


Figure 2: Superposition of $G^{\sim_b}$ in the Figure 1d with $G^{\sim_t}$ in the Figure 1c. Sink equivalence classes are omitted for clarity. $G^{\sim_b}$ is depicted with solid lines, and $G^{\sim_t}$ with dashed lines. Edges in the $Err^{\sim}$ set, i.e., additional edges in $G^{\sim_b}$ induced by $G^{\sim_t}$, are represented with red dotted arrows.

In the Figure 2, we can see part of the virtual $\mathscr{G}^{\sim_t}$ summary: the nodes are depicted with a solid line, and edges both with dashed and red dotted lines.

Depending on the kind of the edge in the $Err^{\sim}$ set, we derive three classifications of the edge, i.e. *connectivity*, *attribute*, and *type*. The *connectivity* category reflects errors of a summary with regards to the structure of the data graph, while the *attribute* and *type* categories with regards to its schema.We illustrate $Err^{\sim}$ with regards to the $\sim_t$ and $\sim_a$ summaries only, since all others are smaller than either as per the $\sqsubseteq$ relation. Thus, any errors these summaries may have are experienced with either $\sim_t$ or $\sim_a$.

Connectivity Error $Err^{\sim}_{conn}$ The connectivity error $Err^{\sim}_{conn}$ captures false positive paths in the $\sim$-summary, i.e., paths that do not exists in $G^{\sim_b}$ but do in $G^{\sim}$. For example, the Figure 2 depicts the false positive path *creator* from the node $[N_0]^{\sim_b}$ to $[N_1]^{\sim_b}$. We illustrate the connectivity error as the chance of following an edge from $Err^{\sim}$ on the virtual graph. We do not consider the sink equivalence class in the connectivity error, since it doesn't provides any outgoing edge. We define $Err^{\sim}_{conn}$ as the set $Err^{\sim}$ minus the edges leading to the sink equivalence class:

$$Err^{\sim}_{conn} = \{(x, \alpha, y) \in Err^{\sim} : y \neq [\emptyset]\}$$

Attribute Error $Err^{\sim}_{attr}$ The attribute error captures false positive edges that impact on the schema of the data graph, due to additional attributes induced by the $\sim$-summary in $G^{\sim_b}$. For example, the node $[N_6]^{\sim_t}$ in Figure 1c implies the existence of a node in G with edges *capital* and *label*, which is not the case. We define $Err^{\sim}_{attr}$ as the set that contains edges $x \xrightarrow{\alpha} y$ of $Err^{\sim}$ which label α is not in the $attributes(x)$ set of the tail x.

$$Err^{\sim}_{attr} = \{(x, \alpha, y) \in Err^{\sim} : \alpha \notin attributes(x)\}$$

Type Error $Err^{\sim}_{type}$ The type error captures false positive edges that impact on the schema of the data graph, due to additional types induced by the $\sim$-summary in $G^{\sim_b}$. For example, the Figure 1b depicts the $\sim_a$-summary of the graph in Figure 1a. The node $[N_3]^{\sim_a}$ contains the data graph nodes $\{N_3, N_4, N_5\}$, since all three have the same set of edges labels, i.e., *label*, *location* and *type*. Then, we may infer from this node the possible existence of a node in G with types *Place* and *City*, which is however not the case. We define $Err^{\sim}_{type}$ as the set that contains additional types from $Err^{\sim}$, i.e., a type edge $x \xrightarrow{\tau} y$ in $Err^{\sim}$, with the head y not in the $types(x)$ set of the tail x.

$$Err^{\sim}_{type} = \{(x, \tau, y) \in Err^{\sim} : y \notin types(x)\}$$

4.3 Edge Precision

A visitor of the virtual $\mathscr{G}^{\sim}$ summary can follow edges from the $G^{\sim_b}$ summary, but also from the $Err^{\sim}$ set. From the $G^{\sim_b}$ point of view, the former are *true* positive edges, while the latter are *false* positive edges. We define $Prec^{\sim}(x)$ the *edge precision* of a node x in $G^{\sim_b}$ as the proportion of the true positives against all the positive edges.

$$\begin{aligned} TP(x) &= \{\forall(\alpha, y) \in \mathscr{L}^A \times V^{\sim_b} : (x, \alpha, y) \in A^{\sim_b}\} \\ FP(x) &= \{\forall(\alpha, y) \in \mathscr{L}^A \times V^{\sim_b} : (x, \alpha, y) \in Err^{\sim}\} \\ Prec^{\sim}(x) &= \frac{|TP(x)|}{|TP(x) \bigcup FP(x)|} \end{aligned}$$

The set $TP(x)$ contains the true positive edges which tail is x. For example, the edge $[N_0]^{\sim_b} \overset{creator}{\rightarrow} [N_2]^{\sim_b}$ is in $TP(x)$, since it does exist in the $\sim_b$-summary in Figure 2. The set $FP(x)$ contains the false positive edges which tail is x. For example, the edge $[N_0]^{\sim_b} \overset{creator}{\rightarrow} [N_1]^{\sim_b}$ is in $FP(x)$, since it is an additional edges in the figure. For example, the edge precision of the node $[N_6]^{\sim_b}$ is equal to $\frac{1}{2}$, since it has the true positive edge *label*, and the false positive edge *capital*. The *probability interpretation* of $Prec^{\sim}(x)$ is the probability that a randomly selected edge is correctly summarized, i.e., a true positive edge. We note the recall is always equal to 1, since there is no false negative edges in a summary.

