Personalized Knowledge Graph Summarization: From the Cloud to Your Pocket

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

Caleb Belth
University of Michigan
Ann Arbor, MI, USA
cbelth@umich.edu

Lukas Faber
Google
Zurich, Switzerland
lukasjfaber@gmail.com

Davide Mottin
Aarhus University
Aarhus, Denmark
davide@cs.au.dk

Emmanuel Müller
B-IT Center
Bonn, Germany
office-mueller@bit.uni-bonn.de

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

***Abstract*—The increasing scale of encyclopedic knowledge graphs (KGs) calls for summarization as a way to help users efficiently access and distill world knowledge. Motivated by the disparity between individuals' limited information needs and the massive scale of KGs, in this paper we propose a new problem called *personalized knowledge graph summarization*. The goal is to construct compact "personal summaries" of KGs containing only the facts most relevant to individuals' interests. Such summaries can be stored and utilized on-device, allowing individuals private, anytime access to the information that interests them most.**

We formalize the problem as one of constructing a sparse graph, or summary, that maximizes a user's inferred "utility" over a given KG, subject to a user- and device-specific constraint on the summary's size. To solve it, we propose GLIMPSE, a summarization framework that provides theoretical guarantees on the summary's utility and is linear in the number of edges in the KG. In an evaluation with real user queries to open-source, encyclopedic KGs of up to one billion triples, we show that GLIMPSE efficiently creates summaries that outperform strong baselines by up to 19% in query answering F1 score.

I. Introduction

Encyclopedic knowledge graphs, which store facts about the world by connecting entities via semantically meaningful relations, have shown to be useful tools for AI tasks including question answering, item recommendation, query expansion, language modeling, and more [38], [40], [37], [9]. Modern knowledge graphs (**KG**s) contain up to billions of entities and relationships, and are continually being augmented with new facts [26]. These increasingly large stores of world knowledge necessitate *summarization*, which reduces large KGs to more concise but still interpretable and query-able forms [22].

This paper proposes the new task of **personalized knowledge graph summarization**, the goal of which is to find a sparse *summary* of a large KG—in essence, a "mini-KG"—containing the facts most relevant to each individual user's interests and queries. We motivate this task by studies in information retrieval and human-computer interaction showing that individuals have limited information capacity [34], [17]. In a KG setting, then, most individuals' information needs will likely cover only small portions of a given large KG [27]. We thus envision compact **personal summaries** containing user-specific facts of interest being stored and accessed on devices like smartphones, intelligent assistants, and in other scenarios where resources are constrained (e.g., network bandwidth, device disk space), an Internet connection is not available, or the user desires privacy in querying. Figure 1 shows an example for a user interested in books and authors. Her personal summary allows for anytime user information access while still supporting KG-powered tasks with high accuracy.

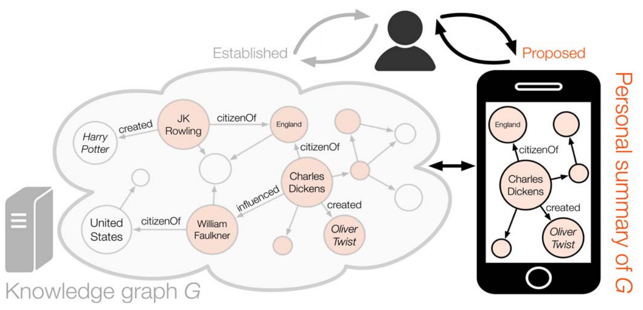


Fig. 1: Personalized KG summarization for a user interested in books and authors. Given seed information about the user's interests over G, GLIMPSE constructs an on-device personal summary of G (i.e., a mini-KG) for anytime information access.

Present work. Our proposed approach to personalized KG summarization, **GLIMPSE** or Graph-based Learning of Personal Summaries, consists of first inferring user preferences over a given KG, then constructing the user's personal summary from these preferences. For the first step, we assume the user demonstrates her interests via her queries to the KG, and use these queries as seeds from which we infer other entities and relations of potential interest to the user. We formalize the second step as an optimization problem in which we maximize the summary's inferred utility to the user, subject to a size constraint corresponding roughly to device resources like disk space. GLIMPSE relies on fast submodular maximization, which to the best of our knowledge has not been used for graph summarization before, to efficiently summarize KGs.

2374-8486/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDM.2019.00063

This paper makes the following contributions:

- *Problem*: We introduce, motivate, and mathematically formulate the problem of personalized KG summarization.
- *Framework*: We propose GLIMPSE, a flexible summarization framework that combines strong theoretical guarantees with the scalability necessary for large KGs.
- *Evaluation*: We analyze GLIMPSE in a direct query answering task using real queries to KGs of up to one billion triples. GLIMPSE personal summaries outperform summaries created by strong baselines by up to 19% in query answering F1 score across various simulated user models. We demonstrate GLIMPSE's consistency across datasets, and provide in-depth analysis of our results.

II. Related work

Graph summarization and sampling. Graph summarization techniques abstract large graphs into smaller ones according to objectives like query efficiency, ease of visualization, and pattern discovery [22], [30], [18], [16]. That said, existing techniques for labeled graphs tend to follow a "one-size-fits-all" approach [4], [32]. Such techniques construct summaries by compressing *all* nodes into "supernodes", with "superedges" connecting pairs of nodes across supernodes. By contrast, in our setting we assume that most facts in a KG are irrelevant to a single user, so a personal summary need not contain all of the KG. Moreover, grouping-based techniques usually compare all nodes pairwise, leading to quadratic-time algorithms that are too slow for our problem setting.

Some graph summarization techniques create summaries preserving specific classes or types of subgraph queries (e.g., path or star queries), though again toward summarizing the entire graph [6], [23]. A few recent approaches also consider user- or task-specific input [1], [19], [15]. That said, no existing work handles the actual *content* of user queries. This calls for new summarization approaches tailored to personalization, as a personal summary cannot be the same for individuals with different interests (as expressed by their past queries).

Because our proposed method selects a subset of edges from G as the summary, graph sampling is also relevant. That said, the goal of graph sampling is not to answer queries tailored to user interests, but rather to preserve graph-specific properties (centralities, motifs) in smaller, representative samples [20].

Knowledge graphs. The KG tasks most related to the present work are fact contextualization and entity ranking or summarization [5], [10], [13], [35], [9]. In these works, a single entity or fact in the knowledge graph is given as input. The output is a set of entities, entity attributes or features, and/or facts that are deemed the most "informative" or "relevant" to the input according to various heuristic criteria. Such methods can be seen as complementary to our work. They usually rely on either feature extraction or PageRank variants, and require human assessment of solution quality and/or manually labeled training data, which can be expensive to obtain. By contrast, we evaluate our method with the accuracy of query answering on the summary. Importantly, we also handle arbitrary query structures, rather than single entities or facts, as input.

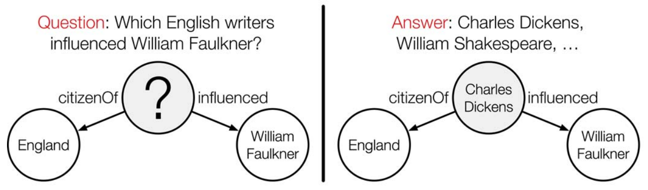

Question: Which English writers influenced William Faulkner?
citizenOf
?
influenced
England
William Faulkner
Answer: Charles Dickens, William Shakespeare, ...
citizenOf
Charles Dickens
influenced
England
William Faulkner

Fig. 2: Example of a query to the `YAGO` knowledge graph (§ IV-A) and one answer, with corresponding natural language and query graph representations.

