Mining Summaries for Knowledge Graph Search

Qi Song [1] Yinghui Wu [1] Xin Luna Dong[2]
[1]Washington State University [2]Amazon, Inc.
{qsong, yinghui}@eecs.wsu.edu lunadong@amazon.com

Abstract—**Mining and searching heterogeneous and large-scale knowledge graphs is very challenging under real-world resource constraints (*e.g.,* memory, response time). In this paper, we study a novel graph summarization framework to facilitate knowledge graph search. 1) We introduce a class of summaries characterized by graph patterns. In contrast to conventional summaries defined by frequent graph patterns, the summaries are capable of adaptively summarize entities with similar neighbors up to a bounded hop. 2) We formulate the computation of graph summarization as a bi-criteria pattern mining problem. Given a knowledge graph G, the problem is to discover k diversified summary patterns that maximizes the informativeness measure. Although this problem is NP-hard, we develop two algorithms to solve this problem: a 2-approximation algorithm, and a *resource-bounded anytime* algorithm to trade-off speed and accuracy, under given resource constraints. 3) We develop query evaluation algorithms by leveraging the graph summarization. These algorithms efficiently compute (approximate) answers with high accuracy by only accessing a small number of summary patterns and their materialized views. Using real-world knowledge graphs, we experimentally verify the effectiveness and efficiency of our parallel algorithms for computing summarizations; and query evaluation guided by summarization.**

I. Introduction

Knowledge graphs are routinely used to represent entities and their relationships in knowledge bases [5], [11]. Unlike relational data, real-world knowledge graphs lack the support of well-defined schema and typing system.

To search knowledge graphs, a number of query processing techniques are proposed [11], [15], [19], [27]. Nevertheless, it is hard for end-users to specify precise queries that will lead to meaningful answers without any prior knowledge of the underlying data graph. Mining and search such knowledge graphs is challenging due to the ambiguity in queries, the inherent computational complexity (*e.g.,* subgraph isomorphism [11], [19]) and resource constraints (*e.g.,* data allowed to be accessed, response time) [8] for large knowledge graphs.

Example 1: Fig. 1 illustrates a sample knowledge graph G of artists and bands. Suppose a music publisher wants to find (artists) who are experts in two genres (genre), acted in a (film), and also collaborated with a band whose manager is located in the same country as the band. This search can be represented as a *graph query* Q [11], [15], [19], [27] as shown in Fig 1. The answer of Q refers to the set of entities typed with artist in the subgraphs of G that are isomorphic to Q. In this example, T.McGraw is the correct answer for Q.

The evaluation of Q over large G is expensive. For example, the ambiguous label "artist" requires the inspection of all the entities having the type. Moreover, it is hard for the users to specify Q without prior knowledge of G.

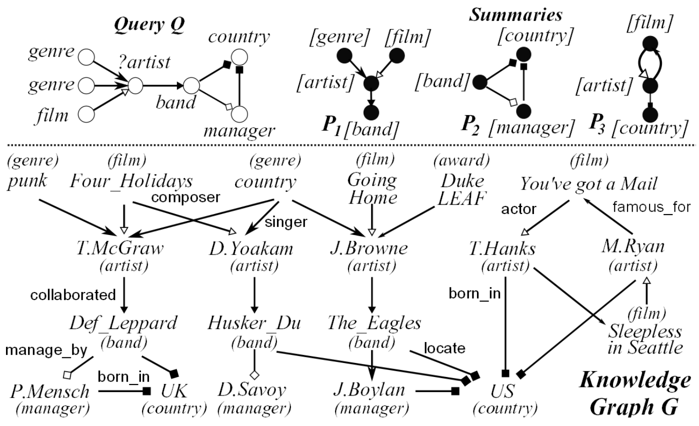


Fig. 1: Knowledge graph, summary patterns, and graph query.

Observe that the graph G can be described by three small graph patterns as *summaries* P_1, P_2, and P_3, as illustrated in Fig. 1. Each pattern abstracts a fraction of G, by summarizing a group of entities as a single node, along with their common neighboring entities in G. For example, P_1 specifies three artists J.Browne, T.McGraw and D.Yoakam in G as a single node artist, who are associated with their band, genre and films as 1 hop neighbors, indicating "musicians"; and P_3 distinguishes the artists T.Hanks and M.Ryan who are associated with only films and country (i.e., "actors"). These concise summaries help the users in understanding G without a daunting inspection of low-level entities.

We may further use these patterns as "views" [7], [14] to speed up knowledge discovery in G. For example, Q can be correctly answered by accessing only the entities and relationships summarized by P_1 and P_2 in G. Indeed, all the matches of Q are contained in those entities. □

This example suggests that summaries as graph patterns can benefit knowledge graph search by suggesting (and can be directly queried as) "materialized views". In addition, such summaries can help users in understanding complex knowledge graphs without inspecting a large mount of data, as well as suggesting meaningful queries in mining tasks.

Although desirable, computing summaries for schema-less, noisy knowledge graph is nontrivial. Conventional graph summaries are defined by frequent subgraph patterns, which capture their isomorphic counterparts in a graph [6], [12], [13], [15]. This can often be an overkill for entities with similar, relevant neighbors up to a certain hop. For example, the two entities J.Browne and T.McGraw, along with their relevant 1 hop neighbors *should* be summarized by a single summary P_1, despite that the two subgraphs induced by these entities are not isomorphic to each other; similarly for the entities T.Hanks and M.Ryan summarized by P_3. Indeed, such cases are commonly seen in schema-less and noisy knowledge graphs. We ask the following questions: 1) *How can we construct summaries in a schema-less knowledge graph?* and 2) *How can we leverage the summaries to support knowledge graph search?*

Contributions. This paper studies a novel graph summarization framework to facilitate knowledge graph search.

1) We introduce a class of graph patterns, namely, *d-summaries*, to summarize similar entities in terms of their labels and neighborhood information up to a bounded hop d. In contrast to conventional summaries defined by frequent subgraph patterns, (a) a d-summary is characterized by a *d-similarity* relation, which induces a lossy representation of similar entities and their d-hop neighbors; (b) it is in PTIME to verify if a graph pattern is a d-summary, without the need for isomorphism checking which is already NP-hard (Section II).

2) We introduce a bi-criteria function to quantify the quality of a summarization for knowledge graphs, which integrates both informativeness and diversity measures. Based on the quality function, we introduce the *diversified graph summarization problem*. Given a knowledge graph G, integers k and d, the problem is to compute a set of k d-summaries that maximizes the bi-criteria quality function.

This problem is (not surprisingly) NP-hard. We show that it already requires verifying all possible d-summaries to generate a summarization with approxiamtion ratio 2. In response, we develop a stream-style *anytime* mining algorithm. Instead of waiting for all summaries to be verified, the algorithm can be interrupted at any time, and provides summaries with desirable quality guarantees, adapting to specific resource bounds (*e.g.,* memory, response time) (Section IV).

3) We further develop a query evaluation algorithm over knowledge graphs for the class of subgraph queries. The algorithm selects and refers to a small set of summaries that best "cover" the query, and fetches entities from the original knowledge graph only when necessary (Section V). This enables a knowledge graph search paradigm that adapts to resource constraints with high quality answers, regardless of how large the original knowledge graph is.

4) Using real-world knowledge bases and synthetic graphs, we experimentally verify the effectiveness and efficiency of our summarization and query-evaluation algorithms (Section VI). We found that 1) It is feasible to compute summarizations over real-world knowledge graphs (300 seconds over a knowledge graph YAGO with 3.9 million nodes and relationships), 2) the summaries effectively support concise, informative and diversified summarization, and 3) the summarization significantly improves querying efficiency (*e.g.,* by 40 times for YAGO).

To the best of our knowledge, this is the first work to employ diversified pattern discovery for approximate summarization and querying large-scale knowledge graphs. We envision that our framework suggests promising tools for accessing, searching, and understanding complex knowledge graphs.

Related work. We categorize the related work as follows.

