APEX^2: Adaptive and Extreme Summarization for Personalized Knowledge Graphs

Zihao Li
University of Illinois Urbana-Champaign
Illinois, USA
zihaoli5@illinois.edu

Dongqi Fu
Meta AI
California, USA
dongqifu@meta.com

Mengting Ai
University of Illinois Urbana-Champaign
Illinois, USA
mai10@illinois.edu

Jingrui He
University of Illinois Urbana-Champaign
Illinois, USA
jingrui@illinois.edu

Abstract

Knowledge graphs (KGs), which store an extensive number of relational facts, serve various applications. Recently, *personalized knowledge graphs* (PKGs) have emerged as a solution to optimize storage costs by customizing their content to align with users' specific interests within particular domains. In the real world, on the one hand, user queries and their underlying interests are inherently evolving, requiring PKGs to adapt continuously; on the other hand, the summarization is constantly expected to be as small as possible in terms of storage cost. However, the existing PKG summarization methods implicitly assume that the user's interests are constant and do not shift. Furthermore, when the size constraint of PKG is extremely small, the existing methods cannot distinguish which facts are more of immediate interest and guarantee the utility of the summarized PKG. To address these limitations, we propose APEX^2, a highly scalable PKG summarization framework designed with robust theoretical guarantees to excel in adaptive summarization tasks with extremely small size constraints. To be specific, after constructing an initial PKG, APEX^2 continuously tracks the interest shift and adjusts the previous summary. We evaluate APEX^2 under an evolving query setting on benchmark KGs containing up to 12 million triples, summarizing with compression ratios $\leq 0.1\%$. The experiments show that APEX outperforms state-of-the-art baselines in terms of both query-answering accuracy and efficiency.

ACM Reference Format:
Zihao Li, Dongqi Fu, Mengting Ai, and Jingrui He. 2025. APEX^2: Adaptive and Extreme Summarization for Personalized Knowledge Graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 27 pages. https://doi.org/10.1145/3690624.3709213

1 Introduction

Knowledge graphs (KGs) have been proven an effective tool for constructing solutions in many application domains, such as healthcare, finance, cyber security, education, question answering, and social network analysis [11, 23, 33, 38–41, 53, 78]. Due to the ever-growing amount of data, encyclopedic knowledge graphs are becoming increasingly large and complex [12, 13], such as DBpedia [2], Freebase [5], Wikidata [62], and YAGO [57]. In contrast, KG users (e.g., individual people, systems, software packages) usually do not have very general interests but only care about a small portion of the whole KG for certain topics. Therefore, personalized knowledge graphs (PKGs) have recently attracted much research attention for balancing storage cost and query-answering accuracy [27, 55, 61]. In brief, a personalized knowledge graph (PKG) is extracted (summarized, compressed, or distilled) from a larger comprehensive knowledge graph. For KG and an individual user, a PKG has a limited size, but contains many entities/triples in the KG that the user is interested in, and can answer their personal queries appropriately. Furthermore, on the application side, each user will have a PKG. Since there might be many individual users, each PKG is expected to store as few facts as possible, to minimize the total storage cost.

As the individual KG's size has been very large, re-summarizing PKG from scratch requires many computational resources, but in

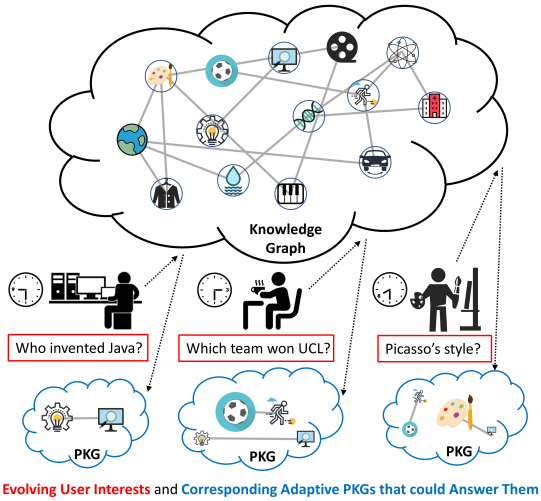


Figure 1: Example of Adaptive PKGs. The entire KG is stored on a cloud server, and the PKG is stored on the user's device. The initial PKG is constructed based on the query "Who invented Java" at 9am. Then, this PKG adapts to the same user's evolving queries at 3pm and 8pm.

KDD '25, August 3–7, 2025, Toronto, ON, Canada

ACM ISBN 979-8-4007-1245-6/25/08
https://doi.org/10.1145/3690624.3709213

the real world, users' personal query interests may shift over time. As shown in Figure 1, in the morning (9am), the user works as a programmer and wonders about software engineering. During the afternoon coffee break (3pm), the same user cares about the UEFA Champions League (soccer games). In the evening (8pm), the same user likes painting and is curious about art. Under these circumstances, on one hand, an outdated summarized PKG might be sub-optimal, since outdated information agglomerates and adversely affects both storage cost and search performance [12]. On the other hand, re-summarizing PKG from scratch at each timestamp causes unaffordable computational complexity. Given the fact that the KG is massively large [2, 5, 57, 62], to make the PKG acceptably small, the compression ratio has to be extremely small. For example, the entire YAGO3 is about 300GB. Even with a 1% compression ratio, the PKG is 3GB, which is still larger than most mobile applications. For the KGs measured in TB, more extreme compression is needed.

Motivated by the above use case, the previous work [12] informally introduced the problem of **adaptive PKG summarization**, which seeks to find a compact summary given the knowledge graph and query history. Take Figure 1 as an example. We expect the adaptive PKG could, during the afternoon coffee break, quickly adapt from software engineering to more sports-related topics, decaying but not eliminating topics on software engineering. Then, at night, the PKG evolves with the user's interests in art topics. Moreover, in this paper, we study how to adaptively summarize the PKG under **extremely small storage constraints**. Theoretically, in Appendix B, we show that the existing PKG summarization methods [27, 55], even if re-run from scratch for the new interested topics, could not incorporate the new interests into the previously summarized PKG when query topics change, or corrupt under extremely small storage constraints.

To address these limitations, we propose APEX^2 (**A**da**p**tive and **Ex**treme Summ**a**rization for **Pe**rsonalized **K**nowledge Graph**s**), which enables summarization to incrementally evolve with user interests over time while satisfying extremely small storage constraints. To the best of our knowledge, this work presents the first adaptive PKG summarization framework tailored for evolving query topics. In brief, given a bunch of queries with different interested topics at different timestamps, APEX^2 works by modeling user interests through a heat diffusion process [9] and maintaining dynamic data structures that allow incremental updates. Under the extremely small storage limitation, APEX^2 incrementally infers the interest scores of the facts and picks the ones with the highest scores (i.e., immediately more interested by the user). Our contributions are summarized as:

- **Problem Formulation and Theoretical Analysis**. We formally formulate the problem of adaptive PKG summarization with storage limitation. We provide theoretical analysis of the adaptability of existing PKG summarization methods. We also prove the efficiency and topic adaptability of our methods.
- **Algorithm**. We propose the adaptive and extreme PKG summarization solution APEX^2 and its variant APEX^2-N to address different circumstances with theoretical guarantees.
- **Experiment Evaluation**. We design extensive experiments under real-world query-answering scenarios on real KGs to show the effectiveness and efficiency of our proposed methods.

Table 1: Table of Notation

Symbol	Definition and Description
$\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$	Knowledge graph being investigated, with entity set $\mathcal{E}$, relation set $\mathcal{R}$ and triple set $\mathcal{T}$
n	number of entities in $\mathcal{G}$ (i.e., $n = \lvert\mathcal{E}\rvert$)
$\mathcal{P} = (\mathcal{E}_p, \mathcal{R}_p, \mathcal{T}_p)$	Personalized Knowledge Graph, with entity set $\mathcal{E}_p$, relation set $\mathcal{R}_p$ and triple set $\mathcal{T}_p$
$\boldsymbol{Q}$	query log by the user
x_{ijk}	triple with entity e_i, e_j and relation r_k (i.e., $x_{ijk} = (e_i, r_k, e_j)$)
$\boldsymbol{A}$	(Sparse $n \times n$) adjacency matrix of $\mathcal{G}$
$\boldsymbol{H}$	(Sparse $n \times n$) heat matrix storing heat of entities (diagonal entries) and triples (non-diagonal entries)
α	damping factor of neighbor
ϵ	tunable tolerance
γ	decay factor
d	diffusing diameter
$\lvert\mathcal{S}\rvert$	number of elements in the set $\mathcal{S}$; Specifically, $\lvert\mathcal{G}\rvert$ represents number of triples in knowledge graph $\mathcal{G}$

The rest of the paper is organized as follows. In Section 2, we introduce the problem setting and background. In Section 3, we introduce how APEX^2 dynamically and incrementally models user's evolving interests. In Section 4, we formally propose APEX^2 and its variant APEX^2-N. We theoretically analyze both APEX^2 and APEX^2-N from multiple aspects. In Section 5, we report the experimental results showing the effectiveness and efficiency of our methods. We discuss related works in Section 6 and conclude in Section 7.

Reproducibility. The code and the download instructions of KG datasets are provided. Refer to Appendix C.4 for more details.

2 Problem Definition

We use calligraphic letters (e.g., $\mathcal{A}$) for sets, bold capital letters for matrices (e.g., $\boldsymbol{A}$), parenthesized superscript to denote the temporal index (e.g., $\boldsymbol{A}^{(t)}$), unparenthesized superscript to denote the power (e.g., $\boldsymbol{A}^k$). For matrix indices, we use $\boldsymbol{A}_{i,j}$ to denote the entry in the i^{th} row and the j^{th} column. The notation used in our proposed APEX is summarized in Table 1.

Knowledge Graph. A knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ is defined by an entity set $\mathcal{E}$, a relation set $\mathcal{R}$ and a triple set $\mathcal{T}$. A triple $x_{ijk} = (e_i, r_k, e_j) \in \mathcal{T}$ is defined by entities e_i, e_j and their relationship r_k. The undirected adjacency matrix $\boldsymbol{A}$ is defined as

$$\boldsymbol{A}_{i,j} = 1 \iff i \neq j \wedge \exists k \; s.t. \; x_{ijk} \; or \; x_{jik} \in \mathcal{T} \tag{1}$$

Personalized Knowledge Graph. A PKG $\mathcal{P}$ of KG $\mathcal{G}$ is defined by an entity set $\mathcal{E}_p \subseteq \mathcal{E}$, a relation set $\mathcal{R}_p \subseteq \mathcal{R}$ and a triple set $\mathcal{T}_p \subseteq \mathcal{T}$. In our problem setting, a PKG is summarized from a KG according to the user's query log $\boldsymbol{Q}$.

Query Log. Since most real queries to KGs consist of only one or two triples [6], assuming we have the full access to the whole KG, we study simple queries with known answers. A **query log** $\boldsymbol{Q}$ consists of a number of queries. Each **query** q consists of a **query entity** e, **query relation** r, and a set of **answer entities** $\mathcal{A}$. A triple $x_{ijk} = (e_i, r_k, e_j)$ in the knowledge graph $\mathcal{G}$ is said to be an **answer triple** to a query q, if $e_i = e \wedge r_k = r \wedge e_j \in \mathcal{A}$. A query may have multiple answer triples. For example, for the query "What movie did Christopher Nolan direct", the query entity is "Christopher Nolan" ("Nolan" for short), and the query relation is

"directed_movie" ("d_m" for short). Suppose in the KG all triples with the head entity "Nolan" and the relation "d_m" are ("Nolan", "d_m", "Interstellar"), ("Nolan", "d_m", "Tenet") and ("Nolan", "d_m", "Oppenheimer"). Then these three triples are answer triples to the query, and the set {"Interstellar", "Tenet", "Oppenheimer"} is the set of answer entities. In this paper, "for triples in q" iterates the answer triples of q; "$e \in q$" iterates all entities accessed by q as the query entity or answer entities; "$\mathcal{Q}^{(t+1)} \setminus \mathcal{Q}^{(t)}$" stands for all queries in $\mathcal{Q}^{(t+1)}$ but not in $\mathcal{Q}^{(t)}$.

Problem. *Adaptive Personalized Knowledge Graph Summarization. "{}" means "a sequence of" here. F1 score is a conventional measure of seaching accuracy, defined in Appendix C.1.*

Input: *(i) a knowledge graph $\mathcal{G}$*
(ii) a sequence of varying end-user query interests, represented by a temporal query log $\{\mathcal{Q}^{(0)}, \mathcal{Q}^{(1)}, \mathcal{Q}^{(2)}, \ldots, \mathcal{Q}^{(T)}\}$
(iii) constant size budget K

Output: *a sequence of personalized knowledge graph $\{\mathcal{P}^{(t)}\}$ of $\mathcal{G}$ for $t \in \{0, 1, \ldots, T-1\}$; each $\mathcal{P}^{(t)}$, whose number of triples $|\mathcal{P}^{(t)}|$ is not greater than K, is able to answer as many queries in $\mathcal{Q}^{(t+1)} \setminus \mathcal{Q}^{(t)}$ as possible and as correctly as possible:*

$$\arg\max_{\{\mathcal{P}^{(t)}\}} \sum_{0 \le t \le T-1} \sum_{q \in \mathcal{Q}^{(t+1)}/\mathcal{Q}^{(t)}} F1(\mathcal{P}^{(t)}, q) \; s.t. \; |\mathcal{P}^{(t)}| \le K \quad (2)$$

In this paper, when the size budget $K \le 1\%$, we say the problem becomes **adaptive and extreme PKG summarization**. The smallest K value that existing PKG summarization methods [27, 55, 61] have explicitly used is 10%. In our experiments, $K \le 0.1\%$.

Heat Diffusion. Heat diffusion process is a natural way to model users' interest [3, 46, 50], as warmer nodes are considered more immediately interesting to the user [12]. Once a query is performed by the user, the queried area will gain more heat, and then globally, the heat of each node will be partially pushed to its neighbors. In this work, we adopt a heat decay-inject-diffuse framework, with a visual example and more details provided in Appendix D.

3 Adaptive PKG Summarization

In the adaptive PKG summarization scenario, compared to previous static PKG summarization [27, 55, 61], the major difference is that the user's interest may be shifting, manifested by the user's evolving query history. The adapting may be simply achieved by re-applying summarization methods from scratch every time the user's query log has evolved. But such a from-scratch solution lacks real-time efficiency. A natural follow-up is: can we reuse the results from the previous summarization and further develop a real-time framework that can evolve incrementally with the user's interest? Moreover, when the storage constraint is extremely small, can we incrementally maintain a descending-order rank of the user's interests in each entity/relation/triple? To address these questions, we propose our solution APEX^2.

The core idea of APEX^2 is to maintain a real-time sparse heat structure that stores the user's interest and incrementally updates its content, then greedily chooses the triples with the highest heat to construct the summarization, even under extremely small storage limitations. To adapt to the user's interests, we introduce a decay factor γ into our framework. This factor is crucial to both adapting effectiveness and efficiency: (i) decaying previous interests gives higher priority to recent queries, which are more likely to represent the user's current interest; (ii) for a set of elements, decaying all of them does not affect their order and will not introduce additional computations to the heat ranking process. γ controls the trade-off between adapting to new interests and retaining relevant information from past queries.

Systematically, our APEX^2 consists of three components for adaptive PKG summarization, i.e., *Dynamic Model of User Interests*, *Incremental Updating*, and *Incremental Sorting*. In brief, *Dynamic Model of User Interests* is proposed to model the user interest dynamically. Then based on the evolving interests modeled, *Incremental Updating* tracks the new interests. *Incremental Sorting* is necessary to construct a high-quality newly summarized PKG, under an extremely small storage constraint. Details of the three components are introduced through Subsections 3.1 – 3.3. Our end-to-end APEX^2 is summarized in Algorithm 4.

3.1 Dynamic Model of User Interests

In order to formulate more conveniently, we **start by freezing at a specific timestamp** T (T **as a constant**), and define $\boldsymbol{q}_{total}$ to store the number of times each entity is accessed as query entity or answer entity in the query log. If accessed as the answer to a query, the marginal value will be weighted by $\frac{1}{\#\ of\ answers}$. We provide a simple example here. Suppose we have 5 entities (indexed 0, 1, 2, 3, 4). The first query is (entity 0, some relation, {entity 1, entity 3}), the second query is (entity 2, some relation, {entity 0, entity 3}), then $\boldsymbol{q}_{\text{total}}$ will be a column vector $(1+0.5, 0.5+0, 0+1, 0.5+0.5, 0+0) = (1.5, 0.5, 1, 1, 0)$. Assuming the user's temporal query log is $\mathcal{Q}^{(t)}$, we use $\mathcal{Q}^{(t)} \setminus \mathcal{Q}^{(t-1)}$ to denote new queries arriving at time t. Additionally, $\mathcal{Q}^{-1} = \emptyset$. Then,

$$\boldsymbol{q}_{\text{total}} = \sum_{t=0}^{T} \boldsymbol{q}^{(t)} = \sum_{t=0}^{T} \sum_{i \in \mathcal{Q}^{(t)} \setminus \mathcal{Q}^{(t-1)}} \boldsymbol{q}_i \quad (3)$$

where for each query $i \in \mathcal{Q}^{(t)} \setminus \mathcal{Q}^{(t-1)}$ with answer set $\mathcal{A}_i$, $\boldsymbol{q}_i$ is a vector with dimension $(|\mathcal{E}| \times 1)$ whose entries are

$$\boldsymbol{q}_i[e] = \begin{cases} 1 & e \text{ is the query entity of query } i \\ \frac{1}{|\mathcal{A}_i|} & e \text{ is an answer entity to query } i \\ 0 & \text{otherwise } (e \text{ is unrelated to query } i) \end{cases} \quad (4)$$

where $e \in \{1, \ldots, |\mathcal{E}|\}$ is the index of entities.

We model user's interest on an entity e in a heat-diffusing style. Define $N_l(e)$ to be the l-hop neighbors of entity e. Additionally $N_0(e) = e$. A **topic** is a sub-area in the KG that has implicit inner connections. Such connections carry semantic meanings for humans (e.g., artistic, physical entities) and are modeled topologically by entities[1]. With the straightforward inspiration that entities near the searched ones are likely to be in the user's interested topics, we model the user's static preference for entities as

$$\Pr(e|\mathcal{Q}) = \sum_{l=0}^{d} \alpha^l \sum_{e_o \in N_l(e)} \sum_{q \in \mathcal{Q}^{(T)}} \mathbb{1}(e_o \in q) \quad (5)$$

[1]Freebase documentation explicitly define topics to correspond to nodes in the KG [55]. https://developers.google.com/freebase/guide/basic_concepts?hl=en

Equivalently, by Equation 3, the vector $\boldsymbol{e}(e) = \Pr(e|Q)$ can be written in matrix expression as

$$\boldsymbol{e} = \boldsymbol{q}_{\text{total}} + \alpha \boldsymbol{A}\boldsymbol{q}_{\text{total}} + \alpha^2 \boldsymbol{A}^2 \boldsymbol{q}_{\text{total}} + \ldots = \sum_{l=0}^{d} \alpha^l \boldsymbol{A}^l \boldsymbol{q}_{\text{total}} \tag{6}$$

and we can use a closed form [34] to calculate the case $d \to +\infty$:

$$\lim_{d \to +\infty} \boldsymbol{e} = \sum_{l=0}^{\infty} \alpha^l \boldsymbol{A}^l \boldsymbol{q}_{\text{total}} = (\boldsymbol{I} - \alpha \boldsymbol{A})^{-1} \boldsymbol{q}_{\text{total}} \tag{7}$$

To model the user's interest in relations, we simply use a frequency-based approach. For a query $i \in Q$,

$$\Pr(r_k|Q) \propto \sum_{i \in Q} (\mathbb{1}(r_k \in i)) \tag{8}$$

where r_k is the k-th relation in KG $\mathcal{G}$, and $\boldsymbol{r}$ is a vector with dimension $(|\mathcal{R}| \times 1)$ as follows

$$\boldsymbol{r} = \sum_{t=0}^{T} \boldsymbol{q}_r^{(t)} = \sum_{t=0}^{T} \sum_{i \in Q^{(t)} \setminus Q^{(t-1)}} \tilde{\boldsymbol{q}}_i \tag{9}$$

where $\tilde{\boldsymbol{q}}_i$ is the one-hot vector with dimension $(|\mathcal{R}| \times 1)$ corresponding to the searched relation in query $i \in Q^{(t)} \setminus Q^{(t-1)}$ at time t,

$$\tilde{\boldsymbol{q}}_i[r] = \begin{cases} 1 & r \text{ is the query relation of query } i \\ 0 & \text{otherwise } (r \text{ is unrelated to query } i) \end{cases} \tag{10}$$

where $r \in \{1, \ldots, |\mathcal{R}|\}$ is the index of relations.

