Adaptive Personalized Knowledge Graph Summarization

Lukas Faber
Hasso Plattner Institute
lukas.faber@student.hpi.de

Tara Safavi
University of Michigan Ann Arbor
tsafavi@umich.edu

Davide Mottin
Hasso Plattner Institute
davide.mottin@hpi.de

Emmanuel Müller
Hasso Plattner Institute
emmanuel.mueller@hpi.de

Danai Koutra
University of Michigan Ann Arbor
dkoutra@umich.edu

ABSTRACT

Knowledge graphs, which are rich networks of entities and concepts connected via multiple types of relationships, have gained traction as powerful structures for natural language understanding and question answering. Although recent research efforts have started to address efficient querying and storage of knowledge graphs, such methods are neither *user-driven* nor *flexible to changes in the data*, both of which are important in the real world. We thus introduce and motivate **adaptive knowledge graph summarization** to create small, personalized knowledge graphs that contain only the information most relevant to an individual user's interests. Such concise summaries may be stored locally on mobile devices, allowing for fast interactive querying, and constantly updated to serve changing user needs and data evolution. In this position paper, we make a case for adaptive knowledge graph summarization, outlining promising approaches toward efficient, personalized knowledge graph management.

1 INTRODUCTION

As more knowledge is extracted from various sources, *knowledge graphs*, or structures that store entities/concepts and their relationships (Fig. 1), are quickly growing. While such large amounts of data—on the order of millions of entities and billions of relationships—enable tasks like natural language understanding and question answering, *graph search* and processing become more challenging. Accordingly, current research in knowledge graphs focuses on efficient information retrieval via indices, or, more recently, on knowledge graph summarization to construct concise but comprehensive versions of the original KG [3, 21].

However, all existing works assume that knowledge graph (KG) queries are handled on the *server side*, where the graph resides, which introduces a delay that comprises interactivity and requires an online connection. Moreover, typical approaches for managing large amounts of data, such as indices, suffer from constant updates to match the rapidly evolving knowledge graphs.

To this end, we argue for the need of *adaptive and personalized knowledge graph summaries*, small local representations of the knowledge graphs that can be stored client-side on mobile devices, such as smartphones, enabling fast offline and online efficient graph search. Such summaries should incrementally evolve with user interests over time while both satisfying storage constraints (e.g., limited capacity of a mobile phone) and limiting expensive, or potentially impossible, accesses to the server. Additionally, by incorporating incremental updates, adaptive graph summarization

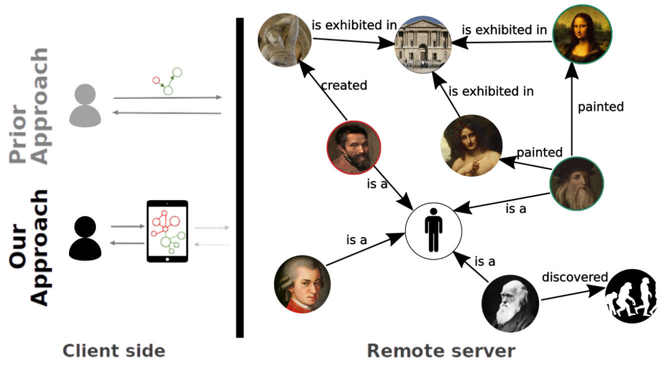

Figure 1: Prior approaches vs. adaptive graph summarization for a user interested in "art". In the latter, if the client-side knowledge graph already contains the answer to the user's query, expensive queries to the server are bypassed.

tackles the problem of consistency in knowledge graphs, for which no previous work exists.

Our approach to the adaptive graph summarization problem is cross-disciplinary, incorporating techniques from databases, information retrieval, and machine learning. In this position paper, we first review related work in the field and discuss how aspects of existing methods from multiple research communities may be used toward adaptive graph summarization. Finally, we highlight promising directions to address three key challenges of adaptive graph summarization:

- **Representation**: What summary representation is both accurate and adherent to user device memory/storage limits?
- **Personalization**: Which regions of the KG should the summary contain to reflect individual user interests and queries?
- **Adaptation**: How can we add, remove, or update data in the summary to meet changing user needs?

2 RELATED WORK

We briefly give a (non-comprehensive) review of related areas of research. For each area, we discuss how its research questions relate to, and differ from, the goals of adaptive graph summarization.