We use $Prec^{\sim}(x)$ as the precision measure for each of the error classifications. We note that for a same node, the edge precision may vary between the classifications. As an example, consider the node $[N_0]^{\sim_b}$ of the Figure 2. The edge precision for the attribute error is equal to 0, since the attributes in both $G^{\sim_b}$ and $\mathscr{G}^{\sim_t}$ graphs for this node are *creator* and *author*. However, the edge precision for the connectivity error is equal to $\frac{3}{4}$. Both edges *author* and *creator* which heads is the $[N_2]^{\sim_b}$ node are true positives. There are also two edges to $[N_1]^{\sim_b}$, but only the edge with attribute *author* is a true positive. Therefore, for the connectivity error, three edges out of four are true positives for the node $[N_0]^{\sim_b}$.

5 Evaluation

In this section, we evaluate the trade-off between the complexity of a summarization algorithm, the volume, and the precision of the graph summary.

5.1 Design

In this section, we describe the design of the evaluation. We first present the environment of our experimental framework. Then, we describe the dimensions of our evaluation.

5.1.1 Environment

The $\sim_b$-summary is the most accurate summary of the data graph, since all incoming and outgoing (and combination of) paths in $G^{\sim_b}$ do exist in the data graph G. In this evaluation, we use the $\sim_{fb}$ as the gold standard summary of a data graph, which ensures that all outgoing paths do exist. The presented summarization algorithms have been implemented on the Hadoop[3] MapReduce framework. Our Hadoop cluster is composed of 10 machines.

5.1.2 Evaluation Dimensions

We present in this section the three dimensions we base this evaluation on, i.e., the *volume*, the *computational complexity* of the summarization algorithm, and the *precision* of the summary.

Summary Volume We measure the amount of data from the gold standard summary that is compressed into the evaluated summary. To this end, we compare the volume of the $\sim$-summary against the volume of the gold standard. We report this comparison as the ratio $G^{\sim} : G^{\sim_{fb}}$ of the former to the later, i.e., $G^{\sim} : G^{\sim_{fb}} = \frac{|V^{\sim}|+|A^{\sim}|}{|V^{\sim_{fb}}|+|A^{\sim_{fb}}|}$.

Computational Complexity The MapReduce implementation of the graph summarization is composed of two separate steps: (1) a *mapping* step, where we assign a node of the data graph to its $\sim$-equivalence class; and (2) a *edges* step, where we compute the edges of $G^{\sim}$. We evaluate the computational complexity of a $\sim$-equivalence by analyzing the CPU time[4] on the *edges* step of the graph summary computation. We do not report on the *mapping* step because the evaluated algorithms have the same complexity, achieving similar times on this step.

Summary Precision With regards to all three error classifications, we evaluate the precision of a summary thanks to the true positive $TP(x)$ edges set, and to the false positive edge set $FP(x)$. We report the precision using two measures, i.e., $P1$ and $P2$. The measure $P1$ reflects the average number of true positive edges from a randomly selected node in the virtual $\mathscr{G}^{\sim}$ summary. The measure $P2$

[3]Hadoop: `http://hadoop.apache.org/`

[4]The cumulated CPU time as reported with the `CPU_MILLISECONDS` counter of Hadoop.

reflects the overall chance for a randomly selected edge in the virtual $\mathscr{G}^{\sim}$ summary of being a true positive.

$$C(c \in V^{\sim}) = \{x \in V^{\sim_b} : x \in c\}$$

$$P1 = \frac{1}{|V^{\sim}|} \times \sum_{c \in V^{\sim}} \frac{1}{|C(c)|} \times \sum_{x \in C(c)} Prec^{\sim}(x)$$

$$P2 = \frac{\sum_{x \in V^{\sim_b}} TP(x)}{\sum_{x \in V^{\sim_b}} TP(x) + FP(x)}$$

The set $C(c)$ is the set of nodes in $G^{\sim_b}$ that are included into the node c in $G^{\sim}$, with regards to the $\sqsubseteq$ relation.

5.2 Datasets

We use in this evaluation several datasets of varying processing complexity. We list in the Table 1 the datasets we use, along with descriptive statistics. The datasets are grouped according to their processing complexity into four categories, i.e., *Low*, *Medium*, *High*, and *Very High*. For each dataset, we present two aspects, i.e., the schema and the structure of the data graph. With regards to the schema complexity, we report the number of unique edge labels $|\mathscr{L}^A|$ and the number of unique types $|\mathscr{L}^T|$ of the data graph. With regards to the structure complexity, we report the size and order of the data graph G, but also of the $\sim_{fb}$-summary $G^{\sim_{fb}}$. The order values omit the number of sink nodes, i.e., nodes without outgoing edges in addition to literal nodes in the RDF data model. The $G^{\sim_{fb}} : G$ column reports the volume ratio of $G^{\sim_{fb}}$ to G as a percentage, where the volume is the sum of the size and order of the graph. We remark that the volume of the $\sim_{fb}$-summary is significantly smaller than the volume of the data graph. This emphasize the performance benefits of using a summary instead of the data graph itself in an application. We note that the bigger the volume ratio, the more complex the data graph is to summarise from a structural point of view. The datasets `bnb` and `world-bank-ld` have a similar volume, however, the volume ratio of the summary for the former is greater than for the latter, i.e., 1.17% to 0.01%. This shows that the structure of `bnb` is more heterogeneous than of `world-bank-ld`.