Then, we are ready to introduce the dynamics (from now T starts to evolve) and decaying factor γ into the problem on both entities and relations. The decaying factor γ controls the trade-off between adapting to new interests and retaining relevant information from past queries. Involving decay factor γ into $\boldsymbol{q}_{\text{total}}, \boldsymbol{e}, \boldsymbol{r}$, while a heat-diffusing style still applies, at timestamp T, temporal expressions of Eqs. 3, 6, 8 become

$$\boldsymbol{q}_{\text{total}}^{(T)} = \sum_{t=0}^{T} \gamma^{T-t} \boldsymbol{q}^{(t)} = \sum_{t=0}^{T} \gamma^{T-t} \sum_{i \in Q^{(t)} \setminus Q^{(t-1)}} \boldsymbol{q}_i \tag{11}$$

$$\boldsymbol{e}^{(T)} = \sum_{l=0}^{d} \alpha^l \boldsymbol{A}^l \boldsymbol{q}_{\text{total}}^{(T)} \tag{12}$$

$$\boldsymbol{r}^{(T)} = \sum_{t=0}^{T} \gamma^{T-t} \boldsymbol{q}_r^{(t)} = \sum_{t=0}^{T} \gamma^{T-t} \sum_{i \in Q^{(t)} \setminus Q^{(t-1)}} \tilde{\boldsymbol{q}}_i \tag{13}$$

As for the objective function, we stick to GLIMPSE's choice for triple preference. Further, we define our objective to be based only on triple preference. The mathematical expressions are

$$\begin{aligned} \Pr(x_{ijk}|Q) &\propto \Pr(e_i|Q)\Pr(r_k|Q)\Pr(e_j|Q) \\ &= \boldsymbol{e}^{(T)}[i]\boldsymbol{r}^{(T)}[j]\boldsymbol{e}^{(T)}[k] \end{aligned} \tag{14}$$

$$\begin{aligned} \phi(\mathcal{P}, Q) &= \sum_{x_{ijk} \in \mathcal{T}_p} \log \Pr(x_{ijk}|Q) \\ &= \sum_{x_{ijk} \in \mathcal{T}_p} \log \boldsymbol{e}^{(T)}[i]\boldsymbol{r}^{(T)}[j]\boldsymbol{e}^{(T)}[k] \end{aligned} \tag{15}$$

in which case the greedy algorithm on triple preference still leads to the optimum in our setting (Please refer to Appendix B for details).

3.2 Incremental Updating

An advantage of our model is that the user's preference on entities and relations at time T can be incrementally updated from the previous timestamp $T-1$. Denoting $\boldsymbol{q}^{(T)} = \sum_{i \in Q^{(T)} \setminus Q^{(T-1)}} \boldsymbol{q}_i$, we derive the incremental updating equations for $\boldsymbol{q}_{\text{total}}^{(T)}$, $\boldsymbol{e}^{(T)}$, and $\boldsymbol{r}^{(T)}$ as follows.

$$\boldsymbol{q}_{\text{total}}^{(T)} = \sum_{t=0}^{T} \gamma^{T-t} \boldsymbol{q}^{(t)} = \gamma \sum_{t=0}^{T-1} \gamma^{T-1-t} \boldsymbol{q}^{(t)} + \boldsymbol{q}^{(T)} = \gamma \boldsymbol{q}_{\text{total}}^{(T-1)} + \boldsymbol{q}^{(T)} \tag{16}$$

$$\begin{aligned} \boldsymbol{e}^{(T)} &= \sum_{l=0}^{d} \alpha^l \boldsymbol{A}^l \boldsymbol{q}_{\text{total}}^{(T)} = \sum_{l=0}^{d} \alpha^l \boldsymbol{A}^l \gamma \boldsymbol{q}_{\text{total}}^{(T-1)} + \sum_{l=0}^{d} \alpha^l \boldsymbol{A}^l \boldsymbol{q}^{(T)} \\ &= \gamma \boldsymbol{e}^{(T-1)} + \sum_{l=0}^{d} \alpha^l \boldsymbol{A}^l \boldsymbol{q}^{(T)} \end{aligned} \tag{17}$$

$$\boldsymbol{r}^{(T)} = \sum_{t=0}^{T} \gamma^{T-t} \boldsymbol{q}_r^{(t)} = \gamma \sum_{t=0}^{T-1} \gamma^{T-1-t} \boldsymbol{q}_r^{(t)} + \boldsymbol{q}_r^{(T)} = \gamma \boldsymbol{r}^{(T-1)} + \boldsymbol{q}_r^{(T)} \tag{18}$$

To model the user's interest in triples, we define $\boldsymbol{H}$ as a sparse 3-dimensional temporal array, implemented using a dictionary,

$$\boldsymbol{H}^{(T)}[i][j][k] = \boldsymbol{e}^{(T)}[i]\boldsymbol{r}^{(T)}[j]\boldsymbol{e}^{(T)}[k] \tag{19}$$

The updating of $\boldsymbol{H}^{(T)}$ from $\boldsymbol{H}^{(T-1)}$ is per-entry conditional. Assume at timestamp T, compared to timestamp $T-1$, if entries i, k in $\boldsymbol{e}^{(T)}$ and entry j in $\boldsymbol{r}^{(T)}$ are not updated (excluding decay), then $\boldsymbol{H}^{(T)}[i][j][k] = \gamma^3 \boldsymbol{H}^{(T-1)}[i][j][k]$. Only entries in $\boldsymbol{H}^{(T)}[i][j][k]$ with any of those being updated need to be recalculated. The number of updated entries is small as we assign d much less than the diameter of $\mathcal{G}$. Therefore, for each timestamp, we multiply $\boldsymbol{H}$ by γ^3 and then do a small-scale update.

3.3 Incremental Sorting

After the user's preference is updated in real-time, we can directly sort the triples by their preferences and then pick the ones with the highest heat until the size budget is reached. However, the sorting algorithms usually cost $O(n \log_2 n)$ complexity [25].

For every new timestamp, excluding decay (which does not change the order), only part of the triples' heat will be updated and recalculated. This indicates that it is possible to reuse the previous order and accelerate the process to get the up-to-date order. For the following problem, we propose an intuitive solution named incremental binary insertion sort, as shown in Algorithm 3.

Problem. *Incremental Sorting*

Input: *(i) a sorted sortable instance $\mathcal{S}$, (ii) a set of changes C, where each change is expressed as a tuple (from, to). For element deleting, the tuple will be (from, None); for element adding, the tuple will be (None, to).*

Output: *sorted $\mathcal{S}$ after applying all changes in C.*

We prove the optimality of our incremental binary insertion sort algorithm by applying it to a simpler pure-adding case where all tuples in C are (None, to).

Theorem 3.1 (Optimality of Incremental Binary Insertion Sort). *Given a sorted instance $\mathcal{S}$ and unsorted instance C with $|\mathcal{S}| =$*

n and $|C| = k$. When $k << n$, any deterministic comparison-based sorting algorithm must perform $\Omega(k \log_2 n)$ new comparisons to sort $\mathcal{S} \cup C$ in the worst case. Incremental Binary Insertion Sort achieves this optimal number of comparisons. (Proof in Appendix E.1)

By applying our incremental sorting algorithm when updating heat, the cost to choose triples with the highest heat becomes $\Omega(klog_2 n)$ instead of $\Omega(nlog_2 n)$ with $k << n$ at most timestamps.

4 Algorithms

4.1 APEX^2 Framework

Our APEX^2 combines user preference modeling, incremental heat updating and incremental sorting as both solutions and optimizations. As shown in Algorithm 4, in Steps 1–7, APEX^2 first initializes the data structures for both heat updating and incremental sorting, then performs pre-computing. Then, in Steps 8–14, for each later timestamp a user inputs new queries, APEX^2 performs macroscopic decaying, incrementally updating and necessary recalculating. After that, the heat of triples gets incrementally sorted and a new PKG is constructed by picking the ones with the highest heat.

The adaptability of APEX^2 is ensured by the decaying operations, and we prove the effectiveness of APEX^2 in Theorem 4.1. As for efficiency, intuitively, though the whole KG is available, APEX^2 only accesses the entities, relations and triples of the user's interest. Moreover, when the heat of a triple is decayed to a small enough value, APEX^2 would switch it to zero and such an out-of-interest triple does not require any further computational resource. These facts show that APEX^2 is highly scalable for large databases. We prove that **the incremental time complexity of APEX^2 is unrelated to the size of KG** in Theorem 4.2. Here, **incremental time complexity** means "time complexity per adapting phase", which is the total time in the adapting phase amortized by the number of for-loop iterations in Steps 8-14. In the following theorems, **connectivity** of an area $\mathcal{V} = (\mathcal{E}_v, \mathcal{R}_v, \mathcal{T}_v)$ is defined as $\frac{\sum_{v \in \mathcal{E}_v} degree(v)}{|\mathcal{E}_v|}$.

Theorem 4.1 (Effectiveness of APEX^2). *Assume two areas (topics) $\mathcal{U}$ and $\mathcal{V}$ with connectivity c_u and c_v are sub-KGs of $\mathcal{G}$, and the user initially queries $\mathcal{U}$ for a times and starts to query $\mathcal{V}$, then APEX^2 takes $\log_\gamma \frac{1}{\frac{A}{B}(1-\gamma^a)+1}$ queries to adapt from $\mathcal{U}$ to $\mathcal{V}$, where $A = (\frac{1-(\alpha c_u)^{d+1}}{1-\alpha c_u})^{\frac{|\mathcal{E}_u|+2|\mathcal{T}_u|}{|\mathcal{E}_u|+3|\mathcal{T}_u|}}$ and $B = (\frac{1-(\alpha c_v)^{d+1}}{1-\alpha c_v})^{\frac{|\mathcal{E}_v|+2|\mathcal{T}_v|}{|\mathcal{E}_v|+2|\mathcal{T}_v|}}$. (Proof in Appendix E.6)*

For each query $q = (e, r, \mathcal{A})$, we can decompose it into a set of sub-queries $q_{sub} = \{(e, r, \{a\}) \forall a \in \mathcal{A}\}$. Similarly, we can decompose a query log Q into a sub-query log. Theorem 4.2 shows the time complexity of APEX^2 is only related to the connectivity of KG and the number of sub-queries the user performed.

Theorem 4.2 (Time Complexity of APEX^2). *The incremental time complexity for APEX^2's adapting phase is $O(c \cdot |\boldsymbol{Q}|^2 \cdot \log_2(c \cdot |\boldsymbol{Q}|))$, where Q is the query log decomposed into sub-queries (each query in Q has only one query entity, one query relation and one answer). $c = \frac{nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|}$, where nnz is the operator outputting the number of non-zero elements in a matrix, $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix of $\mathcal{G}$, $\mathcal{E}$ is the set of entities of $\mathcal{G}$. (Proof in Appendix E.2)*

If set a threshold ϵ_{ths} to eliminate small-enough entries to 0, then after $log_\gamma \epsilon_{ths}$ timestamps, entries introduced by previous queries will be decayed to 0. In this case, the effective number of queries $|\boldsymbol{Q}|$ above can be bounded by a constant $log_\gamma \epsilon_{ths}$. By this operation, **the incremental time complexity is further optimized to** $O(c \cdot \log_2(c))$, where $c = \frac{nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|}$ is the average number of neighbors within d-hops. In other words, the time complexity of APEX^2 updating is only related to the connectivity of KG.

4.2 APEX^2 Variant: APEX^2-N

In APEX^2, we model the user's interest in triples in Eq. 14, where we assign equal weights to entities and relations. However, the user may put more attention on entities than relations when performing queries. For example, a user who searched "what pieces of music did Taylor Swift create" might be more likely to search "what's Taylor Swift's music style" than "what pieces of music did Bill Evens create" later on. In this case, we propose APEX^2-N, a variant of APEX^2 that gives higher weights to entities than relations.

APEX^2-N only incrementally tracks and sorts the heat of entities but not for relations. In other words, APEX^2-N gives weight 1 to entities and 0 to relations. APEX^2-N is designed mainly for adaptive solutions of PKG summarization, and we leave the trade-off between weights on entities and relations to future work. Since APEX^2-N is a variation of APEX^2, we summarize the detailed operations of APEX^2-N in Algorithm 5 in the Appendix. We also give proof of the effectiveness and efficiency of APEX^2-N as follows.

Theorem 4.3 (Effectiveness^2 of APEX-N). *Assume two areas (topics) $\mathcal{U}$ and $\mathcal{V}$ with connectivity c_u and c_v are sub-KGs of $\mathcal{G}$. If a user initially queries $\mathcal{U}$ for a times and starts to query $\mathcal{V}$, then APEX^2-N takes $\log_\gamma \frac{1}{\frac{A}{B}(1-\gamma^a)+1}$ queries to adapt from $\mathcal{U}$ to $\mathcal{V}$, where $A = \frac{1-(\alpha c_u)^{d+1}}{1-\alpha c_u}$ and $B = \frac{1-(\alpha c_v)^{d+1}}{1-\alpha c_v}$. (Proof in Appendix E.7)*

Theorem 4.4 (Time Complexity of APEX^2-N). *The incremental time complexity for APEX^2-N's adapting phase is $O(c \log_2(c|\boldsymbol{Q}|))$, where Q is the query log decomposed into sub-queries (each query in Q has only one query entity, one query relation and one answer). And $c = \frac{nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|}$, where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is the adjacency matrix of $\mathcal{G}$, $\mathcal{E}$ is the set of entity of $\mathcal{G}$, and nnz is the operator outputting the number of non-zero elements in a matrix. (Proof in Appendix E.3)*

Like APEX^2, **the time complexity of APEX^2-N can also be optimized to** $O(c \log_2 c)$ **by setting a threshold value**. In the future, Both APEX^2 and APEX^2-N may be extended to the fully dynamic setting, where the KG itself can evolve. New entities can be reserved as dummy nodes. When a new entity, relation, or triple is added, the initial heat is zero, therefore we only need to update the adjacency matrix and start tracking their heat from the next timestamp. When a triple is deleted, we clear its heat to zero. When an entity or relation is removed, it means there is no triple with that entity, and we can safely clear its heat to zero.

5 Experiments

5.1 Experimental Setup

5.1.1 Datasets. We use YAGO 3 [57], DBPedia 3.5.1 [2], MetaQA [69] and Freebase [5] as knowledge graph datasets in our experiments. The basic information of knowledge graphs is summarized in Table 2. For YAGO, DBPedia and Freebase, we use synthetic queries

that follow the logic and structure of Linked SPARQL Queries' DBpedia data dump[2]. The same format has been used in previous work [55]. For MetaQA, we use the queries provided in the dataset. More details about the datasets can be found in Appendix C.2, and we validate the high quality of our synthetic queries in Appendix C.3.

Table 2: Statistics of Knowledge Graphs

KG	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	$\|\mathcal{T}\|$
YAGO	4,267,316	38	12,403,275
DBPedia	4,616,347	1,043	10,974,936
MetaQA	43,234	13	231,103
Freebase	14,541	237	310,116

5.1.2 Baselines. We choose several state-of-the-art methods as the baselines. We compare sampling-based knowledge graph summarization algorithm (GLIMPSE [55]), merging-based graph summarization algorithm (PEGASUS [27]), workload-based knowledge graph summarization algorithm (iSummary [61]), random walk with restart on knowledge graph (Personalized PageRank [34]), together with our APEX^2 and APEX^2-N. Details of GLIMPSE, PEGASUS and iSummary are provided in section B. The PPR baseline calculates the PageRank vector personalized to $\boldsymbol{q}_{\text{total}}$ and constructs the summarization by continuously adding the most relevant entity.

5.1.3 Re-summarization Interval. By design, the baselines cannot take temporal query logs as inputs. To enable the baselines to handle adaptive PKG summarization problem, we let them output new PKGs after a certain amount R of timestamps. The choice of R affects the performances of baseline methods. If R is small (i.e., $R = 1$ means re-summarize every timestamp), then the re-summarization happens frequently, and the baselines will become very slow. If R is large, then the baseline summaries are outdated for most timestamps. To pick a good R for fair comparisons, we conduct pre-experiments in Appendix C.5 and find that $R = 9$ is a good effectiveness-efficiency trade-off for baselines.

In our design, APEX^2 and APEX^2-N can also take multiple queries at one time by masking the summary updating phase (line 13-14 in Algorithm 4 and line 13-17 in Algorithm 5) for non-summary timestamps. We use R_{APEX} to denote that they update the PKG every R_{APEX} timestamp. By default, $R_{APEX} = 1$, and we provide a comprehensive study on R_{APEX} in Section 5.6.

5.1.4 Metrics. Same as previous research works [8, 55, 58], we use F1 score [20] on the very next query as the metric for searching effectiveness. More details of this can be found in Appendix C.1.

5.2 Experimental Settings

We show the outperformance of APEX^2 and APEX^2-N through auto-regressive[3] style experiments. We set the default hyperparameters $\gamma = 0.5$, $\alpha = 0.3$, $d = 1$ and PageRank restart probability to be 0.85. We set the compression ratio to be $0.000001 = 0.0001\%$ (one in a million) for YAGO and DBPedia, $0.0001 = 0.01\%$ (one in ten thousand) for MetaQA, $0.0005 = 0.05\%$ for Freebase.

Generate user queries. To calculate the average and standard deviation, we simulate 10 users to query the KGs. Following the

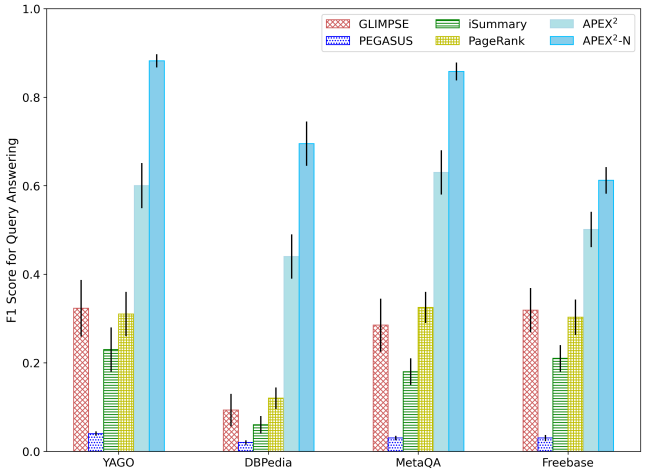


Figure 2: Effectiveness Comparison Under Querying Scenario

norm set by Freebase discussed in section 3.1, we model the abstract concept of topic by "queries with the same query entity"[4]. To simulate a real-world querying scenario with interest shift, for each KG, we generate 200 queries on 20 topics for each user. Each group of 10 consecutive queries are in the same topic. We associate each query t with timestamp $t - 1$. For MetaQA, we first categorize the provided queries into different topics by query entity, then randomly sample 20 distinct query entities. After that, we randomly choose 10 queries on each of the 20 topics. For DBPedia, YAGO and Freebase, we synthetically generate queries by randomly choosing 20 query entities in the KG. Then for each query, we randomly choose 10 relations (with possible multiplicity) that the query entity has. Finally, we include all entities e satisfying (query entity, chosen relation, e) $\in \mathcal{T}$ into the answer set. We study 1-hop simple queries with known answers because in real life there is only a small portion of complex queries [6], which can be decomposed into simple queries.