Graph summarization. The goal of graph summarization is to find a compact but meaningful representation of a large input graph with respect to some goal, like pattern discovery [2, 20], ease of visualization [4, 9] or efficient querying for different kinds of graph queries, like structural or pattern queries [3, 14, 18]. In some approaches, additional nodes are added to reduce edge density [14], whereas others aggregate nodes and edges to reduce the overall graph size [3, 22] or store a template pattern with their matches to use as materialized views [21]. An extensive survey of summarization methods is given in [12]. Existing algorithms mostly build "one-size-fits-all" solutions regardless of the user's interests. While recent work has addressed *domain*-specific graph summarization [8], we aim for small summaries at the granularity of the *user* level.

Database caching. Caching has been successfully employed in many applications, such as CPUs, to speed up data access on previously accessed items. Databases also use caching for accelerating query answering, and recently graph databases have adopted similar strategies for graph queries [15, 17, 23]. In graphs, caching algorithms store small graphs with their database matches, which are used as (partial) results for future queries. The problem we study is similar to caching at the surface, but it differs in that we do not build an explicit match between queries and answers. By contrast, we integrate different answers into one coherent graph summary that answers both previously-asked *and* unseen queries.

Database cracking. Adaptively reorganizing relational data according to the workload at hand motivated the field of database cracking. A column is copied on the first access and continuously reorganized as part of query processing [6]. In this setting, finding a good trade-off between overhead induced on queries and system performance is key [7]. Another challenge is designing the reorganization of data so that the "cracked" index can help answering future queries [5]. We see database cracking as complementary to our problem, with two main differences. The first is that we deal with a specialized data structure—a graph—rather than general relational data. The second is that database cracking builds up one large structure on the server side, whereas we aim for many small, user-specific, client-side structures.

3 CHALLENGES AND APPROACHES

We define a knowledge graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a collection of labeled nodes $\mathcal{V}$ and labeled, directed multi-edges $\mathcal{E}$. The nodes represent entities or concepts (e.g., *Mona Lisa* in the knowledge graph of Fig. 1), and the edges represent relationships between nodes (e.g., the *Mona Lisa* is "exhibited in" the Louvre). We assume that a user asks a sequence of *queries* based on her *changing* interests, as in the scenario below, which need to be handled appropriately by the summarization process.

Scenario. Consider a tourist planning her visit at the Louvre museum and deciding on which art pieces to see. Her quest can be formulated as the pattern query in Fig. 2 for the knowledge graph, which asks for every entity "exhibited in" the Louvre museum. The answers in this example include the paintings *Mona Lisa* or *Bacchus* or the sculpture *Dying Slave*. During her visit in Athens a few days later, her searches change to learn about archaeological sites there.

Adaptive summarization of knowledge graphs requires addressing the three key challenges of *representation*, *personalization*, and *adaption*. In this section, we discuss potential approaches and highlight relevant works for each challenge.

3.1 Summary representation

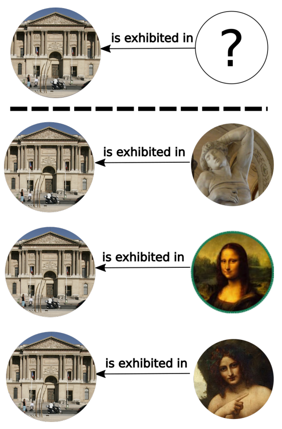

Figure 2: Example query (top) with three answers (bottom three).

While mobile devices are becoming more powerful, their capabilities are still limited compared to traditional computing resources. Because we aim to store data on such mobile devices (i.e., on the client side), we require "memory-aware" summaries that stay strictly below user device storage limits. To closely model these limits, we enforce the summary encoding to stay below a byte constraint. A promising approach to this end is the "supergraph" approach [3, 10, 14, 19, 21] in which nodes and edges are grouped into "supernodes" and "superedges" (Fig. 3). This approach lowers overall storage costs, albeit by introducing some modeling error.

