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

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 summarisation algorithms are suboptimal for this kind of environment as they require multiple iterations over the data graph. We investigate approximate graph summarisation 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 of the algorithms. 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 summarisation technique.

Categories and Subject Descriptors

E.1 [**Data Structures**]: Graphs and networks

Keywords

graph summarisation, summary precision, approximate algorithm, shared-nothing environment, mapreduce

1. INTRODUCTION

With the advent of the Semantic Web, there is a considerable amount of graph-structured data on the Web coming 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 results in 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, the adjacent nodes and the relations between them. In the scenario of graph pattern recommendation, e.g., in a SPARQL query [3], 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 order to deal with such large data graphs, there is a significant research on the concept of *graph summarisation*, 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, [9, 23, 34] propose a graph summarisation approach for optimizing query processing. The authors in [19, 33] leverages a summary for data exploration. A summary is used in [3, 14] to guide users writing graph queries. In practice, voluminous graphs are generally processed using a shared-nothing infrastructure (e.g., MapReduce). However, current graph summarisation 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. To that end, we introduce a model for evaluating the precision of graph summary algorithms compared to a gold standard summary. This precision model takes into account two aspects of the graph summary: the structure and the schema. We then analyse the trade-offs between the efficiency and the precision of the graph summarisation algorithms over 14 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 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 summarisation algorithms. In the Section 4, we introduce the precision model

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

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 each algorithm. We review in the Section 6 related works to the graph summarisation.

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 derived from a data graph.

2.1 Data Graph

The Web of Data is 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 has generally its own schema, ranging from loosely to strictly defined and evolving over time, the data structure does not follow strict rules as in a database. We therefore consider the Web of Data as being *semi-structured* [1], and represent it as a labelled directed graph, that we refer to as a *data graph*. In the following we define a data graph and introduce concepts used in the rest of the paper.

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 $\mathscr{L}^A$. 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 $x \xrightarrow{\alpha} y$, where x is the* source *and y the* target *of the edge. 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.*

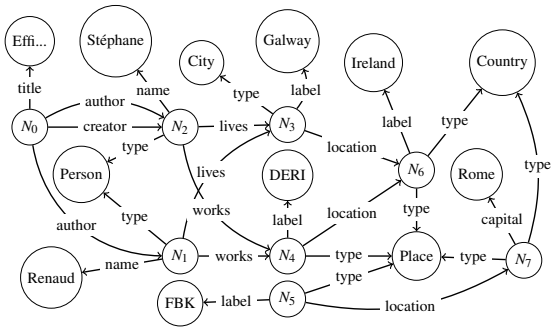
Path. We call a *path* a sequence of edges label $\alpha_1.\cdots.\alpha_n$. A path exists in the data graph G if there is a sequence of nodes $\{v_1,\cdots,v_{n+1}\} \subseteq V$ connected with such labelled edges. The Figure 1a depicts a graph of people and their relations with various places. The 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 $types(x \in V) = \{l_v(c) : \forall (x,\tau,c) \in A\}$ be the set of node labels related to the node x via a type edge. It is assumed for simplicity that all type edges are renamed with a same label, i.e., *type*.

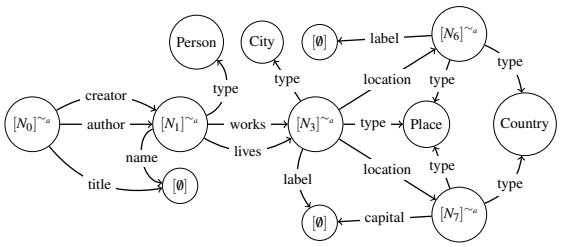
Attribute. We call *attribute* the label of an edge. We define as $attributes(x \in V) = \{\alpha \in \mathscr{L}^A : (x,\alpha,y) \in A\}$ the set of attributes associated with a node x in G.

2.2 Graph Summary

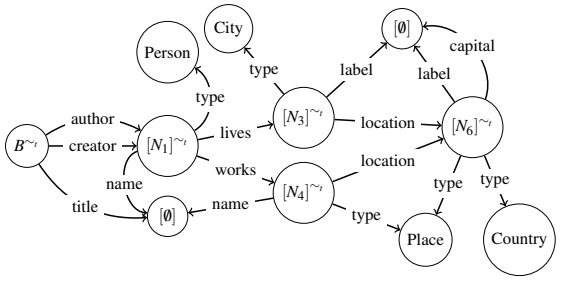
We call a graph summary [3], or *summary*, a graph $G^{\sim}$ that is homomorphic to the data graph G. The author in [29] first proposed the use of a summary in order to reduce the size of the data graph. A summary $G^{\sim}$ is constructed via an