Query Answering Evaluation. After loading KG and the queries of a user, we adaptively summarize the KG. (i) For GLIMPSE, PEGASUS, iSummary, PageRank, we construct an initial summary using the first query, then re-summarize after each $R = 9$ timestamps (i.e., queries). (ii) For APEX^2 and APEX^2-N, they both evolve every timestamp whenever the user performs a new query. We calculate the F1 score of the very next query after re-summarization for all the methods. For example, if at timestamp t the summarization method performs re-summarization or evolving using query $t + 1$, then at timestamp $t + 1$ we search the query $t + 2$ in the new PKG and calculate the F1 score.

5.3 Comparisons

Effectiveness Comparison. We measure the F1 score of each PKG summarization method in the auto-regressive querying scenario for 10 users. We report the mean and standard deviation of sampled timestamps in Figure 2. First, both APEX^2 and APEX^2-N outperform the existing baseline methods on the F1 score in all cases. Second, APEX^2 and APEX^2-N remain highly effective even if the knowledge

[2]https://files.dice-research.org/archive/lsqv2/dumps/dbpedia/

[3]Term borrowed from statistics. We feed PKG adaption methods the very next query for testing, then the same query is used for training. Iterate until all queries are used.

[4]In fact, if a topic is defined in terms of entities, then the ability to adapt entity shift is a sufficient condition for adapting topic shift.

graph is large and the compression rate is extremely small, showing the scalability of our methods. Third, APEX^2-N outperforms all methods, showing the necessity of considering users' attention more on entities than relations. The baseline PEGASUS performs worst, possibly because this method is not designed specifically for knowledge graphs. This provides evidence that traditional graph algorithms should be reconsidered before being applied to the KG domain. In later comparisons, we only compare with GLIMPSE and PageRank baselines because they have acceptable sub-optimal performances.

Efficiency Comparison. We measure the time consumption of PKG summarization methods to produce one summary under the $R = R_{APEX} = 1$ setting. We report the mean and standard deviation in Table 3. PEGASUS and iSummary are not coded in Python. We add an extremely strong baseline in terms of efficiency: the parallel implementation of PageRank "ParallelPR" with walk-length 1 and record its execution time, which is almost the optimal time cost that any diffusion-based algorithm can achieve.

Table 3: Efficiency Comparison (↓) (unit: seconds)

Methods	YAGO	DBPedia	MetaQA	Freebase
GLIMPSE	192.1±27.92	148.4±114.8	1.366±0.089	1.581±0.093
PageRank	22.81±259.7	2.615±0.136	0.032±0.003	0.144±0.011
ParallelPR	1.947±2.061	1.442±0.031	0.016±0.002	0.019±0.002
APEX^2	6.354±5.388	4.655±1.108	0.055±0.035	0.112±0.048
APEX-N^2	2.528±0.502	3.305±0.041	0.018±0.002	0.024±0.003

Our experiments show that both APEX^2 and APEX^2-N are much faster than re-running GLIMPSE for adaptive PKG summarization. Respectively, APEX^2 and APEX^2-N outperform GLIMPSE by **20×** to **30×** and **40×** to **75×**. If we regard ParallelPR as the optimal time complexity, then compared to GLIMPSE, APEX^2 and APEX^2-N get **30×** to **45×** and **80×** to **400×** closer to the optimal. Overall, APEX^2 and APEX^2-N have similar efficiency performance to PageRank (walk length 1), which is a very strong baseline in time complexity. Furthermore, our APEX^2 and APEX^2-N have efficiency close to ParallelPR. Moreover, our experiments show that APEX^2 and APEX^2-N are more scalable because they outperform all the baseline methods in the largest YAGO dataset. Considering APEX^2-N and APEX^2 together, APEX^2-N has better experimental efficiency because it does not consider the user's interest in relationships. This means APEX^2-N could be a good choice for entity interest tracking and suits tasks where the interest in triples is not very necessary.

5.4 Ablation Study

5.4.1 Decay Ablation. In our design, decaying (i.e., a forgetting mechanism), is the key to the adaptive and extreme summarization. Here, we compare the effectiveness under different levels of decaying to further show the importance of decaying. We run the user querying scenario in MetaQA with varying γ values. Other hyperparameters are set to be the same as Section 5.2. We use the same query generated for MetaQA and report both APEX^2 and APEX^2-N's average F1 score of 10 users in Figure 3. Larger γ means lower decay level (less extent of decay). When $\gamma = 1$, the decaying is completely eliminated, and the PKG is full of outdated interests, resulting in low F1 scores of both APEX^2 and APEX^2-N. In general, starting from $\gamma = 0.5$, the effectiveness does not change much until $\gamma = 0.9$, and then the performance gets worse massively from $\gamma = 0.9$ to $\gamma = 1$. It turns out that when the storage space is very limited, decaying the previous interests can pave the way for personalized summarization.

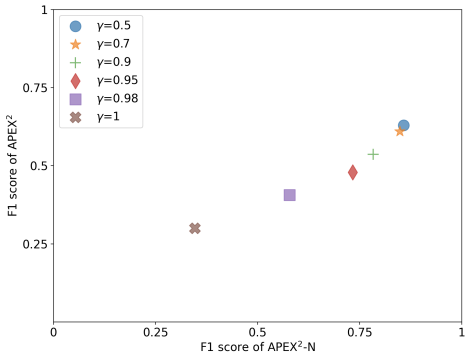


Figure 3: Querying Effectiveness in Multiple Decay Levels

5.4.2 Component Ablation. Aiming to accelerate the computation, after the necessary dynamic modeling of user interests, we designed incremental updating and incremental sorting. These two components serve to accelerate the computation and do not affect the computational results; therefore, to show that all three components of the APEX2^2 framework contribute to the overall efficiency performance, we can compare the mean execution time in seconds shown as follows.

Table 4: APEX^2 Ablations' Efficiency (↓) in seconds

Ablations	YAGO	DBPedia	MetaQA	Freebase
APEX^2-complete	6.354	4.655	0.055	0.110
APEX^2-without-inc-updating	124.155	91.679	0.845	0.912
APEX^2-without-inc-sorting	18.643	15.212	0.124	0.162
APEX^2-dynamic-model-only	167.654	125.687	1.287	1.421

Here, APEX^2-complete is the complete version of APEX^2; APEX^2-without-inc-updating removes incremental updating; APEX^2-without-inc-sorting replaces incremental sorting with normal re-sorting; APEX^2-dynamic-model-only removes both. From the results, all the components contributes to the acceleration, and incremental updating plays a crucial role in design.

5.5 Hyperparameter Study

We study the parameter sensitivity to show our models' robustness using MetaQA dataset. From the result in Figure 4, larger compression ratio leads to better F1 score, but after a certain value F1 score does not change much. This is intuitive as the searching accuracy increases with more triples stored in the PKG. In general, our methods are robust with damping factor and diffusing parameter. A larger diffusing diameter takes more time because more items get non-zero heat. Time per adapting phase increases quadratically with diffusing parameter, because the interested area grows with d.

5.6 Handling Multiple Queries at One Time

As mentioned in Section 5.1.3, APEX^2 and APEX^2-N can re-summarize the KG every R_{APEX} timestamp to increase overall efficiency. We

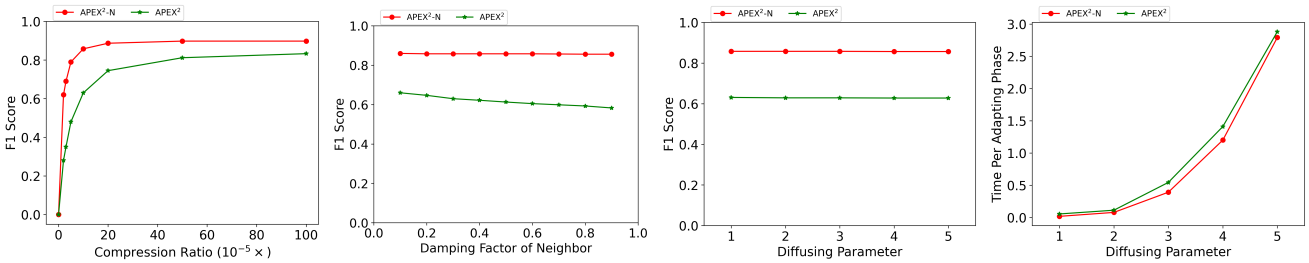


Figure 4: Parameter Study. From left to right: compression ratio κ, damping factor of neighbor α, diffusing diameter d

conduct comprehensive experimental analysis on varying R_{APEX} and report the results in Appendix C.6 due to page limitation. In short, on the effectiveness side, our methods, with varying R_{APEX} of 2, 3, 6, achieved competitive query accuracy (i.e., F1 score) and still outperformed baseline methods. On the efficiency side, the time consumed by the varying R_{APEX} methods is similar to that of $R = 1$. These results suggest that our heat tracking method has a robust performance over timestamps.

5.7 Case Study

We provide a comparative case study to illustrate how our methods work and that they can summarize high-quality PKGs.

5.7.1 Dataset and Query. We do a case study on MetaQA dataset using the queries of user 0 to show in which way our methods are able to adaptively summarize highly interested items into PKG. We use the first 33 queries from the user. Most of the interested part of the KG is shown in Figure 11. In the first 3 groups of 10 queries, the query entities are respectively the actor "Stevan Riley", the movie "The Disappearance of Haruhi Suzumiya" and the movie "LOL".

5.7.2 Settings. We aim to study how APEX^2, APEX^2-N, GLIMPSE and PageRank deal with topic shift. Starting from the 31^{st} query, the user asks (Chad Michael Murray, movie_to_actor, ?), i.e., "which movies did Chad Michael Murray act in" three times. The correct answer entities are "A Cinderella Story", "House of Wax", and "Left Behind". For APEX^2 and APEX^2-N, we use the **same** setting as in the main experiments: both of them evolve every timestamp from the beginning. For GLIMPSE and PageRank, we give them **higher** privileges that they can re-summarize **each new timestamp**, compared to per 9 timestamps in the main experiments. We use the same hyperparameters as the main experiments.

5.7.3 Results. The PKG results are shown in Figure 12, 13, 14, 15. The summarized PKG at timestamp $t-1$ have been fed query t. **(i)** The sub-figure (a) (i.e., the PKGs after the first three groups of queries) of GLIMPSE and PageRank still contains all information about the out-of-interest topic "Blue Blood" and "Stevan Riley", which were accessed in the first 10 queries. **(ii)** For both APEX^2 and APEX^2-N, after the first query on the new topic, the three answer triples (Chad Michael Murray, movie_to_actor, A Cinderella Story), (Chad Michael Murray, movie_to_actor, House of Wax), and (Chad Michael Murray, movie_to_actor, Left Behind) get into the PKG. However, after three times querying on the new topic, the PKG summarized by GLIMPSE only contains (Chad Michael Murray, movie_to_actor, House of Wax), and the PKG summarized by PageRank does not contain any of them. **(iii)** From APEX^2 and APEX^2-N (b) to (d), as topic on "Chad Michael Murray" is queried again and again, the connected group centered at "Chad Michael Murray" grows larger. At timestamp 32 (Figure d), some of its 2-hop neighbors are included, such as (House of Wax, actor_to_movie, Nicolas Cage) in APEX^2 and (A Cinderella Story, director_to_movie, Mark Rosman) in APEX^2-N. The reason why the actor relationship is included first is that, this relation is recently queried many times and has a high relational interest. **(iv)** There are some triples that are not very related to the user queries but are summarized in the GLIMPSE PKG, for example (Onibi, movie_to_language, Japanese), (john lithgow, tag_to_movie, 2010) and (Hercules, movie_to_genre, Animation). The PageRank PKG almost remains the same given three queries on the new topic. Compared to these, APEX^2 and APEX^2-N produce PKGs that have intuitively higher quality in terms of user's interest.

6 Related Work

Graph compression or graph summarization has been a popular research topic in recent years. In 2018, Liu et al. wrote a survey [43] and provided a taxonomy for graph summarization algorithms: static plain graph summarization [30, 32, 67], static labeled graph summarization [4, 55] and dynamic plain graph summarization [29, 56, 59]. Different techniques and metrics have been adopted for graph summarization. Kleindessner et al. [28] proposes using k-center clustering for fair data summarization; SSumM [32] greedily merges supernodes to minimize the reconstruction error; MoSSo [29] approximates the optimal utility by random search; GraphZIP [54] focus on effective summarization for clique structures and compress the graph by decomposing it into a set of cliques; NETCONDENSE [1] shrink the temporal networks by propagating and merging the unimportant node and time-pairs; It is worth noting that macroscopically deep-learning-based methods may not always be a good choice for large-scale graph summarization purposes. This is because a graph-scale embedding requires much larger space than the graph itself, which is already massive and needs to be summarized. However, partial embeddings might still be eligible. Adaptive summarization could be useful in many cases where the computational resource is limited, but the original graph is large. One example topic is Neural Graph Databases, which are considered the next step in the evolution of graph databases [52, 60]. Neural Graph Databases are powered by neural query embedding methods, which take large space complexity to be applied on the whole input graph [47]. Adaptive summarization has the potential to fill this gap by adaptively summarizing the whole input graph into a domain-specific partial graph personalized to the user.

7 Conclusion

In this paper, we propose APEX^2, the first framework for adaptive personalized knowledge graph summarization. We prove the adapting effectiveness, time complexity of APEX^2 and its variant APEX^2-N. Then we find that by adopting APEX^2 in a real-world real-time knowledge graph summarization scenario, much of the storage space can be saved while maintaining high searching effectiveness. We design extensive experiments to show the superiority of APEX^2 over baseline methods.

8 Limitations

Our PKG summarization technique is particularly beneficial in scenarios where (1) users anticipate challenges in communicating with the central server and loading the entire KG, (2) are concerned about their future query privacy and prefer not to send queries directly to the server, and (3) expect a large volume of future queries and want to reduce query response times. In cases where none of these conditions apply, there may be no significant advantage to summarizing a PKG rather than querying the KG directly. For example, if a user queries the KG infrequently, direct querying might be a better choice since querying KG directly takes less time than summarizing it. For instance, querying entire YAGO directly takes approximately 1 second, although querying summarized YAGO costs < 0.0001 second with competitive effectiveness, generating a summary takes around 6 seconds. Making the summarization for infrequent user is interesting and challenging, we would like to explore it in the future work.

9 Broader Impact

In the past decades, graphs and knowledge graphs have been serving various real-world applications, from ranking [22], social network analysis [17, 65, 68, 75], transportation [76], anomaly detection [44, 45, 64, 66, 70, 73], community detection [35, 36] and recommendation [3, 51], to molecular biology [16, 37, 74] and climate sciences [18, 31]. Stepping into the era of large and foundation models [15, 71], graphs have been leveraged in Retrieval Augmented Generation (RAG) [10, 19, 26], which enhances the retrieval and integration of relevant context for improving the quality and relevance of generated responses, and knowledge graph personalization still holds immense potential for enabling more accurate, context-aware, and adaptive decision-making by integrating domain-specific knowledge and leveraging the scalability and representation power of these advanced models. Notably, recent works on personalization [7, 24, 63] and compression [72, 77] of Large Language Models (LLMs) underscore the significance of enhancing adaptability, efficiency, and user-specific customization to better meet diverse application needs.

References

[1] Bijaya Adhikari, Yao Zhang, Sorour E. Amiri, Aditya Bharadwaj, and B. Aditya Prakash. 2018. Propagation-Based Temporal Network Summarization. *IEEE Trans. Knowl. Data Eng.* (2018).

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*

[3] Yikun Ban, Jiaru Zou, Zihao Li, Yunzhe Qi, Dongqi Fu, Jian Kang, Hanghang Tong, and Jingrui He. 2024. PageRank Bandits for Link Prediction. *CoRR* abs/2411.01410 (2024). https://doi.org/10.48550/ARXIV.2411.01410 arXiv:2411.01410

[4] Caleb Belth, Xinyi Zheng, Jilles Vreeken, and Danai Koutra. 2020. What is Normal, What is Strange, and What is Missing in a Knowledge Graph: Unified Characterization via Inductive Summarization. In *WWW 2020.*

[5] Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, Jason Tsong-Li Wang (Ed.).

[6] Angela Bonifati, Wim Martens, and Thomas Timm. 2020. An analytical study of large SPARQL query logs. *VLDB J.* 29, 2-3 (2020), 655–679. https://doi.org/10.1007/s00778-019-00558-9

[7] Marco Braga. 2024. Personalized Large Language Models through Parameter Efficient Fine-Tuning Techniques. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, Grace Hui Yang, Hongning Wang, Sam Han, Claudia Hauff, Guido Zuccon, and Yi Zhang (Eds.). ACM, 3076. https://doi.org/10.1145/3626772.3657657

[8] Qingyu Chen, Alexis Allot, and Zhiyong Lu. 2021. LitCovid: an open database of COVID-19 literature. *Nucleic Acids Res.* (2021).

[9] George Cybenko. 1989. Dynamic Load Balancing for Distributed Memory Multiprocessors. *J. Parallel Distributed Comput.* (1989).

[10] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. *CoRR* abs/2404.16130 (2024). https://doi.org/10.48550/ARXIV.2404.16130 arXiv:2404.16130

[11] Patrick Ernst, Cynthia Meng, Amy Siu, and Gerhard Weikum. 2014. Knowlife: a knowledge graph for health and life sciences. In *2014 IEEE 30th International Conference on Data Engineering.*

[12] Lukas Faber, Tara Safavi, Davide Mottin, Emmanuel Müller, and Danai Koutra. 2018. Adaptive personalized knowledge graph summarization. In *MLG Workshop (with KDD).*

[13] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. 2018. Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO. *Semantic Web* (2018).

[14] William Feller. 1991. *An introduction to probability theory and its applications, Volume 2.* Vol. 81. John Wiley & Sons.

[15] Dongqi Fu, Liri Fang, Zihao Li, Hanghang Tong, Vetle I. Torvik, and Jingrui He. 2024. Parametric Graph Representations in the Era of Foundation Models: A Survey and Position. *CoRR* abs/2410.12126 (2024). https://doi.org/10.48550/ARXIV.2410.12126 arXiv:2410.12126

[16] Dongqi Fu, Liri Fang, Ross Maciejewski, Vetle I. Torvik, and Jingrui He. 2022. Meta-Learned Metrics over Multi-Evolution Temporal Graphs. In *KDD.*

[17] Dongqi Fu, Dawei Zhou, Ross Maciejewski, Arie Croitoru, Marcus Boyd, and Jingrui He. 2023. Fairness-Aware Clique-Preserving Spectral Clustering of Temporal Graphs. In *WWW.*

[18] Dongqi Fu, Yada Zhu, Hanghang Tong, Kommy Weldemariam, Onkar Bhardwaj, and Jingrui He. 2024. Generating Fine-Grained Causality in Climate Time Series Data for Forecasting and Anomaly Detection. *CoRR* abs/2408.04254 (2024). https://doi.org/10.48550/ARXIV.2408.04254 arXiv:2408.04254

[19] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. *CoRR* abs/2312.10997 (2023). https://doi.org/10.48550/ARXIV.2312.10997 arXiv:2312.10997

[20] Cyril Goutte and Éric Gaussier. 2005. A Probabilistic Interpretation of Precision, Recall and *F*-Score, with Implication for Evaluation. In *Advances in Information Retrieval, 27th European Conference on IR Research, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005, Proceedings.*

[21] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016.*

[22] Jingrui He, Hanghang Tong, Qiaozhu Mei, and Boleslaw K. Szymanski. 2012. GenDeR: A Generic Diversified Ranking Algorithm. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger (Eds.). 1151–1159. https://proceedings.neurips.cc/paper/2012/hash/7f24d240521d99071c93af3917215ef7-Abstract.html

[23] Qi He, Jaewon Yang, and Baoxu Shi. 2020. Constructing knowledge graph for social networks in a deep and holistic way. In *Companion Proceedings of the Web Conference 2020.*

[24] Xinrui He, Yikun Ban, Jiaru Zou, Tianxin Wei, Curtiss B Cook, and Jingrui He. 2024. LLM-Forest for Health Tabular Data Imputation. *arXiv preprint arXiv:2410.21520* (2024).