For example, Čebirić et al. [3] group nodes of the same equivalence class—i.e., nodes with shared ingoing and/or outgoing edge types—into supernodes. Song et al. [21] construct *d*-summaries, or generalized graph patterns with concrete matches in the knowledge graph, in which supernodes comprise nodes with identically typed neighborhoods reachable via the same edge labels. In our case, nodes should be considered similar if they share many outgoing edges and are connected to similar nodes. Figure 3 shows an example summary of the knowledge graph in Fig. 1. It contains two supernodes: the two paintings *Mona Lisa* and *Bacchus* form one supernode because they were both painted by Leonardo da Vinci, and the two persons "Mozart" and "Darwin" form the other. Superedges then comprise all pairs of edges between the nodes they contain. The modeling error is thus the surplus edges in the summary that are not present in the original graph.

3.2 Summary personalization

Unlike existing graph summarization techniques, our goal is to create many small graph summaries, each personalized to a different user. The challenge here is identifying the KG region(s) of interest for each user. A simple yet promising approach to this end comes from relational main memory databases: Levandoski et al. [11] identify a small number of tuples to be retained in main memory, with the rest on disk. The tuples in main memory are called "hot" storage and the tuples on disk are "cold" storage, with the "temperature" of a tuple proportional to its frequency of access.

A key limitation of this approach, though, is that it handles every tuple individually. Our aim is to answer queries in a graph format, so we must preserve relationship (edge) information as well. A node's heat must thus incorporate its own access frequency *and* the access frequency of its neighboring nodes. An intuitive way to model this is with heat diffusion processes [13] such that queried nodes gain heat and diffuse that heat to their neighborhood. After

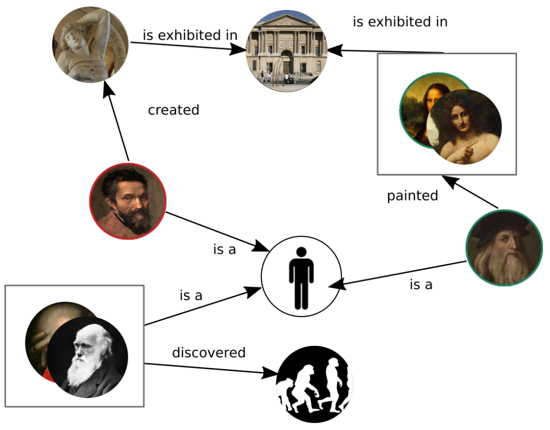

Figure 3: Certain parts of the KG from Fig. 1 are grouped together to form superedges and supernodes.

a query, each of which directly "touches" some group of nodes, the heat of each touched node and its neighborhood increases, whereas untouched nodes lose heat proportional to a decay factor.

3.3 Summary adaptation

Truly personalized summaries must adapt according to *evolving* user interests and needs. Similar to cache eviction [1, 16], we must replace less-useful data with more relevant information, and retain information that will likely prove useful to the user in the future.

Again, the challenge here is that we must not treat nodes individually, but together with their neighborhood. We suggest to optimize the summary around evolving node temperature. Since warmer nodes are considered more immediately "interesting" to the user, the modeling error around these nodes should be minimal. In other words, the supernodes and superedges encompassing these nodes and their adjacent relationships must be more fine-grained, with fewer erroneous edges. Conversely, we accept higher modeling error around cooler nodes because the user is less likely to access them. Temperature is an immediate model for the user's interest: If the interest in some topic decreases, previously interesting nodes will cool down over time and become cold (i.e., uninteresting). At the same time new nodes will become "hot" By continually minimizing modeling error so that there are fewer surplus edges around "hot" nodes, the summary can be adaptively restructured to follow changing user interests.

Example. For the knowledge graph from Fig. 1 with the query from Fig. 2 we could construct the summary in Fig 3. It contains two supernodes. The supernode on the bottom-left contains only cold entities. Hence the introduced errors, such as "Mozart discovered the theory of evolution", hardly matter as the user will probably not query that part. On the other hand, the supernode on the top-right contains hot nodes. We can still create such supernode because the two nodes, Mona Lisa and Baccus, have identical neighborhoods, thus no modeling error is introduced. On the contrary, grouping the nodes Michelangelo and Leonardo da Vinci would introduce errors. As these nodes are warm, since they are closely connected to paintings in the Louvre, no supernode is created.

4 CONCLUSION

In this paper we motivate the need for adaptive graph summarization. Our position is that, for massive graph datasets and in particular knowledge graphs that inherently serve different people in different ways, "one-size-fits-all" graph summaries are less useful to users than adaptive, personalized summaries. We believe that the next frontier of graph summarization lies in personalized and incrementally adapting techniques. Future work in this direction will be a step toward efficient information access and knowledge discovery for all users, regardless of location or online connectivity.