Graph summarization. Graph summarization has been studied to describe the data graph with a small amount of information [13], [18], [21]–[24]. These approaches can be classified as follows: **1)** Graph compression, which aim to compress graphs within a bounded error by minimizing a information complexity measure [13], [18], [21], e.g., Minimum Description Length (MDL), or to reduce the space cost such that the topology of the original data graph can be approximately restored [18], [21]. The algorithm in [13] employs clustering and community detection to describe the data graph with predefined frequent structures (vocabulary) including stars and cliques. **2)** Summarization techniques attempt to construct summaries over attributed graphs, where nodes with similar attributes are clustered in a controlled manner using parameters such as participation ratio [23]. **3)** (Bi)simulation relation is adopted [4] to group paths carrying same labels up to a bounded length. Relaxed bisimulation has also been studied to generate summaries over a set of answer [24]. These work summarizes the entities only when they are pairwise similar, which can be an overkill for knowledge graphs. **4)** Entity summarization [22] generates diversified answers for entity search, instead of summaries for general subgraph queries.

Our work differs from the above works in the following ways: **1)** We introduce *lossy* summaries to facilitate efficient knowledge graph query processing, rather than to restore the exact topology of the graph as in [13], [18], [21]. Moreover, labels of nodes/edges are not considered in these works. **2)** Our algorithms produce summaries for a single knowledge graph rather than for multiple query answers [22], [24]. On the other hand, they can also be applied for diversified result summarization. **3)** Our summarization is measured by both informativeness and diversity, which is more involved than MDL-based approaches. **4)** In contrast to [18], [23], our methods do not rely on auxiliary structure and parameters for preserving the entity and relationships. None of these work addresses diversified summaries as general graph patterns.

Graph clustering. A number of graph clustering approaches have been proposed to group a set of similar graphs [1]. These techniques can not be directly applied for summarizing a single knowledge graph. Frequent subgraph patterns can be mined from a single graph to describe large graphs [6], [13]. Diversified pattern mining is studied for general patterns [25] and graph patterns defined by subgraph isomorphism [9]. Type-based summarization is applied to facilitate keyword search in RDF graphs [15]. In contrast, 1) We capture summaries based on d-similarity rather than subgraph isomorphism [6], [9] or homomorphism [15]; and 2) We study bi-criteria summarization that integrates diversification and informativeness, which is not addressed in [15].

Answering queries using views: View-based query evaluation has been shown to be effective for SPARQL queries [14] and general graph pattern queries [7]. View-based query evaluation typically requires equivalent query rewriting by accessing views defined in the same query language. Our work differs in the following ways: 1) We use d-summaries as views to evaluate graph queries defined by subgraph isomorphism, rather than requiring views and queries to be in the same language; and 2) We develop feasible summarization algorithms as view discovery process. These are not addressed in [7], [14].

II. Knowledge Graphs and summaries

A. Knowledge Graphs and summaries

We start with the notions of knowledge graphs, and then introeduce summaries and summarization for knowledge graphs.

Knowledge graphs. We define a knowledge graph G as a directed labeled graph (V, E, L), where V is a set of nodes, and $E \subseteq V \times V$ is a set of edges. Each node $v \in V$ represents an entity with label $L(v)$ that may carry the content of v such as type, name, and attribute values, as found in knowledge bases and property graphs [11]; and each edge $e \in E$ represents a relationship $L(e)$ between two entities.

We do not assume a standard schema over G, and our techniques will benefit from such a schema, if exists.

Example 2: Fig. 1 depicts a fraction of a typed knowledge graph. Each entity (*e.g.,*J.Browne) has a label that carries its type (*e.g.,*artist), and connects to other typed entities (*e.g.,*band) via labeled relationships (*e.g.,* collaborated). □

We use the following notations: 1) A path ρ in a graph G is a sequence of edges $e_1, \ldots, e_n$, where e_i=(v_i, v_{i+1}) is an edge in G; 2) The *path label* $L(\rho)$ is defined as $L(v_1)L(e_1)\ldots L(v_n)L(e_n)L(v_{n+1})$, *i.e.,* concatenation of all the node and edge labels on the path ρ; and 3) A graph G'=(V', E', L') is a node induced subgraph of G=(V, E, L) if $V' \subseteq V$, and E' consists of all the edges in G with endpoints in V'. It is an edge induced subgraph if it contains $E' \subseteq E$ and all nodes that are endpoints of edges in E'.

Summaries. Given a knowledge graph G, a *summary* P of G is a directed connected graph (V_P, E_P, L_P), where V_P (resp. $E_P \subseteq V_P \times V_P$) is a set of summary nodes (resp. edges). Each node $u \in V_P$ (resp. edge $e \in E_P$) has a label $L_P(u)$ (resp. $L_P(e)$). Each node $u \in V_P$ (resp. $e \in V_E$) represents a non-empty node set $[u]$ (resp. edge set $[e]$) from G.

The *base graph* of P in G, denoted as G_P, refers to the subgraph of G induced by the node set $\bigcup_{u \in V_P} [u]$, and the edge set $\bigcup_{e \in E_P} [e]$, for each $u \in V_P$ and $e \in E_P$. Note that a base graph can be disconnected for a connected summary.

As remarked earlier, a summary should adaptively describe entities with similar neighborhood up to certain hops in G. To capture this, we introduce a notion of d-similarity.

d-similarity. Given a graph pattern P and a graph G, a backward (resp. *forward*) d-similarity relation is a binary relation $R_d^{\uparrow} \subseteq V_P \times V$ (resp. $R_d^{\downarrow} \subseteq V_P \times V$), where

- ∘ $(u, v) \in R_0^{\uparrow}$ and $(u, v) \in R_0^{\downarrow}$ if $L_P(u)$=$L(v)$;
- ∘ $(u, v) \in R_d^{\uparrow}$ if $(u, v) \in R_{d-1}^{\uparrow}$, and for every parent u' of u in P, there exists a parent v' of v in G, such that $L_P(u', u)$=$L(v', v)$ (*i.e.,* edges (u', u) and (v', v) have the same edge label), and $(u', v') \in R_{d-1}^{\uparrow}$;
- ∘ $(u, v) \in R_d^{\downarrow}$ if $(u, v) \in R_{d-1}^{\downarrow}$, and for every child u' of u in P, there exists a child v' of v in G such that $L_P(u, u')$=$L(v, v')$, and $(u', v') \in R_{d-1}^{\downarrow}$.

We define a d-similarity R_d between P and G as the set of node pairs $\{(u, v) | (u, v) \in R_d^{\uparrow} \cap R_d^{\downarrow}\}$. A summary P is a *d-summary*, if for every summary node u and every node $v \in [u]([u] \neq \emptyset)$, $(u, v) \in R_d$.

Intuitively, a d-summary P guarantees that for any incoming (resp. outgoing) path ρ of a summary node u with a bounded length d in P, there must exist an incoming (resp. outgoing) path of each entities summarized in $[u]$ with the same label. That is, P preserves all the neighborhood information up to length d for each summary node u in P. Note that for a given summary P with diameter d_m, $d \leq d_m$,

We now characterize knowledge graph summarization with summaries and base graphs. Given a knowledge graph G and an integer d, we define a *summarization* $\mathcal{S}_G$ of G as a set of d-summaries. In practice, additional mapping structures can be used to trace the base graphs for the summaries.

Example 3: Fig. 1 illustrates a summarization of the knowledge graph G, which contains three 2-summaries P_1, P_2, and P_3. The base graph of P_1 is induced by the entities shown in the table below (the edges are omitted).

summary node	entities
[genre]	{ country, punk }
[film]	{Going Home, Four_Holidays}
[artist]	{J.Browne, D.Yoakam, T.McGraw}
[band]	{The_Eagles, Husker_Du, Def_Leppard }

Indeed, for every path of length bounded by 2 in P_1 (*e.g.,* ρ_p={genre,artist, band}) and for every entity with label genre, there exists a path ρ (*e.g.,* {country, J.Browne, The_Eagles}) in G with the same label as ρ_p. Similarly, one may verify that P_2 summaries the band Def_Leppard and The_Eagles, their associated country and manager in G, and P_3 summaries the films You've got a Mail and Sleepless in Seattle, actors T.Hanks and M.Ryan and their countries.