[25] C. A. R. Hoare. 1961. Algorithm 64: Quicksort. *Commun. ACM* 4, 7 (1961), 321. https://doi.org/10.1145/366622.366644

[26] Bowen Jin, Jinsung Yoon, Jiawei Han, and Sercan Ö. Arik. 2024. Long-Context LLMs Meet RAG: Overcoming Challenges for Long Inputs in RAG. *CoRR* abs/2410.05983 (2024). https://doi.org/10.48550/ARXIV.2410.05983 arXiv:2410.05983

[27] Shinhwan Kang, Kyuhan Lee, and Kijung Shin. 2022. Personalized Graph Summarization: Formulation, Scalable Algorithms, and Applications. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022.*

[28] Matthäus Kleindessner, Pranjal Awasthi, and Jamie Morgenstern. 2019. Fair k-Center Clustering for Data Summarization. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA.*

[29] Jihoon Ko, Yunbum Kook, and Kijung Shin. 2020. Incremental Lossless Graph Summarization. In *KDD 2020.*

[30] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. 2014. VOG: Summarizing and Understanding Large Graphs. In *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014.*

[31] Rémi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Alexander Pritzel, Suman V. Ravuri, Timo Ewalds, Ferran Alet, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Jacklynn Stott, Oriol Vinyals, Shakir Mohamed, and Peter W. Battaglia. 2022. GraphCast: Learning skillful medium-range global weather forecasting. *CoRR* abs/2212.12794 (2022). https://doi.org/10.48550/ARXIV.2212.12794 arXiv:2212.12794

[32] Kyuhan Lee, Hyeonsoo Jo, Jihoon Ko, Sungsu Lim, and Kijung Shin. 2020. SSumM: Sparse Summarization of Massive Graphs. In *KDD 2020.*

[33] Zihao Li, Yuyi Ao, and Jingrui He. 2024. SpherE: Expressive and Interpretable Knowledge Graph Embedding for Set Retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, Grace Hui Yang, Hongning Wang, Sam Han, Claudia Hauff, Guido Zuccon, and Yi Zhang (Eds.). ACM, 2629–2634. https://doi.org/10.1145/3626772.3657910

[34] Zihao Li, Dongqi Fu, and Jingrui He. 2023. Everything Evolves in Personalized PageRank. In *Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023.*

[35] Zihao Li, Dongqi Fu, Hengyu Liu, and Jingrui He. 2024. Hypergraphs as Weighted Directed Self-Looped Graphs: Spectral Properties, Clustering, Cheeger Inequality. *arXiv preprint arXiv:2411.03331* (2024).

[36] Zihao Li, Dongqi Fu, Hengyu Liu, and Jingrui He. 2024. Provably Extending PageRank-based Local Clustering Algorithm to Weighted Directed Graphs with Self-Loops and to Hypergraphs. *arXiv preprint arXiv:2412.03008* (2024).

[37] Zihao Li, Lecheng Zheng, Bowen Jin, Dongqi Fu, Baoyu Jing, Yikun Ban, Jingrui He, and Jiawei Han. 2024. Can Graph Neural Networks Learn Language with Extremely Weak Text Supervision? *arXiv preprint arXiv:2412.08174* (2024).

[38] Jue Liu, Zhuocheng Lu, and Wei Du. 2019. Combining enterprise knowledge graph and news sentiment analysis for stock price prediction. (2019).

[39] Lihui Liu, Yuzhong Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. 2023. Knowledge Graph Question Answering with Ambiguous Query. In *Proceedings of the ACM Web Conference 2023.*

[40] Lihui Liu, Boxin Du, Yi Ren Fung, Heng Ji, Jiejun Xu, and Hanghang Tong. 2021. KompaRe: A Knowledge Graph Comparative Reasoning System. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Virtual Event, Singapore) *(KDD '21)*. Association for Computing Machinery, New York, NY, USA, 3308–3318. https://doi.org/10.1145/3447548.3467128

[41] Lihui Liu, Boxin Du, Jiejun Xu, Yinglong Xia, and Hanghang Tong. 2022. Joint Knowledge Graph Completion and Question Answering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Washington DC, USA) *(KDD '22)*. Association for Computing Machinery, New York, NY, USA, 1098–1108.

[42] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[43] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph Summarization Methods and Applications: A Survey. *ACM Comput. Surv.* (2018).

[44] Zhining Liu, Ruizhong Qiu, Zhichen Zeng, Hyunsik Yoo, David Zhou, Zhe Xu, Yada Zhu, Kommy Weldemariam, Jingrui He, and Hanghang Tong. 2024. Class-Imbalanced Graph Learning without Class Rebalancing. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net. https://openreview.net/forum?id=pPnkpvBeZN

[45] Zhining Liu, Ruizhong Qiu, Zhichen Zeng, Yada Zhu, Hendrik F. Hamann, and Hanghang Tong. 2024. AIM: Attributing, Interpreting, Mitigating Data Unfairness. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, Ricardo Baeza-Yates and Francesco Bonchi (Eds.). ACM, 2014–2025. https://doi.org/10.1145/3637528.3671797

[46] Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. 2008. Mining social networks using heat diffusion processes for marketing candidates selection. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*.

[47] Tharun Medini, Beidi Chen, and Anshumali Shrivastava. 2021. SOLAR: Sparse Orthogonal Learned and Random Embeddings. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.

[48] Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-Value Memory Networks for Directly Reading Documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, Jian Su, Xavier Carreras, and Kevin Duh (Eds.). The Association for Computational Linguistics, 1400–1409. https://doi.org/10.18653/v1/d16-1147

[49] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proc. IEEE* (2016).

[50] Sancheng Peng, Yongmei Zhou, Lihong Cao, Shui Yu, Jianwei Niu, and Weijia Jia. 2018. Influence analysis in social networks: A survey. *J. Netw. Comput. Appl.* (2018).

[51] Yunzhe Qi, Yikun Ban, and Jingrui He. 2023. Graph neural bandits. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1920–1931.

[52] Hongyu Ren, Mikhail Galkin, Michael Cochez, Zhaocheng Zhu, and Jure Leskovec. 2023. Neural Graph Reasoning: Complex Logical Query Answering Meets Graph Databases. *CoRR* abs/2303.14617 (2023).

[53] Mariia Rizun et al. 2019. Knowledge graph application in education: a literature review. *Acta Universitatis Lodziensis. Folia Oeconomica* (2019).

[54] Ryan A. Rossi and Rong Zhou. 2018. GraphZIP: a clique-based sparse graph compression method. *J. Big Data* (2018).

[55] Tara Safavi, Caleb Belth, Lukas Faber, Davide Mottin, Emmanuel Müller, and Danai Koutra. 2019. Personalized Knowledge Graph Summarization: From the Cloud to Your Pocket. In *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8-11, 2019*.

[56] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. 2015. TimeCrunch: Interpretable Dynamic Graph Summarization. In *KDD 2015*.

[57] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*.

[58] Michael M. Tadesse, Hongfei Lin, Bo Xu, and Liang Yang. 2019. Detection of Depression-Related Posts in Reddit Social Media Forum. *IEEE Access* (2019).

[59] Nan Tang, Qing Chen, and Prasenjit Mitra. 2016. Graph Stream Summarization: From Big Bang to Big Crunch. In *SIGMOD 2016*.

[60] James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Y. Levy. 2021. From Natural Language Processing to Neural Databases. *Proc. VLDB Endow.* (2021).

[61] Giannis Vassiliou, Fanouris Alevizakis, Nikolaos Papadakis, and Haridimos Kondylakis. 2023. iSummary: Workload-Based, Personalized Summaries for Knowledge Graphs. In *The Semantic Web - 20th International Conference, ESWC 2023, Hersonissos, Crete, Greece, May 28 - June 1, 2023, Proceedings (Lecture Notes in Computer Science, Vol. 13870)*, Catia Pesquita, Ernesto Jiménez-Ruiz, Jamie P. McCusker, Daniel Faria, Mauro Dragoni, Anastasia Dimou, Raphaël Troncy, and Sven Hertling (Eds.). Springer, 192–208. https://doi.org/10.1007/978-3-031-33455-9_12

[62] Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* (2014).

[63] Stanislaw Wozniak, Bartlomiej Koptyra, Arkadiusz Janz, Przemyslaw Kazienko, and Jan Kocon. 2024. Personalized Large Language Models. *CoRR* abs/2402.09269 (2024). https://doi.org/10.48550/ARXIV.2402.09269 arXiv:2402.09269

[64] Ziwei Wu, Lecheng Zheng, Yuancheng Yu, Ruizhong Qiu, John R. Birge, and Jingrui He. 2024. Fair Anomaly Detection For Imbalanced Groups. *CoRR* abs/2409.10951 (2024). https://doi.org/10.48550/ARXIV.2409.10951 arXiv:2409.10951

[65] Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. 2021. Dynamic knowledge graph alignment. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4564–4572.

[66] Ban Yikun, Liu Xin, Huang Ling, Duan Yitao, Liu Xue, and Xu Wei. 2019. No place to hide: Catching fraudulent entities in tensors. In *The World Wide Web Conference*. 83–93.

[67] Quinton Yong, Mahdi Hajiabadi, Venkatesh Srinivasan, and Alex Thomo. 2021. Efficient Graph Summarization using Weighted LSH at Billion-Scale. In *SIGMOD 2021*.

[68] Zhichen Zeng, Boxin Du, Si Zhang, Yinglong Xia, Zhining Liu, and Hanghang Tong. 2024. Hierarchical Multi-Marginal Optimal Transport for Network Alignment. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (Eds.). AAAI Press, 16660–16668. https://doi.org/10.1609/AAAI.V38I15.29605

[69] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. 2018. Variational Reasoning for Question Answering With Knowledge Graph. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*.

[70] Lecheng Zheng, John R. Birge, Yifang Zhang, and Jingrui He. 2024. Towards Multi-view Graph Anomaly Detection with Similarity-Guided Contrastive Clustering. *CoRR* abs/2409.09770 (2024). https://doi.org/10.48550/ARXIV.2409.09770 arXiv:2409.09770

[71] Lecheng Zheng, Baoyu Jing, Zihao Li, Hanghang Tong, and Jingrui He. 2024. Heterogeneous Contrastive Learning for Foundation Models and Beyond. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, Ricardo Baeza-Yates and Francesco Bonchi (Eds.). ACM, 6666–6676. https://doi.org/10.1145/3637528.3671454

[72] Ming Zhong, Chenxin An, Weizhu Chen, Jiawei Han, and Pengcheng He. 2024. Seeking Neural Nuggets: Knowledge Transfer in Large Language Models from a Parametric Perspective. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net. https://openreview.net/forum?id=mIEHIcHGOo

[73] Dawei Zhou, Kangyang Wang, Nan Cao, and Jingrui He. 2015. Rare Category Detection on Time-Evolving Graphs. In *2015 IEEE International Conference on Data Mining, ICDM 2015, Atlantic City, NJ, USA, November 14-17, 2015*, Charu C. Aggarwal, Zhi-Hua Zhou, Alexander Tuzhilin, Hui Xiong, and Xindong Wu (Eds.). IEEE Computer Society, 1135–1140. https://doi.org/10.1109/ICDM.2015.120

[74] Dawei Zhou, Lecheng Zheng, Dongqi Fu, Jiawei Han, and Jingrui He. 2022. MentorGNN: Deriving Curriculum for Pre-Training GNNs. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, Mohammad Al Hasan and Li Xiong (Eds.). ACM, 2721–2731.

[75] Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. 2020. A Data-Driven Graph Generative Model for Temporal Interaction Networks. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 401–411. https://doi.org/10.1145/3394486.3403082

[76] Xinwen Zhu, Zihao Li, Yuxuan Jiang, Jiazhen Xu, Jie Wang, and Xuyang Bai. 2024. Real-time Vehicle-to-Vehicle Communication Based Network Cooperative Control System through Distributed Database and Multimodal Perception: Demonstrated in Crossroads. *CoRR* abs/2410.17576 (2024). https://doi.org/10.48550/ARXIV.2410.17576 arXiv:2410.17576

[77] Jiaru Zou, Mengyu Zhou, Tao Li, Shi Han, and Dongmei Zhang. 2024. PromptIntern: Saving Inference Costs by Internalizing Recurrent Prompt during Large Language Model Fine-tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, 10288–10305. https://aclanthology.org/2024.findings-emnlp.602

[78] Xiaohan Zou. 2020. A survey on application of knowledge graph. In *Journal of Physics: Conference Series*.

A Pseudo-code of Algorithms

A.1 GLIMPSE

Algorithm 1 The GLIMPSE framework

Require: knowledge graph $\mathcal{G}$; query log $\mathcal{Q}$; size budget K
Ensure: personal summarization $\mathcal{P} \subseteq \mathcal{G}$ with $|\mathcal{T}_p| \leq K$

Compute $\mathcal{T}^{\Delta \neq 0}$ with $\Pr(e|\mathcal{Q})$, $\Pr(x_{ijk}|\mathcal{Q})$
$\mathcal{P} \leftarrow \emptyset$
while $|\mathcal{T}_p| \leq K$ **do**
  Sample a set $\mathcal{S}$ of size $\frac{|\mathcal{T}^{\Delta \neq 0}}{K} \log \frac{1}{\epsilon}$ from $T^{\Delta \neq 0}$
  Select $\widetilde{x}_{ijk} \leftarrow \arg\max_{x_{ijk} \in \mathcal{S}} \Delta_\delta(x_{ijk}|\mathcal{P}, \mathcal{Q})$
  Add triple $\widetilde{x}_{ijk} = (e_i, r_k, e_j)$ to $\mathcal{P}$
end while
return $\mathcal{P}$

A.2 PEGASUS

Algorithm 2 The PEGASUS framework

Require: input graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; size budget K; target node set $\mathcal{T}$; degree of personalization α; parameter for adaptive thresholding β; max number of iterations t_{max}
Ensure: personal summary $\mathcal{P} = (\mathcal{V}_P, \mathcal{E}_P)$ within size budget K

$\mathcal{V}_P \leftarrow \{\{u\} : u \in \mathcal{V}\}$; $\mathcal{E}_P \leftarrow \{\{\{u\}, \{v\}\} : \{u, v\} \in \mathcal{E}\}$
$t \leftarrow 1$; $\theta \leftarrow 0.5$; $\mathcal{L} \leftarrow []$
while $t \leq t_{max}$ and $Size(\mathcal{P}) > K$ **do**
  $C \leftarrow$ generate candidate groups
  for each group $C_i \in C$ **do**
    Greedily merge nodes in C_i with the threshold θ; update $\mathcal{V}_P, \mathcal{E}_P, \mathcal{L}$
  end for
  $\theta \leftarrow [\beta \times |\mathcal{L}|]$-th largest entry in $\mathcal{L}$
  $\mathcal{L} \leftarrow []$; $t \leftarrow t + 1$
end while
if $Size(\mathcal{P}) > K$ **then**
  Sparsify $\mathcal{P}$ further
end if
return $\mathcal{P}$

A.3 Incremental Binary Insertion Sort

Algorithm 3 Incremental Binary Insertion Sort

Require: previously sorted sortable instance $\mathcal{S}$, set of changes C
Ensure: sorted instance $\mathcal{S}$ that is updated as described in C

/* extract unchanged entries in $\mathcal{S}$ by deleting */
for *(from, to)* in C **do**
  if *from* is not None **then**
    pos = binary_search(*from*, $\mathcal{S}$)
    delete($\mathcal{S}$, pos)
  end if
end for
for *(from, to)* in C **do**
  if *to* is not None **then**
    binary_insert(*to*, $\mathcal{S}$)
  end if
end for
return $\mathcal{S}$

A.4 APEX^2

Algorithm 4 APEX^2 framework

Require: Knowledge Graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$; (temporal) user query log $\mathcal{Q}^{(t)}$ on triples; # triples (size budget) K; decay factor γ; diffuse diameter d
Ensure: (temporal) Personal summary $\mathcal{P} = (\mathcal{E}_p^{(t)}, \mathcal{R}_p^{(t)}, \mathcal{T}_p^{(t)}) \subseteq \mathcal{G}$ with $|\mathcal{T}_p| \leq K$

/* Initializing Phase */
$\boldsymbol{H} \leftarrow |\mathcal{E}| \times |\mathcal{R}| \times |\mathcal{E}|$ Sparse Matrix with 0s, $\mathcal{E}_p^{(0)} \leftarrow \emptyset$, $\mathcal{R}_p^{(0)} \leftarrow \emptyset$, $\mathcal{T}_p^{(0)} \leftarrow \emptyset$
for $i = 0$; $i < d$; i++ **do**
  Calculate $(\alpha \boldsymbol{A})^l$ and $\sum_{l=0}^{d} \alpha^l \boldsymbol{A}^l$
end for
Calculate $\boldsymbol{q}_{\text{total}}^{(0)}$, $\boldsymbol{e}^{(0)}$, $\boldsymbol{r}^{(0)}$, $\boldsymbol{H}^{(0)}$ by equation 11, 12, 13 and 19
Choose triples with top-K highest heat to construct $\mathcal{T}_p^{(0)}$
$\mathcal{E}_p^{(0)} \leftarrow \{e \in \mathcal{E} \quad \text{s.t.} \quad \exists x \in \mathcal{T}_p^{(0)}, e \in x\}$ and $\mathcal{R}_p^{(0)} \leftarrow \{r \in \mathcal{R} \quad \text{s.t.} \quad \exists x \in \mathcal{T}_p^{(0)}, r \in x\}$

/* Adapting Phase */
for timestamp $t = 1$; $t \leq T$; t++ **do**
  Decay nonzero elements in $\boldsymbol{H}$ with γ^3
  Incrementally update $\boldsymbol{q}_{\text{total}}$, $\boldsymbol{e}$, $\boldsymbol{r}$
  Recalculate entry $\boldsymbol{H}[i][j][k]$ if $\boldsymbol{e}[i], \boldsymbol{r}[j]$ or $\boldsymbol{e}[k]$ is changed. Meanwhile construct C
  Incrementally sort $\boldsymbol{H}$ using C
  Choose triples with top-K highest heat to construct $\mathcal{T}_p^{(t)}$
  $\mathcal{E}_p^{(t)} \leftarrow \{e \in \mathcal{E} \quad \text{s.t.} \quad \exists x \in \mathcal{T}_p^{(t)}, e \in x\}$ and $\mathcal{R}_p^{(t)} \leftarrow \{r \in \mathcal{R} \quad \text{s.t.} \quad \exists x \in \mathcal{T}_p^{(t)}, r \in x\}$
end for
return $\{\mathcal{P}^{(t)}\}$

A.5 APEX^2-N

Algorithm 5 APEX^2-N Framework

Require: Knowledge Graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$; (temporal) user query log $\mathcal{Q}^{(t)}$; # size budget K; decay factor γ; diffuse diameter d
Ensure: (temporal) Personal summary $\mathcal{P} = (\mathcal{E}_p^{(t)}, \mathcal{R}_p^{(t)}, \mathcal{T}_p^{(t)}) \subseteq \mathcal{G}$ with $|\mathcal{T}_p| \leq K$
/* Initializing Phase */
$\boldsymbol{H} \leftarrow |\mathcal{E}| \times |\mathcal{E}|$ Sparse Matrix with 0s, $\mathcal{E}_p^{(0)} \leftarrow \emptyset$, $\mathcal{R}_p^{(0)} \leftarrow \emptyset$, $\mathcal{T}_p^{(0)} \leftarrow \emptyset$
for $q \in \mathcal{Q}^{(0)}$ **do**
  HeatDiffuse($\boldsymbol{H}, \mathcal{G}, q, d$)
end for
sort non-zero elements in $\boldsymbol{H}$
choose entities e with top-K highest heat to construct $\mathcal{E}_p^{(0)}$
$\mathcal{T}_p^{(0)} \leftarrow \{x_{ijk} \in \mathcal{T} \quad \text{s.t.} \quad i, j \in \mathcal{E}_p^{(0)}, v \in e\}$
$\mathcal{R}_p^{(0)} \leftarrow \{r \in \mathcal{R} \quad \text{s.t.} \quad \exists x_{ijk} \in \mathcal{T}_p^{(0)}, r \in x_{ijk}\}$
/* Updating Phase */
for timestamp $t = 1$; $t \leq T$; t++ **do**
  decay nonzero elements in $\boldsymbol{H}$ with γ
  C = HeatDiffuse($\boldsymbol{H}, \mathcal{G}, \mathcal{Q}^{(t)} \setminus \mathcal{Q}^{(t-1)}, d$)
  incrementally sort $\boldsymbol{H}$ using C
  while $|\mathcal{T}_p^{(t)}| \leq K$ **do**
    add v with highest heat to $\mathcal{E}_p^{(t)}$
    $\mathcal{T}_p^{(t)} \leftarrow \{x_{ijk} \in \mathcal{T} \quad \text{s.t.} \quad i, j \in \mathcal{E}_p^{(t)}\}$
    $\mathcal{R}_p^{(t)} \leftarrow \{r \in \mathcal{R} \quad \text{s.t.} \quad \exists x_{ijk} \in \mathcal{T}_p^{(t)}, r \in x_{ijk}\}$
  end while
end for
return $\{\mathcal{P}^{(t)}\}$

B Analysis of Existing Methods

In this section, we introduce two pioneering solutions on personalized knowledge graph summarization and analyze their adaptability for evolving user query interests, which motivate our adaptive PKG summarization framework, APEX^2, proposed in Section 3.

