COREKG: Coreset-Guided Personalized Summarization of Knowledge Graphs

Sohel Aman Khan[1*], **Raghava Mutharaju**[2†], **Supratim Shit**[1]
[1]Department of CSE, IIIT-Delhi, India
[2]Mehta Family School of Data Science and AI, IIT Palakkad, India
{sohelk, supratim}@iiitd.ac.in, raghava@iitpkd.ac.in

Abstract

Knowledge Graphs (KGs) are extensively used across different domains and in several applications. Often, these KGs are very large in size. Such KGs become unwieldy for tasks such as question answering and visualization. Summarization of KGs offers a viable alternative in such cases. Furthermore, personalized KG summarization is crucial in the current data-driven world as it captures the specific requirements of users based on their query patterns. Since it only maintains relevant information, the personalized summaries of KG are small, resulting in significantly smaller storage requirements and query runtime. In this work, we adapt the coreset theory to create personalized KG summaries. For a given dataset and a user-specific query workload, we present an approach that samples a relevant subset of triples using sensitivity-based importance sampling. We ensure that the subset approximates the characteristics of the full dataset with bounded approximation error. We define sensitivity scores that measure the importance of a triple with respect to a user's query workload, which are then used by our coreset construction algorithm. We explicitly focus on personalized knowledge graph summarization by constructing summaries independently for each user based on their query behaviour. Our evaluation on Freebase, WikiData, and DBpedia shows that COREKG delivers higher query-answering accuracy and structural coverage than the state-of-the-art methods, such as GLIMPSE, PPR, iSummary, PEGASUS and APEX^2 while requiring only a tiny fraction of the original graph.

1 Introduction

In the digital age, numerous sources of information are readily available. Knowledge Graphs (KGs) [Peng *et al.*, 2023] are an effective way to represent and reason over interconnected data. Large-scale KGs such as Wikidata, DBpedia, and Freebase [Bollacker *et al.*, 2008; Mendes *et al.*, 2012] have become central to numerous real-world applications, including search engines, recommendation systems, question-answering (QA), and biomedical informatics. However, the vast size and high connectivity of these KGs introduce challenges for efficient querying, visualization, and reasoning. QA systems often struggle with slow query execution, and visualization tools become cluttered and less interpretable at scale [Wang and Cheng, 2024]. Knowledge graph summarization addresses these issues by helping users understand and utilize the data while preserving key structural and semantic information. However, most summarization methods are focused on capturing the general pattern of the original KG [Jalota *et al.*, 2021; Wang and Cheng, 2024]. As a result, the generated summaries tend to be overly generalized and often fail to accommodate the specific information needs of individual users.

Query-based knowledge graph summarization addresses this issue by constructing compact summaries optimized for a given query workload rather than the global graph structure. It improves query efficiency and answer completeness under reduced graph sizes [Song *et al.*, 2018; Niazmand *et al.*, 2022]. However, such workloads are typically defined at an aggregate level. These summaries do not capture individual user preferences, motivating the need for personalized knowledge graph summarization [Jalota *et al.*, 2021].

Considering the above limitation, personalized summarization of KGs has massive potential because it focuses on user interest. It is also efficient in terms of storage and time. Early research like GLIMPSE [Safavi *et al.*, 2019] uses submodular maximization to retrieve user-specific KG summaries using query logs, which enable device-level KG summaries with theoretical guarantees. However, GLIMPSE assumes full access to each user's query history, which has less explicit control over summary sparsity. ISummary [Vassiliou, 2023] creates efficient, personalized Knowledge Graph summaries by analyzing SPARQL query logs to capture collective user interests, reducing computation while improving scalability and relevance. However, the summarization quality in iSummary depends heavily on workload coverage. Furthermore, the sample size of COREKG ensures controlled budget–risk tradeoffs. We present a coreset-based framework

*Corresponding Author
†Raghava Mutharaju contributed to this work when he was at IIIT-Delhi, India.

named COREKG for personalized Knowledge Graph summarization that addresses the limitations of existing personalized approaches [Feldman *et al.*, 2017; Safavi *et al.*, 2019]. By adapting coreset theory to the KG domain, our method provides theoretical guarantees while maintaining computational efficiency [Braverman *et al.*, 2021]. Our approach generates compact, user-centric KG summaries by sampling triples based on sensitivity scores that capture their importance across user queries [Safavi *et al.*, 2019]. In constructing personalized KG summaries, we focus on queries that contain seed nodes (entities of direct user interest) [Safavi *et al.*, 2019] and we provide theoretical bounds that preserve the original query semantics. A high-level overview of the COREKG pipeline, including preprocessing, sensitivity computation, and sampling, is illustrated in the Figure 1.