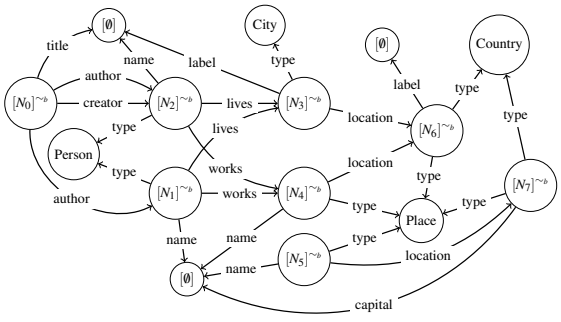
(a) A simple graph. The attribute *type* denotes a type edge.



(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 $\sim$-equivalence.

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 $[x]^{\sim} = \{y \in V : y \sim x\}$ the $\sim$-equivalence class of x. For every edge $x \xrightarrow{\alpha} y$ of G, there is an edge $[x]^{\sim} \xrightarrow{\alpha} [y]^{\sim}$ in $G^{\sim}$.*

For example, let $\sim_t = \left\{(x,y) \in V^2 : types(x) = types(y)\right\}$ 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 nodes $\{N_1, N_2\}$ of G, since both connect to *Person* via the type edge. The concepts *path*, *type*, and *attribute* introduced in the Section 2.1 hold 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 $attributes([N_1]^{\sim_a}) = \{works, lives, name\}$.

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 represented by the sink equivalence class $[\emptyset] = \{x \in V : \nexists(x, \alpha, y) \in A\}$ in $G^{\sim}$.*

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 $B^{\sim}$ 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 $B^{\sim_t}$ 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., $B^{\sim} = \{x \in V : \forall y \in V, x \not\sim y\}$.*

3. GRAPH SUMMARY ALGORITHMS

In this section, we introduce several definitions of an equivalence relation for creating a data graph summary and discuss their overall 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 type *Place*. This equivalence is the only one presented which can assign a node of G to several equivalence classes, e.g., the nodes N_6 and N_7 belong also to a second class since they are associated with the type *Country*.

$\sim_t$-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 $\sim_a$-equivalent if they share the same set of attributes; that is the equivalence relation:

$$\sim_a = \left\{(x,y) \in V^2 : attributes(x) = attributes(y)\right\}$$

The nodes N_3, N_4, and N_5 are $\sim_a$-equivalent because they share the same set of attributes, i.e., $\{label, location, type\}$. The graph in Figure 1b is the $\sim_a$-summary of the data graph in Figure 1a.

$\sim_{at}$-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 same type, i.e., *Person*, and they have the same attributes, i.e., $\{lives, name, type, works\}$.

$\sim_{ioa}$-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:

$$\sim_{ioa} = \left\{(x,y) \in V^2 : \forall s_x \xrightarrow{\alpha_x} x, \forall s_y \xrightarrow{\alpha_y} y, \alpha_x = \alpha_y \wedge x \sim_a y\right\}$$

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 set of attributes, 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:

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

The nodes N_1 and N_2 are $\sim_{at}$-equivalent, but are not according to the relation $\sim_{ioat}$, because only the node N_2 has the incoming attribute *creator*. For $\sim_{ioa}$ and $\sim_{ioat}$, the incoming set of attributes is computed by reversing the direction of edges, i.e., the target becomes the source, and then grouping on the source node.

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

source of an edge to an equivalence class.

$\sim_b$-equivalence. The forward and backward bisimulation [17], that we refer by *bisimulation* in the rest of the paper and is abbreviated with $\sim_b$, is a relation 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:

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

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

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

The $\sim_b$-summary depicted on the Figure 1d assigns a unique equivalence class to each node of the graph in Figure 1a. In this example, the only 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 [27] is generally used. It offers a $O(|A| \times log(|V|))$ complexity for the computation of the $\sim_{fb}$-equivalence. We note that the algorithm starts from an existing partitioning of the data graph, i.e., here a partitioning by $\sim_{ioat}$. The computation of the $\sim_b$-summary is achieved by applying over the $\sim_{fb}$ graph a slightly modified version of the [27] algorithm, in order to account for the incoming edges in the $\sim_{bb}$ definition. We remark that the complexity of all other algorithms are lower than the complexity of $\sim_b$. In addition, the [27] algorithm requires more than one iteration over the data graph, the number of iterations equal to the data graph diameter in the worst case. Iterative algorithms are suboptimal for shared-nothing infrastructures, e.g., MapReduce. The reason is the data graph needs to be read and written at each iteration, leading to an important IO load. Therefore, one pass algorithms can better 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, which we discuss in this section 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, since it is generally used by an application instead of the original data graph for performance reason. 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 equal to the data graph, since they assign a single equivalence class for each node of the data graph.