Personalization. Identifying items of interest to users (web pages, e-commerce products, movies, etc) is a major goal in search and recommender systems. While it is common for recommender systems to use auxiliary information extracted from knowledge graphs toward better recommendations [40], we are not aware of any work that identifies individuals' facts of interest—in other words, the facts themselves are being recommended—in a KG. Furthermore, while there exists a vast literature on personalized web search, such works tend to focus on extracting or learning features of interest at the document or user level [36], [2]. By contrast, we are mainly interested in summary *construction*, so we leave the incorporation of more complex user preference modeling to future work.

That said, many personalization approaches do rely on graph-based methods. The most common approach, personalized PageRank (PPR) [14], returns a ranking of entities in order of their "importance" or "relevance" to a given query. We investigate this approach in our experiments.

III. Methodology

We begin this section with preliminaries. We then outline how we infer user preferences and construct a personal summary from these preferences. Finally, we theoretically analyze our approach. For reference, Table I gives our main symbols.

A. Preliminaries

A **knowledge graph** $G = (E, R, T)$ consists of a set of entities E, a set of relations R, and a set of triples $T \subseteq E \times R \times E$[1]. A triple connecting entities $e_i, e_j \in E$ with relation $r_k \in R$ is denoted $x_{ijk} = (e_i, r_k, e_j)$. From a natural language perspective, triples in a knowledge graph are $\langle$*subject*, *predicate*, *object*$\rangle$ facts. From a graph-theoretical perspective, triples are labeled edges connecting pairs of entity nodes. In this work we assume each question to G is given in **query graph** form (i.e., via semantic parsing [38]), as shown in Figure 2. Each query graph $G_Q = (E_Q, R_Q, T_Q)$, which may be a subgraph of G or may contain elements not in G, is directed, acyclic, and fully connected. One or more entities in G_Q are *variable(s)*, as shown by the **(?)** node in Figure 2. Each variable represents any entity in E. The entities $e \in E$ that replace variables during query answering by matching the

[1]Although KGs are in general incomplete, in the *summary construction* step we assume a closed world (i.e., we add only existing triples to the summary).

TABLE I: Table of main symbols.

Symbol	Meaning
G	Knowledge graph $G = (E, R, T)$ with entity set E, relation set R, and triple set T
e_i	i-th entity in entity set E
r_k	k-th relation in relation set R
x_{ijk}	Triple $(e_i, r_k, e_j) \in T$ with entities $e_i, e_j \in E$, relation r_k
G_Q	Query graph $G_Q = (E_Q, R_Q, T_Q)$ to G
Q_u	Query log $Q_u = (G_Q^1, \ldots, G_Q^n)$ of user u on G
S_u	Personal summary $S_u = (E_u, R_u, T_u) \subseteq G$ of user u
K	Number of triples in personal summary S_u

given query graph pattern are called the *answers* to the query G_Q. Finally, each user u is associated with a sequence of queries, or **query log** $Q_u = (G_Q^1, \ldots, G_Q^n)$ to G.

Queries to G are answered via subgraph isomorphism, which means we find a one-to-one mapping ψ between queried entities E_Q and knowledge graph entities E, if one exists, such that for all triples $(e_i, r_k, e_j) \in T_Q$, there exists a $(\psi(e_i), r_k, \psi(e_j)) \in T$. While subgraph isomorphism is NP-complete in general, most real queries to KGs are small (one or two triples [3]), which makes the isomorphism feasible to compute in practice. That said, another benefit of summarization is faster query answering, since a summary will contain fewer entities/triples than the KG it summarizes.

B. Informal problem statement

Informally, the problem we address may be described as: **Given a knowledge graph G, a user u's past queries to G, and a user-specific resource (device or application) constraint, efficiently infer a personal summary $S_u \subseteq G$ under the given constraint that best captures the user's preferred facts in G, as expressed by her past queries.** In the following sections, we will formalize this problem by defining a notion of a resource-constrained personal summary drawn from a general model of user preferences[2].

C. Inferring user preferences

GLIMPSE consists of two steps. First, we infer entities and relations of potential interest to the user based on her historical queries Q_u. We then construct a summary by maximizing a user-specific utility function drawn from these inferred preferences. In this section, we address the first step.

We model user preferences over G by associating each entity and triple with a probability of user u preferring them, conditioned on u's query history Q_u. We capture preferences for entities and triples separately because of the KG-specific differences in meaning between the two. A user's preference for a single entity e indicates interest in a related group of facts (i.e., any triple containing e or its neighbors), which loosely corresponds to a "topic"[3]. By contrast, a user's preference for a triple x_{ijk} expresses an interest in a single unit of information.

Importantly, note that GLIMPSE is flexible enough to incorporate arbitrary modeling complexities or even explicit user feedback (c.f. [9]). In this paper we focus on an unsupervised, graph-structural user preference model that is highly efficient to compute, but any approach that yields per-user probabilities for both entities and triples in G can be used in its stead.

Entity preference. Let $\Pr(e_i|Q_u)$ be the user's preference for entity $e_i \in E$. Since users often re-seek information [34], we capture the user's historical preference for e_i in $\Pr(e_i|Q_u)$. We also account for the local graph structure around e_i: Since queries come in the form of connected graphs (§ III-A), answers to future queries involving e_i must involve entities e_j directly connected to e_i (i.e., neighbors of e_i). More generally, an interest in a single entity (e.g., *Charles Dickens*) may signal interest in connected entities in the KG (e.g., *Oliver Twist*, *Great Expectations*, *England*, etc).

Denoting the set of all neighbors of e_i in G as $N(e_i) = \{e_j | (e_i, r_k, e_j) \in T\}$, we capture the user's preference with

$$\Pr(e_i|Q_u) \propto \underbrace{\sum_{G_Q \in Q_u} \mathbb{1}_{E_Q}(e_i)}_{\text{historical pref.}} + \underbrace{\gamma \sum_{e_j \in N(e_i)} \mathbb{1}_{E_Q}(e_j)}_{\text{graph structure}}, \quad (1)$$

where $\gamma \in [0, 1]$ controls the influence of neighbors and $\mathbb{1}_X(x)$ is the indicator function, equal to 1 iff $x \in X$ and 0 otherwise. We can generalize (1) to all entities in the KG: Let $\mathbf{q} \in \mathbb{R}^{|E|}$ be the user's seed query vector with $q_i = 1$ if the user queried the i-th entity in E and 0 otherwise, and $\mathbf{M} = \gamma \mathbf{A}\mathbf{D}^{-1} \in \mathbb{R}^{|E| \times |E|}$ be the normalized adjacency matrix of the KG. Then (1) is equivalent to the first two terms of the random walk power series expansion

$$\mathbf{u} = \mathbf{q} + \mathbf{M}\mathbf{q} + \mathbf{M}(\mathbf{M}\mathbf{q}) + \mathbf{M}(\mathbf{M}(\mathbf{M}\mathbf{q})) + \ldots, \quad (2)$$

where $\mathbf{u} \in \mathbb{R}^{|E|}$ is the vector that contains user u's preference for all $|E|$ entities. Note that (2) can be computed efficiently, in time linear in the number of edges in the graph, via sparse matrix-vector multiplications.