Coresets are a form of weighted summarization of a dataset with provable guarantees [Feldman and Langberg, 2011]. In the past couple of decades, coresets have been studied for various problems and applications, including clustering [Feldman *et al.*, 2013; Bachem *et al.*, 2018; Shit *et al.*, 2022], regression [He *et al.*, 2021; Chhaya *et al.*, 2020b; Boutsidis *et al.*, 2013], multilinear algebra [Dasgupta *et al.*, 2009; Chhaya *et al.*, 2020a], non-decomposable functions [Malaviya *et al.*, 2024], federated learning [Huang *et al.*, 2022; Shit *et al.*, 2025] etc. The most popular method for constructing a coreset is importance sampling via a sensitivity framework, which assigns an importance score to data points (triples) commensurate with their contribution to a particular query workload, thereby supporting efficient importance-based sampling. Some works [Feldman *et al.*, 2017; Braverman *et al.*, 2021] prove these ideas yield strong approximation bounds. COREKG extends these ideas to graph triples, and its coreset summary approximates the utility of a complete graph for personalized queries.

To fulfill our objective, we construct a coreset—a smaller, weighted subgraph—from the original knowledge graph, tailored specifically to a user's query workload. Instead of evaluating every possible query association across the complete graph, we sample a subset of relevant triples and assign them statistical weights to correct for sampling probability. This ensures our coreset provides a provable approximation guarantee; it preserves the query-answering semantics of the full graph up to a small, bounded relative error while drastically reducing the data footprint. Our empirical evaluation further demonstrates superior performance in query coverage over real-world KG datasets [Vassiliou, 2023; Safavi *et al.*, 2019], and the method is effective even with limited computational resources [Safavi *et al.*, 2019].

To explicitly capture per-user preferences, COREKG constructs user-centric workloads based on seed entities that represent an individual user's interests. These seed entities are extracted from the query workload, and the summary is generated only from queries that mention at least one of these seeds. By repeating this procedure for different seed sets, COREKG naturally adapts to multiple users, enabling personalized knowledge graph summarization rather than relying on a single global workload. Our main contributions are as follows.

- We propose COREKG, a coreset-guided framework for personalized knowledge graph summarization that adapts coreset theory to graph data and provides provable approximation guarantees (Theorem 4.1) with practical scalability.
- We introduce a sensitivity-based importance sampling formulation that quantifies the relevance of triples with respect to user-specific query workloads, ensuring robustness across diverse personalization.
- Our coreset construction provides a worst-case approximation guarantee on the user-specific workload cost. It is independent of the size of the full graph. All query evaluation on the summary is performed on the coreset.
- We conduct extensive experiments on Freebase, DBpedia, and Wikidata, showing that COREKG consistently achieves higher coverage and F1 Score than baseline methods such as APEX^2, GLIMPSE, PEGASUS, iSummary, and PPR.

2 Related Work

In this section, we review prior studies and approaches that are closely related to our work. We highlight existing methods for knowledge graph summarization, query-driven graph reduction and discuss how COREKG builds upon and differs from these efforts.

Summarization refers to generalizing KGs by removing unnecessary information by merging similar nodes and edges into supernodes and superspecs to generalize the information in the graph. This process removes unnecessary information while preserving the generalization of the information in the graph [Jalota *et al.*, 2021; Kumar and Efstathopoulos, 2018]. Most existing summarization approaches primarily emphasize capturing the overall structure of the original KG [Jalota *et al.*, 2021; Wang and Cheng, 2024].

To fix these limitations, Query-based summarization of knowledge graphs aims to construct compact summaries that maximize response completeness and relevance for a workload of specified target queries. Early research demonstrated that lossy summaries can improve query processing time performance, and enable efficient KG search capacity through summaries, especially relevant to query task needs [Song *et al.*, 2018]. Approaches, such as Grouping-Based and Query-Based Summarization, utilize formal methods that enable the significant compression of graph size while retaining complete query performance [Niazmand *et al.*, 2022]. Lastly, a few relatively recent works are looking at attention, or sampling methods to approach scale issues and maintain query workload performance improvements [Shabani *et al.*, 2024]. Other lines of work explore the structure of the KG, with works on incremental summarization, such as RDFQuotient, enabling the dynamic generation of summaries without the need to resummarize static RDF datasets [Goasdoué *et al.*, 2019]. Summarization oriented towards queries has similarly been explicitly positioned for downstream tasks like question answering (QA). For example, in QA, summarization can be used as either a filtering method for relevance or to rank KG summaries while achieving comparable QA efficacy [Jalota *et al.*, 2021]. More generally, summarization has been positioned as an enabler for efficient graph mining and querying

[Koutra, 2021]. Complementary frameworks also speak to how graph-based summarization techniques could be helpful in support of an abstractive summarization or knowledge extraction pipeline beyond query efficiency [Moro *et al.*, 2023]. All of these studies contribute evidence that query-aware graph summarization is a unique area in KG summarization in which both compactness and completeness can be evaluated. But, this summarization can't fulfill user-specific needs. To fix this problem, KGs are summarized by considering user-specific needs. This summarization is known as personalized KG summarization.