The $\sim_{st}$-equivalence may assign several equivalence classes to a node x, i.e., when $|types(x)| > 1$. The order of a $\sim_{st}$-summary is at most equal to $|V^{\sim_{st}}| = |\mathscr{L}^T|$. However, in the worst case, its size can equal $|A^{\sim_{st}}| = |\mathscr{L}^T|^d$, with d being the diameter of the data graph. Indeed, 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 sensitive to heterogeneous data graph. Indeed, there can be a node in a $\sim_a$-summary for each element of the attribute powerset minus the empty set, i.e., $|V^{\sim_a}| = 2^{|\mathscr{L}^A|} - 1$. 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 volume 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 graph summary is a summary of the structure of the data graph. Therefore, errors in the summary boil down to the presence of invalid 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 $G^{\sim}$ but not in G.

The definition of the precision model requires the introduction of an order relation on the summary. A set of summaries over a same data graph can be ordered using the relation $\sqsubseteq$ [8]. For instance, if we consider the $\sim_b$ and $\sim_t$ summaries in Figures 1d and 1c, the nodes of G in the equivalence classes $[N_1]^{\sim_b}$ and $[N_2]^{\sim_b}$ also belong to the equivalence class $[N_1]^{\sim_t}$. The equivalence classes of $\sim_b$ are then included in the equivalence classes of $\sim_t$. Therefore, the $\sim_b$-summary is inferior to the $\sim_t$-summary, noted as $\sim_b \sqsubseteq \sim_t$.

DEFINITION 4.1 ($\sqsubseteq$ ORDERING). *Let $\sim_1$ and $\sim_2$ be two equivalences on a data graph 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:* $\forall [x]^{\sim_1} \in V^{\sim_1}, \exists [x]^{\sim_2} \in V^{\sim_2}, [x]^{\sim_1} \subseteq [x]^{\sim_2}$.

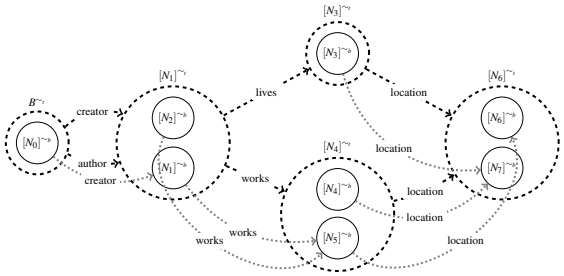


Figure 2: The graph $\mathscr{G}^{\sim_t}$ based on the summaries of Figures 1c and 1d. Sink equivalence classes are omitted for clarity. $G^{\sim_b}$ is depicted with solid lines, and $G^{\sim_t}$ with dashed lines. The edges in the $Err^{\sim_t}$ set are represented with gray dotted arrows, and edges in $G^{\sim_t}$ with dashed arrows.

As per the recursive definition of the $\sim_b$-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 inferior than any of the other presented summaries. Relations between two $\sim$-equivalence classes can be inferred between the included $\sim_b$-equivalence classes. The set $Err^{\sim}$ of inferred edges are invalid as they do not originally exist in $G^{\sim_b}$, and thus neither in G.

DEFINITION 4.2 (SUMMARY ERROR $Err^{\sim}$). *The set $Err^{\sim}$ is the set of edges between nodes of $G^{\sim_b}$ that are inferred from $G^{\sim}$ as per the $\sqsubseteq$ relation.*

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

We note that $Err^{\sim}$ is equal to all possible combinations of nodes pair on G, keeping only the ones that exists in $G^{\sim}$ but not in $G^{\sim_b}$.