5.3 Results

We evaluate and compare in this section the different graph summary algorithm according to the volume, the computational complexity, and the precision. Then, we discuss the trade-offs according to these three dimensions.

5.3.1 Graph Summary Volume

The Table 2a reports the volume ratio of a $\sim$-summary to the gold standard $\sim_{fb}$-summary. We report also the mean μ for each category complexity of the datasets. The algorithms based only on the type information, i.e., $\sim_{st}$ and $\sim_{t}$, exhibit a high compression of the gold standard. With regards to the gold standard volume, the volume ratio is under 5% on the low and medium datasets, and at most half on the

Dataset	Schema		G		$G^{\sim_{fb}}$		$G^{\sim_{fb}} : G$
	$\lvert\mathscr{L}^T\rvert$	$\lvert\mathscr{L}^A\rvert$	$\lvert V\rvert$	$\lvert A\rvert$	$\lvert V^{\sim_{fb}}\rvert$	$\lvert A^{\sim_{fb}}\rvert$	
Very High							
dbpedia [19]	288 524	22 369	65 042 837	233 051 608	6 884 121	33 990 765	13.71%
High							
twc-logd [25]	450	10 060	3 398 947	67 505 792	5 565	121 873	0.18%
enipedia [22]	128	267	413 520	4 463 566	1037	21 419	0.46%
Medium							
b3kat [18]	10	30	85 795 956	592 778 746	782 056	3 230 936	0.59%
ecs [21]	14	120	167 390	955 112	8 232	51 494	5.32%
lobid [23]	19	46	124 691 274	625 941 644	51 529	746 099	0.11%
bnb [26]	27	53	12 246 306	89 733 453	146 956	1 046 714	1.17%
datos [24]	23	143	7 412 312	58 048 932	360 822	2 504 262	4.38%
gnd [20]	22	35	962 930	7 940 373	9 664	88 875	1.11%
eures [12]	18	49	288 862	4 146 421	2 205	37 052	0.89%
Low							
europeana [9]	5	58	5 559 452	40 773 834	4 792	72 125	0.17%
world-bank-ld [3]	4	174	11 210 832	84 345 613	293	6 987	0.01%
cordis [11]	7	63	729 780	7 101 623	2 245	36 783	0.50%
ny-times [33]	2	38	22 662	345 888	65	794	0.23%

Table 1: Descriptive statistics of the datasets. The Schema column reports the number of unique edges labels $|\mathscr{L}^T|$ in the dataset G, and also the number of unique types $|\mathscr{L}^A|$. The column G reports the order and the size of the data graph. The column $G^{\sim_{fb}}$ reports the order and the size of the $\sim_{fb}$-summary of the data graph G. The $G^{\sim_{fb}} : G$ column reports the volume ratio of $G^{\sim_{fb}}$ to G as a percentage.

more complex datasets. On average, we note that the volume of a summary with $\sim_{st}$ is slightly higher than with $\sim_t$. The reason is a node of the data graph can be in several $\sim_{st}$-equivalence class, leading to an increase of the size of the summary. The $\mathscr{L}^A$ feature appears to be better for a summarization algorithm than the $\mathscr{L}^T$ feature. Indeed, the volume ratio of $\sim_a$ and $\sim_{ioa}$ remain stable from the medium to high datasets, i.e., from 41.85 to 47.99 (resp., from 50.65 to 66.13). However, this is not the case with the $\sim_{st}$ and $\sim_t$ algorithms. We can observe from this a correlation between the volume ratio and the size of $\mathscr{L}^T$. We note that the $\sim_a$ algorithm actually achieves the lowest ratio on the `dbpedia` dataset. This shows that the use of attributes on `dbpedia` is homogeneous across types. The volume ratio of $\sim_{ioa}$ is on average on par with the ratio of $\sim_a$, indicating a certain homogeneity of incoming edges in the data graph. By using both type and attribute information as with $\sim_{at}$, we remark that this can lead to a significant increase of the summary volume. Indeed, the combination of the features on `b3kat` and `lobid` exhibit a much higher ratio than when taken separately, e.g., 55.87 and 84.06 respectively with $\sim_{at}$, against 27.43 and 49.83 with $\sim_a$. This can be explained by a sparse usage of attributes with a type, creating an explosion of combinations. In general, the type feature seems less reliable for an algorithm than the attribute feature.