GLIMPSE [Safavi *et al.*, 2019] develops a submodular optimization theory for the personalized KG summarization for data from query logs. iSummary [Vassiliou, 2023] proposed summarization through workloads that generalize across aggregate user behaviours. Yet, limitation exist for both studies. GLIMPSE assumes each individual has a dense set of histories for generating results, and iSummary relies on the conclusive coverage of SPARQL workloads. The PEGASUS [Kang *et al.*, 2022] method generates summary graphs for individuals in linear time and uses a personalized cost function to generate summaries through greedily combining supernodes to act as underlying groups that allows for expected query answering under a high compression, however it is still a heuristic, which lacks slight approximation, and it assumes the target set of nodes are static, reducing its applicability in rapidly changing and dynamic environments, where direction and interests are constantly evolving. The APEX^2 [Li *et al.*, 2025] method modulates personalized assistance in Knowledge Graphs incrementally while modeling a more diverse set of evolving query workloads. It generates summaries in real time based on modeled heat diffusion, while adhering to tight constraints that ensure provable guarantees. This is achieved under the circumstance where the adaptive current workload generated by the querying node provides for the activity of interactions. However, APEX^2 [Li *et al.*, 2025] includes complete reliance on representative query workloads, and can fail on queries that are sparse or extremely skewed. APEX^2 also assumes that modeling heat diffusion behaves as expected between the workload nodes based on topological proximity, and does not expect semantic relationships as captured in modeling KGs. COREKG circumvents both challenges and gaps by utilizing coreset sampling to provide theoretical guarantees while accommodating sparse or skewed query logs.

Recommendation systems usually personalize suggestions based on user preferences that can be inferred from query histories. For example, model behaviour studies, such as [Čebirić *et al.*, 2015; Cheng *et al.*, 2011; Gunaratna *et al.*, 2015], are designed to characterize user interests using SPARQL queries to extract relevant entities and paths to create personalized summaries. Additionally, some diversity-aware methods, including FACES [Gunaratna *et al.*, 2015], are designed to balance informative against representation diversity in entity-centric summaries.

Personalized PageRank (PPR) [Haveliwala *et al.*, 2003] presents a method for designing the ranking method for user-centered exploration utilizing the graph. However, these methods often do not provide any explicit controls for summary size or utility. The COREKG includes an advancement over these approaches by introducing a generalized sampling framework that incorporates user-specific queries to lay the groundwork for explicit personalizability capabilities.

3 Preliminaries

The preliminaries here provide the foundation for understanding our methodology and analysis.

Knowledge Graph (KG): A Knowledge Graph (KG) [Peng *et al.*, 2023] is a structured representation of data, where entities are modeled as nodes and their relationships as labeled edges. Formally, a KG is defined as $G = (V, E, T)$, where V is the set of nodes (entities), E is the set of edge labels (relations), and $T \subseteq V \times E \times V$ is the set of RDF triples representing relational facts. SPARQL is commonly used to access and manipulate these triples, enabling complex pattern matching and information retrieval. The query workload on a KG consists of sets of SPARQL queries that reflect typical information needs, varying in complexity, frequency, and the types of operations they perform.

Seed node: A seed node is an entity that represents a user's interest and is used to guide personalization. In this paper, Seed nodes are used to filter the global query log to construct a user-specific query workload.

Personalized KG summarization: A Personalized Knowledge Graph summary [Safavi *et al.*, 2019; Vassiliou, 2023; Kang *et al.*, 2022; Li *et al.*, 2025] refers to a summary extracted from a knowledge graph that is tailored to a user's interests and preferences.

In this setting, seed nodes represent a user's interests by identifying the entities around which personalization is performed. In our work, seed entities define user-specific query workloads that guide the selection of relevant triples. As a result, the generated KG summary focuses on the parts of the graph that are most important to the user's information needs [Vassiliou, 2023]. Example: If a user frequently queries about *workplaces* and *industries*, the chosen seeds may include entities such as *Bob* and *XCorp*. A personalized summary will then prioritize triples related to these seeds, such as: (Bob, worksAt, XCorp) and (XCorp, industry, Tech)

Coreset: A Coreset [Bachem *et al.*, 2017] is a small, weighted subset of data that approximates a cost function with provable error bounds. We adapt this concept to Knowledge Graphs. Given a KG $G = (V, E, T)$ with triples T, and a query workload Q, we define the cost of answering Q on the full graph as $\text{cost}(G, Q) = \sum_{t \in T} \sum_{q \in Q} f_q(t)$, where $f_q(t) \in 0, 1$ indicates whether triple t is relevant to query q.

A coreset $C = (V', E', T')$, where $V' \subseteq V$, $E' \subseteq E$, and $T' \subseteq T$, is a weighted subset of triples. The coreset is constructed by sampling triples according to their importance and then assigning them appropriate weights. Each triple $t \in C$ receives a weight $w(t)$ that compensates for its sampling probability, ensuring unbiased estimation of the full cost. The weighted cost over the coreset is defined as $\text{cost}(C, Q) = \sum_{t \in C} \sum_{q \in Q} w(t) \cdot f_q(t)$. Formally, if each triple t is sampled with probability $p(t)$ and the total number of samples is m, the weight is $w(t) = \frac{1}{m \cdot p(t)}$. This weight $w(t)$ ensures that the expected contribution of each