We introduce the graph $\mathscr{G}^{\sim}$ which is $G^{\sim_b}$ augmented with inferred edges from the $Err^{\sim}$ set. This graph is used for computing the set of true and false positive edges in Section 4.3. The Figure 2 depicts the graph $\mathscr{G}^{\sim_t}$ based on the $\sim_t$ and $\sim_b$ summaries of Figures 1c and 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 $Err^{\sim_t}$ set are represented with gray dotted arrows. As per the $\sqsubseteq$ relation, the nodes $[N_1]^{\sim_b}$ and $[N_2]^{\sim_b}$ are included into the node $[N_1]^{\sim_t}$. Similarly, the nodes $[N_4]^{\sim_b}$ and $[N_5]^{\sim_b}$ are within the node $[N_4]^{\sim_t}$. Since the edge $[N_1]^{\sim_t} \xrightarrow{works} [N_4]^{\sim_t}$ exists in $G^{\sim_t}$, edges labelled *works* can then be inferred from the nodes $[N_1]^{\sim_b}$ and $[N_2]^{\sim_b}$ to $[N_5]^{\sim_b}$. Edges from the $Err^{\sim_t}$ set cause the nodes $[N_1]^{\sim_b}$, $[N_5]^{\sim_b}$, and $[N_6]^{\sim_b}$ to be connected. This generates the invalid path *works.location.capital* that exists in $G^{\sim_t}$ but not in $G^{\sim_b}$ nor G.

4.2.2 Classification of Errors

Depending on the kind of edge in the $Err^{\sim}$ set, we derive three categories, *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 the three categories in the following with regards to the $\sim_t$ and $\sim_a$ summaries only. Because the presented equivalences are inferior to the previous two as per the $\sqsubseteq$ relation, any error experienced with either $\sim_t$ or $\sim_a$ may also occur with others.

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

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

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 of $Err^{\sim}$ in which the attribute is not in the set $attributes(x)$ of the source 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, in the Figure 1b 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 attributes, i.e., *label*, *location* and *type*. Then, we may infer from that 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 $Err^{\sim}$, in which the target y is not in the set $types(x)$ of the source x:

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

4.3 Edge Precision

A visitor of the $\mathscr{G}^{\sim}$ graph can follow edges from the $G^{\sim_b}$ summary, but also from the $Err^{\sim}$ set. From a $G^{\sim_b}$ 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

over all the positive edges.

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

The sets $TP(x)$ and $FP(x)$ contain the true and false positive edges which source is x, respectively. For example, in Figure 2, the edge $[N_0]^{\sim_b} \overset{creator}{\rightarrow} [N_2]^{\sim_b}$ is in $TP(x)$, since it does exist in $G^{\sim_b}$. The edge $[N_0]^{\sim_b} \overset{creator}{\rightarrow} [N_1]^{\sim_b}$ is in $FP(x)$, since it does not exist in $G^{\sim_b}$. In total, this results that $Prec^{\sim}([N_0]^{\sim_b}) = \frac{3}{4}$. The *probability interpretation* of $Prec^{\sim}(x)$ is the probability that a randomly selected edge is correctly summarised. We note the recall is always equal to 1, since there is no false negative edges in the presented algorithms.

We use $Prec^{\sim}(x)$ as the precision measure for each of the classification of errors. We note that for a same node, the edge precision may vary between the three categories. As an example, consider the node $[N_0]^{\sim_b}$ in Figure 2. The edge attribute precision 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 connectivity precision is equal to $\frac{3}{4}$.

5. EVALUATION

In this section, we evaluate the trade-off between the performance of a summarisation 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 summarisation 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 three dimensions we use for evaluating a summary, i.e., the *volume*, the *performance* of the algorithm, and the *precision*.

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., $G^{\sim} : G^{\sim_{fb}} = \frac{|V^{\sim}|+|A^{\sim}|}{|V^{\sim_{fb}}|+|A^{\sim_{fb}}|}$.

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

Algorithm Performance. The MapReduce implementation of the graph summarisation 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 performance 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 classification of errors, we evaluate the precision of a summary thanks to the true and false positive edges set $TP(x)$ and $FP(x)$. 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 $P2$ reflects the overall chance for a randomly selected edge in the virtual $\mathscr{G}^{\sim}$ summary of being a true positive.

$$P1 = \frac{1}{|V^{\sim}|} \times \sum_{c \in V^{\sim}} \frac{1}{|C(c)|} \times \sum_{x \in C(c)} Prec^{\sim}(x)$$
$$P2 = \frac{\sum_{x \in V^{\sim_b}} TP(x)}{\sum_{x \in V^{\sim_b}} TP(x) + FP(x)}$$

The set $C(c \in V^{\sim}) = \{x \subset V^{\sim_b} : x \in c\}$ is the set of nodes in $G^{\sim_b}$ that are included into the node c in $G^{\sim}$.

5.2 Datasets

We use in this evaluation several datasets of various complexity. We list in the Table 1 the datasets along with some descriptive statistics. The datasets are grouped according to their complexity into four categories, i.e., *Low*, *Medium*, *High*, and *Very High*. We note the mean of measurements in a category as μ_L, μ_M, μ_H, and μ_{VH}, respectively. 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 G and $G^{\sim_{fb}}$. The order values omit the number of sink nodes, i.e., nodes without outgoing edges which includes 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 `wb` 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 `wb`.