Note that P_1 cannot summarize T.Hanks, as the latter has no path to a band as suggested in P_1. □

B. Verification of d-summaries

We next study the verification of d-summaries. Given a summary P and a knowledge graph G, the verification problem is to determine if P is a d-summary of G, and if so, identify the *largest* base graph G_P of P in G.

In contrast to its counterpart defined by frequent subgraphs (NP-hard), the verification of d-summaries is *tractable*.

Lemma 1: *Given a summary P=(V_P, E_P, L_P) and a graph G=(V, E, L), it is in $O(|V_P|(|V_P| + |V|)(|E_P| + |E|) + |G|)$ time to verify if P is a d-summary of G, and if so, also finds the largest base graph G_P of P in G.* □

As a proof of Lemma. 1, we outline an algorithm that determines if P is a d-summary in polynomial time. (1) For each node $u \in V_P$, the algorithm initializes a set $[u]$ with all the nodes v where $(u, v) \in R_0^{\uparrow}$. (2) It then iteratively computes backward d-similarity $R_i^{\uparrow}$ for each edge $(u', u) \in E_P$, by removing the nodes in $[u']$ that are not in $R_{i-1}^{\uparrow}$ by definition. This step repeats until i=d. The forward d-similarity can be similarly computed. (3) If for every node $u \in V_P$, $[u] \neq \emptyset$, P is verified as a d-summary. Otherwise, whenever $[u]$ becomes $\emptyset$, it determines that P is not a d-summary.

The algorithm keeps the following invariants: (1) For any pair $(u, v) \in R_d^{\uparrow} \cap R_d^{\downarrow}$, $v \in [u]$ when it terminates. (2) If a node v is removed from $[u]$ at any time, then $(u, v) \notin R_d$. Hence it correctly computes the largest R_d and G_P. To see the complexity, observe that (1) it takes $O((|V_P|+|V|)(|E_P|+|E|)$ time to verify forward and backward d-similarity for a single summary node in V_P, and the total verification time is in $O(|V_P|(|V_P| + |V|)(|E_P| + |E|)$; and (2) it takes $O(|G|)$ time to induce G_P.

Remarks. d-summaries differ from several other graph patterns as follows. (1) Frequent graph patterns [6], [12], [15] capture their isomorphic counterparts in G. However, this requires (a) expensive isomorphism checking (NP-hard), and (2) excessive number of redundant patterns. (2) (Bi)simulation-based patterns [4], [24] enforce pairwise entity equivalence induced by the label equivalence of their associated paths. These semantics, while fit better for data supported by fixed entity schema (*e.g.,* XML data), can be an overkill for schema-less knowledge graphs, where similar entities that are not pairwisely equivalent are quite common (e.g., J.Browne, D.Yoakam, and T.McGraw in Fig. 1). (3) Dual-simulation [16] relax the pairwise similarity relation by requiring all the paths with arbitrary length in a pattern to have a matched path. This can also be restrictive to capture entities having similar relevant neighbors up to a bounded hop (1 hops neighbors for J.Browne and T.McGraw). In contrast, the parameter d can be tuned to adapt to such relevant neighbors in d-summaries. (4) Neighborhood based summaries [3] merge overlapped neighbors. In contrast, a d-summary can group entities that share no neighbors (*e.g.,*The_Eagles and Def_Leppard in Fig. 1).

The notations of this paper are summarized in Table I.

III. Quality of Summarization

In this section, we introduce a bi-criteria metric that captures the quality of knowledge graph summarization in terms of both informativeness and diversity.

A. Informative Summaries

The interestingness of a summary P can be quantified by its informativeness, which should capture (1) summary size, and (2) the total amount of information (entities and their relationships) it encodes in a knowledge graph G . We define the informativeness function $I(\cdot)$ of a summary P as:

$$I(P) = \frac{|P|}{b_P} * \mathsf{supp}(P, G)$$

where 1) $|P|$ refers to the size of a summary P, *i.e.,* total

Symbol	Definition
G=(V, E, L)	knowledge graph G
P=(V_P, E_P, L_P)	summary P as a graph pattern
G_P	base graph of summary P in G
$\mathcal{S}_G$	summarization of G
$\mathsf{card}(\mathcal{S}_G)$	cardinality of $\mathcal{S}_G$
$\mathsf{supp}(P, G)$	support of summary P in G
$I(P)$	informativeness of summary P; $I(P)$=$\|P\| * \mathsf{supp}(P, G)$
$\mathsf{diff}(P_i, P_j)$	distance between summaries P_i and P_j
$F(\mathcal{S}_G)$	bi-criteria quality function of $\mathcal{S}_G$

TABLE I: Notations

number of nodes and edges in P, 2) b_p is a size bound to normalize $|P|$, which can be specified as a recognition budget (*i.e.,* the largest summary size a user can understand) [22], and (3) the support $\mathsf{supp}(P, G)$ is defined as $\frac{|G_P|}{|G|}$, where $|G_P|$ (resp. $|G|$) refers to the size (*i.e.,* total number of entities and relationships) in G_P (resp. G). Intuitively, the function $I(\cdot)$ favors larger summaries that have higher support.

B. Summary Diversification

A second challenge is to avoid redundancy among the summaries for knowledge graphs [22]. The summaries can be redundant due to: 1) common "sub-summaries"; and 2) they summarize many common entities and relationships.

Maximal summaries. Most pattern mining tasks employ maximal patterns (patterns with no super-pattern that is more frequent) [26] to avoid redundancy as large common patterns. This carries over for summaries as graph patterns. Given graph G, a d-summary P of G is *maximal* if $\mathsf{supp}(P) \geq \mathsf{supp}(P')$ for every d-summaries P' derived by adding an edge to P.

Difference of Summaries. To cope with the summary redundancy due to commonly summarized entities, we define a distance function diff for two summaries P_1 and P_2 as

$$\mathsf{diff}(P_1, P_2) = 1 - \frac{|V_{G_{P_1}} \cap V_{G_{P_2}}|}{|V_{G_{P_1}} \cup V_{G_{P_2}}|}$$

where $V_{G_{P_1}} = \bigcup_{u \in V_{P_1}} [u]$ (resp. $V_{G_{P_2}} = \bigcup_{u \in V_{P_2}} [u]$); that is, it measures the Jaccard distance between the set of entities summarized by P_1 and P_2 in their base graphs.

One may verify that diff is a metric, *i.e.,* for any three d-summaries P_1, P_2 and P_3, $\mathsf{diff}(P_1, P_2) \leq \mathsf{diff}(P_1, P_3) + \mathsf{diff}(P_2, P_3)$. We quantify entity set difference as a more important factor of summary difference than edge set difference. Label/type difference of the entities can also be applied to quantify weighted V_{G_P} in diff.

Example 4: Consider the 2-summaries P_1-P_3 of the graph G (with 45 entities and edges) in Fig. 1. Let the summary size bound b_p=8, we observe the following. (1) The size of the base graph $|G_{P_1}|$ is 20. Hence, $\mathsf{supp}(P_1, G)$=$\frac{20}{45}$. the informativeness of P_1 $I(P_1)$=$\frac{7}{8} * \frac{20}{45}$=0.39. Similarly, $I(P_2)$=$\frac{6}{8} * \frac{12}{45}$=0.20, and $I(P_3)$=$\frac{6}{8} * \frac{11}{45}$=0.18. One may verify that P_1 is also a maximal summary within size 8. (2) $\mathsf{diff}(P_1, P_2)$=1-$\frac{2}{14}$=0.86, where they summarize two common entities {The_Eagles, Def_Leppard}). Similarly, $\mathsf{diff}(P_1, P_3)$= 1.00, and $\mathsf{diff}(P_2, P_3)$=0.90. □

C. Diversified Knowledge Graph Summarization

Good summarizations should cover diverse concepts in a knowledge graph with informative summaries. We introduce a bi-criteria function F that integrates informativeness $I(\cdot)$ and distance $\mathsf{diff}(\cdot)$ functions. Given a summarization $\mathcal{S}_G$ for knowledge graph G, F is defined as:

$$F(\mathcal{S}_G) = (1-\alpha) \sum_{P_i \in \mathcal{S}_G} I(P_i) + \frac{\alpha}{\mathsf{card}(\mathcal{S}_G)-1} \sum_{P_i \neq P_j \in \mathcal{S}_G} \mathsf{diff}(P_i, P_j)$$

where 1) $\mathsf{card}(\mathcal{S}_G)$ refers to the number of summaries it contains; and 2) $\alpha (\in [0,1])$ is a tunable parameter to trade-off informativeness and diversification. Note that we scale down the second summation (diversification) in $F(\mathcal{S}_G)$ which has $\frac{\mathsf{card}(\mathcal{S}_G)(\mathsf{card}(\mathcal{S}_G)-1)}{2}$ terms, to balance out the fact that the first summation (informativeness) has $\mathsf{card}(\mathcal{S}_G)$ terms.