5.3.2 Computational Complexity

The Table 2b reports the CPU time in *ms* of the edges step in the graph summarization computation. The times are based on a single run of an algorithm for a certain dataset. Due to an error in the computation, we don't report the times for $\sim_{at}$ in this table. On the medium and high datasets, the time taken by the $\sim_{st}$ algorithm in the edge step is higher than any other algorithm. This highlights the property of the $\sim_{st}$ algorithm of assigning more than one $\sim_{st}$-equivalence class to a single node of that data graph. As

Dataset	$G^{\sim} : G^{\sim_{fb}}$ (in %)					
	$G^{\sim_{st}}$	$G^{\sim_t}$	$G^{\sim_a}$	$G^{\sim_{at}}$	$G^{\sim_{ioa}}$	$G^{\sim_{ioat}}$
dbpedia	45.00	51.81	4.71	59.10	6.41	60.56
twc-logd	24.27	23.99	35.69	48.96	47.13	60.13
enipedia	11.35	11.20	60.28	76.14	85.14	98.76
μ	*17.81*	*17.60*	*47.99*	*62.55*	*66.13*	*79.45*
b3kat	0.02	0.13	27.43	55.87	27.99	56.49
ecs	0.90	0.55	23.78	31.73	29.22	37.22
lobid	0.09	0.26	49.83	84.06	63.32	97.92
bnb	0.03	0.03	19.95	20.67	20.53	21.26
datos	0.02	0.01	44.59	44.60	44.74	44.76
gnd	0.32	0.32	36.37	40.91	78.41	85.12
eures	0.27	0.20	95.62	95.71	95.93	95.98
μ	*0.23*	*0.22*	*42.51*	*53.37*	*51.45*	*62.68*
europeana	0.09	0.09	72.57	72.57	72.57	72.57
world-bank-ld	3.02	2.55	91.09	91.09	100.00	100.00
cordis	0.25	0.25	77.13	77.15	85.71	85.72
ny-times	4.19	4.19	86.15	86.15	100.00	100.00
μ	*1.89*	*1.77*	*81.73*	*81.74*	*89.57*	*89.57*

(a) Volume ratio comparison. For each category of dataset complexity, we report the mean μ of the volume ratio.

Dataset	$\sim_{st}$	$\sim_t$	$\sim_a$	$\sim_{at}$	$\sim_{ioa}$	$\sim_{ioat}$
dbpedia	203 914 640	73 754 210	116 698 410	-	102 065 040	99 506 800
twc-logd	12 916 030	9 064 630	11 009 020	-	10 468 330	10 507 360
enipedia	2 331 170	2 451 350	1 862 870	-	1 800 940	1 928 800
μ	*7 623 600,00*	*5 757 990,00*	*6 435 945,00*	-	*6 134 635,00*	*6 218 080,00*
b3kat	153 671 180	169 688 740	153 913 260	-	139 000 290	148 529 710
ecs	1 217 620	1 609 670	1 295 370	-	1 459 700	1 291 470
lobid	219 378 920	201 144 890	220 755 490	-	198 931 260	196 103 920
bnb	30 609 340	27 971 020	25 154 300	-	22 940 100	23 117 820
datos	10 351 980	12 869 960	10 363 740	-	9 126 940	9 108 480
gnd	2 368 920	2 853 300	2 250 990	-	2 117 530	2 223 320
eures	1 734 400	1 742 290	2 097 560	-	2 046 310	2 150 850
μ	*59 904 622,86*	*59 697 124,29*	*59 404 387,14*	-	*53 660 304,29*	*54 646 510,00*
europeana	9 955 670	12 267 870	10 359 330	-	9 390 360	9 339 710
world-bank-ld	14 972 490	18 018 890	14 598 010	-	12 846 900	12 828 300
cordis	2 369 730	2 879 820	2 257 780	-	2 374 730	2 253 350
ny-times	718 720	1 051 320	755 740	-	850 130	761 990
μ	*7 004 152,50*	*8 554 475,00*	*6 992 715,00*	-	*6 365 530,00*	*6 295 837,50*

(b) Computational complexity comparison. We report the CPU time, in *ms*, of the edges step in the graph summarization computation. For each category of dataset complexity, we report the mean μ of the CPU time. Due to an error in the computation, we don't report the times for $\sim_{at}$.

Table 2: Comparison of the volume and the algorithm efficiency.

a consequence, we need to compute in the edges step all the possible combinations of edges between equivalence classes. We can see in the Table 2a that the type-based candidates $\sim_{st}$ and $\sim_t$ provide the smallest volume. However, we observe in the Table 2b that the candidates providing the largest ratio are actually the fastest in general, i.e., $\sim_{ioa}$ and $\sim_{ioat}$. This suggests that summaries that are closest to the gold standard as per the *sqsubseteq* exhibit a distribution of edges that is less skewed, achieving thus a lower computational complexity on a shared-nothing environment such as Hadoop.

5.3.3 Graph Summary Precision

We discuss in this section the precision of a graph summary with regards to the connectivity error first, then to the type and attribute error next. We do not report the precision in any error classification for the dbpedia dataset. The reason is we were unable to evaluate the precision on it due to performance issues.