Extending (1) and (2) to neighbors of neighbors spreads user preference from queried entities across paths in the KG, which captures compositional facts centered around topic entities. For example, users interested in *Charles Dickens* may also be interested in the *United Kingdom* more generally. Spreading preference across paths in the KG allows us to capture this interest, since the entity *Charles Dickens* is connected to the entity *United Kingdom* by the two-step path (*Charles Dickens*, *citizenOf*, *England*) and (*England*, *constituentOf*, *United Kingdom*). We investigate such paths further in § V.

Triple preference. To capture the user's preference for triple or fact $x_{ijk} = (e_i, r_k, e_j) \in T$, we incorporate its entities and relation. We follow the standard conditional independence assumption in graph mining and KG learning (c.f. [8], [26]):

$$\Pr(x_{ijk}|Q_u) \propto \Pr(e_i|Q_u)\Pr(r_k|Q_u)\Pr(e_j|Q_u), \quad (3)$$

We compute $\Pr(r_k|Q_u)$ as the proportion of queries in query log Q_u containing relation r_k. As mentioned earlier, both

[2]For privacy reasons, we model each user's preferences individually, and leave privacy-preserving collaborative approaches for future directions.

[3]For instance, the Freebase KG documentation explicitly refers to each entity as a topic: https://developers.google.com/freebase/guide/basic_concepts

$\Pr(e_i|Q_u)$ and $\Pr(r_k|Q_u)$ can modeled with more complex relevance features, either extracted or learned. However, as discussed in the next section, our focus is summary construction, so we leave more complex user modeling for future work.

D. Constructing the summary

The second step of our approach is to construct the personal summary. Given the user preference model described in the previous section, let $\Pr(S_u|Q_u)$ be our estimate of how well a constructed summary $S_u = (E_u, R_u, T_u)$ captures the user's inferred preferences, conditioned on Q_u:

$$\Pr(S_u|Q_u) \propto \prod_{e \in E_u} \underbrace{\Pr(e|Q_u)}_{\text{“topic” pref.}} \prod_{x_{ijk} \in T_u} \underbrace{\Pr(x_{ijk}|Q_u)}_{\text{fact pref.}}. \quad (4)$$

Using the above as an objective function, we formalize the optimization problem corresponding to summary construction:

Problem 1 (Personalized KG summarization)**.** *Given (**1**) a knowledge graph G, (**2**) a user u and her query history Q_u to G, and (**3**) a number of triples K, find the personal summary $S_u = (E_u, R_u, T_u) \subseteq G$ of K triples that maximizes the log-likelihood of* $\Pr(S_u|Q_u)$*:*

$$\arg\max_{S_u \subseteq G} \log \Pr(S_u|Q_u) \ \textit{s.t.}\ |T_u| \leq K. \quad (5)$$

The constraint K roughly corresponds to a resource constraint like device disk space or latency requirements. For example, say an offline KG-powered application has 10MB of available storage on a user's mobile device. Given a KG of 10 million triples that requires 1GB of storage, a value of $K \approx 100\,000$ might be appropriate. Overall, K is expected to be relatively small, as most users' information needs are limited, and especially so on the mobile devices or applications where we imagine personal summaries being used [34], [17], [27]. We discuss K in more depth in § V.

A utility perspective. Exactly optimizing (5) would be computationally infeasible, as solving it would require enumerating all size-k subsets of the KG's triple set T, leading to a complexity of $O\binom{|T|}{K}$. That said, instead of immediately resorting to heuristics we reformulate (5) to lead to a tractable, approximation with theoretical guarantees. The key intuition is to restate the likelihood maximization problem in (5) as a *utility maximization* problem, where the utility function to be maximized is nonnegative. We exploit this nonnegativity to show that our utility function is submodular (discussed in the next section), which allows us to devise a near-optimal approximation algorithm.

Define $\phi : (S_u; Q_u) \to \mathbb{R}^+$ as a function that captures the **“utility” of personal summary S_u over a *non-personalized*** S_α that includes every entity and triple in the summary with a constant, small uniform probability $0 < \alpha \ll 1$:

$$\begin{aligned}\phi(S_u; Q_u) &= \log \Pr(S_u|Q_u) - \log \Pr(S_\alpha) \\ &= \sum_{e \in E_u} \log \frac{\Pr(e|Q_u)}{\alpha} + \sum_{x_{ijk} \in T_u} \log \frac{\Pr(x_{ijk}|Q_u)}{\alpha}, \quad (6)\end{aligned}$$

where $P(\cdot|Q_u) \geq \alpha > 0$; in other words, α can be seen as the smallest non-zero probability $\alpha = P(\cdot|Q_u)$. Note that the summations above only include entities and triples with $P(\cdot|Q_u) > 0$, since those with $P(\cdot|Q_u) = 0$ bring no extra improvement to S_u.

Given ϕ, we restate (5) as a utility maximization problem:

$$\arg\max_{S_u \subseteq G} \phi(S_u; Q_u) \text{ s.t. } |T_u| \leq K. \quad (7)$$

A near-optimal approximation. While (7) is close to (5), we can show that the objective function in the former is submodular, which will allow us to near-optimally approximate the solution to (7). Intuitively, submodularity is a “diminishing returns” property of set functions. Formally, for set X and subsets $A \subseteq B \subseteq X$ and element $x \in X \backslash B$, let $\Delta_F(x|A) = F(A \cup \{x\}) - F(A)$ be the marginal utility gained in F by adding x to A. The function F is submodular if $\Delta_F(x|A) \geq \Delta_F(x|B)$ everywhere, i.e., the marginal gain of adding x to the result set diminishes as the result set grows. Importantly, a greedy algorithm that chooses the item with the highest marginal gain in iterations yields a solution that is guaranteed to be within a $\geq (1 - \frac{1}{e})$ fraction away from the (unknown) optimal solution's value when maximizing a nonnegative monotone submodular function under cardinality constraints [25]. The practical implication can be seen as similar to convexity in continuous optimization problems: Under certain conditions, submodularity admits optimization algorithms that yield solutions with theoretically guaranteed bounds on optimality, rather than arbitrarily bad solutions.

The key to our solution is to view $(E \cup R)$ as our “ground set” of elements, and consider each triple $x_{ijk} = (e_i, r_k, e_j) \in T$ as a three-element “subset” of this ground set. From this perspective, we can prove that the utility function ϕ is submodular over the triples T of G. The important implication is that continually **choosing the triple with the highest marginal utility** $\Delta_\phi(x_{ijk}|S_u, Q_u)$, up to K triples, near-optimally solves (7):

Theorem 1. *Equation* (7) *has a* $(1 - \frac{1}{e})$*-approximation.*

Proof. Let $S_u^{(1)} \subseteq S_u^{(2)} \subseteq G$ be two personal summaries of a knowledge graph G, and let triple $x_{ijk} = (e_i, r_k, e_j) \in G \backslash S_u^{(2)}$. Consider that (**1**) if either (or both) entities e_i or e_j are contained in $S_u^{(1)}$, by necessity those entities must also be contained in $S_u^{(2)}$, since $S_u^{(1)} \subseteq S_u^{(2)}$. Conversely, however, (**2**) if either (or both) e_i or e_j are *not* contained in $S_u^{(1)}$, they may still be contained in $S_u^{(2)}$ for the same reason. Therefore, in case (**1**), when adding the triple x_{ijk} to the smaller $S_u^{(1)}$, any entity e *not* in $S_u^{(2)}$ will result in a marginal gain of at least $\Delta_\phi(x_{ijk}|S_u^{(1)}, Q_u) = \log \frac{\Pr(e|Q_u)}{\alpha}$ *more* than the corresponding gain to $S_u^{(2)}$, since e is already contained in $S_u^{(2)}$. In case (**2**), where the entities contained within $S_u^{(1)}$ and $S_u^{(2)}$ are the same with respect to x_{ijk}, the marginal gains $\Delta_\phi(x_{ijk}|S_u^{(1)}, Q_u) = \Delta_\phi(x_{ijk}|S_u^{(2)}, Q_u)$. Therefore, the function ϕ is submodular over the triples x_{ijk} of the knowledge graph G, which means that optimizing (7)