B.1 Preliminary

B.1.1 GLIMPSE. GLIMPSE [55] is a sampling-based method to summarize personalized KG. It first infers user preferences over the KG, then constructs a personal summary by maximizing an inferred utility by sampling. Denoting an entity e's neighbor set as $N(e)$. Then for a query log $\mathcal{Q}$, GLIMPSE captures the user's preference as

$$\Pr(e|\mathcal{Q}) \propto \sum_{q \in \mathcal{Q}} (\mathbb{1}(e \in q) + \alpha \sum_{e_o \in N(e)} \mathbb{1}(e_o \in q)) \tag{20}$$

where $\Pr(x|\mathcal{Q})$ stands for speculative preference on x given the query log, $\mathbb{1}$ is an indicator function, i.e., $\mathbb{1}(\mathcal{X}) = 1 \iff \mathcal{X} = \mathit{True}$, and α denotes the damping factor of neighbors $N(e)$ in the given knowledge graph $\mathcal{G}$. This equation assumes user's preference is more on the searched entity and can be generally pushed to its neighboring entities. The preference for relationship is defined by the query frequency, normalized by the total number of queries,

$$\Pr(r|\mathcal{Q}) \propto \frac{\sum_{q \in \mathcal{Q}} (\mathbb{1}(r \in q))}{|\mathcal{Q}|} \tag{21}$$

Then, following the standard conditional independence assumption [21, 49], GLIMPSE assumes user's preference for a triple to be proportional to a multiplication form as

$$\Pr(x_{ijk}|\mathcal{Q}) \propto \Pr(e_i|\mathcal{Q}) \Pr(r_k|\mathcal{Q}) \Pr(e_j|\mathcal{Q}) \tag{22}$$

After inferring user's preference, GLIMPSE constructs the PKG $\mathcal{P} = (\mathcal{E}_p, \mathcal{R}_p, \mathcal{T}_p)$ by maximizing the following utility,

$$\phi(\mathcal{P}, \mathcal{Q}) \propto \sum_{e \in \mathcal{E}_p} \log \Pr(e|\mathcal{Q}) + \sum_{x_{ijk} \in \mathcal{T}_p} \log \Pr(x_{ijk}|\mathcal{Q}) \tag{23}$$

In the optimizing phase, GLIMPSE continuously calculates the marginal utility of triples x_{ijk} and greedily samples triples with the highest marginal utilities[5] $\Delta_\phi(x_{ijk}|\mathcal{P}, \mathcal{Q})$,

$$\mathcal{T}^{\Delta \neq 0} \triangleq \{x_{ijk} \in \mathcal{G} \quad \text{s.t.} \quad \Delta_\phi(x_{ijk}|\mathcal{P}, \mathcal{Q}) \neq 0\} \tag{24}$$

where $\mathcal{T}^{\Delta \neq 0}$ is the set of triples with nonzero marginal utility for any given KG $\mathcal{G}$ and the corresponding PKG $\mathcal{P}$. "$\triangleq$" means "defined as". The sampling process stops when the number of triples in the summarization reaches the restriction or bound of size. The overall GLIMPSE framework is shown in Algorithm 1.

B.1.2 PEGASUS. PEGASUS [27] is a merging-based method for summarizing personalized graphs (not specific to knowledge graphs). It determines how to merge supernodes and superedges by minimizing the reconstruction error, during which more consideration will be put on a given set of targeted nodes. Formally, given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a set of target nodes $\mathcal{T}$, and a space budget k, PEGASUS aims to find a summarized graph $\mathcal{P} = (\mathcal{V}_P, \mathcal{E}_P)$ of $\mathcal{G}$ that is personalized to $\mathcal{T}$ while satisfying the budget k. The general logic is that attributes (edge connectivities) near the target set $\mathcal{T}$ are more likely to be retained in $\mathcal{P}$ than those far from the target set. The optimization of PEGASUS is based on the weighted reconstruction error $RE^{(\mathcal{T})}$,

$$RE^{(\mathcal{T})}(\mathcal{P}) = \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} \boldsymbol{W}_{ij}^{(\mathcal{T})} |\boldsymbol{A}_{ij}^{(\mathcal{G})} - \boldsymbol{A}_{ij}^{(\hat{\mathcal{G}})}| \tag{25}$$

where $\hat{\mathcal{G}} = (\mathcal{V}, \hat{\mathcal{E}})$ is the reconstructed graph from $\mathcal{P}$. $\boldsymbol{A}_{ij}$ stands for the adjacency matrix. $\mathcal{T}$ is the given target node set as interests. $\boldsymbol{W}_{ij}^{(\mathcal{T})}$ is the personalized weight of each pair of nodes (i, j). $\boldsymbol{W}_{ij}^{(\mathcal{T})}$ depends on the pairwise distance to the target nodes,

$$W_{ij}^{(\mathcal{T})} = \frac{\alpha^{-(D(i,\mathcal{T})+D(j,\mathcal{T}))}}{Z} \tag{26}$$

where $\alpha > 1$ controls the degree of personalization, Z is the constant that makes the average weight 1, and

$$D(u, \mathcal{T}) = \min_{t \in \mathcal{T}} \#hops(u, t) \tag{27}$$

is the minimum number of hops between node u and any node in $\mathcal{T}$. The overall PEGASUS framework is shown in Algorithm 2.

[5] Generally, $\Delta_F(x|\mathcal{S}) = F(\mathcal{S} \cup \{x\}) - F(\mathcal{S})$ is the marginal utility gained in the set function F by adding x to $\mathcal{S}$. Here $\Delta_\phi(x_{ijk}|\mathcal{P}, \mathcal{Q}) = \phi(x_{ijk} \cup \mathcal{P}, \mathcal{Q}) - \phi(\mathcal{P}, \mathcal{Q})$

B.1.3 iSummary. iSummary [61] summarizes knowledge graphs according to weights and seed nodes provided by users. iSummary tries to find a PKG by solving an NP-complete Steiner tree problem. The authors formulate the λ/κ-Personalized Summary problem as the following. Given (1) a knowledge graph $G = (V, E)$; (2) a non-negative weight assignment to all nodes representing user preferences; (3) λ seed nodes; (4) a number κ ($\lambda \leq \kappa$). The λ/κ-Personalized Summary problem aims to find the smallest maximum-weight tree $G' = (V', E') \in G$ that includes the κ most preferred nodes. In the original iSummary paper [61], the form of KG and the summarization problem is different from ours and the seminal work [55].

Given a SPARQL query log, iSummary works by first determining which entity the users are interested in the most (the λ seed nodes), then finding the shortest paths connecting them and including more facts in the summary. The constructed PKG is a maximum-weight tree that includes the κ most preferred nodes, which means the storage cost is much more than the cost of κ nodes themselves. We convert our scenario to the λ/κ Personalized Summary problem as the following. (1) we use our knowledge graphs $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$; (2) we use the heat on nodes as the user preferences; (3) we use the explicitly queried nodes as seed nodes, the number of which is λ; (4) we sweep κ from λ so that the summarized PKG from iSummary satisfies the storage limitation. By the time this paper is submitted, the source code of iSummary hasn't been open-sourced yet[6]. Therefore, we implemented iSummary on our own based on the pseudo-code [61] provided in the original paper.

B.2 Proof of non-adaptability or corruption

Here, we show that neither GLIMPSE [55] or PEGASUS [27] can deal with interest-evolving user queries very well. Our discussions here are under the circumstance that the summarized PKG is already full. Due to the page limit, we briefly introduce the proof idea and flow. The detailed and formal proof is placed in the appendix.

For GLIMPSE [55], the triples with higher utility will be more likely to be included in the summarization. We show that once the user's interest has shifted from a previous topic to a new topic, GLIMPSE can adapt to new interest only if the volume of new queries is considerably large, with respect to the volume of queries on previous interests. From this perspective, GLIMPSE is not very agile for quick and ad-hoc new interest queries.

Theorem B.1 (Non-adaptability of GLIMPSE in PKG Summarization). *In adaptive PKG summarization setting, GLIMPSE could not swiftly adapt to user's new interest after the previous summarized PKG reaches the size budget. (Proof in Appendix E.4)*

For PEGASUS [27], according to Eq. 26 and Eq. 27, a historical target node j permanently gives a lower bound to all W_{ij}. In other words, if a node i is close to a target node j, node i (as well as the nodes that are close to node i) will be given high weights. Then, we show that once these nodes (e.g., i and its k-hop neighbors) gets high weights and are considered important, they are hard to be replaced by the nodes of later new interests. This means that though the user's interest may already shift to other topics, previous topics still have high weights and may occupy the summary storage.

[6]https://anonymous.4open.science/r/iSummary-47F2/

Theorem B.2 (Non-adaptability of PEGASUS in PKG Summarization). *In adaptive PKG summarization setting, PEGASUS could not effectively evolve with user's new interest after the previous summarized PKG reaches the size budget. (Proof in Appendix E.5)*

For iSummary [61], under the extremely small storage limitation in our experiment scenario, even if we set $\kappa = \lambda$, the summarized PKG is still too large in terms of size. In this case, even though iSummary finds the shortest paths, those paths will not be used, and the PKG simply stores the explicitly searched nodes that have the highest heat. In the extreme summarization case, iSummary becomes a simple caching algorithm and does not have the ability to infer the user's interests. In other words, iSummary corrupts under extremely small storage constraints since $\lambda > \kappa$, i.e., the number of seed nodes exceeds the maximum number of nodes to summarize.

C Experiment Details

C.1 Metric: F1 Score

For a query q with query entity e_i, query relation r_k and a set of answers $\mathcal{A}$, we define the answer sets in the KG $\mathcal{G}$ and PKG $\mathcal{P}$ are, respectively, $\mathcal{A}_{kg} = \{e_j \ s.t. \ \exists x_{ijk} = (e_i, r_k, e_j) \in \mathcal{T}\}$ and $\mathcal{A}_{pkg} = \{e_j \ s.t. \ \exists x_{ijk} = (e_i, r_k, e_j) \in \mathcal{T}_p\}$. Then we have

$$TP(True\ Positive) = |\mathcal{A}_{pkg} \cap \mathcal{A}_{kg}| \tag{28}$$

$$FP(False\ Positive) = |\mathcal{A}_{pkg} \setminus \mathcal{A}_{kg}| \tag{29}$$

$$FN(False\ Negative) = |\mathcal{A}_{kg} \setminus \mathcal{A}_{pkg}| \tag{30}$$

$$precision = \frac{TP}{TP + FP}, \ recall = \frac{TP}{TP + FN} \tag{31}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{32}$$

"Precision" calculates the fraction of relevant instances among the summarized instances, while "Recall" is the fraction of relevant instances that were summarized[7]. Taking both of them into account, F1 score provides a single number that reflects the model's overall search performance.

In our adaptive setting, we adapt historical PKG to the new query, wishing to make the adaption accurate. In this way, the very next query is the most valuable to test the utility of the current PKG, as future queries may shift again. Moreover, we report the mean and std over multiple single next queries, showing the robustness.

C.2 Datasets

YAGO3 is a huge semantic knowledge base, derived from Wikipedia WordNet and GeoNames. Currently, the whole YAGO3 has knowledge of more than 10 million entities and contains more than 120 million facts about these entities[8]. The whole YAGO3 is over 200GB. Due to the computational resource limitation, we use the **core YAGO3 facts** that contains 4.2 million entities and 12.4 million triples, which is a 40% sub-YAGO3.

[7]Resource: https://en.wikipedia.org/wiki/Precision_and_recall
[8]Resource: https://datahub.io/collections/yago

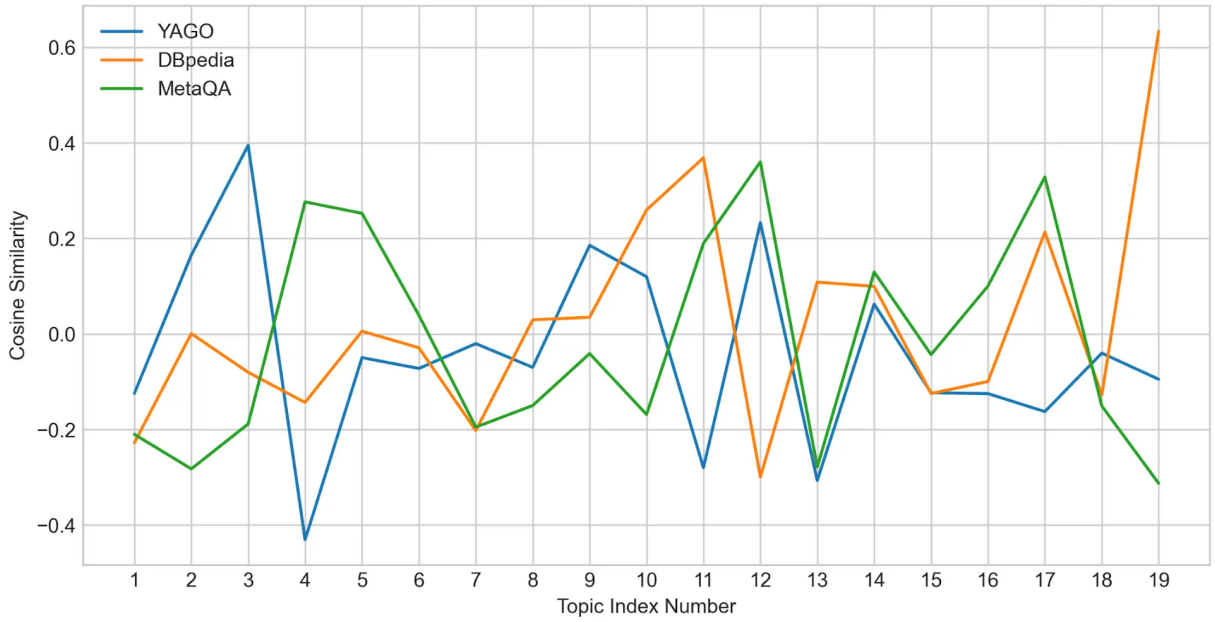


Figure 5: Cosine similarity between consecutive query topics in different datasets for one user. The cosine similarity is computed from Roberta [42] text embeddings.

DBpedia was created by extracting semantically-structured information from Wikipedia and other data sources. Specifically, we use the **DBpedia ontology**, which is the "heart of DBpedia"[9]. The DBpedia Ontology is a shallow, cross-domain ontology, which has been manually created based on the most commonly used infoboxes within Wikipedia[10].

MetaQA consists of a movie ontology derived from the WikiMovies Dataset[11]. WikiMovies dataset, introduced by Miller et al.[48], is constructed along with a graph-based KB consisting of entities and relations, with the guarantee that each query can be answered by the KB. We use the **whole MetaQA** dataset and the "Vanilla" branch of its query data.

Freebase is a large collaborative knowledge base created by the Freebase community. We use FB15K237, a well-developed and commonly used sub-knowledge-graph of the originally retired Freebase data dump[12].

C.3 Quality Analysis of Synthetic Queries

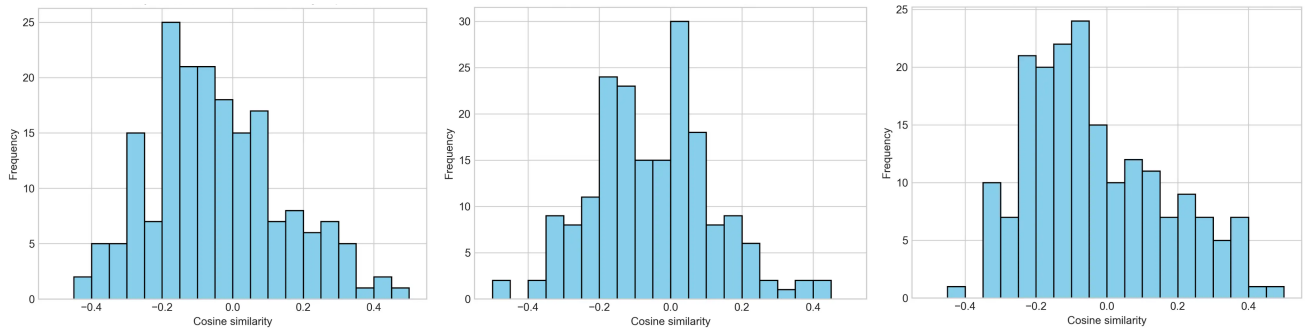
To the best of our knowledge, there is no paired user query log and KG dataset, because the public user query datasets are anonymized by the publisher and do not have user identification. Therefore, it is impossible to extract one specific user's consecutive queries. Nonetheless, we conduct our experiments over our synthetic but realistic query logs over benchmark KGs. In this section, we further validate that our synthetic queries are of high quality.

One piece of evidence is that our sampled queries follow the logic and structure of Linked SPARQL Queries' DBpedia anonymous data dump[13]. In both LSQ and our queries, multiple query relations on the same query head appear in a batch and get answered sequentially. Then query relations start from another entity head and repeat such a pipeline.

We further conduct a study on the semantic of sampled topics. As discussed in Section 5, for each dataset, we sample 10 users and 200 queries for each user. To be specific, for each user, we uniformly randomly sampled 20 entities (i.e., topics) from all the entities in the KG. The user querying topics are evolving, which is exactly the reason why we need adaptive personalized KG summarization. Therefore, we measure the semantic distances of each pair of consecutive topics. For example, sample a random user in the real-world MetaQA dataset, and we can get the query topic history as *['Stevan Riley', 'The Disappearance of Haruhi Suzumiya', 'LOL', 'Chad Michael Murray', 'Orson Scott Card', 'Gérard Lanvin', 'menahem golan', '...All the Marbles', 'Operator 13', 'Heaven Help Us', 'peter pan', 'Mark Saltzman', 'Denise Dillaway', 'Hell Drivers', 'drummer', 'Beware of Pity', 'The Cruel Sea', 'The Great Gabbo', 'Takuya Kimura', 'The Secrets']*. From this log of topics, we can see the topic changes from different genre movies to different directors and even novels. Using the widely-used language model RoBERTa [42], we can compute the normalized embedding vector for each pair of consecutive topics. Their cosine similarities are [-0.2100357562303543, -0.2824622392654419, -0.18836577236652374, 0.2763717472553253, 0.2524639368057251, 0.03907020390033722, -0.19497732818126678, -0.14980322122573853, -0.041029710322618484, -0.16855204105377197, 0.18958419561386108,

[9]Resource: https://www.dbpedia.org/resources/ontology/
[10]Resource: http://wikidata.dbpedia.org/services-resources/ontology
[11]Resource: https://paperswithcode.com/dataset/metaqa
[12]Resource: https://paperswithcode.com/dataset/fb15k-237

[13]https://files.dice-research.org/archive/lsqv2/dumps/dbpedia/



(a) YAGO (b) DBpedia (c) MetaQA

Figure 6: The distribution of cosine similarities between consecutive query topics in our synthetic queries. From left to right: YAGO, DBpedia, MetaQA. Our synthetic queries cover diverse topic-evolving scenarios.