The Table 3 reports the precision results $P1$ and $P2$ for the connectivity error. For each category of dataset complexity, we report the mean μ of the connectivity precision. The algorithms based only on the type feature, i.e., $\sim_{st}$ and $\sim_t$, provide a low connectivity precision. Indeed, they show on average a connectivity precision below 20% according to $P1$, i.e., $\mu = 0.1876$ for the high dataset complexity. Algorithms based on the attribute feature only provide also a low precision on medium and high categories. On the low category, the attribute feature exhibit a better precision than the type, i.e., 0.5608 against 0.3528 on average. However, when the type attribute features are combined in $\sim_{at}$, it provides a significant increase of the precision. According to $P1$, we reach on average a 50% connectivity precision on the high category, and at least 20% on the medium category. We remark that the incoming attribute in $\sim_{ioa}$ is an important feature to increase the precision. It provides a precision of 30% on the medium category up to 48% on high, against 15% down to 10%, respectively. Overall, we can achieve a good average connectivity precision with $\sim_{ioat}$ according to $P1$. However, the overall

Dataset	$Err^{\sim_{st}}_{conn}$		$Err^{\sim_{t}}_{conn}$		$Err^{\sim_{a}}_{conn}$		$Err^{\sim_{at}}_{conn}$		$Err^{\sim_{ioa}}_{conn}$		$Err^{\sim_{ioat}}_{conn}$	
	$P1$	$P2$	$P1$	$P2$	$P1$	$P2$	$P1$	$P2$	$P1$	$P2$	$P1$	$P2$
twc-logd	0.0080	0.0001	0.1709	0.0000	0.1247	0.0154	0.5572	0.0228	0.2700	0.0219	0.7463	0.0276
enipedia	0.0071	0.0005	0.2044	0.0070	0.0608	0.0335	0.4382	0.1988	0.7067	0.3069	0.9972	0.3718
μ	*0.0075*	*0.0003*	*0.1876*	*0.0035*	*0.0928*	*0.0244*	*0.4977*	*0.1108*	*0.4884*	*0.1644*	*0.8717*	*0.1997*
b3kat	0.0000	0.0000	0.0625	0.0000	0.4382	0.0001	0.5162	0.0001	0.4395	0.0001	0.5217	0.0001
ecs	0.0710	0.0001	0.0662	0.0000	0.0084	0.0049	0.0160	0.0053	0.0887	0.0093	0.0772	0.0104
lobid	0.0355	0.0000	0.0235	0.0001	0.2329	0.0047	0.3322	0.0072	0.2790	0.0136	0.3906	0.0218
bnb	0.0044	0.0000	0.0025	0.0001	0.0037	0.0012	0.0136	0.0013	0.0091	0.0013	0.0247	0.0013
datos	0.0135	0.0000	0.0483	0.0000	0.0597	0.0000	0.0921	0.0000	0.1618	0.0000	0.1630	0.0000
gnd	0.0073	0.0001	0.0073	0.0001	0.1566	0.0091	0.1802	0.0131	0.7408	0.0617	0.8601	0.0880
eures	0.0203	0.0005	0.3268	0.0002	0.1211	0.1139	0.3037	0.2486	0.5419	0.4317	0.5422	0.4317
μ	*0.0217*	*0.0001*	*0.0767*	*0.0001*	*0.1458*	*0.0192*	*0.2077*	*0.0394*	*0.3230*	*0.0740*	*0.3685*	*0.0791*
europeana	0.4450	0.0546	0.4450	0.0546	0.5702	0.4922	0.5702	0.4922	0.5702	0.4922	0.5702	0.4922
world-bank-ld	0.6080	0.0476	0.5750	0.0476	0.9382	0.8711	0.9382	0.8711	1.0000	1.0000	1.0000	1.0000
cordis	0.0083	0.0143	0.0083	0.0143	0.1590	0.1266	0.1592	0.1269	0.3868	0.2602	0.3868	0.2602
ny-times	0.3830	0.0798	0.3830	0.0798	0.5756	0.6813	0.5756	0.6813	1.0000	1.0000	1.0000	1.0000
μ	*0.3611*	*0.0491*	*0.3528*	*0.0491*	*0.5608*	*0.5428*	*0.5608*	*0.5429*	*0.7393*	*0.6881*	*0.7393*	*0.6881*

Table 3: Connectivity precision comparison. For each category of dataset complexity, we report the mean μ of the connectivity precision.

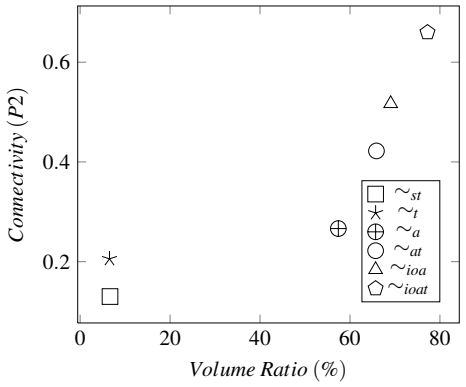
	$P1$	$P2$	$P1$	$P2$	$P1$	$P2$	$P1$	$P2$	$P1$	$P2$	$P1$	$P2$
	$Err^{\sim_{st}}_{type}$		$Err^{\sim_{t}}_{type}$		$Err^{\sim_{a}}_{type}$		$Err^{\sim_{at}}_{type}$		$Err^{\sim_{ioa}}_{type}$		$Err^{\sim_{ioat}}_{type}$	
μ_{high}	0.6576	0.0000	1.0000	1.0000	0.9453	0.0000	1.0000	1.0000	0.9580	0.0000	1.0000	1.0000
μ_{medium}	0.7114	0.3687	1.0000	1.0000	0.8944	0.5492	1.0000	1.0000	0.9190	0.6175	1.0000	1.0000
μ_{low}	1.0000	1.0000	1.0000	1.0000	0.9999	0.9989	1.0000	1.0000	0.9999	0.9998	1.0000	1.0000
	$Err^{\sim_{st}}_{attr}$		$Err^{\sim_{t}}_{attr}$		$Err^{\sim_{a}}_{attr}$		$Err^{\sim_{at}}_{attr}$		$Err^{\sim_{ioa}}_{attr}$		$Err^{\sim_{ioat}}_{attr}$	
μ_{high}	0.5625	0.0387	0.8460	0.0325	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
μ_{medium}	0.4934	0.2321	0.5540	0.2584	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
μ_{low}	0.7568	0.5177	0.7512	0.4929	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 4: Schema precision comparison. We report the mean μ of the attribute and type precision over three datasets categories, i.e., low, medium, and high.

