2016 IEEE 16th International Conference on Data Mining

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 knowledge graphs is challenging under real-world resource constraints such as response time. This paper studies a framework that discover to facilitate knowledge graph search. 1) We introduce a class of summaries characterized by graph patterns. In contrast to conventional summaries defined by frequent subgraphs, 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 summaries that maximizes the informativeness measure. Although this problem is NP-hard, we show that it is 2-approximable. We also introduce an online mining algorithm that trade-off speed and accuracy, under given resource constraints. 3) We develop query evaluation algorithms that make use of the summaries as views. These algorithms efficiently compute (approximate) answers with high accuracy, and only refer to a small number of summaries. Our experimental study verifies that online mining over large knowledge graphs is feasible, and can suggest bounded search in knowledge graphs.**

I. Introduction

Knowledge graphs have been commonly used to represent and manage knowledge bases [3], [7]. Real-world knowledge graphs, unlike familiar relational data, lack the support of well-defined schema and typing system. It is often hard to identify *relevant* data that lead to meaningful answers without any prior knowledge of the underlying graph. Knowledge search is also challenging due to the query ambiguity and resource constraints (*e.g.,* data allowed to be accessed, response time) [6].

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 [7] 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.

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 summaries help the users in understanding G without a daunting inspection of low-level entities. Better still, they suggest small "relevant" data for evaluating the query Q. Indeed, Q can be correctly answered by accessing only the entities summarized by P_1 and P_2 in G. □

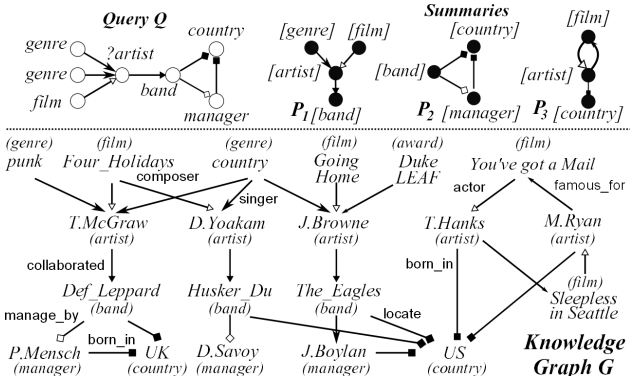


Fig. 1: Knowledge graph, summary patterns, and graph query.

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 [4], [8]. 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. *How can we construct summaries in a schema-less knowledge graph?* Moreover, *How can we leverage the summaries to support knowledge graph search?*

In this work, we exploit summarization of knowledge graphs to facilitate efficient query evaluation. We introduce a class of graph patterns, to summarize similar entities in terms of their labels and neighborhood information up to a bounded hop. We present a stream-style mining algorithm to discover a set of diversified summaries, and introduce query evaluation algorithm using summaries as "views".

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 [8], [11], [13]. These approaches can be classified as graph

2374-8486/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDM.2016.24

compression, attribute summarization, (Bi)simulation relation based summarization and entity summarization. In contrast to prior work, **1)** We introduce *lossy* summaries to facilitate efficient knowledge graph query processing, rather than to restore the exact graph [8], [11]. **2)** We measure summaries by both informativeness and diversity, which is more involved than MDL-based approaches [8]. **3)** In contrast to [11], our summaries do not require tuning effort on parameters to achieve approximate summarization. None of these work addresses diversified summaries as graph patterns.

Answering queries using views: View-based query evaluation has been shown to be effective for SPARQL queries [9] and general graph pattern queries [5]. 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 [5], [9].

II. Knowledge Graphs and summaries

A. Knowledge Graphs and summaries

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 [7]; 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.

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. In practice, additional mapping structures can be used to trace the base graphs for the summaries.

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, for any incoming (resp. outgoing) path ρ of a summary node u with a bounded length d in a summary 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, $d \leq d_m$,

Given a knowledge graph G and an integer d, we define a *summarization* $\mathcal{S}_G$ of G as a set of d-summaries.