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

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 [13]	288 524	22 369	65 042 837	233 051 608	6 884 121	33 990 765	13.71%
High							
twc-logd [32]	450	10 060	3 398 947	67 505 792	5 565	121 873	0.18%
enipedia [7]	128	267	413 520	4 463 566	1037	21 419	0.46%
Medium							
b3kat [2]	20	30	85 795 956	592 778 746	782 056	3 230 936	0.59%
ecs [35]	24	120	167 390	955 112	8 232	51 494	5.32%
lobid [12]	19	46	124 691 274	625 941 644	51 529	746 099	0.11%
bnb [31]	27	53	12 246 306	89 733 453	146 956	1 046 714	1.17%
datos [26]	23	143	7 412 312	58 048 932	360 822	2 504 262	4.38%
gnd [24]	22	35	962 930	7 940 373	9 664	88 875	1.11%
eures [16]	18	49	288 862	4 146 421	2 205	37 052	0.89%
Low							
europeana [11]	5	58	5 559 452	40 773 834	4 792	72 125	0.17%
wb [4]	4	174	11 210 832	84 345 613	293	6 987	0.01%
cordis [15]	7	63	729 780	7 101 623	2 245	36 783	0.50%
ny-times [30]	2	38	22 662	345 888	65	794	0.23%

Table 1: Descriptive statistics of the datasets. The Schema column reports the number in G of unique edges labels $|\mathscr{L}^A|$ and of unique types $|\mathscr{L}^T|$. 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.

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 with respect to these three dimensions.

5.3.1 Graph Summary Volume

The Table 2a reports the volume ratio between $G^{\sim}$ and the gold standard $G^{\sim_{fb}}$-summary. We report also the mean μ for each category of datasets complexity. The algorithms that are based only on the type information, i.e., $\sim_{st}$ and $\sim_t$, exhibit a higher compression compared to 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 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 that 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 attribute feature appears to be better for a summarisation algorithm than the type feature. Indeed, the volume ratio of $\sim_a$ and $\sim_{ioa}$ remains stable with *Medium* (42.51 and 51.45, resp.) and *High* (47.99 and 66.13, resp.) datasets. However, this is not the case with the $\sim_{st}$ and $\sim_t$ algorithms. We can observe 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 in `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` exhibits 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 $\sim_a$. We can conclude that a sparse usage of attributes with a type creates an explosion of combinations. In general, the type feature is less stable than the attribute feature.

5.3.2 Algorithm Performance

The Table 2b reports the CPU time in *ms* of the *edges* step in the graph summarisation computation. The reported times are the average of two runs of an algorithm for a certain dataset. The $\sim_{st}$ and $\sim_t$ algorithms are subject to have a blank equivalence class if there is no type edge, while others algorithms do not. For performance reason reason, such node are filtered from the resulting summary. Therefore, reported run times in Table 2b from $\sim_{st}$ and $\sim_t$ cannot be compared to others. On the *Medium*, *High*, and *Very High* datasets, the time taken by the $\sim_{st}$ algorithm in the *edges* step is higher than $\sim_t$. 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 a consequence, we need to compute all the possible combinations of edges between equivalence classes. We note that the incoming attribute feature in $\sim_{ioa}$ and $\sim_{ioat}$ does not imply a higher runtime when compared to $\sim_a$ and $\sim_{at}$.

5.3.3 Graph Summary Precision

We discuss in this section the precision of a graph summary with regards to the connectivity first, then to the type and attribute 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. While the performance evaluation did not account for the blank equivalence class, we do consider it for the precision evaluation.

The Table 3 reports the connectivity precision results $P1$ and $P2$. 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 of 25% according to $P1$, i.e., $\mu_H = 0.2414$. Algorithms based on the attribute feature only provide also a low precision on *Medium* and *High* categories. On the *Low* category, the attribute feature exhibits a better precision than the type feature, i.e., $\mu_L = 0.5579$ against $\mu_H = 0.3617$. However, when the type and 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 (0.5124) on the *High* category for $\sim_{at}$, and at least 20% on *Medium*. We remark that the incoming attribute in $\sim_{ioa}$ is an important feature that increases the precision. The $\sim_{ioa}$-equivalence provides a precision of 30% on the *Medium* up to 50% on *High*, whereas $\sim_a$ reaches 15% on *Medium* and 10% on *High*. Overall, we can achieve a good average connectivity precision with $\sim_{ioat}$ according to $P1$. However, the overall precision $P2$ is very low on the datasets of *Medium* and *High* complexities. 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 means μ for a category of dataset complexity of the type and attribute precision results $P1$ and $P2$, i.e., regarding the schema of the summary. The algorithms $\sim_{st}$ and $\sim_t$ based on the type feature provide an attribute precision above at least 60% for $P1$ ($\mu_M = 0.5927$ for $\sim_{st}$). On the contrary, the attribute feature reports a