Based on the quality metrics, we next introduce a graph summarization problem for knowledge graphs.

Diversified graph summarization. Given a knowledge graph G, integers k and d, and a size budget b_p, the *diversified knowledge graph summarization* problem is to compute a summarization $\mathcal{S}_G$ of G as a top k summary set, where

- ◦ each summary in $\mathcal{S}_G$ is a maximal d-summary with size bounded by b_p; and
- ◦ the overall quality function $F(\mathcal{S}_G)$ is maximized.

Example 5: Set b_p=8 and α=0.1, a top-2 diversified summarization $\mathcal{S}_G$ of G (Fig. 1) is $\{P_1, P_2\}$, with total quality score $F(\mathcal{S}_G)$= $0.9 * (0.39 + 0.20)$+$0.1 * 0.86$ = 0.62. □

This problem is (not surprisingly) intractable. One may verify that it is NP-hard, by constructing a reduction from the maximum dispersion problem [10] that is known to be NP-complete. In the next section, we present algorithms to solve the summarization problem with desirable quality and efficiency guarantees.

IV. Discovering summarization

We next study feasible mining algorithms for diversified knowledge graph summarizations.

A. Approximability

Finding optimal summarization by mining and verifying all k-subsets of summaries is not practical for large G, especially under resource constraints such as memory and response time. One may want to resort to faster approximation algorithms that find sub-optimal d-summaries with provable quality guarantees. Let $\mathcal{S}_G^*$ denote the optimal summarization that maximizes the quality function F. For any given knowledge graph G, an ϵ-approximation mining algorithm returns a summarization $\mathcal{S}_G$, such that $F(\mathcal{S}_G) \geq \frac{F(\mathcal{S}_G^*)}{\epsilon}$ ($\epsilon \geq 1$). We show that diversified knowledge summarization is approximable, by introducing such an algorithm.

Approximation. The algorithm, denoted as $\mathsf{approxDis}$, invokes a mining procedure sumGen to discover all the maximal d-summaries $\mathcal{C}_P$. It then greedily adds a summary pair $\{P, P'\}$ from $\mathcal{C}_P$ to $\mathcal{S}_G$ that maximally improves a function $F'(\mathcal{S}_G)$, where F' is defined as

$$F'(P, P') = (1-\alpha)(I(P) + I(P')) + \alpha * \mathsf{diff}(P, P')$$

That is, F' is obtained by rounding down the original function F, which guarantees an approximation ratio for F. This step is repeated $\lfloor \frac{k}{2} \rfloor$ times to obtain top-k d-summaries $\mathcal{S}_G$. If k is odd, it selects an additional summary P that maximizes $F(\mathcal{S}_G \cup \{P\})$ after $\lfloor \frac{k}{2} \rfloor$ rounds of selection.

Procedure sumGen. The procedure sumGen follows existing A-priori graph pattern mining (*e.g.,* [6]) to discover d-summaries. The difference is that it invokes the polynomial-time verification procedure (Lemma 1; see Section II-B) to verify if a graph pattern P is a d-summary of G, and if so, checks if P is maximal. It adds all maximal d-summaries to set $\mathcal{C}_P$ and return it to $\mathsf{approxDis}$.

Analysis. The diversified summarization over the maximal summaries set $\mathcal{C}_P$ can be reduced to the max-sum diversification problem [10]. The latter is 2-approximation if the difference function is a metric. The algorithm $\mathsf{approxDis}$ adopts a greedy strategy to select summaries that maximizes a rounded down function F', which simulates a 2-approximation algorithm for the max-sum diversification constrained by the difference metric diff.

B. Anytime Summarization

The main drawback of the algorithm $\mathsf{approxDis}$ is that it needs to *wait* until all the maximal summaries are mined to compute the summarization. In contrast to frequent subgraph mining, the verification is no longer a major bottleneck for d-summaries. This indicates that we can quickly select maximal d-summaries in an online fashion, over a *stream* of summary candidates. We next develop an *anytime* summarization algorithm with tunable performance, controlled by two parameters on time and space, respectively.

Anytime measures. Given a problem I and a function $\mathcal{J}$ to measure the quality of a solution, an algorithm $\mathcal{A}$ is an *anytime* [2] of I *w.r.t.* $\mathcal{J}$ if: a) $\mathcal{A}$ returns an answer $\mathcal{A}(I)_t$ when it is interrupted at any time t; and b) $\mathcal{J}(\mathcal{A}(I)_{t'}) \geq \mathcal{J}(\mathcal{A}(I)_t)$ for $t' \geq t$, i.e., quality of results improve with more time.

To measure the quality of anytime output of $\mathcal{A}$, we introduce a notation of *anytime approximation*.

Anytime approximation. An optimal anytime algorithm $\mathcal{A}^*$ will return locally optimal answer $\mathcal{A}^*(I)_t$ at any time t, given the fraction of the input accessed upto time t. We say an anytime algorithm $\mathcal{A}$ is an *anytime ϵ-approximation* algorithm with respect to I and $\mathcal{J}$, if at *any* time t, the answer $\mathcal{A}(I)_t$ returned by $\mathcal{A}$ approximates the answer $\mathcal{A}^*(I)_t$ with a fixed approximation ratio ϵ.

Anytime approximation is desirable: it verifies that the answer produced by $\mathcal{A}$ has guaranteed quality compared to the best answer one can obtain from the "seen" input at any time t. Note that it is "weaker" than an anytime quality guarantee with respect to the globally optimal answer (over the entire

```
Algorithm streamDis
Input: a graph $G$, integer $k$,
       threshold $l_p$, $b_p$, time bound $t_{max}$;
Output: summarization $\mathcal{S}_G$.
1.  Initialization: $\mathcal{S}_G$:=$\emptyset$; $\mathcal{C}_P$:= $\emptyset$; termination:=false; and $L$:=$\emptyset$;
2.  while termination $\neq$ true do
      // fetch a new summary(bounded by $b_p$) from summary stream
3.    summary $P_t$ := sumGen $(G, k)$;
4.    $\mathcal{C}_P$:=$\mathcal{C}_P \cup \{P_t\}$;
5.    for each $L_i \in \mathcal{L}$ do
6.      Update top $l_p$ summary pairs in $L_i$ that maximizes $F'(\cdot)$;
7.    Update $\mathcal{S}_G$ with top $\lfloor\frac{k}{2}\rfloor$ summary pairs in $\mathcal{L}$;
8.    if no new summary can be generated
      or time-bound reaches $t_{max}$ then
9.      termination:=true;
10. return $\mathcal{S}_G$;
```

Fig. 2: Algorithm streamDis

input) [2]. The latter typically requires the prior knowledge of error distribution. We do not make such assumptions, and defer this study to future work.

We present the main result of this section below.

Theorem 2: *There exists an anytime 2-approximation algorithm that computes a diversified summarization for a knowledge graph, which takes (measured by input size) (1) $O(N_t * b_p(b_p+|V|)(b_p+|E|)+\frac{k}{2}N_t^2)$ time, and (2) $O(k*N_t+|\mathcal{S}_G|)$ space, where N_t is the number of summaries it verified when interrupted, and $|\mathcal{S}_G|$ refers to the total size of summaries and their base graphs.* □

In a nutshell, the anytime algorithm maintains a top-k set $\mathcal{S}_G$, and keeps track of a bounded number of summaries that can potentially improve the summarization quality $F(\mathcal{S}_G)$. Instead of waiting for all the summaries to be generated by sumGen, it operates on a *summary stream*. It updates $\mathcal{S}_G$ whenever a new summary that can improve $F(\mathcal{S}_G)$ is identified in the stream, by referring to a set of cached summaries. When interrupted, it returns the up-to-date summarization $\mathcal{S}_G$.