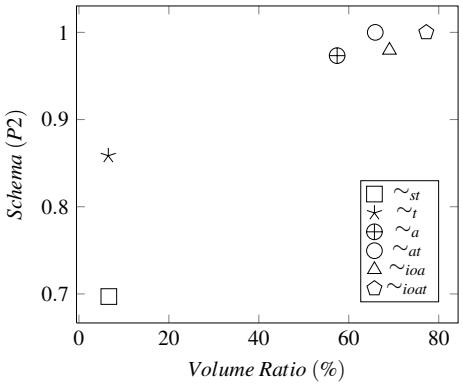
precision $P2$ is very low on the complex datasets of the medium and high categories. This suggests that few nodes of the summary have a high out-degree, creating a combinatorial explosion of false positive edges. This will be investigated in future work.

The Table 4 reports the precision results $P1$ and $P2$ for the type and attribute errors, i.e., regarding the schema of the summary. the table contains the mean μ of the attribute and type precision over three datasets categories, i.e., low, medium, and high. The $\sim_{st}$ algorithm provides no false positive type edge on the low datasets only. On the low and medium datasets, the type feature only shows at most 84% of true positive attributes, i.e., $P1 = 0.8460$ for $Err^{\sim_{t}}_{attr}$. On the contrary, the attribute feature reports a good type precision, reaching on average at least 90% of true positive attribute for $\sim_a$, e.g., $P1 = 0.8944$ on the medium datasets. Incoming attributes do not increase much the type precision, since the type precision of $\sim_a$ stays on par with $\sim_{ioa}$. The $\sim_{at}$ algorithm provides a perfect summarization of the schema. Again, the significant differences between $P1$ and $P2$ suggests on more time that few nodes of the summary contains a high out-degree, creating a combinatorial explosion of false positive edges.

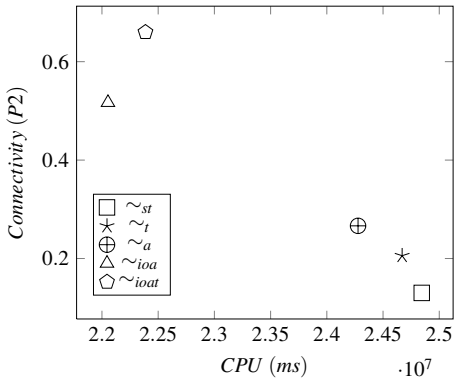
In conclusion, we can see that a combination of both type and attribute features is necessary to achieve a good precision. The results show that taking incoming attributes as a feature of the summarization algorithm is important for connectivity precision, but not for the schema. However, the reported connectivity precision is still low. We observe that the precision $P2$ leads to very low precision values which is caused by a few summary nodes with a high out-degree. This indicates that the model of $P2$



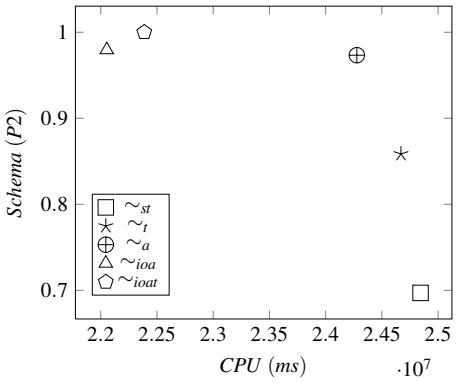
(a) Connectivity precision versus volume ratio.



(b) Schema precision versus volume ratio.



(c) Connectivity precision versus computational complexity.



(d) Schema precision versus computational complexity.

Figure 3: Efficiency and precision trade-offs of the candidate summary algorithms. The values are taken as the average across all dataset categories.

is not appropriate for measuring the precision of a summary in terms of connectivity and schema.

5.3.4 Trade-Offs

We report in Figure 3a the trade-off between the average connectivity precision and the average volume ratio across all datasets among all the algorithms. We can distinguish two groups of algorithms, the type algorithms, i.e., $\sim_t$ and $\sim_{st}$, and the attribute algorithms. The type algorithms provide the best volume ratio, but also the worse precision. In the attribute group, the volume ratio among algorithms is close to each others, but their precision differs greatly, with $\sim_{ioat}$ ahead. This suggests that in term of trade-off between connectivity precision and volume, $\sim_{ioat}$ is the best candidate. We report in Figure 3b the trade-off between the average schema precision and the average volume ratio across all datasets among all the algorithms. Again we can distinguish the same two groups. However, in the attribute group, the precision does not differ too much among the candidates, each one being either equal or very close to 1. In the type group, the $\sim_t$ algorithm provides a quite reasonable precision for a very small volume ratio. This suggests that if precision is primordial, the $\sim_{at}$ algorithm is the best candidate, providing a

perfect schema precision for the smallest volume. However, if volume is primordial, and imperfection can be tolerated, then the $\sim_t$ algorithm is the best candidate.