Example 2: Fig. 1 illustrates a summarization of the knowledge graph G that contains three 2-summaries P_1, P_2, and P_3. The base graph of P_1 is induced by the entities as follows: [genre] = {country, punk}, [film] = {Going Home, Four_Holidays}, [artist] = {J.Browne, D.Yoakam, T.McGraw}, and [band] ={The_Eagles, Husker_Du, Def_Leppard }. Similarly, P_2 summaries the band Def_Leppard and The_Eagles, their associated country and manager, and P_3 summaries the films You've got a Mail and Sleepless in Seattle, actors T.Hanks and M.Ryan and their countries. P_1 cannot summarize T.Hanks, as the latter has no path to a band as suggested in P_1. □

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, to compute the *largest* base graph of P in G. In contrast to its counterpart defined by frequent subgraphs (NP-hard), the verification of d-summaries is tractable, as verified below.

Lemma 1: *Given a summary P=(V_P, E_P, L_P) and a graph G=(V, E, L), the verification problem is in $O(|V_P|(|V_P| + |V|)(|E_P| + |E|))$ time.* □

We present the detailed proof in [1].

Remarks. d-summaries differ from several other graph patterns like frequent graph patterns, (Bi)simulation-based, dual-simulation-based and neighborhood-based summaries. More details are shown in the full version [1].

B. Interestingness measure

We characterize the interestingness of the summaries in terms of both informativeness and diversity.

Informativeness. $I(P)$ 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 $I(\cdot)$ as:

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

where 1) $|P|$ refers to the size of a summary P, *i.e.,* total number of nodes and edges in P, and (2) 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). In practice, $|P|$ can be normalized by a summary size bound b_p, which can be specified as a recognition budget (*i.e.,* the largest summary size a user can understand) [13], Intuitively, the function $I(\cdot)$ favors larger summaries that have higher support in G, constrained by a size budget b_p.

Summary Diversification. A second challenge is to avoid redundancy among the summaries [13]. The redundancy may due to: 1) common "sub-summaries"; and 2) common entities summarized by two summaries.

Maximal summaries. Most pattern mining tasks employ maximal patterns (patterns with no super-pattern that is more frequent) 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.

C. Diversified Knowledge Graph Summarization

Good summarizations should be both informative and diversified. 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.

III. Discovering Summarization

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

A 2-approximation. We first outline an algorithm, denoted as approxDis, that discovers all maximal d-summaries $\mathcal{C}_P$ with approximation ratio 2. Given graph G, it invokes a mining procedure (denoted as sumGen) to enumerate and verify all maximal d-summaries. 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

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

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. One may verify approxDis identifies $\mathcal{C}_P$ with approximation ratio 2. We present the details of approxDis and sumGen in [1].

The algorithm approxDis needs to *wait* until all the summaries are verified, which may not be feasible when G is large. We can do better. Lemma 1 indicates that the verification cost is not a major bottleneck (contrast this to its frequent subgraph counterpart [4]). This suggests an "stream-style" mining process over a *stream* of (quickly verified) summaries. Better still, (1) the algorithm can be interrupted to report the summarization upon request; and (2) It approximates the optimal answer over the "seen" summaries.

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

Auxiliary structure. The algorithm maintains 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. 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.

Stream-style Summarization. The algorithm, denoted as streamDis, is illustrated in Fig. 2. Given G, integer k, and two threshold b_p and l_p, 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) It invokes sumGen to fetch a newly generated summary P_t (line 3). The procedure sumGen is modified to return one verified summary at a time, instead of waiting and returning a set of summaries in a batch.

2) It updates $\mathcal{C}_P$ and the list $\mathcal{L}$ (lines 4-6) 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

Algorithm streamDis

Input: a graph G, integer k,
threshold l_p, b_p;
Output: summarization $\mathcal{S}_G$ upon request.