Algorithm 1 The GLIMPSE framework. All **OPT** comments refer to the optimizations discussed in § III-E.

```
   Input: Knowledge graph $G$, user query log $Q_u$, # triples $K$
   Output: Personal summary $S_u \subseteq G$ with $|T_u| \leq K$
1: Compute $T^{\Delta\neq 0}$ with $\Pr(e|Q_u)$, $\Pr(x_{ijk}|Q_u)$   ▷ § III-C, OPT1
2: $S_u \leftarrow \emptyset$
3: while $|T_u| \leq K$ do
4:     Sample set $A$ of size $\frac{|T^{\Delta\neq 0}|}{K}\log\frac{1}{\epsilon}$ from $T^{\Delta\neq 0}$   ▷ OPT2
5:     Select $\tilde{x}_{ijk} \leftarrow \arg\max_{x_{ijk}\in A} \Delta_\phi(x_{ijk}|\, S_u; Q_u)$   ▷ OPT3
6:     Add triple $\tilde{x}_{ijk} = (e_i, r_k, e_j)$ to $S_u$
7: return personal summary $S_u$
```

by continually selecting the triple with the highest marginal utility $\Delta_\phi(S_u; Q_u)$ results in a $(1 - \frac{1}{e})$ approximation of the unknown optimal solution. ☐

E. The GLIMPSE framework

The naive greedy algorithm discussed in the previous section updates up to $|T|$ marginal utilities of triples, and there are up to K iterations, leading to $O(K|T|)$ complexity. This is too slow for KGs with $|T|$ on the order of millions or billions.

Fast summary construction. We exploit special properties of personalization and submodularity to construct a personal summary in time linear in the number of triples $|T|$ in G, *while retaining near-optimal approximation guarantees*. Our first optimization, **OPT1**, is specific to the domain of personalization and relies on the intuition that most triples in the original KG will be of no interest or relevance to a single given user. Let $T^{\Delta\neq 0}$ be the set of triples with nonzero marginal utility for any given knowledge graph G and corresponding S_u:

$$T^{\Delta\neq 0} \triangleq \{x_{ijk} \in G \text{ s.t. } \Delta_\phi(x_{ijk}|S_u; Q_u) \neq 0\}. \quad (8)$$

Now, we only need to update the marginal utilities of triples in $T^{\Delta\neq 0}$. This is because ϕ is submodular, so the triples that start with zero marginal utility can never increase in value as S_u grows. In the experiments (§ V-C) we show that this optimization alone leads to a speedup of over *thirteen thousand times* compared to a non-optimized version of GLIMPSE.

For **OPT2**, we extend [24]: Given a ground set of n triples, we sample a set A of size $\frac{n}{K}\log\frac{1}{\epsilon}$ per iteration, $0 < \epsilon \ll 1$, and update the marginal utilities for only the sampled triples, then pick the highest-valued triple out of that sample. We combine this with "lazy selection" (**OPT3**): In each iteration, we take the top-valued item from the previous iteration and recompute its marginal utility. If it remains top-ranked, we do not need to recompute the marginal utility of other items, since by submodularity those values cannot increase.

GLIMPSE overview and analysis. We outline the GLIMPSE framework given in Algorithm 1. First, we compute the set of nonzero-marginal utility triples $T^{\Delta\neq 0}$ from all $\Pr(e|Q_u)$ and $\Pr(x_{ijk}|Q_u)$ (§ III-C and OPT1). Then, in each iteration we sample a set A of size $\frac{|T^{\Delta\neq 0}|}{K}\log\frac{1}{\epsilon}$ from the precomputed set $T^{\Delta\neq 0}$ (OPT2), and add the triple x_{ijk} of maximal marginal utility from A to personal summary S_u, using the lazy greedy approach to select the highest-valued triple if possible (OPT3). We continue until $|T_u| = K$.

Theorem 2. *GLIMPSE is $O(|T|)$.*

Proof. The user preference inference step is linear in $|T|$ using sparse matrix-vector multiplication (Eq. (2)). Then, for the summary construction step, GLIMPSE consists of K iterations, each of which updates $\frac{|T^{\Delta\neq 0}|}{K}\log\frac{1}{\epsilon}$ marginal utilities of sampled triples $x_{ijk} \in A$. Therefore, its runtime complexity is $O(|T| + K\frac{|T^{\Delta\neq 0}|}{K}\log\frac{1}{\epsilon}) = O(|T| + \log\frac{1}{\epsilon}|T^{\Delta\neq 0}|) = O(|T|)$, since $|T^{\Delta\neq 0}| \ll |T|$ and $\log\frac{1}{\epsilon}$ is a constant. ☐

Theorem 3. *GLIMPSE constructs a summary that is a $\big(1 - \frac{1}{e^{(1-\epsilon)}}\big)$-approximation to the (unknown) optimal personal summary S_u^* for $0 < \epsilon \ll 1$, in expectation.*

Proof. As shown in [24], the expected marginal gain of OPT2 for a single triple x_{ijk} is at least

$$\mathbb{E}[\Delta_\phi(x_{ijk}|S_u; Q_u)] = \frac{1-\epsilon}{K} \sum_{x_{ijk}\in S_u^*\setminus S_u} \Delta_\phi(x_{ijk}|S_u; Q_u), \quad (9)$$

where S_u^* is the (unknown) summary that optimally solves (7). Now, a fact of submodularity, which was proven for ϕ in Theorem 1, is that $\sum_{x_{ijk}\in S_u^*\setminus S_u} \Delta_\phi(x_{ijk}|S_u; Q_u) \geq \phi(S_u^*; Q_u) - \phi(S_u; Q_u)$, because the sum of individual marginal utilities for each triple must be greater than the total value of those triples grouped as a set, due to diminishing returns. By consequence, combining this fact with the result of (9),

$$\mathbb{E}[\Delta_\phi(x_{ijk}|S_u; Q_u)] \geq \frac{1-\epsilon}{K}\big[\phi(S_u^*; Q_u) - \phi(S_u; Q_u)\big].$$

Using the above, it can be shown by induction on K that

$$\begin{aligned}\mathbb{E}[\phi(S_u; Q_u)] &\geq \phi(S_u^*; Q_u) - \big(1 - \frac{1-\epsilon}{K}\big)^K \phi(S_u^*; Q_u) \\ &= \big(1 - \frac{1}{e^{(1-\epsilon)}}\big)\phi(S_u^*; Q_u),\end{aligned}$$

where the last line follows from $e^x \geq (1 + \frac{x}{n})^n$. ☐

IV. Data

In this section we describe the KGs and queries used in our experiments, as well as the various user models we study.

A. Knowledge graphs

We use three large encyclopedic knowledge graphs in our experiments: **`DBPedia`** 3.5.1[4], **`YAGO`** 3[5], and a subset of **`Freebase`**[6], all detailed in Table II. All three KGs contain over ten million triples, spanning topics like music, movies, sports, etc. We do not perform any extra preprocessing on the RDF dumps. Following standard practice [21], we treat each KG as bidirectional by including inverse relations.