We report in Figure 3c (resp., 3d) the trade-off between the average connectivity precision (resp., average schema precision) and the average CPU time across all datasets among all the algorithms. We can distinctively see that $\sim_{ioa}$ and $\sim_{ioat}$ provide the best connectivity and schema precision, as well as the best CPU time. This suggests that if CPU time is important, these algorithms are the best candidates, and in addition they will also provide the best precision.

6 Related Work

In various Computer Science areas, the information is represented as and is analysed through graphs. Graphs are used to capture the social relationship between people. Graphs represent processus and their transition in concurrency. Graphs are used to structure data in a flexible way, e.g., using the Object Exchange Model [31] or, more recently, in the Resource Description Framework [16].

A common challenge encountered when analysing a graph is its size. A large number of nodes or edges reduce the scalability and increase the running time of applied algorithms. A solution is to translate the graph into another smaller graph, while preserving the structure. The smaller graph is then seen as a summary of the original data graph. [27] introduce the concept of bisimulation to define equivalent processes in concurrent systems. The relational coarsest partition [30] is generally the algorithm used for computing a bisimulation on a graph. With a similar perspective in reducing the size of the graph, DataGuides [7] is the first work to propose a summary of the structure of the graph as an index, which is then used for improving the execution of queries.

However, the DataGuides construction algorithm and the bisimulation are computationally expensive. Also, in presence of data with a complex structure, the size of the summary can be as large as the data graph, or even larger with DataGuides. This issue is discussed in [8], where some constraints of DataGuides are relaxed, e.g., the existence of a path in the data graph.

The complexity of both real-world data graphs and summarisation algorithms highlight the need for approximate summaries, i.e., summaries ideally smaller and simpler to compute, at the price of presence of errors with regards to the data graph. [28] extend the work on DataGuides, using the notion of bisimulation to build a structure index. The authors propose to reduce further the size of a summary by defining the type of query to answer. However, it assumes some knowledge about the structure of the data graph in order to specify the query type. Also, any change in the query requirements necessitate to rebuild the summary. [13, 14] simplify the definition of bisimulation on a graph by limiting the length of a path to k hops. In [4], the authors vary the maximum length of a path per node, depending on the query load of the system. The summarisation approach proposed in [29] is grounded on Information Theory for the representation of a data graph. Under the Minimum Description Length principle, the best representation is the one that is the smallest in size. The representation is composed of two components: a graph summary and a set of edge corrections. In association with the graph summary, the set of edge of corrections contains the edges that were created or deleted by the summarisation

process. Using this representation, it is possible to re-create the original graph. In addition, the authors propose a graph representation with a user-defined bounded error ε. The computed representation has at most ε errors compared to original graph; however, the loss in precision is balanced by the reduced size of the graph representation. [35] uses a similar approach to [13] for the purpose of improving the partitioning of RDF data and the execution of queries. [34] propose an approach based on bisimulation that provides more or less detailed summaries, which the detail level is up to the user. However, this assumes that the end user has some knowledge about the data. We focus instead on an approach with no prior knowledge requirement.

These approaches of the summary seek to reduce the complexity of algorithms, or the size of the summary. However, the proposed approaches as well as applications of a summary [5, 10, 15], do not scale with large data graph, composed of billions of nodes and edges. [17] propose an implementation of bisimulation over MapReduce. The iteration aspect of the bisimulation necessitate to read and write the data graph across the network of computing machines. However, this creates an important IO load on the network. In this paper, we are interested instead in summarization algorithms on such a shared-nothing environment, which computation complexity scale gracefully regardless of the structure heterogeneity of the data graph.

7 Conclusion

In this paper, we have investigated several approximate graph summarisation algorithms that are efficient to compute on a shared-nothing infrastructure such as Hadoop. To that end, we have introduced a precision model for evaluating the accuracy of graph summary algorithms compared to a gold standard summary. This precision model takes into account two aspects of the graph summary: the structural summary and the schema summary. We then analyse the trade-offs between the efficiency and the precision of the graph summarisation algorithms. We have performed the evaluation of the algorithms over 15 real-world datasets of various size and complexity. The experimental results show that it is possible to approximate quite accurately the gold standard graph summary but with a much lower space and time complexity. The experimental results also provide initial evidences about the applicability of the algorithms in different context, e.g., schema summarisation or structure summarisation. Future work will concentrate on increasing the connectivity precision. We have to investigate the feasability of new approaches that will increase the connectivity precision while being as efficient as the approaches presented in this paper. We will also investigate the impact of sampling techniques on the efficiency and precision of graph summarisation algorithms.

References

[1] S. Abiteboul. Querying Semi-Structured Data. In *Proceedings of the 6th International Conference on Database Theory*, pages 1–18, 1997.