1. Initialization: $\mathcal{S}_G$:=$\emptyset$; $\mathcal{C}_P$:= $\emptyset$; termination:=false; and L:=$\emptyset$;
2. **while** there is a next d-summary in the stream **do**
   // fetch a new summary (bounded by b_p) from the 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** there is a request of summarization **then**
9. **return** $\mathcal{S}_G$;
10. **return** $\mathcal{S}_G$;

Fig. 2: Algorithm streamDis

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) It incrementally updates the top-k summaries $\mathcal{S}_G$ (lines 7), with top $\lfloor \frac{k}{2} \rfloor$ pairs of summaries with maximum quality $F'(\cdot)$ from the list set $\mathcal{L}$. 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 $\mathcal{S}_G$.

At any time, it returns the current $\mathcal{S}_G$ upon request (lines 8-9). The above process is repeated until no new pattern can be fetched from sumGen.

Example 3: 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. (1) In round 1, 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. (2) 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={<(P_2, P_3),0.43}, L_3= {<(P_3, P_2),0.43}, and $\mathcal{S}_G$ as $\{P_2, P_3\}$. (3) In round 3, it discovers P_1, and verifies $F'(P_1, P_2)$= 0.62 and $F'(P_1, P_3)$=0.61. 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 updates $\mathcal{S}_G$ to $\{P_1, P_2\}$. 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 approximates the optimal answer with approximation ratio 2 over the "seen" summaries at any time. For complexity, it 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 has verified upon interrupted, and $|\mathcal{S}_G|$ refers to the total size of summaries and their base graphs. We present the detailed analysis in [1].

Remarks. "Anytime approximation" is desirable and, nevertheless, "weaker" than an anytime quality guarantee *w.r.t.* to the optimal answer over the entire input [2]. The latter requires the prior knowledge of error distribution. We do not make such assumptions, and defer this study to future work.

IV. Knowledge Graph Search with Summaries

We next show that d-summaries can suggest relevant data and support fast knowledge graph search within bounded resource, by developing such a query evaluation algorithm.

"Summaries+Δ" Scheme. Given a query Q, a knowledge graph G and a summarization $\mathcal{S}_G$, our query evaluation algorithm, denoted as evalSum, 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.

Summary Selection. Given a query Q and a summarization $\mathcal{S}_G$, the summary selection aims 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. To this end, the selection procedure greedily adds the summaries $\mathcal{P}$ that "maximally" covers Q, and have small base graphs in G. 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); (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 process can be done efficiently in $O(\mathsf{card}(\mathcal{S}_G) b_q (b_q + |V_p|)(b_q + |E_p|))$ 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. We present the detailed proof in [1].

V. Experimental Evaluation

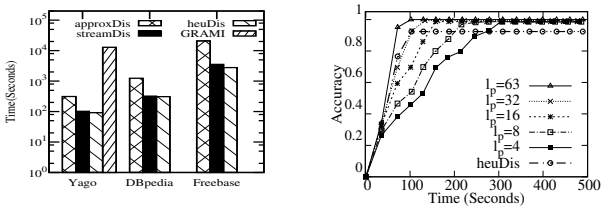
Using real-world and synthetic knowledge graphs, we conducted three sets of experiments to evaluate 1) Performance of the summary mining algorithms approxDis and streamDis; 2) Effectiveness of the algorithm evalSum for query evaluation; and 3) Effectiveness of summaries, using a case study.

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

[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) Real-world datasets (b) streamDis: quality vs. time

Fig. 3: Performance of summarization

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 the following algorithms in Java:

1) Summarization algorithms approxDis and streamDis, compared with: a) heuDis, a counterpart of streamDis that maintains $\mathcal{S}_G$ over pattern streams following [10]. Each time a summary P is seen, 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, that uses an open-source tool [4] 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; 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 [12]. 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[5]. 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".