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_H	*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_M	*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
wb	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_L	*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	77 643 685	52 608 760	99 839 325	104 255 540	102 540 080	105 685 365
twc-logd	6 974 425	4 213 100	6 542 530	6 557 955	6 519 540	6 649 275
enipedia	1 518 265	1 314 045	1 381 875	1 327 200	1 460 110	1 416 315
μ_H	*4 246 345*	*2 763 572*	*3 962 202*	*3 942 577*	*3 989 825*	*4 032 795*
b3kat	137 235 825	128 601 525	135 458 935	135 080 175	136 010 320	135 805 145
ecs	946 635	812 700	983 015	921 185	946 450	946 015
lobid	187 415 830	161 954 355	204 759 505	205 217 415	204 657 490	205 605 435
bnb	25 310 415	19 193 380	21 927 785	21 776 020	21 757 485	21 568 175
datos	8 271 535	8 215 710	8 387 465	8 389 250	8 478 590	86 804
gnd	1 510 455	1 596 060	1 569 565	1 563 460	1 580 720	1 596 630
eures	1 378 185	1 223 370	1 448 975	1 468 045	1 472 430	1 473 545
μ_M	*51 724 125*	*45 942 442*	*53 505 035*	*53 487 935*	*53 557 640*	*52 440 249*
europeana	7 995 115	7 871 920	8 327 500	8 362 750	8 498 575	8 345 030
wb	10 508 190	10 592 170	10 340 025	10 412 450	10 291 955	10 309 730
cordis	1 642 030	1 536 155	1 620 345	1 605 850	1 711 790	1 627 270
ny-times	594 285	549 680	563 200	566 740	563 110	580 405
μ_L	*5 181 727*	*5 140 988*	*5 208 950*	*5 227 250*	*5 254 718*	*5 212 401*

(b) Performance comparison. We report the CPU time in *ms* of the *edges* step in the graph summarisation computation. For each category of dataset complexity, we report the mean μ of the CPU time.

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

Dataset	$Err^{\sim_{st}}_{con}$		$Err^{\sim_t}_{con}$		$Err^{\sim_a}_{con}$		$Err^{\sim_{at}}_{con}$		$Err^{\sim_{ioa}}_{con}$		$Err^{\sim_{ioat}}_{con}$	
	*P*1	*P*2	*P*1	*P*2	*P*1	*P*2	*P*1	*P*2	*P*1	*P*2	*P*1	*P*2
twc-logd	0.0107	0.0003	0.1846	0.0002	0.1444	0.0175	0.5671	0.0232	0.2886	0.0222	0.7572	0.0280
enipedia	0.0059	0.0005	0.2982	0.0046	0.0632	0.0283	0.4576	0.1047	0.7294	0.1394	0.9991	0.1512
μ_H	*0.0083*	*0.0004*	*0.2414*	*0.0024*	*0.1038*	*0.0229*	*0.5124*	*0.0640*	*0.5090*	*0.0808*	*0.8781*	*0.0896*
b3kat	0.3162	0.0000	0.2304	0.0000	0.4975	0.0001	0.5199	0.0001	0.4973	0.0001	0.5217	0.0001
ecs	0.0710	0.0003	0.0663	0.0002	0.0145	0.0042	0.0167	0.0045	0.1240	0.0070	0.1030	0.0076
lobid	0.0806	0.0000	0.0779	0.0001	0.2480	0.0035	0.3020	0.0048	0.2924	0.0072	0.3483	0.0094
bnb	0.0121	0.0001	0.0262	0.0001	0.0091	0.0013	0.0181	0.0013	0.0178	0.0013	0.0299	0.0014
datos	0.0206	0.0000	0.0618	0.0000	0.0339	0.0000	0.0653	0.0000	0.1273	0.0000	0.1285	0.0000
gnd	0.0086	0.0001	0.0086	0.0001	0.1959	0.0091	0.2158	0.0131	0.6438	0.0617	0.7027	0.0880
eures	0.0205	0.0018	0.3617	0.0010	0.1145	0.1084	0.2696	0.2227	0.4833	0.3844	0.4835	0.3844
μ_M	*0.0757*	*0.0003*	*0.1190*	*0.0002*	*0.1591*	*0.0181*	*0.2011*	*0.0352*	*0.3123*	*0.0660*	*0.3311*	*0.0701*
europeana	0.3641	0.0345	0.3641	0.3944	0.5429	0.3944	0.5429	0.3944	0.5429	0.3944	0.5429	0.3944
wb	0.6935	0.0476	0.6320	0.0476	0.9142	0.8479	0.9142	0.8479	0.9680	0.9687	0.9680	0.9687
cordis	0.0064	0.0094	0.0064	0.0094	0.1619	0.0347	0.1620	0.0347	0.4203	0.0690	0.4203	0.0690
ny-times	0.3830	0.0798	0.3830	0.0798	0.6126	0.6813	0.6126	0.6813	1.0000	1.0000	1.0000	1.0000
μ_L	*0.3617*	*0.0428*	*0.3464*	*0.1328*	*0.5579*	*0.4896*	*0.5579*	*0.4896*	*0.7328*	*0.6080*	*0.7328*	*0.6080*