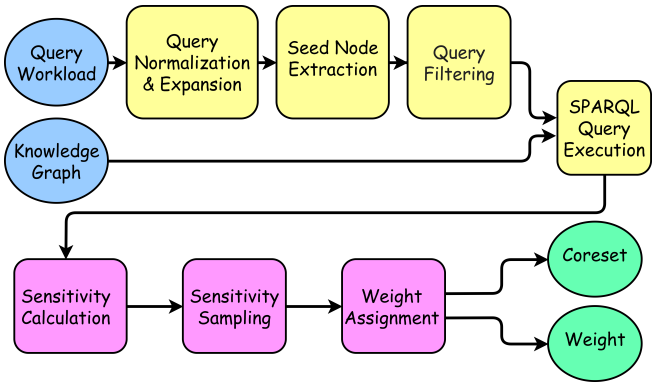

Figure 1: COREKG Framework

triple in C matches its true contribution in G, thereby removing sampling bias. Intuitively, triples that are sampled with lower probability receive higher weights, since they represent a larger portion of the original graph, whereas frequently sampled triples receive smaller weights. The coreset satisfies $\left|\text{cost}(G, Q) - \text{cost}(C, Q)\right| \leq \varepsilon \cdot \text{cost}(G, Q)$ with this weighting and high probability. This definition is generic and applies to any query workload Q. In the COREKG framework, it is applied to personalized query workloads derived from individual users, resulting in a user-specific coreset.

4 Methodology

We present COREKG (Figure 1), a coreset-based importance sampling approach for personalized knowledge graph summarization. The method produces compact, query-relevant subgraphs that approximate the behaviour of the full graph with respect to a user's query workload while guaranteeing bounded approximation error. Here, triples are scored based on how frequently or critically they contribute to a user-specific query workload, where more relevant triples are assigned higher sampling probabilities to preserve informative structure. COREKG employs sensitivity-based sampling to prioritize triples by their explicit contribution to query answering, ensuring preservation of user-relevant structural and semantic information. It operates as a per-user summarization framework, where user modeling, workload construction, sensitivity estimation, and coreset sampling are performed independently for each user and conditioned on their personalized query workload.

4.1 Preprocessing Pipeline

In COREKG, preprocessing provides user modeling by representing each user through a small set of seed entities extracted from query history. These seeds define a personalized query workload that filters relevant queries from the global log and enables user-specific knowledge graph summarization. The preprocessing pipeline comprises four stages: query normalization and expansion, seed extraction, query filtering, and SPARQL execution.

Query Normalization and Expansion: In order to maintain semantic consistency within the query workload, we need to first normalize all SPARQL queries. Normalization consists of two stages: First, we take the original SPARQL query and eliminate duplicate PREFIX declarations to remove unnecessary repeated boilerplate in the process. Second, we expand all prefixed terms to complete URIs using a fixed prefix map via regular expression matching and replacement. This ensures that all relations and entities are consistently and unambiguously represented across the query set. We keep fully expanded queries for future processing.

Seed Node Extraction: After we have normalized the query workload, we extract the unique entities from the triple patterns that we can use as the seed nodes that can facilitate personalized summaries. For every unique query, we parse the triple patterns and locate URIs and useful tokens while omitting variables and literals. We use regular expressions in order to extract the URIs while filtering the SPARQL syntax.

Query Filtering: We apply final filtering to the expanded query set to keep only the queries that mention one of the sampled seed nodes. This filtering step enables the summary construction to begin from the most appropriate and well-requested sections of the knowledge graph based on actual user queries. The filtered queries can be used to execute SPARQL queries against the KG and guide the development of the personalized summaries.

SPARQL Query Execution: Preprocessed SPARQL queries are executed using Apache Jena Fuseki[1], an open-source SPARQL server, with knowledge graphs stored in the TDB2 backend for disk-based indexing and query optimization. Filtered queries are evaluated through Fuseki's SPARQL endpoint using the ARQ engine, which performs internal indexing and query rewriting for efficient retrieval. Result sets are streamed and parsed to extract relevant triples for sensitivity computation in COREKG. This deployment supports concurrent execution, persistent endpoints, and low-latency access for large graphs such as Freebase, DBpedia, and Wikidata. This provides a scalable and reproducible integration between query preprocessing and coreset.

Each distinct seed set uniquely identifies a user, and all subsequent stages of the framework operate exclusively on the corresponding user-specific workload.

4.2 Sensitivity Score

As shown in Algorithm 1 (Lines 4–6), COREKG computes a user-specific sensitivity score for each triple based on its contribution to the queries issued by an individual user. Let u denote a user characterized by a seed entity set extracted from a query set. All sensitivity computations are then performed with respect to filtered user-specific (i.e., personalized) workload Q. Formally, the sensitivity of a triple t for a given user u is defined as:

$$s(t) = \sum_{q \in Q} \frac{1}{|T_q|} \cdot \mathbb{I}[t \in T_q],$$

where T_q denotes the set of triples relevant to query $q \in Q$, and $\mathbb{I}[t \in T_q]$ is an indicator function that returns 1 if t contributes to the answer of q, and 0 otherwise. T_q denotes the set of triples matched by the triple patterns of query q under