[2] S. Campinas, T. Perry, D. Ceccarelli, R. Delbru, and G. Tummarello. Introducing rdf graph summary with application to assisted sparql formulation. In *DEXA Workshops*, pages 261–266, 2012.

[3] S. Capadisli. World bank linked data. `http://datahub.io/dataset/world-bank-linked-data`, May 2013.

[4] Q. Chen, A. Lim, and K. W. Ong. D(k)-index: an adaptive structural summary for graph-structured data. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 134–144, New York, NY, USA, 2003. ACM.

[5] K. Christodoulou, N. W. Paton, and A. A. A. Fernandes. Structure inference for linked data sources using clustering. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*, EDBT '13, pages 60–67, New York, NY, USA, 2013. ACM.

[6] J.-C. Fernandez. An implementation of an efficient algorithm for bisimulation equivalence. *Sci. Comput. Program.*, 13(2-3):219–236, Apr. 1990.

[7] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 436–445, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[8] R. Goldman and J. Widom. Approximate dataguides. In *In Proceedings of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats*, pages 436–445, 1999.

[9] B. Haslhofer and A. Isaac. Europeana linked open data. `http://datahub.io/dataset/europeana-lod`, May 2013.

[10] M. Jarrar and M. Dikaiakos. A query formulation language for the data web. *Knowledge and Data Engineering, IEEE Transactions on*, 24(5):783–798, 2012.

[11] A. Jentzsch. Community research and development information service (cordis). `http://datahub.io/dataset/fu-berlin-cordis`, May 2013.

[12] A. Jentzsch. European employment services (eures). `http://datahub.io/dataset/fu-berlin-eures`, May 2013.

[13] R. Kaushik, P. Bohannon, J. F. Naughton, and H. F. Korth. Covering indexes for branching path queries. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, SIGMOD '02, pages 133–144, New York, NY, USA, 2002. ACM.

[14] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Exploiting local similarity for indexing paths in graph-structured data. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 129–140, 2002.

[15] S. Khatchadourian and M. P. Consens. Explod: Summary-based exploration of interlinking and rdf usage in the linked open data cloud. In *ESWC (2)*, pages 272–287, 2010.

[16] G. Klyne and J. J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. World Wide Web Consortium, Recommendation REC-rdf-concepts-20040210, February 2004.

[17] Y. Luo, Y. de Lange, G. H. L. Fletcher, P. De Bra, J. Hidders, and Y. Wu. Bisimulation Reduction of Big Graphs on MapReduce. In *Proc. BNCOD*, Oxford, UK, to appear 2013.

[18] Bavarian State Library, Bavarian Library Union, Cooperative Library Network Berlin-Brandenburg. B3kat - library union catalogues of bavaria, berlin and brandenburg. `http://datahub.io/dataset/b3kat`,

May 2013.

[19] DBpedia Team - http://wiki.dbpedia.org/Imprint. Dbpedia. http://datahub.io/dataset/dbpedia, May 2013.

[20] Deutsche Nationalbibliothek and German Library Networks. Gemeinsame normdatei (gnd). http://datahub.io/dataset/dnb-gemeinsame-normdatei, May 2013.

[21] ECS / Southampton University. Ecs southampton eprints. http://datahub.io/dataset/southampton-ecs-eprints, May 2013.

[22] Enipedia Team @ Energy and Industry Section, TBM, Delft University of Technology. Enipedia - energy industry data. http://datahub.io/dataset/enipedia, May 2013.

[23] North Rhine-Westphalian Library Service Center (hbz). lobid. bibliographic resources. http://datahub.io/dataset/lobid-resources, May 2013.

[24] Ontology Engineering Group, Facultad de Informtica, Universidad Politcnica de Madrid. datos.bne.es. http://datahub.io/dataset/datos-bne-es, May 2013.

[25] Tetherless World Constellation, Rensselaer Polytechnic Institute. Twc: Linking open government data. http://datahub.io/dataset/twc-logd, May 2013.

[26] The British Library Metadata Services. British national bibliography (bnb) - linked open data. http://datahub.io/dataset/bluk-bnb, May 2013.

[27] R. Milner. *Communication and Concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[28] T. Milo and D. Suciu. Index structures for path expressions. In *Proceedings of the 7th International Conference on Database Theory*, ICDT '99, pages 277–295, London, UK, UK, 1999. Springer-Verlag.

[29] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 419–432, New York, NY, USA, 2008. ACM.

[30] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, Dec. 1987.

[31] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *Proceedings of the Eleventh International Conference on Data Engineering*, ICDE '95, pages 251–260, Washington, DC, USA, 1995. IEEE Computer Society.

[32] D. Park. Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI-Conference on Theoretical Computer Science*, pages 167–183, London, UK, UK, 1981. Springer-Verlag.

[33] E. Sandhaus and R. Larson. New york times - linked open data. http://datahub.io/dataset/nytimes-linked-open-data, May 2013.

[34] Y. Tian, R. A. Hankins, and J. M. Patel. Efficient aggregation for graph summarization. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 567–580, New York, NY, USA, 2008. ACM.

[35] T. Tran, G. Ladwig, and S. Rudolph. Rdf data data partitioning and query processing using structure indexes. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints), 2012.