0.3601977825164795, -0.278546005487442, 0.12963563203811646, -0.04322625696659088, 0.10026416927576065, 0.3284454941749573, -0.15075044333934784, -0.3128302991390228]. Such semantic distances range wide and are diverse, validating that our sample user query logs cover diverse topic-evolving scenarios and are of high quality.

We also conduct the visualization for all users in every dataset to show their query topic shifts. The figures of the cosine similarity frequency of all topic shifts for all datasets are shown in Figure 6. Here, "frequency" means, among a total of $190 = 10 \times (20 - 1)$ topic changes, i.e., number of users times topic evolves per user, how many times the cosine similarity between consecutive topics falls into a specific range. In general, we can observe that the various kinds of topics evolve, which demonstrates that our query synthesis and our adaptive summarization setting are challenging. Under such a realistic setting, our proposed methods achieve effectiveness and efficiency outperformance than baseline methods.

C.4 Reproducibility

C.4.1 Code. The code for the main experiment is provided at https://github.com/Violet24K/APEX2. The data of DBpedia and MetaQA is provided within our submission code, but YAGO3 is not due to its large size. You need to manually download YAGO3 follow our instruction if you want to play with it. Our code only supports ".gz" Compressed Archive Folders, so you need to convert using gzip if the downloaded files are in other format.

C.4.2 YAGO3. https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/downloads/. Download YAGO Themes -> CORE- > yagoFacts (all facts of YAGO that hold between instances). You need to right click "Download TSV" icon, then click "Save link as..." to download.

C.4.3 DBpedia. Provided within our code. Original download link: http://downloads.dbpedia.org/3.5.1/en/. Specifically, download the "mappingbased_properties_en.nt" file.

C.4.4 MetaQA. Provided within our code. Original download link: https://github.com/yuyuz/MetaQA. The entire MetaQA dataset can be downloaded from the link provided in their README.md.

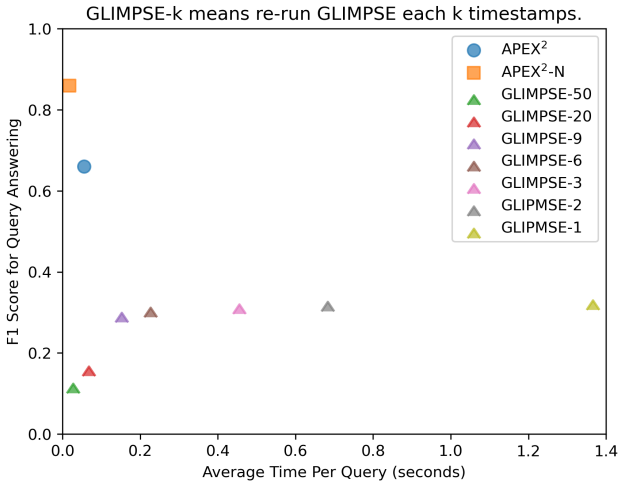


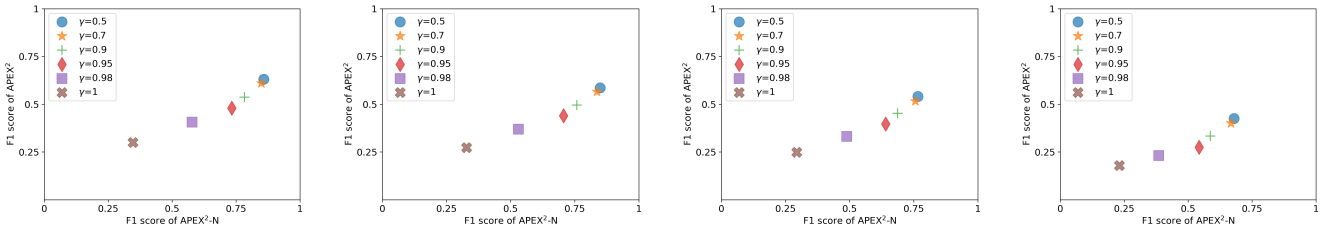
Figure 7: Our pre-experiment to determine that $R = 9$ is a good trade-off for accuracy and efficiency.

C.4.5 Freebase. https://paperswithcode.com/dataset/fb15k-237. The entire FB15K237 dataset can be downloaded by clicking "Homepage".

C.4.6 Environments. We run all our experiment on a Windows 10 machine with Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz and 32GB RAM. For other different platforms such as Linux, you may need to reset the path and encoding in code/src/path.py file. The Python version is 3.8.13 in our experiment.

C.5 Pre-experiments to Pick Baseline Re-summary Interval

For the baseline methods, we vary their R (re-summarization interval) hyperparameter. We observe that when $R = 9$, their query answering accuracy does not drop much compared to $R = 1$. But if R exceeds 9, their query answering accuracy drops significantly. An example curve for GLIMPSE on the MetaQA dataset is shown in Figure 7. Therefore, we pick the Pareto-optimal $R = 9$ for the baselines.



(a) $R_{APEX} = 1$ **(b)** $R_{APEX} = 2$ **(c)** $R_{APEX} = 3$ **(d)** $R_{APEX} = 6$

Figure 8: Querying Effectiveness in Multiple Decay Levels for different R_{APEX}.

C.6 Experiments Adjusting R_{APEX}

APEX^2 and APEX^2-N can re-summarize the KG every R_{APEX} timestamp to increase overall efficiency, denoted by APEX^2-R_{APEX} and APEX^2-N-R_{APEX}. The other experiment settings are the same as the main experiments. The mean performances of the F1 score and per-query time consumption on MetaQA and YAGO are shown in Table 5 and 6.

Table 5: The performance of APEX^2-R_{APEX} and APEX^2-N-R_{APEX} with varying R_{APEX} on MetaQA.

Methods	F1 Score (↑)	Time Consumption Per Query (↓)
APEX^2-1	0.631	0.055
APEX^2-2	0.586	0.049
APEX^2-3	0.540	0.044
APEX^2-6	0.426	0.036
APEX^2-N-1	0.858	0.018
APEX^2-N-2	0.849	0.011
APEX^2-N-3	0.766	0.007
APEX^2-N-6	0.680	0.004

Table 6: The performance of APEX^2-R_{APEX} and APEX^2-N-R_{APEX} with varying R_{APEX} on YAGO.

Methods	F1 Score (↑)	Time Consumption Per Query (↓)
APEX^2-1	0.601	6.354
APEX^2-2	0.562	4.947
APEX^2-3	0.524	3.433
APEX^2-6	0.475	2.695
APEX^2-N-1	0.882	2.528
APEX^2-N-2	0.861	1.691
APEX^2-N-3	0.772	1.052
APEX^2-N-6	0.667	0.689

From the results, by adjusting R_{APEX}, we can balance between the searching accuracy and the overall time consumption. By choosing a larger R_{APEX} and processing multiple queries together with less granularity, APEX^2 and APEX^2-N can execute faster, but sacrifice interest tracking accuracy.

APEX^2-R_{APEX} and APEX^2-N-R_{APEX} are similar to APEX^2 and APEX^2-N, except that the re-summarization interval is different. Because we only masked the summary updating phase in some timestamps, at the timestamps where there is no mask, the output PKGs are identical to those from APEX^2 and APEX^2-N, respectively. Therefore, the results and analysis from our case study still applies for APEX^2-R_{APEX} and APEX^2-N-R_{APEX}. Additionally, we conduct the decay ablation studies and hyperparameter studies on $R_{APEX} \in \{1, 2, 3, 6\}$ to further study these faster variants. The results are shown in Figure 8 and Figure 9. From the ablation results, the decaying mechanism controlled by γ is still crucial for APEX^2-R_{APEX} and APEX^2-N-R_{APEX}, as setting γ close to 1 will decrease their performances. From the hyperparameter study results, APEX^2-R_{APEX} and APEX^2-N-R_{APEX} demonstrate similar trends when adjusting compression ratio κ, damping factor of neighbor α, diffusing diameter d. Overall, the effectiveness of APEX^2-R_{APEX} and APEX^2-N-R_{APEX} are robust to the damping factor of neighbor α, diffusing diameter d.

C.7 Handling Very Large Knowledge Graphs

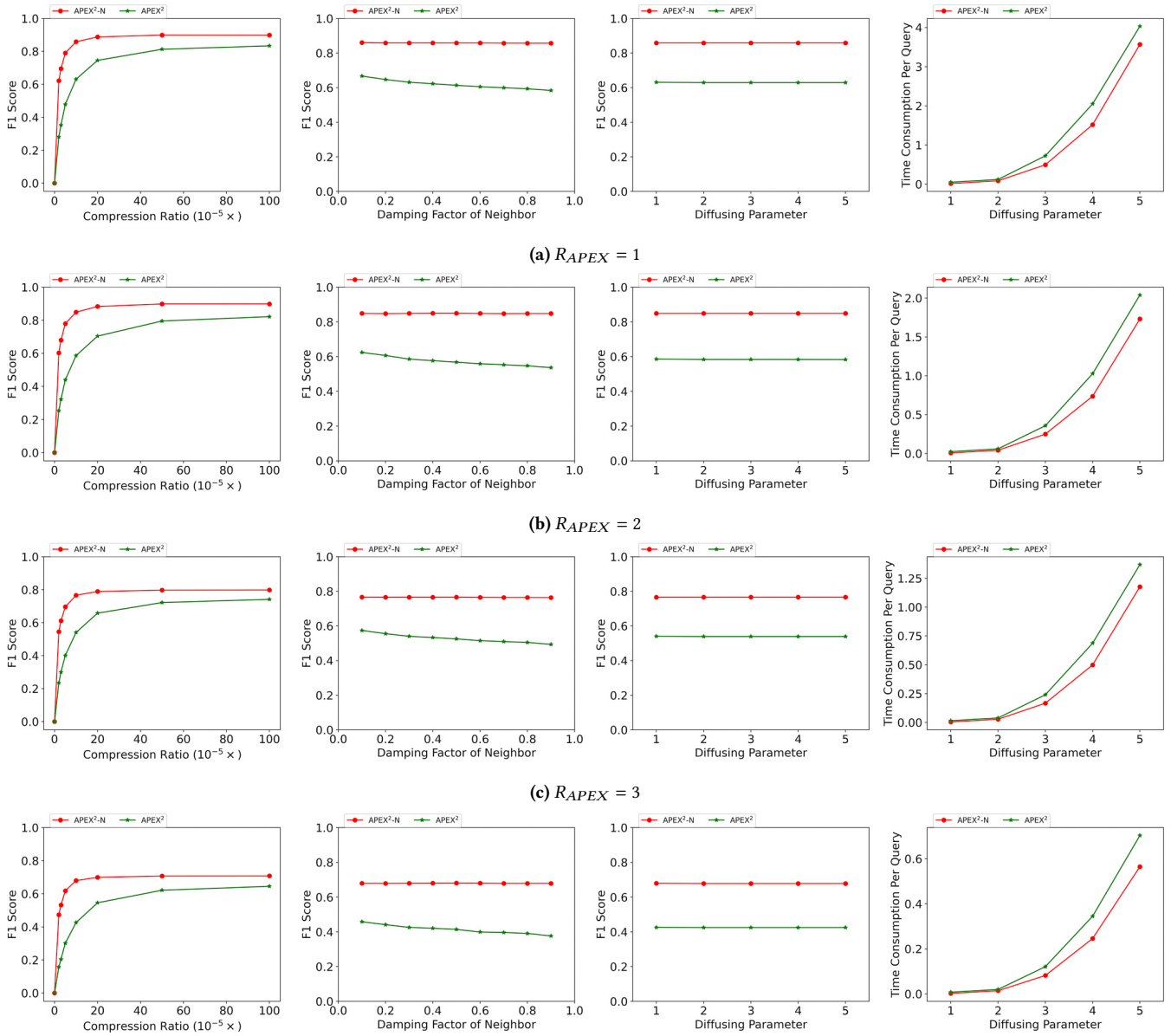
Theoretically, we give scalability proofs in Theorems 4.2 and 4.4. The complexity is further optimized to be independent of graph size by setting an eliminating threshold. In the main experiments, we verify the scalability of our algorithms by two large KGs containing 10M triples. To further demonstrate the scalability of our algorithms, we conduct an experiment using a freebase subset of 30M triples with 0.001% compression ratio. The mean results are reported in Table 7.

Table 7: Results Summarizing Freebase subset of 30,000,000 triples.

Methods	F1	time (seconds)
APEX^2	0.4941	8.28
APEX-N^2	0.6889	2.56
GLIMPSE	0.3273	284.86
PageRank	0.1448	17.2

D Visual Aid for Heat Diffusion Mechanism

We use a heat model to simulate and track user's interest from realistic perspectives of human interest phenomenons. **(i)** When a human searches for one thing over KG or the Internet, it means, in most cases, he or she is interested in that thing. **(ii)** Human interest transits from one thing to its related things. For example, when a user shows interest in a specific movie, such interest may extend to related aspects such as the actors, director, or genre of the film. **(iii)** Over time, as humans tend to forget, their interest in a particular thing should naturally decline. From these facts and observations,



(a) $R_{APEX} = 1$

(b) $R_{APEX} = 2$

(c) $R_{APEX} = 3$

(d) $R_{APEX} = 6$

Figure 9: Parameter Study adjusting different R_{APEX}. From left to right: compression ratio κ, damping factor of neighbor α, diffusing diameter d.

we adopt a heat decay-inject-diffuse framework for heat diffusion, as shown in Figure 10, which shows an illustration of user's interest after each timestamp.

Initially (Timestamp 0), the user has not interacted with the KG, and the KG is all-white, i.e., we cannot infer if the user is interested in any entity. Then, at timestamp 1, first, there is a *decay* over entities, resulting from all-white to still all-white. Second, the user searches *(Interstellar, directed_by, ?)*, and gets the answer triple *(Interstellar, directed_by, Nolan)*. Thus, we *inject* heat into entities *Interstellar* and *Nolan*, as marked by the dashed-line box in Timestamp 1, Figure 10. Third, there is a global *push* from all entities to its neighbors. Therefore, the entities connected to *Interstellar* and *Nolan* get partial heat.

At Timestamp 2, first, there is heat decaying happening for all entities, and therefore entities such as *Interstellar, McConaughey, Zimmer, London* change to lighter gray colors. Second, the user searches *(Tenet, has_tag, ?)* and gets the answer triple *(Tenet, has_tag, Action)*. Thus, we *inject* heat into entities *Tenet* and *Action*, as marked

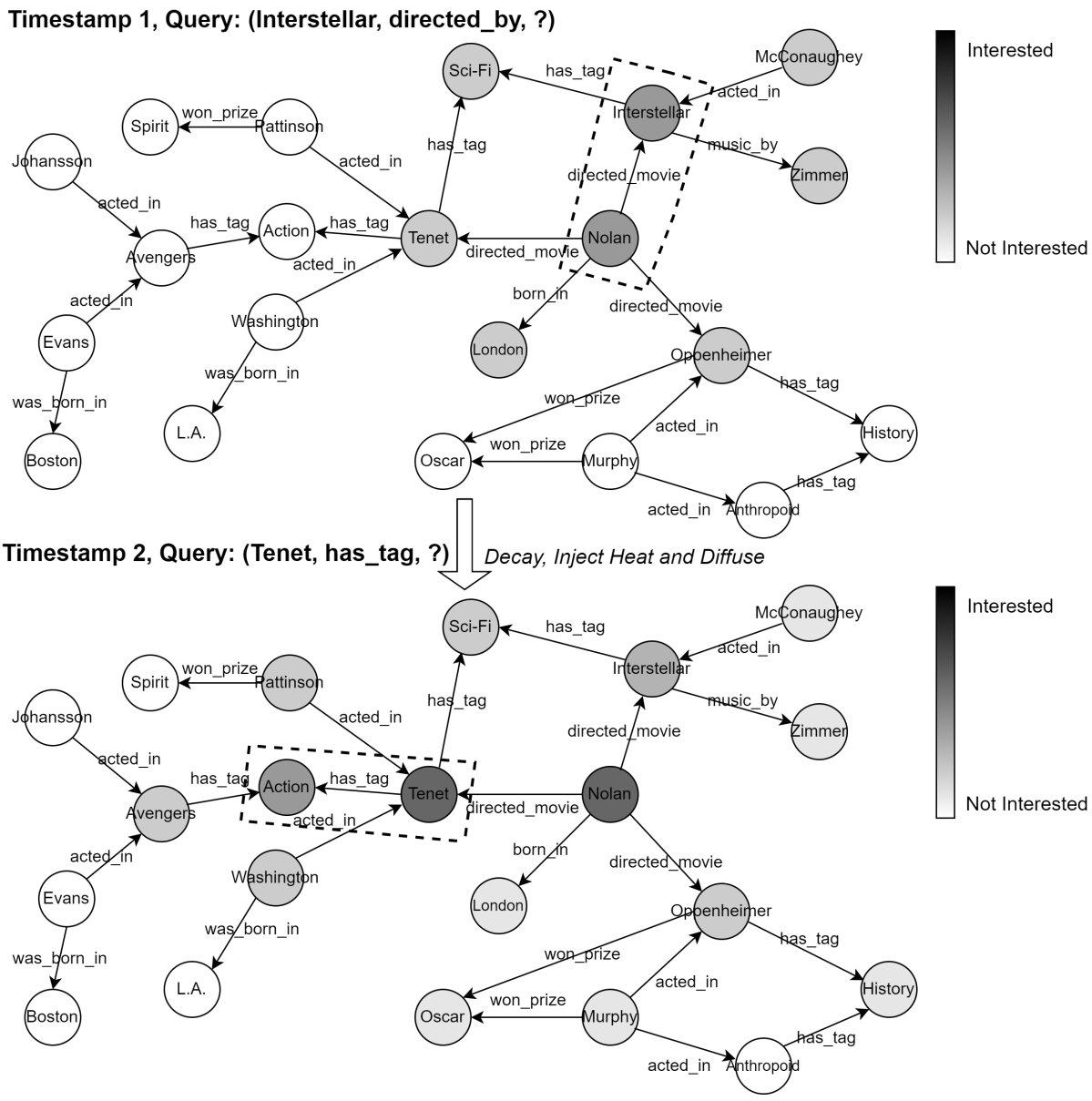


Figure 10: We adopt a heat decay-inject-diffuse framework for heat diffusion. For each timestamp, first, all the heat (i.e., interest) will be decayed; then, an amount of new heat will be injected to the searched entities (marked by the dashed-line boxes); third, a global push is executed, where the heat of each entity will be partially pushed to its neighbors.

by the dashed-line box in Timestamp 2. Third, global push is conducted and entities such as *Pattinson, Averngers, Washington, Oscar, Murphy, History* get pushed partial heat.

For future timestamps, the decay-inject-diffuse mechanism will work similarly as a simulation of user interest tracking.

E Proof of Theorems

E.1 Proof of Theorem 4.1

Proof. All the sorting algorithms must output a permutation of $\mathcal{S} \cup C$. With the fact that there is only one correct permutation that satisfies the sorted condition (the "deterministic" one), reconsider the sorting process to be eliminating the possible permutations by new comparisons. Note that using the prior knowledge in the sorted $\mathcal{S}$ does not count toward new comparisons.

With the input $\mathcal{S}$ and C, let $\mathcal{V}$ be the set of these inputs that are consistent with the answers to all comparisons made so far. Initially, $\mathcal{V}$ contains all v that is a permutation of $\mathcal{S} \cup C$ and does not violate any order in $\mathcal{S}$. Each new comparison (*is* $a > b$?) will split $\mathcal{V}$ into two groups: the ones answering "YES" and the ones answering "NO". With the fact that only one group will be the next $\mathcal{V}$, setting the correct answer to the larger group will guarantee $|\mathcal{V}_{next}| \geq \frac{|\mathcal{V}_{current}|}{2}$.