[4]https://wiki.dbpedia.org/services-resources/datasets/data-set-35/data-set-351

[5]https://old.datahub.io/dataset/yago

[6]https://developers.google.com/freebase/ – last available version

TABLE II: Knowledge graphs used in our experiments.

	# Entities $\|E\|$	# Relations $\|R\|$	# Triples $\|T\|$
DBPedia	2 026 781	1 043	10 964 261
YAGO	5 155 416	72	19 635 755
Freebase	115 765 760	269 984	1 000 000 000

B. Queries

Real queries. The **WebQuestionsSP** dataset [39] provides manually parsed mappings from natural language questions to structured query graphs for several thousand simple real questions to the Freebase knowledge graph. Each question has a natural language representation and a corresponding query graph representation. Each query graph consists of a *topic entity* and a short path of predicates (an *inferential chain*) leading from the topic to one or more answer entities, with optional *constraints* to limit the number of intermediate answers at any point along the inferential chain.

As our experiments model users querying KGs according to topics of interest (discussed further in § IV-C), we manually categorized 153 WebQuestionsSP queries into five high-level topics according to what we observed in the dataset: “history”, “travel”, “art”, “geography”, and “pop culture”. To ensure consistency, we had three independent assessors annotate each query with a single topic out of the five choices. The inter-annotator agreement using Fleiss’ kappa [7], which quantifies the degree of agreement over that expected by chance, was 85.6% (“almost perfect” agreement). The subset of WebQuestionsSP queries used in our experiments consist of 1.07 triples each, on average.

Synthetic queries. We generate queries to DBPedia and YAGO following standard procedure in KG query answering evaluation tasks (c.f. [11], [12]). Following the structure of the queries in WebQuestionsSP, we start with a topic entity e_i, a path length ℓ, and a number of constraints c. We then follow the procedure to generate a query G_Q:

1) Follow a path without self-loops of ℓ steps starting from entity e_i by uniformly choosing the next entity e_j to visit, and add each encountered relation in the path to the query’s relation set R_Q.
2) After arriving at the query’s answers, add up to c $\langle$relation, argument$\rangle$ constraint pairs to R_Q, E_Q respectively such that the answer set of G_Q remains non-empty.

Since most real queries on KGs contain few triples [31], [3], we limit $\ell \leq 2$ and $c \leq 4$. The synthetic queries to DBPedia/YAGO are more complex than those to Freebase, consisting of 2.05 triples each on average.

C. User querying models

As we are not aware of any publicly available dataset with individual users’ queries to KG, we resort to user simulation with realistic assumptions. Since user behavior is a complex matter, we simulate real behaviors reported in the information retrieval literature and attempt to be consistent with related works that simulate users [29], [6], [32]. We assume that each user is interested in t topics. At any point in time, the user may ask a query from any of their topics of interest, with a small probability of asking “random” or “off-topic” queries. Note that we define topics differently on Freebase versus DBPedia/YAGO: For the latter, since we do not have high-level conceptual topics, we randomly select a set of topic entities (e.g., *Charles Dickens*, *JK Rowling*) from the given KG, and generate queries of varying size and shape anchored around each topic entity. This in effect models two kinds of user behavior: *High-level conceptual* querying for Freebase, and *low-level structural* querying for DBPedia and YAGO.

Given a number of topics t, a number of queries n, and a random query probability p, we simulate a user’s query history Q_u as follows:

1) Uniformly sample a set of t topic entities in the case of DBPedia/YAGO or high-level topic categorizations (“art”, “history”, etc) in the case of Freebase.
2) Uniformly generate a multinomial distribution D specifying the proportion of topics in the log.
3) For each topic, (a) if the KG is DBPedia or YAGO, select a path length $\ell \in \{1, 2\}$ and randomly select a number of constraints $c \in 1 \ldots 4$, then generate a query G_Q centered around the current topic entity, following § IV-B. With probability p, re-generate G_Q with a randomly chosen topic entity. (b) If the KG is Freebase, select a query G_Q categorized into the current topic. With probability p, set G_Q equal to a randomly chosen query from the query database. (c) Add G_Q to Q_u.

Within this model we also simulate *re-retrieval*, a well-documented phenomenon whereby users repeat queries [34], [17]. We limit the percentage of query reuse in our user simulations according to statistics reported in real log analyses [34], from around 20% for the simulations with more topics to 50% for the simulations with fewer topics.

V. Evaluation

Our evaluation focuses on the following questions:

Q1 How well do GLIMPSE personal summaries answer user queries under various conditions and constraints?
Q2 Can GLIMPSE handle large real knowledge graphs?
Q3 How do changes in parameters affect GLIMPSE?

We implemented all methods[7] in Python3 on a single machine with a 6-core 3.50GHz Intel Xeon CPU and 1TB of RAM.

A. Experimental setup

Baselines. As discussed in § II, there is no existing method directly addressing our problem. As such, we modify existing baselines from the literature and introduce a new one:

- **PPR** [14]: As discussed in § II, most graph-based personalization methods rely on variants of personalized PageRank (PPR) to find the entities most relevant to a set of queries. To compare to this method, we perform random walks with restart on the knowledge graph, varying the walk length

[7] Code and data: https://github.com/tsafavi/glimpse-summary

TABLE III: GLIMPSE consistently outperforms competitors across the two user models defined in § V-B: Average F1 score for all methods, knowledge graphs, and user querying models following the settings in § V-A. All averages are over 15 simulated users per KG and querying model. Top performer per experiment in bold. In the GLIMPSE column, the value in parentheses denotes the number of percentage points improvement over the best baseline. $^\blacktriangle$: significant improvement by GLIMPSE over the best baseline for a two-sided t-test at $p < 0.01$.

User model	*Dataset*	TCM	CACHE	PPR-1	PPR-2	PPR-5	PPR-10	GLIMPSE (+ improve.)
Few topics ($t \in 2 \dots 5$)	`DBPedia`	0.687 ± 0.09	0.684 ± 0.09	0.693 ± 0.09	0.846 ± 0.09	0.824 ± 0.09	0.819 ± 0.09	$\mathbf{0.980 \pm 0.02}^\blacktriangle$ (+0.134)
	`YAGO`	0.539 ± 0.11	0.558 ± 0.10	0.549 ± 0.08	0.672 ± 0.08	0.659 ± 0.08	0.653 ± 0.08	$\mathbf{0.814 \pm 0.11}^\blacktriangle$ (+0.142)
	`Freebase`	0.678 ± 0.06	0.707 ± 0.05	0.469 ± 0.05	0.486 ± 0.05	0.499 ± 0.04	0.499 ± 0.04	$\mathbf{0.724 \pm 0.06}$ (+0.017)
Many topics ($t \in 5 \dots 10$)	`DBPedia`	0.585 ± 0.08	0.603 ± 0.08	0.650 ± 0.08	0.782 ± 0.07	0.765 ± 0.08	0.764 ± 0.08	$\mathbf{0.971 \pm 0.03}^\blacktriangle$ (+0.189)
	`YAGO`	0.526 ± 0.07	0.546 ± 0.07	0.552 ± 0.08	0.685 ± 0.07	0.673 ± 0.07	0.670 ± 0.07	$\mathbf{0.768 \pm 0.11}^\blacktriangle$ (+0.082)
	`Freebase`	0.542 ± 0.07	0.577 ± 0.05	0.345 ± 0.05	0.339 ± 0.05	0.350 ± 0.05	0.354 ± 0.05	$\mathbf{0.593 \pm 0.06}$ (+0.016)

in $\{1, 2, 5, 10\}$, and take the subgraph of K edges (triples) induced by the top-ranked entities as S_u. The initial seed PPR value of each entity in the KG is its query frequency in the log Q_u. In our tables and figures, PPR-n refers to an n-step random walk. We implemented PPR using the linear-time power method.