[1]https://jena.apache.org/documentation/fuseki2/

Algorithm 1 COREKG

Require: KG $G = (V, E, T)$; Query set Q; #samples m
Ensure: Weighted summary $C \subseteq T$ with weights $w(t)$
1: **for all** $q \in Q$ **do**
2: $T_q \leftarrow \{t \mid \forall t$ relevant to query $q\}$
3: **end for**
4: **for all** $t \in T$ **do**
5: $s(t) \leftarrow \sum_{q \in Q} \frac{1}{|T_q|} \cdot \mathbb{I}[t \in T_q]$ ▷ Triple sensitivity
6: **end for**
7: $S \leftarrow \sum_{t \in T} s(t) = |Q|$
8: **for all** $t \in T$ **do**
9: $p(t) \leftarrow \frac{s(t)}{S}$ ▷ Sampling probability
10: **end for**
11: Sample m triples iid from T using distribution p
12: Let C be the sampled triples
13: **for all** $t \in C$ **do**
14: $w(t) \leftarrow \frac{S}{m \cdot s(t)} = \frac{|Q|}{m \cdot s(t)}$
15: **end for**
16: **return** subset C and weight function w

standard SPARQL evaluation. This definition ensures that the sensitivity of a triple is explicitly conditioned on the interests of user u, as encoded by their personalized query workload. Intuitively, triples that appear frequently in a given user's queries receive higher sensitivity scores. Similarly, triples that occur in queries with small answer supports are emphasized and are therefore more likely to be included in the user's summary. As a result, different users induce different sensitivity distributions over the same underlying graph, leading to distinct personalized summaries. The total sensitivity for user u is defined as $S = \sum_{t \in T} s(t)$.

The total sensitivity S with respect to a fixed user workload Q, is bounded on $|Q|$. This result simplifies normalization and guarantees that sensitivity-based sampling remains unbiased at the user level. Once sensitivities are computed, they are normalized into a user-specific sampling distribution. For each triple $t \in T$, the probability of sampling it for user u is given by $p(t) = \frac{s(t)}{S}$. This normalization ensures that the summary construction process is directly aligned with the query behaviour of each individual user rather than with the global query log.

We emphasize that sensitivity is not a universal measure of importance, but rather a quantity that varies depending on the user. Consequently, the same triple may be assigned different sensitivity values across users, depending on their respective query workloads. This user-conditioned notion of sensitivity constitutes the core mechanism through which personalization is realized in COREKG.

4.3 Coreset Construction

Given a user u and the corresponding personalized query workload Q, we construct a coreset via sensitivity-based sampling, as detailed in Algorithm 1 (Lines 11–15). Each sampled triple is assigned a weight to correct for sampling bias and ensure unbiased cost estimation. Algorithm 1 is executed independently for each user using their respective workload Q, resulting in a separate, user-specific coreset.

To correct for the sampling bias introduced by preferential selection, each sampled triple is assigned a weight: $w(t) = \frac{1}{m \cdot p(t)} = \frac{S}{m \cdot s(t)}$ where m is the sample size. This weighting guarantees that the cost computed from the coreset is unbiased, i.e., the expected cost equals the true cost of the full data. The weights also ensure bounded variance,

COREKG conditions the sampling process on a user-specific query workload. For each user, the integration of filtered workload Q enables COREKG to generate truly personalized summaries. This construction is repeated independently for each user, producing user-specific summaries with independent approximation guarantees.

Theorem 4.1 (Coreset Size)**.** *Let C and w be the subset and weight function returned from the algorithm 1 over query workload Q, and let $X = \mathrm{cost}(G, Q)$ be the true cost on the full graph. Then, with probability at least $1 - \delta$ we have, $|\mathrm{cost}(C, Q) - \mathrm{cost}(G, Q)| \leq \epsilon \cdot \mathrm{cost}(G, Q)$ provided the number of samples m satisfies:*

$$m \geq \frac{8|Q|}{\epsilon^2} \cdot \log\left(\frac{1}{\delta}\right).$$

The above theorem can be proved by applying Bernstein's inequality and sensitivity analysis [Khan *et al.*, 2026].

5 Evaluation

We do an extensive empirical evaluation of real-world datasets to assess the quality of our summarization method. The evaluation is performed on real-world KGs with SPARQL queries. For evaluation, we divide queries into train and test with a ratio of 80:20. Our focus is to test the generality and scalability of our proposed approach. Query coverage is used as the evaluation metric. The implementation of this methodology is available at https://github.com/SohelKResearch/COREKG.

5.1 Experimental Design and Personalization

All experiments are conducted in a personalized knowledge graph summarization setting. Each user is modeled by a profile defined as a set of seed entities. Each seed entity is extracted from the train query workload, which represents user interests. For each dataset, we construct 15 user profiles. Each user profile consists of 5 seed entities sampled uniformly at random from non-variable entities appearing in training queries. These seeds define a personalized workload by retaining only queries that mention at least one seed entity. To ensure fair comparison, all baseline methods are constrained to produce summaries with the same size budget, measured as the number of retained triples. Our coreset sampling size is determined by theoretical bounds from coreset theory, guaranteeing approximation quality independent of the original KG size. This design ensures that observed performance differences stem from summarization quality rather than summary size. Reported results are averaged across users. COREKG performs user-specific preprocessing, including query filtering and workload construction, prior to sensitivity computation and coreset generation, ensuring fully

personalized summaries. To ensure fair comparison, all models are evaluated exclusively on the same test queries associated with each user profile.