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

	$Err^{\sim_{st}}_{type}$		$Err^{\sim_t}_{type}$		$Err^{\sim_a}_{type}$		$Err^{\sim_{at}}_{type}$		$Err^{\sim_{ioa}}_{type}$		$Err^{\sim_{ioat}}_{type}$	
	*P*1	*P*2	*P*1	*P*2	*P*1	*P*2	*P*1	*P*2	*P*1	*P*2	*P*1	*P*2
μ_H	1.0000	1.0000	1.0000	1.0000	0.9675	0.1424	1.0000	1.0000	0.9760	0.2151	1.0000	1.0000
μ_M	1.0000	1.0000	1.0000	1.0000	0.9222	0.7136	1.0000	1.0000	0.9415	0.7753	1.0000	1.0000
μ_L	1.0000	1.0000	1.0000	1.0000	0.9999	0.9995	1.0000	1.0000	0.9999	0.9999	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_H	0.7826	0.0843	0.9446	0.2398	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
μ_M	0.5927	0.2798	0.6607	0.2925	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
μ_L	0.7586	0.5217	0.7529	0.4968	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Table 4: Schema precision comparison.

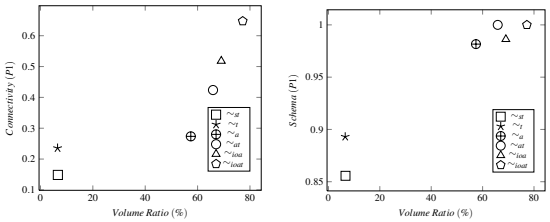
good type precision, reaching on average at least 90% for $\sim_a$, i.e., $P1 = 0.9222$ on the *Medium* datasets. Incoming attributes do not increase much the type precision, since the type precision of $\sim_a$ stays on par with $\sim_{ioa}$. The $\sim_{at}$ algorithm provides a perfect summarisation of the schema. Again, the significant differences between $P1$ and $P2$ suggests one 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 summarisation algorithm is important for the connectivity precision, but not for the schema. However, there is place for improvement for overall connectivity precision especially on certain datasets. 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 $P2$ is not appropriate for measuring the precision of a summary in terms of connectivity and schema.

5.3.4 Trade-Offs

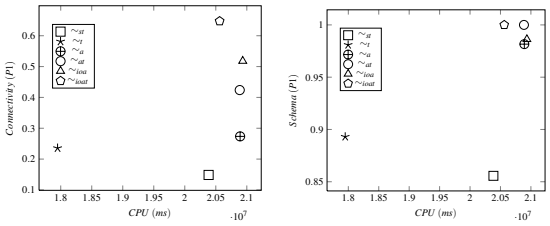
We report in Figure 3a the trade-off between the average connectivity precision and the average volume ratio across all datasets among all the algorithms. We can distinguish two groups of algorithms, the type algorithms, i.e., $\sim_t$ and $\sim_{st}$, and the attribute algorithms. The type algorithms provide the best volume ratio, but also the worst 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 terms 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 the 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 that some 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. Among the attribute-based algorithms $\sim_a$, $\sim_{at}$, $\sim_{ioa}$ and $\sim_{ioat}$, the latter is the one that achieves the best runtime with the highest precision. Among the type-based algorithms, $\sim_t$ achieves a lower runtime and a higher precision than $\sim_{st}$. If the schema precision is primordial and a low connectivity precision can be tolerated, $\sim_t$ is the best candidate as it provides a high schema precision, with the best CPU time. On the contrary, if the connectivity is primordial, $\sim_{ioat}$ is the best candidate, at the cost of a longer runtime.