Below we first introduce the auxiliary structure used by the algorithm, followed by the actual algorithm.

Auxiliary structure. Our anytime algorithm maintains the following: 1) a set $\mathcal{C}_P$ of the maximal summaries verified by sumGen; and 2) a set $\mathcal{L}$ of ranked lists, one list L_i for each maximal summary $P_i \in \mathcal{C}_P$. Each list L_i caches the top-l_p ($n \in [1, k-1]$) summary pairs (P_i, P_j) in $\mathcal{C}_P$ that have the highest $F'(P_i, P_j)$ score, where $F'(\cdot)$ refers to the revised quality function (see Section IV-A). It bounds the size of the list L_i based on a tunable parameter l_p, which can be adjusted as per the available memory.

Algorithm. Given G, integer k, and two threshold b_p and l_p, the algorithm, denoted as streamDis, computes a summerization $\mathcal{S}_G$ as follows (see Fig. 2). It first initializes $\mathcal{S}_G$, $\mathcal{C}_P$, $\mathcal{L}$, and a flag termination (set as false) for the termination condition (line 1). It then iteratively conducts the following steps.

1) Invokes sumGen to fetch a newly generated summary P_t. Note that the procedure sumGen can be easily modified to return a single summary after evaluation, instead of returning a set of summaries in a batch.

2) Updates $\mathcal{C}_P$ and the list $\mathcal{L}$ (lines 5-7) based on the newly fetched summary P_t. For each summary $P_i \in \mathcal{C}_P$, it computes the quality score $F'(P_i, P_t)$, and updates the top-l_p list L_i of P_i by replacing the lowest scoring pair (P_i, P') with (P_i, P_t), if $F'(P_i, P')<F'(P_i, P_t)$.

3) Incrementally updates the top-k summaries $\mathcal{S}_G$ (lines 5-6). Greedily selects top $\lfloor\frac{k}{2}\rfloor$ pairs of summaries with maximum quality $F'(\cdot)$ from the list set $\mathcal{L}$, and adds the summaries to $\mathcal{S}_G$. If $|\mathcal{S}_G| < k$, a summary $P \in \mathcal{C}_P \setminus \mathcal{S}_G$ that maximizes the quality $F(\mathcal{S}_G \cup \{P\})$ is added to $\mathcal{S}_G$.

The above process is repeated until the termination condition is satisfied (lines 8-9): a) no new summary can be discovered in sumGen; or b) running time reaches the time-bound t_{max}. The up-to-date $\mathcal{S}_G$ is then returned.

Example 6: Consider the sample graph G in Fig. 1. Let b_q=8, k=2, d=2 and α=0.1. streamDis computes a summarization $\mathcal{S}_G$ as follows. In the first round, it invokes sumGen to discover a maximal 2-summary, *e.g.,* P_3, and initializes $\mathcal{C}_P$ and $\mathcal{S}_G$ with P_3. In round 2, it discovers a new 2-summary P_2, verifies $F'(P_2, P_3)$ as $0.9*(0.20+0.18)+0.1*0.90 = 0.43$, and updates L_2, L_3 and $\mathcal{S}_G$ as shown below.

round	L	$\mathcal{C}_P$	$\mathcal{S}_G$
2	L_2={<(P_2, P_3),0.43} L_3={<(P_3, P_2),0.43}	$\{P_2, P_3\}$	$\{P_2, P_3\}$
3	L_1={<(P_1, P_2),0.62} L_2={<(P_2, P_1),0.62} L_3={<(P_3, P_1),0.61}	$\{P_1, P_2, P_3\}$	$\{P_1, P_2\}$

In round 3, it discovers summary P_1, and updates the top-1 entries in each list of $\mathcal{L}$. It verifies the pairwise quality scores as $F'(P_1, P_2)$= 0.62 (see Example 5) and $F'(P_1, P_3)$=$0.9*(0.39+0.18)+0.1*1.00$ = 0.61. The new top elements in the lists L_1, L_2, and L_3 are hence updated to (P_1, P_2), (P_2, P_1) and (P_3, P_1) respectively. Hence, it replaces $\{P_2, P_3\} \in \mathcal{S}_G$ with $\{P_1, P_2\}$, and updates the auxiliary structures as follows.

As all the maximal summaries within size 8 are discovered, streamDis terminates and returns $\mathcal{S}_G$=$\{P_1, P_2\}$. □

Analysis. The algorithm streamDis is an anytime 2-approximation algorithm. (1) It can be interrupted at any time-bound t_{max} to return $\mathcal{S}_G$; and 2) the set $\mathcal{S}_G$ is incrementally updated such that the overall quality $F(\mathcal{S}_G)$ monotonically increases (line 5). Let $\mathcal{C}_{P_t}$ be the set of summaries generated upto time t, and $\mathcal{S}^*_{G_t}$ be the optimal summarization over the cached summaries $\mathcal{C}_{P_t}$. When l_p=k-1, streamDis simulates its 2-approximation counterpart approxDis by accessing all cached summaries $\mathcal{C}_{P_t}$, and produces a summarization $\mathcal{S}_{G_t}$ where $F(\mathcal{S}_{G_t}) \geq \frac{F(\mathcal{S}^*_{G_t})}{2}$ for a given time t (see Appendix).

V. Knowledge Graph Search with Summaries

Knowledge graph search is the cornerstone for advanced knowledge mining tasks. A query Q is typically represented as a graph (V_q, E_q, L_q) [11], [15], [19], [27]. Given a knowledge graph G=(V, E, L), the answer $Q(G)$ of Q in G refers to the set of all the subgraphs of G that are isomorphic to Q.

We show that d-summaries can support fast knowledge graph search under resource bounds (*e.g.,* number of entities to be accessed). This is especially desirable in large-scale graph mining with limited computational resources [8].

"Summaries+Δ" Scheme. Given a query Q, a knowledge graph G and a summarization $\mathcal{S}_G$ of d-summaries, our query evaluation algorithm, denoted as evalSum, only refers to select d-summaries in $\mathcal{S}_G$ and their base graphs as "materialized views" [7], and fetches additional data in G only when necessary. The algorithm has the following two steps.

- It selects a set of summaries from $\mathcal{S}_G$ with "materialized" base graphs that contains the potential answers of Q as much as possible, and
- It refers to the base graphs to compute the (partial) answer $Q(G)$, and fetches bounded amount (Δ) of data from G to complete the computation of $Q(G)$, only when necessary.

We next introduce the summary selection strategy in our "summaries+Δ" scheme.

Summary Selection. Ideally, a query Q can be evaluated by only accessing the base graphs of the summaries in $\mathcal{S}_G$ as materialized views. We say a query Q is *covered* by $\mathcal{S}_G$, if every edge in $Q(G)$ is contained in one or more base graphs of the summaries in $\mathcal{S}_G$. Given a query Q and a summarization $\mathcal{S}_G$, the *summary selection* problem is to find a set $\mathcal{P}$ of n summaries in $\mathcal{S}_G$, such that the maximum fraction of Q is covered by $\mathcal{P}$, with a bounded total size of base graphs B.

The result below connects d-similarity with query coverage.

Lemma 3: *A query Q is covered by a set of d-summaries $\mathcal{S}_G$, if and only if $\bigcup_{P_i \in \mathcal{S}_G} Q_{P_i} = Q$, where Q_{P_i} refers to the maximum base graph induced by the d-similarity between each summary $P_i \in \mathcal{S}_G$ and Q.* □

We defer the detailed proof of Lemma 3 in Appendix.