While COREKG is evaluated under static workload settings, the sensitivity formulation is additive over queries. This allows incremental updates as new queries arrive without recomputing from scratch. This shows that COREKG naturally supports streaming coresets. We focus on static workloads for fair comparison with prior methods such as GLIMPSE and iSummary.

5.2 Datasets

We evaluate COREKG on three widely used real-world knowledge graphs: Freebase, DBpedia, and Wikidata. These datasets differ substantially in scale, schema richness, and domain coverage. These datasets allow us to assess the robustness of personalized knowledge graph summarization under diverse and realistic query workloads.

Freebase [Bollacker *et al.*, 2008][2]: It is a large-scale knowledge base developed by Metaweb, containing a large number of RDF triples describing entities and their relationships across diverse domains. The KG Size is 1.9 billion with a query size of 4,737. For query workloads, we use SPARQL queries from the WebQSP dataset [Yih *et al.*, 2016], which consists of natural language questions paired with corresponding SPARQL queries over Freebase. In our experiments, we retain only the SPARQL queries. These queries exhibit non-trivial structures, including multiple triple patterns and joins, making them suitable for evaluating query-driven summarization methods.

DBpedia [Mendes *et al.*, 2012][3]: It is a large-scale knowledge graph extracted from structured content in Wikipedia. We use the latest DBpedia core dataset, which contains a large number of RDF triples spanning a wide range of entity types and relations. The KG Size is 0.53 billion with a query size of 82,519. We use SPARQL query logs from the Linked SPARQL Queries (LSQ) dataset [Stadler *et al.*, 2024][4]. The raw LSQ logs are processed using the `lsq-clean` toolkit[5] to remove malformed, duplicated, and template-generated queries. The resulting workload contains diverse query shapes, including star-shaped and multi-join patterns, reflecting realistic usage of DBpedia in practice.

Wikidata[6]: It is a large, collaboratively maintained knowledge graph curated by the Wikimedia community. The KG Size is 25.33 billion with a query size of 30,154. We use the latest full RDF dump, which contains a large number of triples describing entities, properties, and statements across a wide range of domains. For Wikidata query workloads, we use SPARQL queries from the LC-QuAD dataset 2.0, obtained from the publicly available release[7]. LC-QuAD provides complex question–query pairs designed to test reasoning over Wikidata.

5.3 Evaluation Metrics

We assess summarization methods using two key metrics: structural coverage reflecting how well the summary preserves query patterns, and the F1 score evaluating answer-retrieval accuracy.

Coverage: Given a personalized summary S for a user s, a query workload Q, and two weights w_n and w_p for nodes and edges, respectively, we define the coverage as:

$$\text{Coverage}(Q, S, s) = \frac{1}{n} \sum_{s \in q_i} \left(w_n \frac{\text{snodes}(S, q_i)}{\text{nodes}(q_i)} + w_p \frac{\text{sedges}(S, q_i)}{\text{edges}(q_i)} \right)$$

Here, $\text{nodes}(q_i)$ and $\text{edges}(q_i)$ represent the total number of nodes and edges in query q_i, while $\text{snodes}(S, q_i)$ and $\text{sedges}(S, q_i)$ denote the number of those nodes and edges that are also included in the summary S. In our setup, we assign equal importance to nodes and edges by setting $w_n = 0.5$ and $w_p = 0.5$.

F1 Score: The F1 score measures the query-answering accuracy of a summary S by comparing its retrieved answers to the ground-truth answers obtained from the full knowledge graph. For each query, we compute true positives (TP) as answers returned by both S and the full KG, false positives (FP) as answers returned by S but not found in the full KG, and false negatives (FN) as answers present in the full KG but missing from S. From these, we calculate precision $P = \frac{\text{TP}}{\text{TP+FP}}$ and recall $R = \frac{\text{TP}}{\text{TP+FN}}$, and use them to derive the F1 score as $\text{F1} = 2\,\frac{P\,R}{P+R}$.

Coverage and F1 Score are computed independently for each user-specific workload and then averaged across users.