(a) Connectivity precision versus volume ratio.

(b) Schema precision versus volume ratio.



(c) Connectivity precision versus algorithm performance.

(d) Schema precision versus algorithm performance.

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

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, to represent processus and their transition in concurrency, or to structure data in a flexible way, e.g., using the Object Exchange Model [28] or, more recently, the Resource Description Framework [20].

A common challenge encountered when analysing a graph is its volume. A large number of nodes or edges reduces the scalability and increases the running time of applied algorithms. A solution is to translate the graph into another smaller graph, while preserving its structure. The smaller graph is then considered as a summary of the original data graph. [22] introduces the concept of bisimulation to define equivalent processes in concurrent systems. The relational coarsest partition [27] is generally the algorithm used for computing a bisimulation on a graph. With a similar perspective in reducing the size of the graph, DataGuides [9] is the first work to improve query execution through the use of an index on the summary of the structure of the graph.

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 [10], 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 errors with regards to the original data graph. [23] extends 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 specification requires to rebuild the summary. In [17, 18] the definition of bisimulation on a graph is simplified by limiting the length of a path to k hops. In [5], the authors vary the maximum length of a path per node, depending on the query load of the system. The authors of [25] propose a summarisation based on the Information Theory principle of Minimum Description Length. Although the original data graph can be re-created from the summary, a user-defined bounded error parameter can be used to balance the summary volume with the loss in precision. The authors of [34] uses a similar approach to [17] for the purpose of improving the partitioning of RDF data and the execution of queries. [33] proposes an approach based on bisimulation that provides more or less detailed summaries.However, the level of detail is left to the user; this assumes some prior knowledge about the data structure and content. In this paper, we focus instead on approaches that require no prior knowledge.

Current approaches of the summary computation seek to reduce the complexity of algorithms or the volume of the summary. However, the existing approaches and applications [6, 14, 19], do not scale to large data graphs composed of billions of nodes and edges. [21] proposes 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, therefore increasing the runtime of the algorithm. In this paper, we are interested instead in summarisation algorithms optimised for shared-nothing environment, in which the 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 environment 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 14 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 con-

nectivity 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.

Acknowledgement

This material is based upon works supported by the European FP7 project LOD2 (257943) and the Irish Research Council for Science, Engineering and Technology.

References

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

[2] C. L. N. B.-B. Bavarian State Library, Bavarian Library Union. B3kat - library union catalogues of bavaria, berlin and brandenburg. `http://datahub.io/dataset/b3kat`, May 2013.

[3] 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.

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

[5] 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.

[6] 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.

[7] E. T. . Energy and D. U. o. T. Industry Section, TBM. Enipedia - energy industry data. `http://datahub.io/dataset/enipedia`, May 2013.

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

[9] 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.

[10] 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.

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

[12] N. R.-W. L. S. C. (hbz). lobid. bibliographic resources. `http://datahub.io/dataset/lobid-resources`, May 2013.

[13] D. T. http://wiki.dbpedia.org/Imprint. Dbpedia. `http://datahub.io/dataset/dbpedia`, May 2013.

[14] 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.

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

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] Y. Luo, Y. Lange, G. Fletcher, P. Bra, J. Hidders, and Y. Wu. Bisimulation reduction of big graphs on mapreduce. In G. Gottlob, G. Grasso, D. Olteanu, and C. Schallhart, editors, *Big Data*, volume 7968 of *Lecture Notes in Computer Science*, pages 189–203. Springer Berlin Heidelberg, 2013.

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

[23] 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.

[24] D. Nationalbibliothek and G. L. Networks. Gemeinsame normdatei (gnd). `http://datahub.io/dataset/dnb-gemeinsame-normdatei`, May 2013.

[25] 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.

[26] U. P. d. M. Ontology Engineering Group, Facultad de InformÃątica. datos.bne.es. `http://datahub.io/dataset/datos-bne-es`, May 2013.

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

[28] 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.

[29] 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.

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

[31] T. B. L. M. Services. British national bibliography (bnb) - linked open data. `http://datahub.io/dataset/bluk-bnb`, May 2013.

[32] R. P. I. Tetherless World Constellation. Twc: Linking open government data. `http://datahub.io/dataset/twc-logd`, May 2013.

[33] 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.

[34] 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.

[35] E. . S. University. Ecs southampton eprints. `http://datahub.io/dataset/southampton-ecs-eprints`, May 2013.