The algorithm must reduce $|\mathcal{V}|$ to 1 in order to get the output, and the total number of the initial $\mathcal{V}$ can be attained by counting the number of ways to insert C (k distinct elements) into $\mathcal{S}$ (n already-sorted elements). Use the formula from *Stars and Bars* [14] ($\mathcal{S}$ contains stars, and C contains bars, permitting neighboring bars), the total number of ways is

$$\binom{k+n-1}{n-1} \cdot k! = \frac{(k+n-1)!}{(n-1)! \cdot k!} \cdot k! = \frac{(n+k-1)!}{(n-1)!} \tag{33}$$

Thus, when $k << n, \log_2 \frac{(n+k-1)!}{(n-1)!} = \Omega(klog_2 n)$ is the worst case complexity. The Incremental Binary Insertion Sort has complexity $\log_2 n$ for each element in C to do a binary insertion sort into $\mathcal{S}$, therefore its complexity is $\Omega(klog_2 n)$. □

E.2 Proof of Theorem 4.3

Proof. In the initializing phase, the computation of sparse matrix multiplication, $\alpha \boldsymbol{A}$, summation $\sum_{l=0}^{d} (\alpha \boldsymbol{A})^l$ and sparse matrix-vector multiplication $\boldsymbol{q}_{\text{total}}^{(0)}$, $\boldsymbol{e}^{(0)}$, $\boldsymbol{r}^{(0)}$ needs to be done only one time. Choosing top-K heat triples requires a one-time sort.

In the updating phase, the decaying requires $O(nnz(\boldsymbol{H}^{(0)})$. The update of $\boldsymbol{q}_{\text{total}}^{(t)}$, $\boldsymbol{r}^{(t)}$ respectively takes at most $O(nnz(\boldsymbol{q}_{\text{total}}^{(t)})) < O(2|\mathcal{Q}|)$, $O(nnz(\boldsymbol{r}_{\text{total}}^{(t)})) < O(|\mathcal{Q}|)$. The update of $\boldsymbol{e}^{(t-1)}$ requires $O(nnz(\boldsymbol{e}_{\text{total}}^{(t)}) + 2|\mathcal{E}|) < O(2|\mathcal{E}||\mathcal{Q}|)$ at most and $O(nnz(\boldsymbol{e}_{\text{total}}^{(t-1)}) + \frac{2nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|}) < O(2 \cdot \frac{nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|} \cdot |\mathcal{Q}|)$ on average.

Respectively, for each timestamp $\boldsymbol{e}[i], \boldsymbol{r}[j]$ and $\boldsymbol{e}[k]$ has on average 2, 1, and $\frac{2nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l))}{|\mathcal{E}|}$ entries updated, therefore $\boldsymbol{H}$ has on average $O(3 \cdot 2 \cdot 2 \cdot \frac{nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|} \cdot |\mathcal{Q}|^2) = O(12 \cdot \frac{nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|} \cdot |\mathcal{Q}|^2)$ entries changed. The incremental sorting algorithm on $\boldsymbol{H}$ will take $O(12 \cdot \frac{nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|} \cdot |\mathcal{Q}|^2 \cdot \log_2 nnz(\boldsymbol{H})) < O(12 \cdot \frac{nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|} \cdot |\mathcal{Q}|^2 \cdot \log_2(12 \cdot \frac{nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|} \cdot |\mathcal{Q}|^3)) = O(36 \cdot \frac{nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|} \cdot |\mathcal{Q}|^2 \cdot \log_2(12 \cdot \frac{nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|} \cdot |\mathcal{Q}|))$. □

E.3 Proof of Theorem 4.5

Proof. The calculations in the initializing phase needs to be done only one time. With heat diffuse to be "equally push to neighbors", each HeatDiffuse operation requires $O(\frac{2nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|})$ calculations.

In the updating phase, the decaying requires $O(nnz(\boldsymbol{H}^{(0)})$. $\boldsymbol{H}$ has at most $\frac{2nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|}$ entries changed and an incremental sort requires $O(\frac{2nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|} \log_2 nnz(\boldsymbol{H})) < O(\frac{2nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|} \cdot \log_2(\frac{2nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|}|\mathcal{Q}|))$. The construction of $\mathcal{T}_p$ and $\mathcal{R}_p$ requires approximately $O(K^2)$ which is a constant in our problem setting. Therefore the most cost-demanding operation takes $O(c \cdot \log_2(c|\mathcal{Q}|))$ where $c = \frac{2nnz(\sum_{l=0}^{d}(\alpha \boldsymbol{A})^l)}{|\mathcal{E}|}$. □

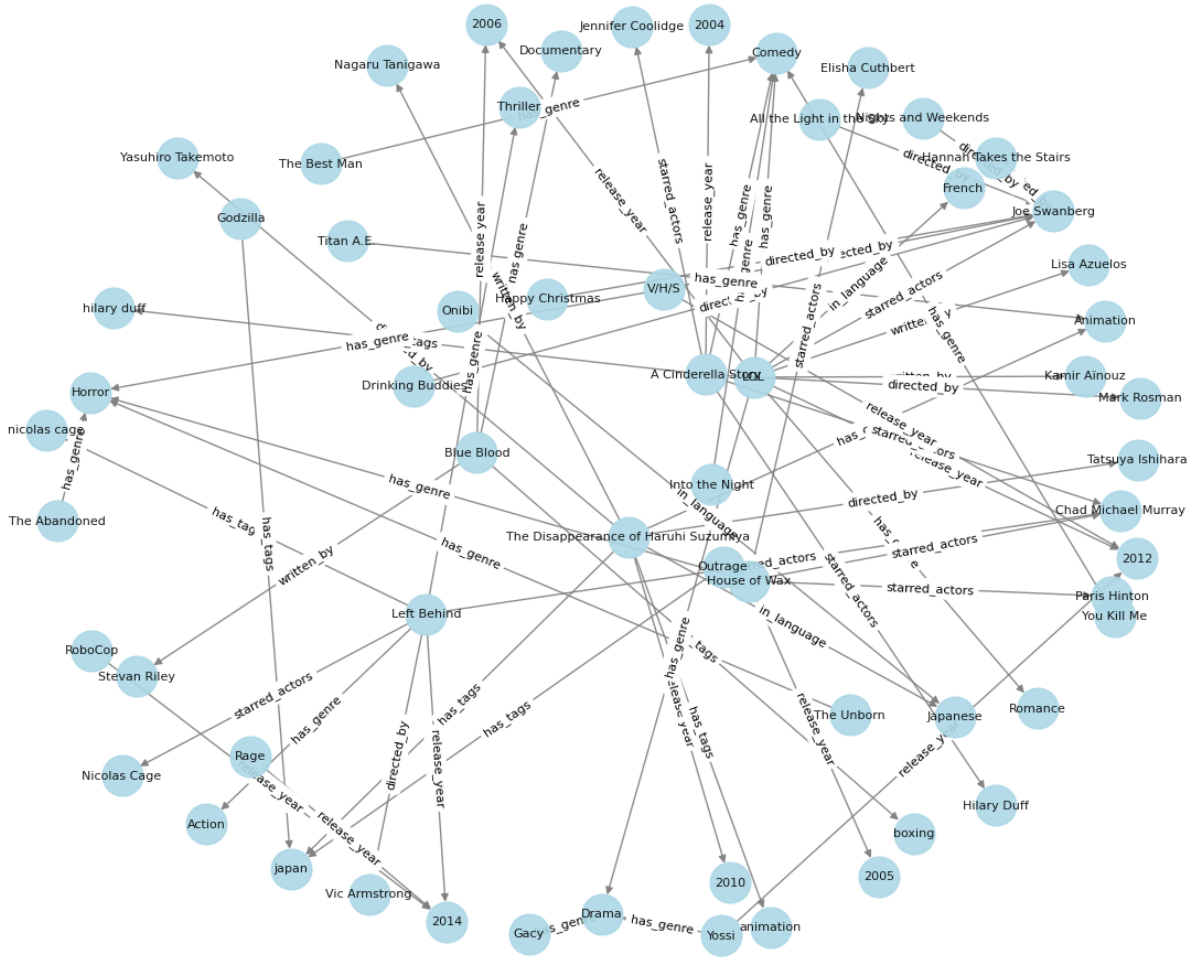
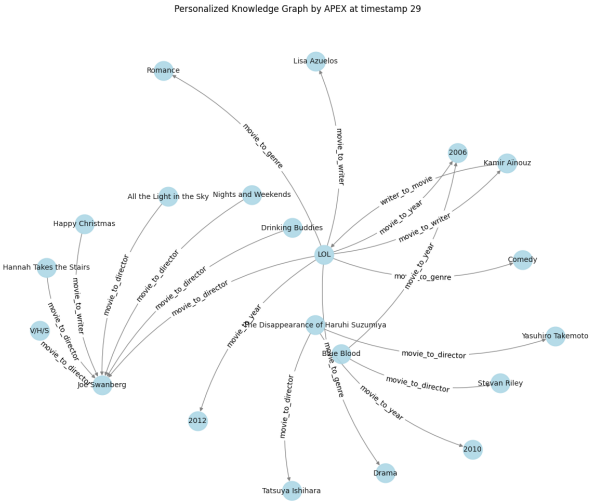
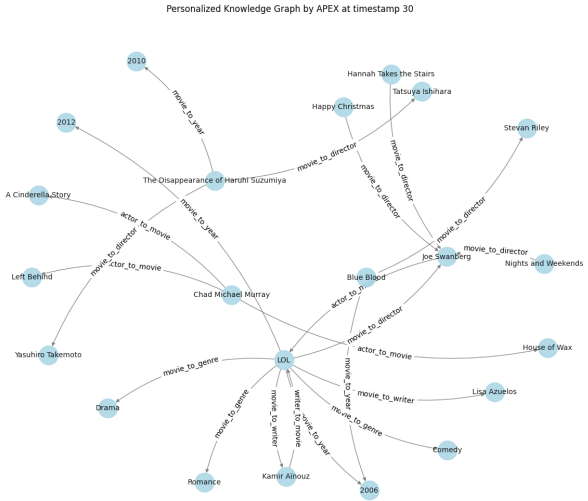


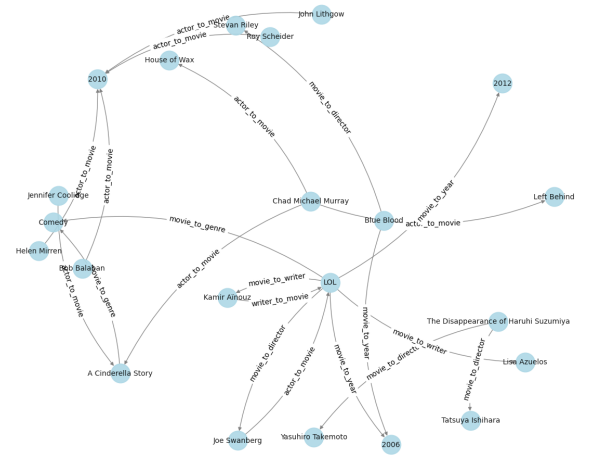
Figure 11: The specific part of MetaQA Knowledge Graph within the interest of our case study. The summarization operates on the whole KG, not only this small portion of KG.



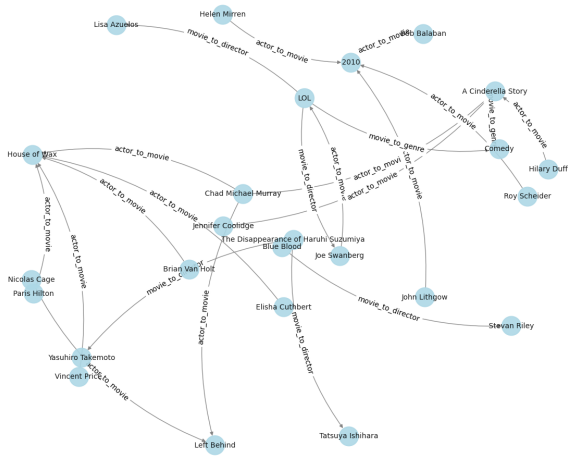
(a) PKG Before the New Topic



(b) PKG after the First Query on the New Topic

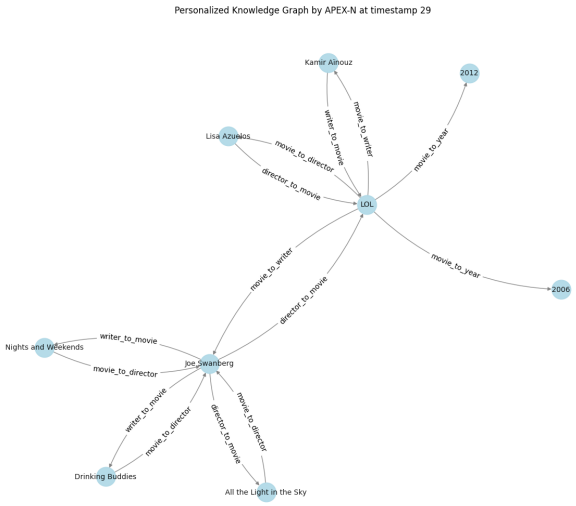


(c) PKG after the Second Query on the New Topic

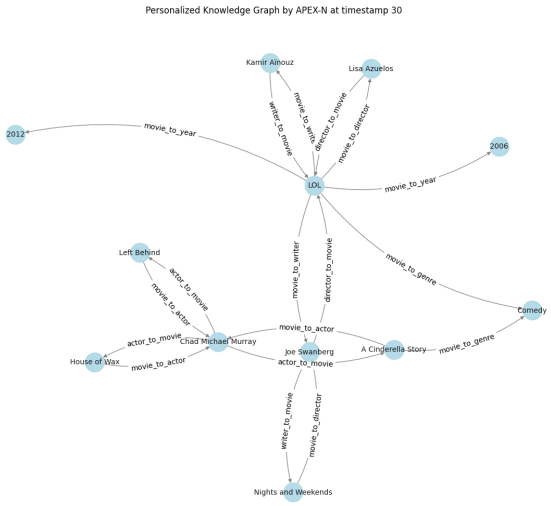


(d) PKG after the Third Query on the New Topic

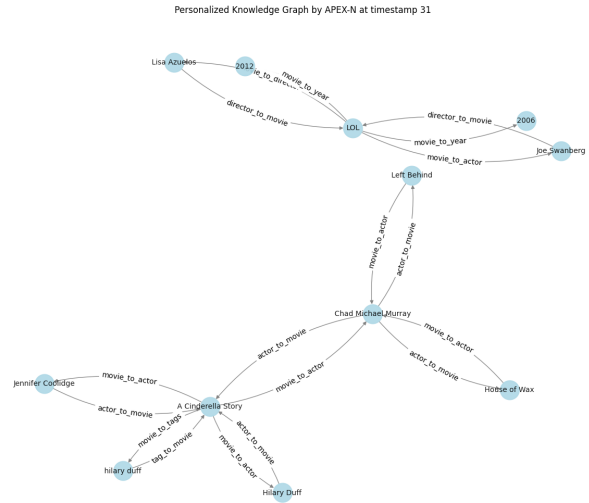
Figure 12: Case Study on APEX^2



(a) PKG Before the New Topic



(b) PKG after the First Query on the New Topic

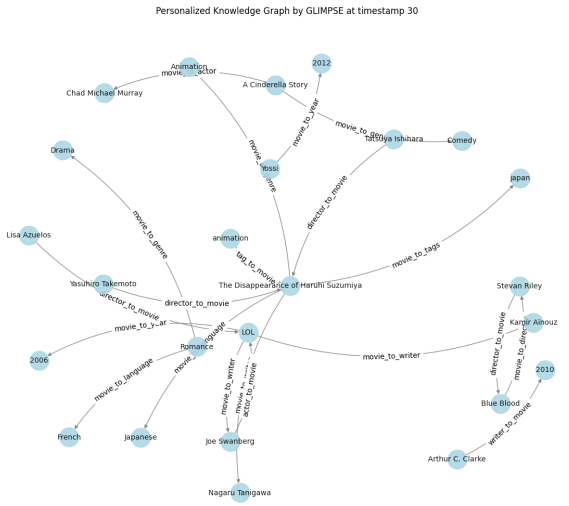


(c) PKG after the Second Query on the New Topic

(d) PKG after the Third Query on the New Topic

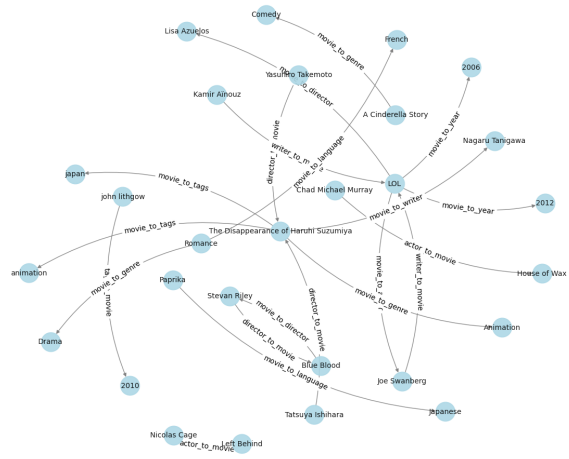
Figure 13: Case Study on APEX^2-N

(a) PKG Before the New Topic



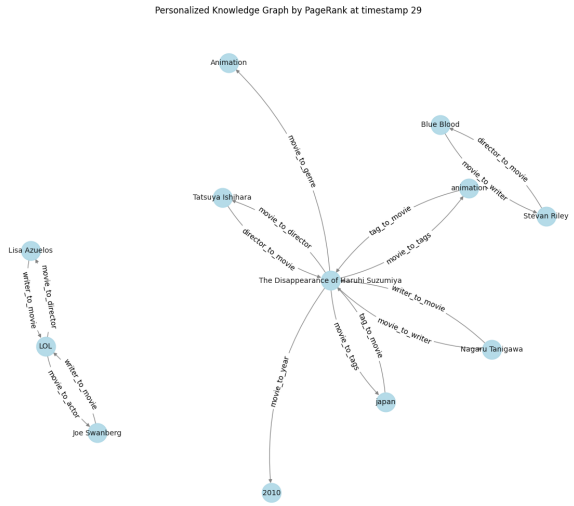
(b) PKG after the First Query on the New Topic

(c) PKG after the Second Query on the New Topic

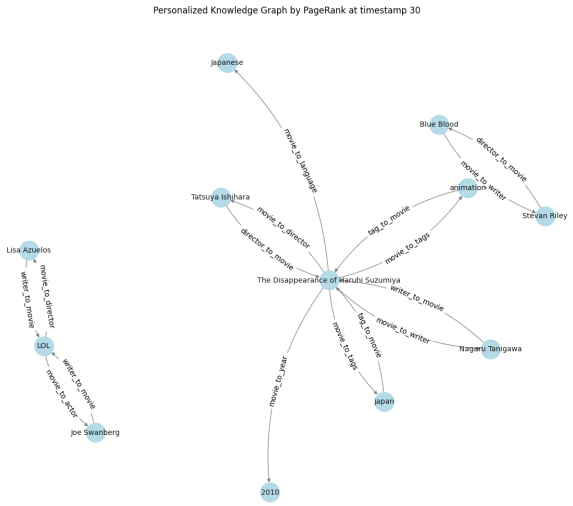


(d) PKG after the Third Query on the New Topic

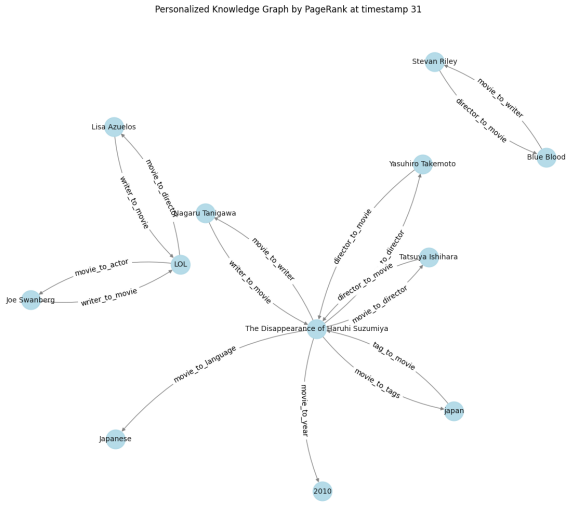
Figure 14: Case Study on GLIMPSE



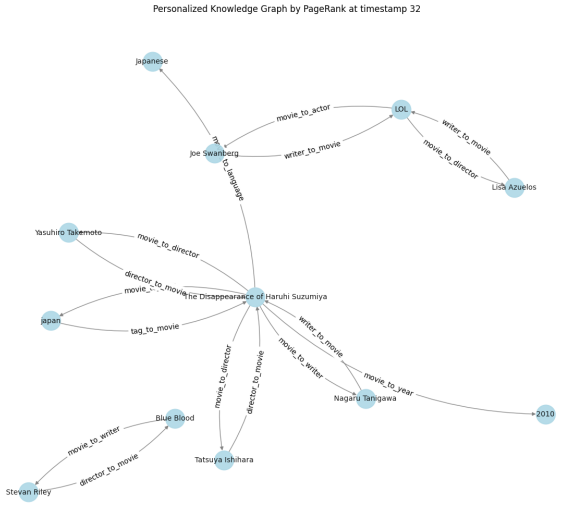
(a) PKG Before the New Topic



(b) PKG after the First Query on the New Topic



(c) PKG after the Second Query on the New Topic



(d) PKG after the Third Query on the New Topic

Figure 15: Case Study on PageRank

E.4 Proof of Theorem 3.1

Proof. Here, we give the proof for the non-adaptability of GLIMPSE [55].