Based on Lemma 3, the selection procedure, denoted as Select-Sum, uses a greedy strategy to add the summaries $\mathcal{P}$ that maximally covers Q, and have small base graphs in G. To this end, it dynamically updates a rank $r(P) = \frac{|E_{Q_P} \setminus E_c|}{|G_P|}$ for the summaries in $\mathcal{S}_G$, where (1) E_{Q_P} refers to the edge set of the base graph Q_P, induced by the d-similarity between the summary P and query Q (as a graph) (Lemma 3); (2) E_c refers to the edges of Q that has been "covered", *i.e.,* already in a base graph of a selected summary $P' \in \mathcal{P}$. In each round of selection, a summary with highest $r(P)$ is added to $\mathcal{P}$, and the ranks of the remaining summaries in $\mathcal{S}_G$ are dynamically updated. The process repeats until n patterns are selected, or the total size of the base graphs reaches B.

The selection procedure Select-Sum is efficient: it takes $O(\mathsf{card}(\mathcal{S}_G) b_q (b_q + |V_p|)(b_q + |E_p|))$ time, where b_q and $|V_q|$, $|E_q|$ are typically small. Better still, it guarantees the approximation ratio $(1 - \frac{1}{e})$ for optimal summaries under budget B, by reducing the summary selection to the budgeted maximum coverage problem.

Δ-Validation. Once the summaries $\mathcal{P}$ are selected, the algorithm evalSum (1) identifies Q_1 in Q covered by $\mathcal{P}$, (2) evaluates Q_1 by only accessing the base graphs of the summaries in $\mathcal{P}$, using existing query evaluation algorithm based on subgraph isomorphism (*e.g.,* [20]); and 3) If necessary, refines the matches for Q by visiting additional nodes and edges in G, up to a bound Δ. Note that evalSum never visits more than B+Δ nodes and edges in G. In practice, both B and Δ can be tuned to adapt to the actual resource bounds.

Remarks. View-based query evaluation [7] studies interactions between queries and views in the same language. In contrast, we verify that d-summaries can be mined and selected efficiently for query evaluation even with different semantics (*e.g.,* subgraph isomorphism), as verified by our experimental study (Section VI). The summaries can be used to support fast graph mining under other matching semantics that are more "strict" than d-similarity. We defer this study in future work.

VI. Experimental Evaluation

Using real-world and synthetic knowledge graphs, we conducted three sets of experiments to evaluate 1) Performance of the summary mining algorithms approxDis and streamDis; 2) Effectiveness of the algorithm evalSum for query evaluation; and 3) Effectiveness of summaries, using a case study.

Experimental Setting. We used the following setting.

Datasets. We use three real-life knowledge graphs: 1) *DBpedia*[1] consists of 4.86M nodes and 15M edges, where each entity carries one of the 676 labels (*e.g.,*'Settlement', 'Person', 'Building'); 2) *YAGO*[2], a sparser graph compared to *DBpedia* with 1.54M nodes and 2.37M edges, but contains more diversified (324343) labels; and 3) *Freebase* (version 14-04-14)[3], with 40.32M entities, 63.2M relationships, and 9630 labels.

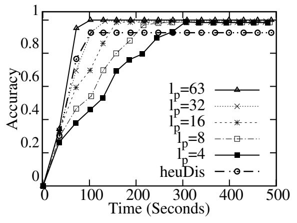
We employ *BSBM*[4] e-commerce benchmark to generate synthetic knowledge graphs over a set of products with different types, related vendors, consumers, and views. The generator is controlled by the number of nodes (up to $60M$), edges (up to $152M$), and labels drawn from an alphabet Σ of 3080 labels.

Queries. To evaluate evalSum algorithm, we generated 50 subgraph queries Q=(V_q, E_q, L_q) over real-world graphs with size controlled by $(|V_p|, |E_p|)$. We inspected meaningful queries posed on the real-world knowledge graphs, and generated queries with labels drawn from their data (domain, type, and attribute values). For synthetic graphs, we generated 500 queries with labels drawn from BSBM alphabet. We generate queries with different topologies (star, trees, and cyclic patterns) and sizes (ranging from (4,6) to (8,14)).

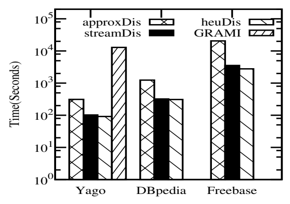
Algorithm. We implemented all the algorithms in Java:

1) Summarization algorithms approxDis and streamDis, are

[1]http://dbpedia.org
[2]http://www.mpi-inf.mpg.de/yago
[3]http://freebase-easy.cs.uni-freiburg.de/dump/
[4]http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/

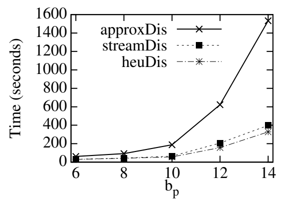


(a) streamDis: quality vs. time

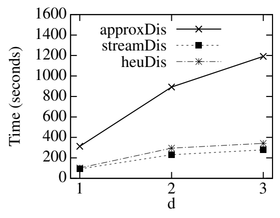


(b) Real-world datasets

Fig. 3: Anytime Performance



(a) Summarization: Varying b_p



(b) Summarization: Varying d

Fig. 4: The impact of bound b_p and d

compared with: a) heuDis, an anytime heuristic counterpart of streamDis that incrementally maintains a diversified summarization $\mathcal{S}_G$ over the stream of maximal summries following [17]. Each time a new summary P is seen by sumGen, it swaps out a summary P' in $\mathcal{S}_G$ if $F(\mathcal{S}_G \setminus \{P'\} \cup \{P\}) > F(\mathcal{S}_G)$; and b) GRAMI, an open-source graph pattern mining tool [6] to discover frequent subgraph patterns as summaries. Here the base graph is computed as the union of all the subgraphs isomorphic to the summary in G.

2) Query evaluation algorithm evalSum, compared with the following variants: a) evalRnd, that performs random selection instead of using Select-Sum; b) evalGRAMI,that employs frequent graph patterns mined by GRAMI; and c) evalNo that evaluates Q by directly employing an optimized subgraph isomorphism algorithm in [20]. We also allow a resource bound Δ to be posed on evalRnd and evalGRAMI as in our "summary+Δ" scheme, to allow them to return approximate answers by fetching at most Δ additional data from G.

We ran all our experiments on a linux machine powered by an Intel 2.4 GHz CPU with 128 GB of memory. We ran each experiment 5 times and report the averaged results.

Overview of Results. We summarize our findings below.

1) It is feasible to summarize large real-world graphs with d-summaries (**Exp-1**). Our algorithm streamDis produces high-quality summarization (*e.g.,* at least 99% accurate with respect to its 2-approximation counterpart approxDis) within a smaller time budget (90 seconds), on YAGO with 3.91 million entities and relationships. It is orders of magnitude faster than summarizing by mining frequent subgraph patterns (GRAMI).
2) The d-summaries significantly improves the efficiency of query evaluation (**Exp-2**). For example, evalSum is 40 times faster than evalNo (without using summarization) over YAGO. It is 2.5 times faster than its counterpart using frequent subgraph patterns as views. Moreover, the summary selection is effective: evalSum outperforms evalRnd (that randomly select summaries) by 2 times, using at most 64 summaries. Finally, it does not take much additional cost ($\Delta \leq 5\%$ of graph size) to find exact answers.
3) Our case study shows that summarization captured by d-summaries is concise, and provides a good coverage for diversified entities (**Exp-3**).

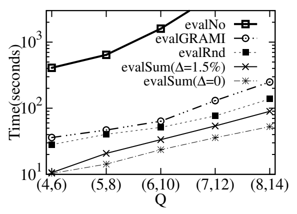
We next report the details of our findings.

Exp-1: Effectiveness of summary discovery. We fixed parameter α=0.5 for diversification, k=64, the summary size bound b_p=6, number of hops $d = 1$ and l_p=k-1 for this experiment, unless otherwise specified. In addition, we set a support threshold θ=0.005 to find frequent maximal patterns in the summary mining procedure sumGen used by approxDis, streamDis, and heuDis. For the real-life datasets, we also excluded "overly general" (top 2% frequent) labels such as "thing", "place", and "person".