5.4 Baseline Comparison

We compare COREKG with several baseline methods that vary in personalization strategies and their ability to preserve query-relevant information. Table 1 reports F1 scores reflecting query-answering accuracy and workload coverage. Graph-proximity-based methods such as PPR and GLIMPSE rely on random walks or submodular selection over query logs; although they achieve moderate coverage in some cases, their low F1 scores indicate loss of essential relational structure, particularly for complex SPARQL queries under small summary budgets or sparse histories. iSummary improves scalability by learning from aggregate workloads but limits personalization, resulting in consistently low coverage and F1 scores. PEGASUS and APEX^2 improve coverage using heuristic node merging and adaptive heat diffusion. However, PEGASUS lacks formal guarantees and may lose fine-grained semantics, leading to inconsistent F1 scores on complex datasets such as Wikidata. APEX^2 assumes dense query activity and alignment between semantic relevance and graph proximity, which does not always hold in practice. In contrast, COREKG consistently achieves the highest F1 scores while maintaining strong coverage. It avoids reliance on structural heuristics and instead provides explicit approximation guarantees through sensitivity-based coreset sampling that selects triples according to their contribution to user-specific query workloads. As a result, COREKG delivers ro-

[2]https://developers.google.com/freebase

[3]https://databus.dbpedia.org/dbpedia/collections/latest-core

[4]http://lsq.aksw.org/

[5]https://github.com/sparqeology/lsq-clean?tab=readme-ov-file

[6]https://dumps.wikimedia.org/wikidatawiki/entities/

[7]https://huggingface.co/datasets/mohnish/lc_quad/blob/main/data.zip

bust and controllable performance even under sparse, skewed, or evolving workloads.

Method	F1(%)			Coverage(%)		
	Freebase	DBpedia	WikiData	Freebase	DBpedia	WikiData
COREKG	**56.35**	**58.64**	**52.10**	**66.92**	**68.07**	**67.46**
APEX^2	49.76	47.52	45.29	56.21	42.06	65.18
iSummary	21.83	19.38	30.64	23.16	12.55	39.72
PEGASUS	3.18	2.46	2.90	63.03	66.32	33.84
GLIMPSE	16.21	32.68	28.94	31.39	34.62	11.93
PPR	8.64	11.39	9.75	10.93	5.04	4.61

Table 1: F1 scores and Coverage across datasets

To further analyse the behaviour of different methods under varying summary budgets, we report performance across multiple budget settings. COREKG consistently outperforms all baselines across the entire budget range. At a budget of 1000 triples (DBpedia), COREKG achieves 30.76% coverage, and 30.12% F1 and APEX^2 achieves 11.72% coverage and 18.26% F1. PEGASUS achieves 32.80% coverage but only 1.72% F1, indicating poor query relevance. At a budget of 30000 triples (DBpedia), COREKG achieves 66.72% coverage and 57.83% F1, while APEX^2 reaches 44.48% coverage and 47.12% F1. PEGASUS achieves 65.32% coverage and 2.68% F1 and shows negligible improvement beyond this point, indicating early convergence. In the table 1 we use a budget of 60000. Finally, at a budget of 100000 triples (DBpedia), COREKG achieves 74.8% coverage and 60.72% F1, while APEX^2 reaches 50.83% coverage and 49.82% F1, and PEGASUS has already converged earlier and does not improve further. Methods such as iSummary and GLIMPSE also show minimal gains beyond earlier budgets.

Overall, COREKG consistently outperforms all baselines across budgets, demonstrating its ability to preserve query-relevant structure.

5.5 Time & Space Complexity Analysis

The COREKG framework includes preprocessing stages such as query normalization, seed node extraction, and query filtering. Let L denote the average characters in a query and Q denote the number of queries. Query normalization and seed node extraction require scanning each query using regular expressions, resulting in $O(QL)$ time. The query filtering operates in $O(QL)$ time through URI token extraction and set-based intersection with the seed set, and the corresponding space complexity is $O(QL)$. SPARQL query execution is independent of COREKG and is externally handled using Apache Jena Fuseki. Thus, its time and space complexity depend on the underlying SPARQL engine, indexing strategy, and query structure. Sensitivity computation requires $O(TQ)$ with $O(T)$ space, where T is the total number of triples. Probability normalization, sampling, and weight assignment require $O(T)$ time and $O(m)$ space, where m is the coreset sample size.

6 Ablation Study

To evaluate the role of each component of COREKG, we perform an Ablation study in which we investigate the effect of each component on the overall performance while maintaining the same size summary.

First, COREKG-Uniform, which replaces sensitivity-based sampling with uniform sampling, collapses to near-zero performance, achieving almost 0% F1 across all datasets and negligible coverage. This confirms that uniform random sampling is ineffective for identifying relevant triples. This indicates the need to use query-based sensitivity to preserve meaningful query structure.

Second, COREKG-Global, which removes user-specific conditioning and constructs summaries using a global workload, performs significantly worse than the full model. It achieves F1 scores of 39.78% (Freebase), 44.06% (DBpedia), and 37.63% (Wikidata), with corresponding coverage scores of 47.34% (Freebase), 51.24% (DBpedia), and 49.17% (Wikidata). COREKG-Global performed worse than the full COREKG model, demonstrating that to provide personalized query handling, we must include user-specific query behaviours as a fundamental component.

Finally, COREKG-Unweighted treated each sample from the coreset equally and assigned each sample an equal weight. It shows a substantial degradation in performance. COREKG-Unweighted achieving F1 scores of 10.82% (Freebase), 11.53% (DBpedia), and 19.38% (Wikidata), and coverage scores of 14.06% (Freebase), 16.68% (DBpedia), and 14.72% (Wikidata). This means that it could not adequately handle sampling and thus produce an overall very low coverage. It has been demonstrated through the ablation study that Weighted Coreset Sampling is effective in reducing sampling bias and preserving query semantics across workloads.