We define "average connectivity of an area (sub knowledge graph) $\mathcal{V} = (\mathcal{E}_v, \mathcal{R}_v, \mathcal{T}_v)$" as $\frac{\sum_{v\in\mathcal{E}_v} degree(v)}{|\mathcal{E}_v|}$.

We then define "average connectivity of an area (sub knowledge graph) $\mathcal{V} = (\mathcal{E}_v, \mathcal{R}_v, \mathcal{T}_v)$" as $\frac{\sum_{v\in\mathcal{E}_v} degree(v)}{|\mathcal{E}_v|}$.

Assume two areas $\mathcal{U}$ and $\mathcal{V}$ with connectivity c_u and c_v.

Let the user initially query $\mathcal{U}$ for a times, i.e., the query set $|Q_u| = a$.

For each query, $\sum_{e\in\mathcal{E}_u} \Pr(e, Q_u)$ is expected to increase $2(1 + \alpha c_u)$. Then, for a many queries, the total increase is $2a(1 + \alpha c_u)$.

Assume the user's interest shifts to $\mathcal{V}$ and queries $\mathcal{V}$ for b times. In the same way, the total increase is $2b(1 + \alpha c_v)$. s Assume the total query is Q, where $Q_u \subseteq Q$ and $Q_v \subseteq Q$.

Based on Eq. 21, we have $\frac{\sum_{r_k\in\mathcal{R}_u} Pr(r_k|Q)}{\sum_{r_k\in\mathcal{R}_v} Pr(r_k|Q)} = \frac{a}{b}$. Moreover, we can derive the relation of expectation $\frac{\mathbb{E}_{r,u}}{\mathbb{E}_{r,v}} = \frac{a|\mathcal{R}_v|}{b|\mathcal{R}_u|}$, where $\mathbb{E}_{r,u}$ is the average time being queried for each relation in $\mathcal{R}_u$ and $\mathbb{E}_{r,u} \times |\mathcal{R}_u| = \sum_{r_k\in\mathcal{R}_u} Pr(r_k|Q)$. Also, $\mathbb{E}_{e,u}$ is defined as the average time being queried for each entity in $\mathcal{E}_u$. $\mathbb{E}_{r,v}, \mathbb{E}_{e,v}$ are defined similarly.

To adapt to the user's interest shift, the summarized PKG should have $\phi(\mathcal{U}, Q) < \phi(\mathcal{V}, Q)$. According to Eq. 23, it equals to

$$\begin{aligned}
\sum_{e\in\mathcal{E}_u} \log\Pr(e|Q) + \sum_{x_{ijk}\in\mathcal{T}_u} \log\Pr(x_{ijk}|Q) &< \sum_{e\in\mathcal{E}_v} \log\Pr(e|Q) + \sum_{x_{ijk}\in\mathcal{T}_v} \log\Pr(x_{ijk}|Q) \\
\prod_{e\in\mathcal{E}_u} \Pr(e|Q) \prod_{x_{ijk}\in\mathcal{T}_u} \Pr(x_{ijk}|Q) &< \prod_{e\in\mathcal{E}_v} \Pr(e|Q) \prod_{x_{ijk}\in\mathcal{T}_v} \Pr(x_{ijk}|Q) \\
\prod_{e\in\mathcal{E}_u} \Pr(e|Q) \prod_{x_{ijk}\in\mathcal{T}_u} \lambda\Pr(e_i|Q)\Pr(r_k|Q)\Pr(e_j|Q) &< \prod_{e\in\mathcal{E}_v} \Pr(e|Q) \prod_{x_{ijk}\in\mathcal{T}_v} \lambda\Pr(e_i|Q)\Pr(r_k|Q)\Pr(e_j|Q)
\end{aligned} \tag{34}$$

We approximate this inequality by evaluating the expectations of items on both sides.

$$\begin{aligned}
\prod_{e\in\mathcal{E}_u} \mathbb{E}_{e,u} \prod_{x_{ijk}\in\mathcal{T}_u} \lambda\mathbb{E}_{e,u}\mathbb{E}_{r,u}\mathbb{E}_{e,u} &< \prod_{e\in\mathcal{E}_v} \mathbb{E}_{e,v} \prod_{x_{ijk}\in\mathcal{T}_v} \lambda\mathbb{E}_{e,v}\mathbb{E}_{r,v}\mathbb{E}_{e,v} \\
\lambda^{|\mathcal{T}_u|}\mathbb{E}_{e,u}^{|\mathcal{E}_u|}\mathbb{E}_{e,u}^{2|\mathcal{T}_u|}\mathbb{E}_{r,u}^{|\mathcal{T}_u|} &< \lambda^{|\mathcal{T}_v|}\mathbb{E}_{e,v}^{|\mathcal{E}_v|}\mathbb{E}_{e,v}^{2|\mathcal{T}_v|}\mathbb{E}_{r,v}^{|\mathcal{T}_v|} \\
\lambda^{|\mathcal{T}_u|}\mathbb{E}_{e,u}^{|\mathcal{E}_u|+2|\mathcal{T}_u|}\mathbb{E}_{r,u}^{|\mathcal{T}_u|} &< \lambda^{|\mathcal{T}_v|}\mathbb{E}_{e,v}^{|\mathcal{E}_v|+2|\mathcal{T}_v|}\mathbb{E}_{r,v}^{|\mathcal{T}_v|} \\
\lambda^{|\mathcal{T}_u|}(\frac{2a(1+\alpha c_u)}{|\mathcal{E}_u|})^{|\mathcal{E}_u|+2|\mathcal{T}_u|}\mathbb{E}_{r,u}^{|\mathcal{T}_u|} &< \lambda^{|\mathcal{T}_v|}(\frac{2b(1+\alpha c_v)}{|\mathcal{E}_v|})^{|\mathcal{E}_v|+2|\mathcal{T}_v|}\mathbb{E}_{r,v}^{|\mathcal{T}_v|} \\
\lambda'^{|\mathcal{T}_u|}(\frac{2a(1+\alpha c_u)}{|\mathcal{E}_u|})^{|\mathcal{E}_u|+2|\mathcal{T}_u|}(a|\mathcal{R}_v|)^{|\mathcal{T}_u|} &< \lambda'^{|\mathcal{T}_v|}(\frac{2b(1+\alpha c_v)}{|\mathcal{E}_v|})^{|\mathcal{E}_v|+2|\mathcal{T}_v|}(b|\mathcal{R}_u|)^{|\mathcal{T}_v|}
\end{aligned} \tag{35}$$

If $\mathcal{U}$ and $\mathcal{V}$ have similar size, i.e. $|\mathcal{E}_u| \approx |\mathcal{E}_v|$ and $|\mathcal{R}_u| \approx |\mathcal{R}_v|$, then we can further approximate by

$$\begin{aligned}
(a(1+\alpha c_u))^{|\mathcal{E}|+2|\mathcal{T}|}a^{|\mathcal{T}|} &< (b(1+\alpha c_v))^{|\mathcal{E}|+2|\mathcal{T}|}b^{|\mathcal{T}|} \\
a^{|\mathcal{E}|+3|\mathcal{T}|}(1+\alpha c_u)^{|\mathcal{E}|+2|\mathcal{T}|} &< b^{|\mathcal{E}|+3|\mathcal{T}|}(1+\alpha c_v)^{|\mathcal{E}|+2|\mathcal{T}|} \\
a(\frac{(1+\alpha c_u)}{(1+\alpha c_v)})^{\frac{|\mathcal{E}|+2|\mathcal{T}|}{|\mathcal{E}|+3|\mathcal{T}|}} &< b
\end{aligned} \tag{36}$$

It means that b needs to be roughly the same scale with a to finish the interest shift and completes the proof. □

E.5 Proof of Theorem 3.2

Proof. Here, we give the proof for the non-adaptability of PEGASUS [27].

Similar to proof on in E.5 for GLIMPSE [55], we also assume two areas $\mathcal{U}$ and $\mathcal{V}$ with connectivity c_u and c_v. Let the user initially query $\mathcal{U}$ for a times. The query set $|Q_u| = a$. The user's interest then shifts to $\mathcal{V}$ and queries $\mathcal{V}$ for b times.

For PEGASUS here, we show that the historical search on $\mathcal{U}$ will permanently give high weights to some edges in $\mathcal{U}$, and hence these summarized items in $\mathcal{U}$ (with high weights) are hard to get replaced by items of later interests.

Consider two search queries in $\mathcal{U}$ search u_1 and u_2 (i.e., target nodes), then the edge u_1u_2 gets weight $\frac{1}{Z}$ since $D(u_1, \mathcal{T}) = D(u_2, \mathcal{T}) = 0$. The edge between u_2 and any u_1's 1-hop neighbor u_1' gets the weight at least $\frac{\alpha^{-1}}{Z}$ because $D(u_1', \mathcal{T}) <= 1$ and $D(u_2, \mathcal{T}) = 0$. Similarly, we can infer that the edge between any u_1's x-hop neighbor and u_2's any y-hop neighbor gets weight $\frac{\alpha^{-(x+y)}}{Z}$.

Thus, no matter how the user searches in $\mathcal{V}$, these weights remain unchanged. The newly assigned weights in $\mathcal{V}$ will not exceed $\frac{\alpha^{-1}}{Z}$. But from the previous search, many edges in $\mathcal{U}$ can have weight $\frac{\alpha^{-1}}{Z}$, which means that those items (in $\mathcal{U}$ with high weights) are hard to be replaced and will still appear in the summarized graph even though the user's interest has shifted to $\mathcal{V}$. □

E.6 Proof of Theorem 4.2

Proof. First, we assume the query set and its volume $|\mathcal{Q}_u| = a$, and the user's interest shifts to $\mathcal{V}$ and queries $\mathcal{V}$ for b times. Then, we denote the total query is Q, where $Q_u \subseteq Q$ and $Q_v \subseteq Q$. Define $\mathbb{E}_{e,u}$ and $\mathbb{E}_{e,v}$ as the average time being queried for each entity in $\mathcal{E}_u$ and $\mathcal{E}_v$, respectively; and $\mathbb{E}_{r,u}$ and $\mathbb{E}_{r,v}$ as the average time being queried for each relation in $\mathcal{R}_u$ and $\mathcal{R}_v$, respectively.

$$\begin{aligned}\sum_{x_{ijk}\in\mathcal{T}_u}\log\Pr(x_{ijk}|\mathcal{Q}) &< \sum_{x_{ijk}\in\mathcal{T}_v}\log\Pr(x_{ijk}|\mathcal{Q})\\ \prod_{x_{ijk}\in\mathcal{T}_u}\Pr(e_i|\mathcal{Q})\Pr(r_k|\mathcal{Q})\Pr(e_j|\mathcal{Q}) &< \prod_{x_{ijk}\in\mathcal{T}_v}\Pr(e_i|\mathcal{Q})\Pr(r_k|\mathcal{Q})\Pr(e_j|\mathcal{Q})\\ \mathbb{E}_{e,u}^{|\mathcal{E}_u|+2|\mathcal{T}_u|}\mathbb{E}_{r,u}^{|\mathcal{T}_u|} &< \mathbb{E}_{e,v}^{|\mathcal{E}_v|+2|\mathcal{T}_v|}\mathbb{E}_{r,v}^{|\mathcal{T}_v|}\end{aligned} \tag{37}$$

Considering the decay, the expectations are computed as

$$\begin{aligned}|\mathcal{E}_u|\mathbb{E}_{e,u} &= \gamma^b\sum_{t=0}^{a-1}\gamma^t 2(\sum_{i=0}^{d}(\alpha c_u)^i) = \frac{\gamma^b-\gamma^{a+b}}{1-\gamma}\cdot 2\cdot\frac{1-(\alpha c_u)^{d+1}}{1-\alpha c_u}\\ |\mathcal{E}_v|\mathbb{E}_{e,v} &= \sum_{t=0}^{b-1}\gamma^t 2(\sum_{i=0}^{d}(\alpha c_v)^i) = \frac{1-\gamma^b}{1-\gamma}\cdot 2\cdot\frac{1-(\alpha c_v)^{d+1}}{1-\alpha c_v}\\ |\mathcal{R}_u|\mathbb{E}_{r,u} &= \gamma^b\sum_{t=0}^{a-1}\gamma^t = \frac{\gamma^b-\gamma^{a+b}}{1-\gamma},\ |\mathcal{R}_v|\mathbb{E}_{r,v} = \sum_{t=0}^{b-1}\gamma^t = \frac{1-\gamma^b}{1-\gamma}\end{aligned} \tag{38}$$

Plug these above into Eq. 37. Then, assuming $\mathcal{U}$ and $\mathcal{V}$ have similar size, i.e. $|\mathcal{E}_u| \approx |\mathcal{E}_v|$ and $|\mathcal{R}_u| \approx |\mathcal{R}_v|$ gives

$$(\frac{\gamma^b-\gamma^{a+b}}{1-\gamma}\cdot\frac{1-(\alpha c_u)^{d+1}}{1-\alpha c_u}\cdot\frac{2}{|\mathcal{E}_u|})^{|\mathcal{E}_u|+2|\mathcal{T}_u|}(\frac{\gamma^b-\gamma^{a+b}}{|\mathcal{R}_v|(1-\gamma)})^{|\mathcal{T}_u|} < (\frac{1-\gamma^b}{1-\gamma}\cdot\frac{1-(\alpha c_v)^{d+1}}{1-\alpha c_v}\cdot\frac{2}{|\mathcal{E}_v|})^{|\mathcal{E}_v|+2|\mathcal{T}_v|}(\frac{1-\gamma^b}{|\mathcal{R}_u|(1-\gamma)})^{|\mathcal{T}_v|} \tag{39}$$

If $\mathcal{U}$ and $\mathcal{V}$ have similar size, i.e. $|\mathcal{E}_u| \approx |\mathcal{E}_v|$ and $|\mathcal{R}_u| \approx |\mathcal{R}_v|$, then we can further approximate by

$$\begin{aligned}((\gamma^b-\gamma^{a+b})\cdot\frac{1-(\alpha c_u)^{d+1}}{1-\alpha c_u})^{|\mathcal{E}_u|+2|\mathcal{T}_u|}(\gamma^b-\gamma^{a+b})^{|\mathcal{T}_u|} &< ((1-\gamma^b)\cdot\frac{1-(\alpha c_v)^{d+1}}{1-\alpha c_v})^{|\mathcal{E}_v|+2|\mathcal{T}_v|}(1-\gamma^b)^{|\mathcal{T}_v|}\\ (\gamma^b-\gamma^{a+b})\cdot(\frac{1-(\alpha c_u)^{d+1}}{1-\alpha c_u})^{\frac{|\mathcal{E}_u|+2|\mathcal{T}_u|}{|\mathcal{E}_u|+3|\mathcal{T}_u|}} &< (1-\gamma^b)\cdot(\frac{1-(\alpha c_v)^{d+1}}{1-\alpha c_v})^{\frac{|\mathcal{E}_v|+2|\mathcal{T}_v|}{|\mathcal{E}_v|+2|\mathcal{T}_v|}}\\ b &> \log_\gamma\frac{1}{\frac{A}{B}(1-\gamma^a)+1}\end{aligned} \tag{40}$$

where $A = (\frac{1-(\alpha c_u)^{d+1}}{1-\alpha c_u})^{\frac{|\mathcal{E}_u|+2|\mathcal{T}_u|}{|\mathcal{E}_u|+3|\mathcal{T}_u|}}$ and $B = (\frac{1-(\alpha c_v)^{d+1}}{1-\alpha c_v})^{\frac{|\mathcal{E}_v|+2|\mathcal{T}_v|}{|\mathcal{E}_v|+2|\mathcal{T}_v|}}$. And $\log_\gamma\frac{1}{\frac{A}{B}(1-\gamma^a)+1} < \log_\gamma\frac{1}{\frac{A}{B}+1}$ further gives us a bound $b < \log_\gamma\frac{1}{\frac{A}{B}+1}$. □

E.7 Proof of Theorem 4.4

Proof. First, we assume query set $|\mathcal{Q}_u| = a$, and the user's interest shifts to $\mathcal{V}$ and queries $\mathcal{V}$ for b times. Also, we denote the total query is Q, where $Q_u \subseteq Q$ and $Q_v \subseteq Q$.

$$\begin{aligned}\sum_{e\in\mathcal{E}_u}\log\Pr(e|\mathcal{Q}) &< \sum_{e\in\mathcal{E}_v}\log\Pr(e|\mathcal{Q})\\ \prod_{e\in\mathcal{E}_u}\Pr(e|\mathcal{Q}) &< \prod_{e\in\mathcal{E}_v}\Pr(e|\mathcal{Q})\\ \mathbb{E}_{e,u}^{|\mathcal{E}_u|} &< \mathbb{E}_{e,v}^{|\mathcal{E}_v|}\\ (\frac{\gamma^b-\gamma^{a+b}}{1-\gamma}\cdot 2\cdot\frac{1-(\alpha c_u)^{d+1}}{1-\alpha c_u}\cdot\frac{1}{|\mathcal{E}_u|})^{|\mathcal{E}_u|} &< (\frac{1-\gamma^b}{1-\gamma}\cdot 2\cdot\frac{1-(\alpha c_v)^{d+1}}{1-\alpha c_v}\cdot\frac{1}{|\mathcal{E}_v|})^{|\mathcal{E}_v|}\end{aligned} \tag{41}$$

If $\mathcal{U}$ and $\mathcal{V}$ have similar size, i.e. $|\mathcal{E}_u| \approx |\mathcal{E}_v|$, then we can further approximate by

$$(\gamma^b-\gamma^{a+b})\cdot\frac{1-(\alpha c_u)^{d+1}}{1-\alpha c_u} < (1-\gamma^b)\cdot\frac{1-(\alpha c_v)^{d+1}}{1-\alpha c_v} \iff b > \log_\gamma\frac{1}{\frac{A}{B}(1-\gamma^a)+1} \tag{42}$$

where $A = \frac{1-(\alpha c_u)^{d+1}}{1-\alpha c_u}$ and $B = \frac{1-(\alpha c_v)^{d+1}}{1-\alpha c_v}$. Knowing that $\log_\gamma\frac{1}{\frac{A}{B}(1-\gamma^a)+1} < \log_\gamma\frac{1}{\frac{A}{B}+1}$ gives us a bound $b < \log_\gamma\frac{1}{\frac{A}{B}+1}$ □