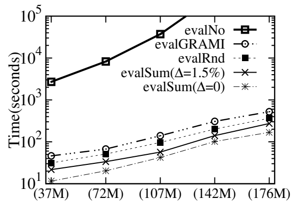
Anytime performance. We evaluate the impact of time constraint t and size constraint l_p (the number of summary pairs stored for each summary in streamDis), on the anytime performance of streamDis and heuDis. We define the *anytime accuracy* of streamDis as $\frac{F(\mathcal{S}_{G_t})}{F(\mathcal{S}_G)}$, where $\mathcal{S}_{G_t}$ refers to the summaries returned by streamDis at time t, and $\mathcal{S}_G$ refers to the one returned by approxDis. The accuracy of heuDis is defined similarly. Specifically, we report the "convergence" time of streamDis and heuDis when the accuracy reaches 99%, for a fair comparison with approxDis and GRAMI.

Fig. 3(a) shows the anytime accuracy of streamDis and heuDis over YAGO. 1) The quality of summarization returned by streamDis increases monotonically as t and l_p increases; 2) Convergence speed of streamDis to near-optimal summarization improves with increasing l_p values as more summary pairs are stored and compared. Remarkably, streamDis converges in less than 100 seconds when l_p=63; and 3) heuDis converges faster than streamDis, but stops at accuracy 0.9 on average. These results verify that streamDis provides a principled way to trade-off accuracy and time, and converges early by processing a small number of summaries. Remarkably, streamDis converges after processing 50 patterns instead of 280 patterns in total when l_p=63.

Efficiency of Summarization. We evaluate the efficiency of approxDis, streamDis, heuDis, and GRAMI over the real-world knowledge graphs. For the two anytime algorithms streamDis and heuDis, we report their convergence time. For GRAMI, we carefully adjusted a support threshold to allow the generation of patterns with similar label set and size to those from approxDis. As shown in Fig.3(b), 1) streamDis and approxDis are both orders of magnitude faster than GRAMI. The latter does not run to completion within 10 hours over both DBpedia and Freebase; 2) Performance of streamDis is comparable to that of heuDis, and streamDis is 3-6 times faster than approxDis with comparable accuracy; 3) streamDis is feasible over large knowledge graphs. For example, it takes less than 100 seconds to produce high-quality summaries by verifying only 64 summaries for YAGO.



(a) evalSum: Varying $|Q|$



(b) evalSum: Varying $|G|$

Fig. 5: Efficiency of evalSum

We also evaluated the scalability of our mining algorithms using larger synthetic graphs, by varying $|G|$ from $(10M, 27M)$ to $(60M, 152M)$ (not shown). The algorithms streamDis and heuDis scale well with larger $|G|$, and are less sensitive to increasing $|G|$ due to their speed of covergence. streamDis is feasible over large graphs. Remarkably, it summarizes a knowledge graph of size $(60M, 152M)$ within 1 hours. In contrast, GRAMI does not run to completion in 10 hours even with graphs of size $(10M, 27M)$.

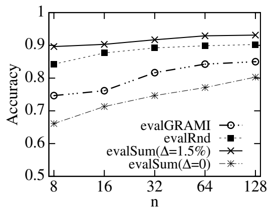
Varying b_p. Using the settings in Fig. 3(a) over YAGO, we varied the summary size threshold b_p from 6 to 14 (node # + edge#). As shown in Fig.4(a), all the algorithms take more time with larger values of b_p, as more candidate patterns are examined and verified. Remarkably, on average streamDis is 4 times faster than approxDis with 90% accuracy.

Varying d. Using the same setting over YAGO, we varied d from 1 to 3. Fig. 4(b) shows that all the algorithms take more time with larger d, as expected. Additionally, the convergence time of streamDis and heuDis are less sensitive to increasing of d when compared with approxDis.

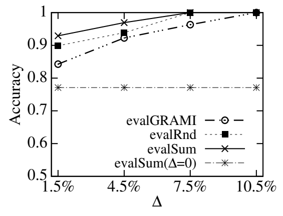
Exp-2: Effectiveness of evalSum. We evaluate the efficiency of evalSum, and compare it with evalSum (Δ=0), evalRnd, evalGRAMI, and evalNo.

Varying $|Q|$. We set n = 64 and card ($\mathcal{S}_G$)=500, and varied the query size $|Q|$ from $(4, 6)$ to $(8, 14)$ over YAGO. Fig. 5(a) tells us the following: 1) By leveraging 2-summaries, evalSum and evalSum (Δ=0) find answers in less than 60 seconds, and improves the efficiency of evalNo by 40 and 50 times respectively; evalNo does not terminate within 10^4 seconds for queries with 6 nodes. 2) On average, evalSum and evalSum (Δ=0) are 2.5 and 4 times faster than evalGRAMI. This demonstrates that the d-summaries are more effective than using frequent subgraph patterns as "views"; 3) Our summary selection strategy is effective: evalSum is 2 times faster than evalRnd that employs random selection. In all cases, it takes less than 10 seconds to select the summaries.

Varying $|G|$. We evaluate the scalability of evalSum with large synthetic graphs. We set $|Q|$=(6,10), card ($\mathcal{S}_G$)=500, and varied the size of synthetic graph $|G|$ from (10M,27M) to (50M,126M). We make the following observations from Fig. 5(b): 1) all algorithms take longer time for large $|G|$ as expeceted; (2) evalSum and evalSum (Δ=0) scale better than all the other algorithms. evalSum is reasonably efficient: when $|G|$=(10M,27M), evalSum takes 35 seconds.



(a) Accuracy *w.r.t.* n



(b) Accuracy *w.r.t.* Δ

Fig. 6: Accuracy of evalSum

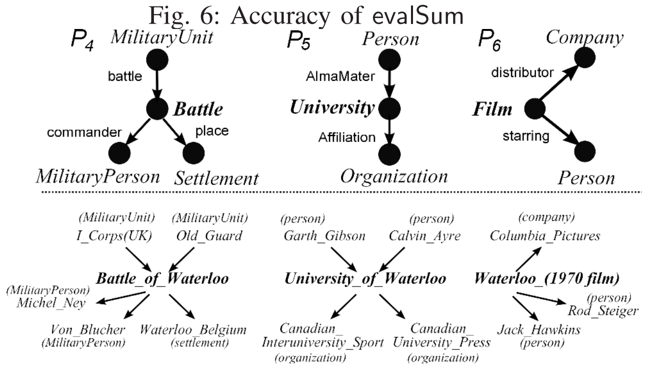


Fig. 7: Real-life Summaries: DBpedia.

Accuracy. We evaluated the accuracy of the query answers produced by evalSum (Δ=0), evalRnd, and evalGRAMI. Let $Q(G)_A$ be the set of node and edge matches returned by a query evaluation algorithm A, and $Q(G)$ the exact match set. We define the accuracy of algorithm A as the Jaccard similarity $\frac{Q(G)_A \cap Q(G)}{Q(G)_A \cup Q(G)}$. For evalNo, the accuracy is 1. As shown in Fig 6(a) and 6(b), all algorithms perform better with larger n, and evalSum achieves highest accuracy with $\Delta = 1.5\%$. Remarkably, evalSum can get 100% accuracy with 7.5% of original graph, however, evalGRAMI needs more data compared to evalSum.

Query diversity. We also compared the performance of evalSum (Δ=1.5%) with evalNo over three categories of queries over YAGO: 1) **Frequent**, which carries most frequent labels in G; 2) **Diversified**, where the query node labels range over a diversified set of labels; and 3) **Mixed** that combines queries uniformly sampled from the two categories. The table below shows the results, where C (resp. C_{ISO}) refers to the total number of nodes and edges (including summaries) visited by evalSum (resp. evalNo).

	Time(seconds)	Accuracy	$\frac{C}{C_{ISO}}$	$\frac{C}{\lvert G\rvert}$
Diversified	31.63	0.9341	0.2761	0.0395
Frequent	39.87	0.9443	0.1768	0.0762
Mixed	33.09	0.9319	0.2098	0.0514

evalSum takes more time for **Frequent** queries due to large candidates for frequent labels and visits more entities with different labels for **Diversified** queries. In general, it visits no more than 27% (resp. 8%) data visited by evalNo (resp. $|G|$) to achieve accuracies not less than 93%.