- **TCM** [33]: TCM is one of the only graph summarization methods that is both fast enough to handle massive graphs and flexible enough to be adapted to a (semi-)personalized setting. Briefly, TCM is a sketching method that maps each node in a given graph to a supernode via one or more hash functions. To "personalize" this method, given a query log Q_u, we randomly hash only the entity IDs of entities appearing in Q_u, as well as neighbors of those entities. We set the number of supernodes in the summary to K.
- **CACHE**: To further evaluate how GLIMPSE fares in comparison with a frequency-based "caching" strategy, we devised a method that sorts all entities in the user's query history Q_u by their query frequency, then adds the neighborhoods of entities in descending order of query frequency to S_u until $|T_u| = K$.

We also made a consistent effort to compare GLIMPSE to non-personalized KG summarization methods [4], [32] with the original implementations, but they failed to run on our large-scale KGs (Table II), motivating our choice to take a direction orthogonal to grouping-based summarization (§ II).

Evaluation metrics. We evaluate a personal summary S_u with the average F1 score of query answering on S_u for a given log Q_u (see § III-A for algorithmic details on graph-based query answering). For true positives TP, false positives FP, and false negatives FN, the F1 score $F1 = 2{\cdot}P{\cdot}R/(P + R)$ is defined as the harmonic mean of precision $P = TP/(TP + FP)$ and recall $R = TP/(TP + FN)$. In our setting, TP is the number of query answers in S_u that are also in G, FP is the number of query answers in S_u that are not in G, and FN is the number of query answers in G that are not in S_u.

Settings. Unless otherwise stated, we use the following default parameters. We set $\epsilon = 10^{-3}$ following our analysis in § V-D. For all methods involving random walks, we set $\gamma = 0.85$ following [28]. We use the first 50% of queries from each simulated user's log to create the personal summary S_u, then compute the average F1 score of answering the held-out queries on S_u. In § V-B we use the first 100 million triples of the `Freebase` RDF dump to make running multiple simulations per method feasible. We use larger subsets of `Freebase`, up to one billion triples, in other experiments.

B. Query answering on GLIMPSE summaries

We address question **Q1** by exploring different models of user interests. Each simulation consists of 15 users asking 200 queries. We end the section with an in-depth discussion.

Varying topical interests. Here we study how performance changes with the number of topical interests per simulated user. Table III displays F1 score averages and standard deviations for two aggregate user models: **few topics**, which corresponds to 2-5 topics t of interest per user, and **many topics**, which corresponds to 5-10 topics t of interest per user. Here, all summaries have K equal to 10% of the number of triples $|T|$ of the original KG (we vary K separately in § V-B).

For each user model and dataset, we denote a significant improvement by GLIMPSE over the best baseline for a two-sided t-test at $p < 0.01$ with $^\blacktriangle$. In the GLIMPSE column, the value in parentheses denotes the number of percentage points improvement of GLIMPSE over the best baseline. The relatively high standard deviations are due to the randomness in simulation, reflecting the variability of real users.

The findings per user model may be summarized as follows:

- **Few topics** ($t \in 2 \dots 5$): GLIMPSE outperforms the strongest baselines by around 14% on `DBPedia` and `YAGO`, and 2% on `Freebase`. GLIMPSE's improvement over the best baselines on `DBPedia` and `YAGO` are significant at the $p < 0.01$ level. The PPR methods generally perform second best after GLIMPSE, and specifically PPR-2, which we analyze more in our discussion and § V-D. We find that TCM's performance is overall low due to the existence of "super-edges" between supernodes, which leads to false positives in query answering. CACHE's performance depends on whether the user queries are simple and can be answered within the neighborhoods of queried entities, which is true only for the `Freebase` queries, hence its stronger performance on this dataset.
- **Many topics** ($t \in 5 \dots 10$): Here it is expected that personalization methods should perform poorer, since query reuse is lower and the simulated users' information needs are less focused. The bottom three rows of Table III show the average F1 scores for this model. Here, GLIMPSE outperforms the strongest competitor by 19%, 8%, and 2% on `DBPedia`,

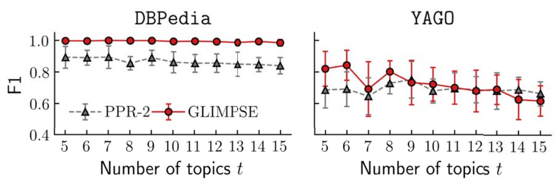
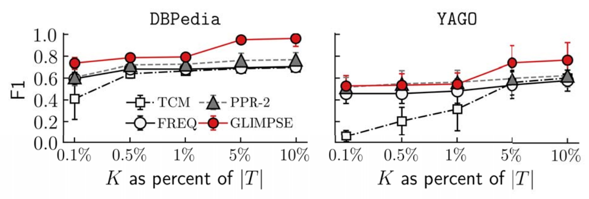
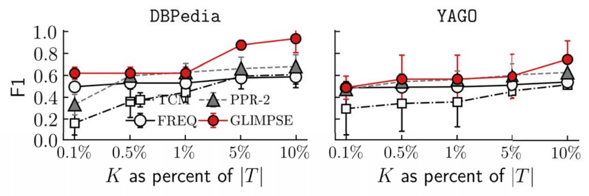


Fig. 3: Comparing GLIMPSE and its closest competitor PPR-2 by varying the number of topics of interest, averaged over 15 simulated users each. GLIMPSE consistently outperforms PPR-2 on `DBPedia`, significant at $p < 0.01$. It is also comparable to or better than PPR-2 on `YAGO` for 10-15 topics.



(a) Few topics model



(b) Many topics model

Fig. 4: GLIMPSE consistently outperforms baselines across constraints: Performance comparison varying K as a percentage of the number of triples $|T|$ in the original KG across user models.

`YAGO`, and `Freebase`, respectively. Interestingly, for both user models PPR-2 usually substantially outperforms PPR-1—by over 15% on `DBPedia`, for example—but adding more steps to PPR exhibits a diminishing returns effect. We believe that this is because most real queries to KGs contain few triples, as demonstrated by the literature [31], [3]. We study this effect further in § V-D.

Figure 3 compares the performance of GLIMPSE to its closest competitor on `DBPedia` and `YAGO`, PPR-2, when the number of topics is varied between 5 and 20. We find that GLIMPSE consistently outperforms PPR-2 on `DBPedia`. The same is true on `YAGO` for 5-10 topics. While GLIMPSE's performance slightly decreases on `YAGO` for $10-15$ topics, it still performs comparably with PPR-2. We discuss the differences between `DBPedia` and `YAGO` and interpret these findings further at the end of this section.

Varying the constraint K. As discussed in § III-D, the value of K may depend on the user, the amount of device space, or the application scenario. Here, we experiment with different values of K, where we vary K as a percentage of the number of triples $|T|$ in the KG, to observe how the F1 score changes with different levels of constraints. We show results in Figure 4 across user models for `DBPedia` and `YAGO`, using only PPR-2 among the PPR methods since PPR-2 consistently performs the best. As expected, across all methods, the query answering F1 score increases with K, since a larger K means the summary S_u has more capacity for facts. That said, GLIMPSE still performs well, for example at nearly 80% average F1 on `DBPedia` at just 0.1% of the number of triples $|T|$. For reference, this corresponds to a GLIMPSE summary that uses just 3.6 MB of memory, for a KG that originally uses around 2600 MB memory.