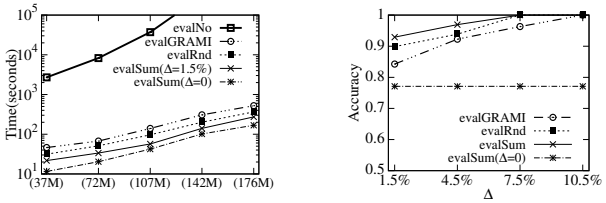
Efficiency of Summarization. We evaluate the efficiency of approxDis, streamDis, heuDis, and GRAMI over the real-world knowledge graphs. For the two anytime algorithms streamDis and heuDis, we report their convergence time. For GRAMI, we carefully adjusted a support threshold to allow the generation of patterns with similar label set and size to those from approxDis. As shown in Fig.3(a), 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.

Using larger synthetic graphs, we evaluated the scalability of streamDis, by varying $|G|$ from $(10M, 27M)$ to $(60M, 152M)$ (not shown, see details in [1]). The algorithms streamDis and heuDis scale better with larger $|G|$ compared with GRAMI due to their speed of convergence. In contrast, GRAMI does not run to completion in 10 hours with graphs of size $(10M, 27M)$.

Anytime performance. We evaluate the "anytime" performance of streamDis and heuDis. We define the *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(b) shows the accuracy of streamDis and heuDis over YAGO *w.r.t.* time t and cache bound l_p. 1) The quality of summaries returned by streamDis increases monotonically as

[5]Source code: https://github.com/songqi1990/KnowGraphSum



(a) evalSum: Varying $|G|$ (b) Accuracy *w.r.t.* Δ

Fig. 4: Efficiency and Accuracy of evalSum

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.

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

Varying $|G|$. We evaluated 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. 4(a): 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.

Accuracy. We evaluated the accuracy of the query answers produced by four evaluation algorithms. 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 4(b), all algorithms perform better with larger Δ, 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.

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. 5, which distinguish

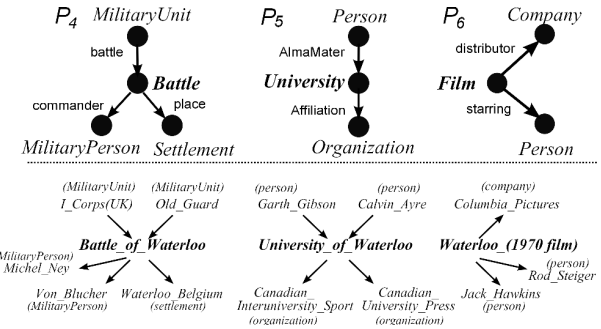


Fig. 5: Real-life Summaries: DBpedia.

"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 [13] that find diversified facts of a single entity.

VI. 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] Full version. http://eecs.wsu.edu/∼qsong/Files/paper/ICDM2016Full.pdf.
[2] B. Arai, G. Das, D. Gunopulos, and N. Koudas. Anytime measures for top-k algorithms. In *VLDB*, pages 914–925, 2007.
[3] 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.
[4] 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.
[5] W. Fan, X. Wang, and Y. Wu. Answering graph pattern queries using views. In *ICDE*, 2014.
[6] W. Fan, X. Wang, and Y. Wu. Querying big graphs within bounded resources. In *SIGMOD*, 2014.
[7] G. Kasneci, F. M. Suchanek, G. Ifrim, M. Ramanath, and G. Weikum. Naga: Searching and ranking knowledge. In *ICDE*, pages 953–962, 2008.
[8] D. Koutra, U. Kang, J. Vreeken, and C. Faloutsos. Vog: Summarizing and understanding large graphs. In *SDM*, 2014.
[9] W. Le, S. Duan, A. Kementsietsidis, F. Li, and M. Wang. Rewriting queries on SPARQL views. In *WWW*, 2011.
[10] E. Minack, W. Siberski, and W. Nejdl. Incremental diversification for very large sets: a streaming-based approach. In *SIGIR*, 2011.
[11] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *SIGMOD*, 2008.
[12] 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.
[13] 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.