Exp-3: Case study. We performed case studies to test the number of summaries needed to "cover" all the entity types for 50 sampled ambiguous keywords from DBpedia (*e.g.,* "waterloo", "Tesla", "Avatar"). Each keyword has on average 4 different types. We observed that for highly diverse summarization (*e.g.,* $\alpha = 0.9$), less number of summaries (*e.g.,* $k = 9$) are needed to cover all the entity types. For all cases, it takes at most 15 summaries to cover all the types for each keyword. In contrast, most of the summaries from GRAMI are redundant small patterns, and cannot cover the entity types of keywords even when k=64.

Three real-life 2-summaries for keyword "waterloo" discovered from DBpedia are shown in Fig. 7, which distinguish "waterloo" as Battle entities (P_1), University (P_2), and Films (P_3). These summaries suggest intermediate keywords as enhanced queries (*e.g.,*Military Person); and can also suggest answers for *e.g.,* Précis queries [22] that find diversified facts of a single entity.

VII. Conclusions

We proposed a class of d-summaries, and developed feasible summary mining algorithms that summarize large, schema-less knowledge graphs. We also developed efficient query evaluation algorithm by selecting and accessing a small number of summaries and their base graphs. Our experimental results verified that our algorithms efficiently generate concise summaries that significantly reduces query evaluation cost in schema-less knowledge graphs. Our future work is to enable query suggestion and resource-bounded query evaluation by referring to summaries, for more types of query classes.

Acknowledgment

This research is supported by the National Science Foundation under grant BIGDATA-1633629 and Google Faculty Research Award.

References

[1] C. C. Aggarwal and H. Wang. A survey of clustering algorithms for graph data. In *Managing and Mining Graph Data*, pages 275–301. 2010.
[2] B. Arai, G. Das, D. Gunopulos, and N. Koudas. Anytime measures for top-k algorithms. In *VLDB*, pages 914–925, 2007.
[3] G. Buehrer and K. Chellapilla. A scalable pattern mining approach to Web graph compression with communities. In *WSDM*, 2008.
[4] Q. Chen, A. Lim, and K. W. Ong. D (k)-index: An adaptive structural summary for graph-structured data. In *SIGMOD*, pages 134–144, 2003.
[5] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, 2014.
[6] M. Elseidy, E. Abdelhamid, S. Skiadopoulos, and P. Kalnis. Grami: Frequent subgraph and pattern mining in a single large graph. *Proceedings of the VLDB Endowment*, 7(7):517–528, 2014.
[7] W. Fan, X. Wang, and Y. Wu. Answering graph pattern queries using views. In *ICDE*, 2014.
[8] W. Fan, X. Wang, and Y. Wu. Querying big graphs within bounded resources. In *SIGMOD*, 2014.
[9] W. Fan, X. Wang, Y. Wu, and J. Xu. Association rules with graph patterns. *Proceedings of the VLDB Endowment*, 8(12):1502–1513, 2015.
[10] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *WWW*, 2009.
[11] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. Naga: Searching and ranking knowledge. In *ICDE*, pages 953–962, 2008.
[12] N. S. Ketkar, L. B. Holder, and D. J. Cook. Subdue: Compression-based frequent pattern discovery in graph data. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, pages 71–76. ACM, 2005.
[13] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos. Vog: Summarizing and understanding large graphs. In *SDM*, 2014.
[14] W. Le, S. Duan, A. Kementsietsidis, F. Li, and M. Wang. Rewriting queries on SPARQL views. In *WWW*, 2011.
[15] W. Le, F. Li, A. Kementsietsidis, and S. Duan. Scalable keyword search on large rdf data. *TKDE*, 26(11):2774–2788, 2014.
[16] S. Ma, Y. Cao, W. Fan, J. Huai, and T. Wo. Capturing topology in graph pattern matching. *Proceedings of the VLDB Endowment*, 5(4):310–321, 2011.
[17] E. Minack, W. Siberski, and W. Nejdl. Incremental diversification for very large sets: a streaming-based approach. In *SIGIR*, 2011.
[18] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *SIGMOD*, 2008.
[19] B. Quilitz and U. Leser. Querying distributed RDF data sources with sparql. In *ESWC*, pages 524–538, 2008.
[20] X. Ren and J. Wang. Exploiting vertex relationships in speeding up subgraph isomorphism over large graphs. *Proceedings of the VLDB Endowment*, 8(5):617–628, 2015.
[21] M. Riondato, D. Garcia-Soriano, and F. Bonchi. Graph summarization with quality guarantees. In *ICDM*, pages 947–952, 2014.
[22] M. Sydow, M. Pikuła, and R. Schenkel. To diversify or not to diversify entity summaries on rdf knowledge graphs? In *Foundations of Intelligent Systems*. 2011.
[23] Y. Tian, R. Hankins, and J. Patel. Efficient aggregation for graph summarization. In *SIGMOD*, 2008.
[24] Y. Wu, S. Yang, M. Srivatsa, A. Iyengar, and X. Yan. Summarizing answer graphs induced by keyword queries. *Proceedings of the VLDB Endowment*, 6(14):1774–1785, 2013.
[25] D. Xin, H. Cheng, X. Yan, and J. Han. Extracting redundancy-aware top-k patterns. In *SIGKDD*, 2006.
[26] X. Yan and J. Han. Closegraph: mining closed frequent graph patterns. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 286–295, 2003.
[27] S. Yang, Y. Wu, H. Sun, and X. Yan. Schemaless and structureless graph querying. *PVLDB*, 7(7):565–576, 2014.

Appendix: Proofs

Cost Analysis of streamDis. It takes $O(N_t(b_p + |V|)(b_p + |E|))$ time for the procedure sumGen to generate and verify a total of N_t d-summaries with size bounded by b_q at time t; and $O(\frac{k}{2}|\mathcal{C}_P|^2)$ time to compute $\mathcal{S}_G$ using the greedy strategy, where $\mathcal{C}_P$ refers to the candidate set of maximal d-summaries, also bounded by N_t. Hence, the time complexity of streamDis is in $O(N_t(b + |V|)(b + |E|) + \frac{k}{2}N_t^2)$.

It suffice for streamDis to store the top $k-1$ pattern pairs in each list $L_i \in \mathcal{L}$ that maximizes $F'(P_i, P_j)$ to achieve anytime 2-approximation. Indeed, a) streamDis identifies at most $\lfloor \frac{k}{2} \rfloor$ pairwise disjoint summaries each time a new summary is returned by sumGen; and b) for each new summary, at most $2(\lfloor \frac{k}{2} \rfloor - 1)$ summary pairs are replaced in $\mathcal{L}$. Putting (a) and (b) together, a top k summaries can be derived by aggregating all summaries cached in the lists, which guarantees the 2-approximation ratio over "seen" summaries. Hence, the space cost is at most $O(k * N_t + |\mathcal{S}_G|)$. Here $|\mathcal{S}_G|$ is required for "bookkeeping" the summaries and base graphs.

Proof of Lemma 3. 1) **If**. Assume $\bigcup_{P_i \in \mathcal{S}_G} Q_{P_i} = Q$. For each edge e=(u, v) in Q, there exists a summary $P_i \in \mathcal{S}_G$ such that e is in Q_P. Hence there exists an edge e_p=(u_p, v_p) in P_i such that $(u_p, u) \in R_d$ and $(v_p, v) \in R_d$, where R_d is the d-similarity between Q and P. For any edge e'=(u', v') in the answer $Q(G)$ where e is mapped to, one can verify that $(u_p, u') \in R'_d$ and $(v_p, v') \in R'_d$, where R'_d is the d-similarity between P_i and G. Hence Q is covered by $\mathcal{S}_G$ by definition. 2) **Only If**. We prove the Only If condition by contradiction. Assume Q is covered but there exists an edge e in Q not covered by any d-summary. Then there exists at least one match of e not included in any base graph, contradicting to the assumption that Q is covered.

The above analysis completes the proof of Lemma 3.