Discussion. While certain baselines can perform as well as GLIMPSE under specific conditions, GLIMPSE is better at generalizing across datasets and user models. For example, the baseline CACHE performs relatively well on the `Freebase` logs where the queries have on average around one triple, but cannot handle the more complex queries to `DBPedia` and `YAGO`. Another example is PPR-2, which is the closest competitor to GLIMPSE on `DBPedia` and `YAGO`. PPR-2 performs comparably to GLIMPSE on `YAGO` when the number of topics t is high or the constraint K is tighter (Figures 3 and 4b). We hypothesize that this is because `YAGO` has relatively few relations and contains several extremely high-degree entities to which many other entities are connected. For example, most *Person* entities in `YAGO` are connected by the *hasGender* relation to one of the *Male* or *Female* entities. PPR includes all entities connected to these high-degree hubs, thereby answering many queries connected to these hubs. However, not all KGs have this structure: `DBPedia`, for example, does not, and GLIMPSE consistently and significantly outperforms PPR-2 here. Moreover, the generated queries to `YAGO` are less complex than those on `DBPedia` due to the fact that `YAGO` contains very few unique relations, and therefore each entity in `YAGO` has fewer outgoing/incoming edges (Table II).

Importantly, GLIMPSE performs substantially better in particular over all PPR variants on the real `Freebase` queries, whereas PPR is usually a stronger baseline on `DBPedia` and `YAGO`. We believe this is due to important inherent properties of KGs, which contain both low-level structure (entity-entity links) and high-level concepts (subjective topics). Recall that the `Freebase` queries were manually grouped by high-level topic (§ IV-B). Therefore, two queries from the same user in the same topic (e.g., "geography" or "history") may *not* be close distance-wise in the graph, which is the only type of query similarity that PPR can capture. In fact, querying behavior without structural coherence or locality in the KG can be considered *adversarial* to personalization methods that only extract paths or subgraphs. By contrast, because GLIMPSE selects triples one at a time for the summary S_u, it is more robust. It includes different facts in the summary S_u that are not necessarily structurally close in the graph. Therefore, GLIMPSE relies on but does not over-emphasize the graph's structure, and **can handle both the lower-level structure and the higher-level concepts over the KG**.

C. Scalability of GLIMPSE

In this section, we address question **Q2**, which concerns the scalability of GLIMPSE.

TABLE IV: Comparison of GLIMPSE runtime on a subset of `Freebase` with and without the optimizations discussed in § III-E. Evidently, the optimizations are necessary for GLIMPSE to be feasible on encyclopedic knowledge graphs.

	GLIMPSE	With OPT1 only	With OPT2+3 only
Runtime (seconds)	2.11 ± 0.08	15487.93 ± 978.05	28980.46 ± 416.38
Relative to GLIMPSE	1×	7340×	13734×

Benefits of optimization. Here we compare GLIMPSE as it is described in § III-E and Algorithm 1 to non-optimized versions of GLIMPSE. In our first experiment, we only use OPT1, the precomputation of $T^{\Delta \neq 0}$. In the second experiment, we only use OPT2 and OPT3, the random sampling of triples and lazy updating of marginal values. These experiments were all run on a relatively small subset ($|T| = 1\,000\,000$) of the `Freebase` knowledge graph, averaged over 3 simulated users asking 100 queries across two topics. We simulate fewer users here because runtime varies very little from one user to another. Table IV summarizes our results. Evidently, the optimizations are necessary for GLIMPSE to be viable in the real world. GLIMPSE with both optimizations takes only 2 seconds on average. By contrast, only using OPT1, GLIMPSE takes on average 4.5 hours and is $> 7\,000\times$ slower than the fully optimized version of GLIMPSE. Only using OPT2 and OPT3, GLIMPSE takes on average *8 hours* and is $> 13\,000\times$ times slower than fully optimized GLIMPSE. In particular, these results demonstrate that OPT1, our optimization tailored to personalization, allows GLIMPSE to operate at scale.

Runtime. Figure 5 shows how summarizing KGs with GLIMPSE scales for progressively larger subsets of `Freebase`, each one an order of magnitude larger than the previous, up to one billion triples. These values are averages over three simulated users asking 100 queries each across two topics. Our results confirm that GLIMPSE is indeed practical on web-scale data. For instance, GLIMPSE takes only *two minutes* on average for a 10 million-triple subset of `Freebase`. Figure 6 compares the summarization runtime of GLIMPSE, PPR-2, CACHE, and TCM, averaged over all experiments in § V-B (as all PPR variants took around the same amount of time, only PPR-2 is shown). From the figure it can be seen that our versions of TCM and CACHE are the fastest, sublinear in the number of triples in G, because they only consider the queried entities and neighbors of those queried entities. However, TCM and CACHE are usually not competitive baselines. On the other hand, the stronger performers PPR and GLIMPSE are linear in the number of triples. While GLIMPSE is slightly slower than PPR due to the

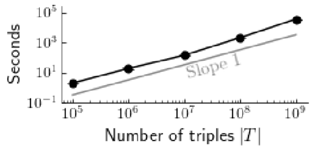


Fig. 5: GLIMPSE scalability (seconds) on `Freebase`.

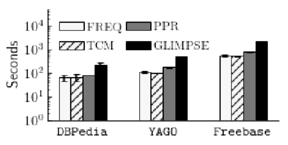
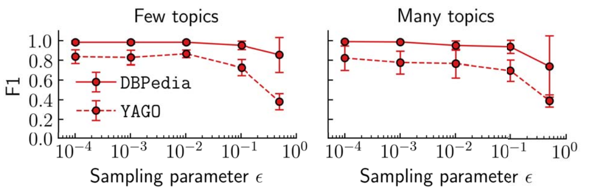
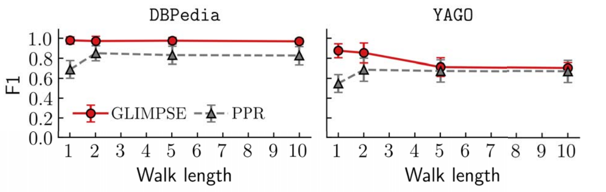


Fig. 6: Summarization time on all KGs.



(a) Varying the sampling parameter ϵ from OPT2.



(b) Varying the random walk length on GLIMPSE and PPR.

Fig. 7: Parameter analysis of GLIMPSE and competitors.

extra marginal value updates (§ III-E), the extra computation pays off in accuracy, as demonstrated in § V-B.

D. Parameter analysis

Finally, we address question **Q3**: How do variations in parameters affect GLIMPSE's performance?

Sampling parameter ϵ. Here we study how varying the sampling parameter ϵ, which in turn changes the expected theoretical performance of GLIMPSE (Theorem 3), affects query answering accuracy. Figure 7a shows the results on `DBPedia` and `YAGO` for $\epsilon \in [10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 0.5]$ for both user querying models. We find that for very small values of ϵ (10^{-2} or less) GLIMPSE performs well with fairly stable results, although we observe slight variance and noise due to the randomness of sampling. However, with a large value of ϵ, the performance degrades rapidly and the results are less stable, as evidenced by the large standard deviations in the plot for $\epsilon = 0.5$. This is to be expected. The larger the ϵ, the smaller the sampled set, which makes it more likely for sub-optimal triples to be chosen for S_u. Therefore, we recommend a value of $\epsilon = 10^{-2}$ or less for good performance.