5 ACKNOWLEDGEMENTS

This work was supported by a fellowship within the FITweltweit programme of the German Academic Exchange Service (DAAD), and the University of Michigan.

REFERENCES

[1] Sorav Bansal and Dharmendra S Modha. 2004. CAR: Clock with Adaptive Replacement.. In *FAST*, Vol. 4. 187–200.
[2] Gregory Buehrer and Kumar Chellapilla. 2008. A Scalable Pattern Mining Approach to Web Graph Compression With Communities. ACM, 95–106.
[3] Šejla Čebirić, François Goasdoué, and Ioana Manolescu. 2015. Query-oriented summarization of RDF graphs. *PVLDB* 8, 12 (2015), 2012–2015.
[4] Cody Dunne and Ben Shneiderman. 2013. Motif Simplification: Improving Network Visualization Readability with Fan, Connector, and Clique Glyphs. In *SIGCHI*. ACM, 3247–3256.
[5] Felix Halim, Stratos Idreos, Panagiotis Karras, and Roland HC Yap. 2012. Stochastic database cracking: Towards robust adaptive indexing in main-memory column-stores. *PVLDB* 5, 6 (2012), 502–513.
[6] Stratos Idreos, Martin L Kersten, Stefan Manegold, et al. 2007. Database Cracking.. In *CIDR*, Vol. 7. 68–78.
[7] Stratos Idreos, Stefan Manegold, Harumi Kuno, and Goetz Graefe. 2011. Merging what's cracked, cracking what's merged: adaptive indexing in main-memory column-stores. *PVLDB* 4, 9 (2011), 586–597.
[8] Di Jin and Danai Koutra. 2017. Exploratory Analysis of Graph Data by Leveraging Domain Knowledge. In *ICDM*. 187–196.
[9] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. [n. d.]. VoG: Summarizing and Understanding Large Graphs. SIAM.
[10] Kristen LeFevre and Evimaria Terzi. 2010. GraSS: Graph structure summarization. In *SDM*. 454–465.
[11] Justin J Levandoski, Per-Åke Larson, and Radu Stoica. 2013. Identifying hot and cold data in main-memory databases. In *ICDE*. 26–37.
[12] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph Summarization Methods and Applications: A Survey. *ACM Comput. Surv.* 51, 3, Article 62 (June 2018), 34 pages.
[13] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. 2008. Mining social networks using heat diffusion processes for marketing candidates selection. In *CIKM*. 233–242.
[14] Antonio Maccioni and Daniel J Abadi. 2016. Scalable pattern matching over compressed graphs via dedensification. In *KDD*. 1755–1764.
[15] Michael Martin, Jörg Unbehauen, and Sören Auer. 2010. Improving the performance of semantic web applications with SPARQL query caching. In *ESWC*. 304–318.
[16] Nimrod Megiddo and Dharmendra S Modha. 2003. ARC: A Self-Tuning, Low Overhead Replacement Cache.. In *FAST*, Vol. 3. 115–130.
[17] Nikolaos Papailiou, Dimitrios Tsoumakos, Panagiotis Karras, and Nectarios Koziris. 2015. Graph-aware, workload-adaptive sparql query caching. In *KDD*. 1777–1792.
[18] Matteo Riondato, David García-Soriano, and Francesco Bonchi. 2014. Graph Summarization With Quality Guarantees. IEEE, 947–952.
[19] Matteo Riondato, David García-Soriano, and Francesco Bonchi. 2017. Graph summarization with quality guarantees. *Data Min. Knowl. Discov.* 31, 2 (2017), 314–349.
[20] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2015. TimeCrunch: Interpretable Dynamic Graph Summarization.
[21] Qi Song, Yinghui Wu, and Xin Luna Dong. 2016. Mining summaries for knowledge graph search. In *ICDM*. 1215–1220.
[22] Nan Tang, Qing Chen, and Prasenjit Mitra. 2016. Graph stream summarization: From big bang to big crunch. In *SIGMOD*. 1481–1496.
[23] Jing Wang, Nikos Ntarmos, and Peter Triantafillou. 2017. GraphCache: a caching system for graph queries. *EDBT* (2017).