Random walk length. Recall that in § III-C our user entity preference model can be interpreted as a random walk with restart controlled by restart parameter γ. In these final experiments, we model the user's preference distribution over G with varying-length random walks, and compare these results to our variants of PPR. Figure 7b shows that across all KGs, GLIMPSE's performance with random walks either plateaus or else decreases, whereas PPR's performance with longer random walks increases from one to two, and then plateaus. Thus, random walks on KGs longer than two steps appear to result in diminishing returns. We believe these findings are KG-specific, due to the unique structure of real queries to a knowledge graph. Assuming that such queries are "localized" in the graph and do not span more than a few triples, which has been demonstrated in several analyses of real KG queries [31], [3], longer walks may add more complexity than necessary.

VI. Conclusion

Motivated by the disparity between the massive scale of encyclopedic knowledge graphs and the relatively limited information needs of individuals, this paper proposes personalized knowledge graph summarization. Toward this goal we devise GLIMPSE, and demonstrate its empirical and theoretical strengths. That said, there are many possibilities for future work. For example, online (incremental) methods may prove useful as facts are updated, KGs are augmented, and user interests evolve. Future work could also make use of the rich semantics provided by ontologies, as well as contextual user cues (e.g., location, preferred language), as is common in traditional ad-hoc web search [36]. As such, we believe our work opens up many possibilities for emerging approaches to accessing and managing world knowledge.

Acknowledgments

The authors thank Edgar Meij for his helpful feedback. This material is based upon work supported by the National Science Foundation under Grant No. IIS 1845491, Army Young Investigator Award No. W911NF1810397, an Adobe Digital Experience award, an Amazon research faculty award, and an NSF Graduate Research Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF or other funding parties.

References

[1] Sorour E Amiri, Bijaya Adhikari, Aditya Bharadwaj, and B Aditya Prakash. Netgist: Learning to generate task-based network summaries. In *ICDM*, pages 857–862. IEEE, 2018.

[2] Paul N Bennett, Ryen W White, Wei Chu, Susan T Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. Modeling the impact of short-and long-term behavior on search personalization. In *SIGIR*, pages 185–194. ACM, 2012.

[3] Angela Bonifati, Wim Martens, and Thomas Timm. An analytical study of large sparql query logs. *VLDB*, 11(2):149–161, 2017.

[4] Šejla Čebirić, François Goasdoué, and Ioana Manolescu. Query-oriented summarization of rdf graphs. *VLDB*, 8(12):2012–2015, 2015.

[5] Gong Cheng, Thanh Tran, and Yuzhong Qu. Relin: relatedness and informativeness-based centrality for entity summarization. In *ISWC*, pages 114–129. Springer, 2011.

[6] Wenfei Fan, Jianzhong Li, Xin Wang, and Yinghui Wu. Query preserving graph compression. In *SIGMOD*, pages 157–168. ACM, 2012.

[7] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[8] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864. ACM, 2016.

[9] Yu Gu, Tianshuo Zhou, Gong Cheng, Ziyang Li, Jeff Z Pan, and Yuzhong Qu. Relevance search over schema-rich knowledge graphs. In *WSDM*, pages 114–122. ACM, 2019.

[10] Kalpa Gunaratna, Krishnaparasad Thirunarayan, and Amit Sheth. Faces: diversity-aware entity summarization using incremental hierarchical conceptual clustering. In *AAAI*, 2015.

[11] Kelvin Guu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. In *EMNLP*, pages 318–327, 2015.

[12] Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. Embedding logical queries on knowledge graphs. In *NeurIPS*, pages 2030–2041, 2018.

[13] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. Dynamic factual summaries for entity cards. In *SIGIR*, pages 773–782. ACM, 2017.

[14] Taher Haveliwala, Sepandar Kamvar, and Glen Jeh. An analytical comparison of approaches to personalizing pagerank. Technical report, Stanford, 2003.

[15] Di Jin and Danai Koutra. Exploratory analysis of graph data by leveraging domain knowledge. In *ICDM*. IEEE, 2017.

[16] Di Jin, Ryan A. Rossi, Eunyee Koh, Sungchul Kim, Anup Rao, and Danai Koutra. Latent network summarization: Bridging network embedding and summarization. In *KDD*. ACM, 2019.

[17] Maryam Kamvar, Melanie Kellar, Rajan Patel, and Ya Xu. Computers and iphones and mobile phones, oh my!: a logs-based comparison of search users on different devices. In *WWW*, pages 801–810. ACM, 2009.

[18] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. VoG: Summarizing and Understanding Large Graphs. In *SDM*, pages 91–99. SIAM, 2014.

[19] K. Ashwin Kumar and Petros Efstathopoulos. Utility-driven graph summarization. *VLDB*, 12(4):335–347, December 2018.

[20] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *KDD*, pages 631–636. ACM, 2006.

[21] Xi Victoria Lin, Richard Socher, and Caiming Xiong. Multi-hop knowledge graph reasoning with reward shaping. In *EMNLP*, pages 3243–3253, 2018.

[22] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. Graph summarization methods and applications: A survey. *ACM CSUR*, 51(3):62, 2018.

[23] Antonio Maccioni and Daniel J Abadi. Scalable pattern matching over compressed graphs via dedensification. In *KDD*, pages 1755–1764. ACM, 2016.

[24] Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *AAAI*, pages 1812–1818, 2015.

[25] George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[26] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[27] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. Industry-scale knowledge graphs: Lessons and challenges. *Commun. ACM*, 62(8):36–43, July 2019.

[28] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[29] Filip Radlinski, Robert Kleinberg, and Thorsten Joachims. Learning diverse rankings with multi-armed bandits. In *ICML*, pages 784–791. ACM, 2008.

[30] Matteo Riondato, David García-Soriano, and Francesco Bonchi. Graph summarization with quality guarantees. *DMKD*, 31(2):314–349, 2017.

[31] Muhammad Saleem, Muhammad Intizar Ali, Aidan Hogan, Qaiser Mehmood, and Axel-Cyrille Ngonga Ngomo. Lsq: the linked sparql queries dataset. In *ISWC*, pages 261–269. Springer, 2015.

[32] Qi Song, Yinghui Wu, and Xin Luna Dong. Mining summaries for knowledge graph search. In *ICDM*, pages 1215–1220. IEEE, 2016.

[33] Nan Tang, Qing Chen, and Prasenjit Mitra. Graph stream summarization: From big bang to big crunch. In *SIGMOD*, pages 1481–1496. ACM, 2016.

[34] Jaime Teevan, Eytan Adar, Rosie Jones, and Michael AS Potts. Information re-retrieval: repeat queries in yahoo's logs. In *SIGIR*, pages 151–158. ACM, 2007.

[35] Nikos Voskarides, Edgar Meij, Ridho Reinanda, Abhinav Khaitan, Miles Osborne, Giorgio Stefanoni, Prabhanjan Kambadur, and Maarten de Rijke. Weakly-supervised contextualization of knowledge graph facts. In *SIGIR*, pages 765–774. ACM, 2018.

[36] Ryen W White, Paul N Bennett, and Susan T Dumais. Predicting short-term interests using activity-based search context. In *CIKM*, pages 1009–1018. ACM, 2010.

[37] Chenyan Xiong and Jamie Callan. Query expansion with freebase. In *ICTIR*, pages 111–120. ACM, 2015.

[38] Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL*, volume 1, pages 1321–1331, 2015.

[39] Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *ACL*, volume 2, pages 201–206, 2016.

[40] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *KDD*, pages 353–362. ACM, 2016.