Based on our ablation analysis, we conclude that all elements of COREKG are important and complementary to its success.

7 Limitations

While COREKG demonstrates strong performance across datasets, several limitations remain. First, the effectiveness of sensitivity-based sampling depends on the representativeness of the user-specific query workload. When workloads are sparse, highly skewed, or dominated by a narrow set of seed entities, the resulting summaries may overfit to observed queries and fail to capture emerging or under-represented structural patterns via sensitivity analysis.

The theoretical analysis in this work guarantees preservation of the aggregate workload cost across the query set. It does not provide guarantees for per-query answers. These properties depend on correlations among triples and are outside the scope of the current coreset formulation. Extending the framework to per-query or structure-aware guarantees is an important direction for future work.

8 Conclusion

COREKG generates high-quality personalized knowledge graph summaries using sensitivity-based sampling over user-specific query workloads. The summary preserves query-relevant structure while achieving high coverage and F1 scores. Overall, COREKG is a strong approach for personalized KG summarization.

Acknowledgments

The authors would like to acknowledge the partial support of the Infosys Center for Artificial Intelligence (CAI), IIIT-Delhi in this work. Supratim acknowledges the kind support from Anusandhan National Research Foundation (ECRG/2024/000959). The work was partly supported by their generous fund.

References

[Bachem *et al.*, 2017] Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017.

[Bachem *et al.*, 2018] Olivier Bachem, Mario Lucic, and Andreas Krause. Scalable k-means clustering via lightweight coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1119–1127, 2018.

[Bollacker *et al.*, 2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[Boutsidis *et al.*, 2013] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near-optimal coresets for least-squares regression. *IEEE transactions on information theory*, 59(10):6880–6892, 2013.

[Braverman *et al.*, 2021] Vladimir Braverman, Dan Feldman, Harry Lang, Adiel Statman, and Samson Zhou. Efficient coreset constructions via sensitivity sampling. In *Asian Conference on Machine Learning*, pages 948–963. PMLR, 2021.

[Čebirić *et al.*, 2015] Šejla Čebirić, François Goasdoué, and Ioana Manolescu. Query-oriented summarization of rdf graphs. In *British International Conference on Databases*, pages 87–91. Springer, 2015.

[Cheng *et al.*, 2011] Gong Cheng, Thanh Tran, and Yuzhong Qu. Relin: relatedness and informativeness-based centrality for entity summarization. In *International semantic web conference*, pages 114–129. Springer, 2011.

[Chhaya *et al.*, 2020a] Rachit Chhaya, Jayesh Choudhari, Anirban Dasgupta, and Supratim Shit. Streaming coresets for symmetric tensor factorization. In *International Conference on Machine Learning*, pages 1855–1865. PMLR, 2020.

[Chhaya *et al.*, 2020b] Rachit Chhaya, Anirban Dasgupta, and Supratim Shit. On coresets for regularized regression. In *International conference on machine learning*, pages 1866–1876. PMLR, 2020.

[Dasgupta *et al.*, 2009] Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W Mahoney. Sampling algorithms and coresets for lp regression. *SIAM Journal on Computing*, 38(5):2060–2078, 2009.

[Feldman and Langberg, 2011] Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 569–578, 2011.

[Feldman *et al.*, 2013] Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1434–1453. SIAM, 2013.

[Feldman *et al.*, 2017] Dan Feldman, Sedat Ozer, and Daniela Rus. Coresets for vector summarization with applications to network graphs. In *International Conference on Machine Learning*, pages 1117–1125. PMLR, 2017.

[Goasdoué *et al.*, 2019] François Goasdoué, Pawel Guzewicz, and Ioana Manolescu. Incremental structural summarization of rdf graphs. In *EDBT 2019-22nd International Conference on Extending Database Technology*, 2019.

[Gunaratna *et al.*, 2015] Kalpa Gunaratna, Krishnaparasad Thirunarayan, and Amit Sheth. Faces: diversity-aware entity summarization using incremental hierarchical conceptual clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015.

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

[He *et al.*, 2021] Daojing He, Runmeng Du, Shanshan Zhu, Min Zhang, Kaitai Liang, and Sammy Chan. Secure logistic regression for vertical federated learning. *IEEE Internet Computing*, 2021.

[Huang *et al.*, 2022] Lingxiao Huang, Zhize Li, Jialin Sun, and Haoyu Zhao. Coresets for vertical federated learning: Regularized linear regression and k-means clustering. *Advances in Neural Information Processing Systems*, 35:29566–29581, 2022.

[Jalota *et al.*, 2021] Rricha Jalota, Daniel Vollmers, Diego Moussallem, and Axel-Cyrille Ngonga Ngomo. Laurenknowledge graph summarization for question answering. In *2021 IEEE 15th International Conference on Semantic Computing (ICSC)*, pages 221–226. IEEE, 2021.

[Kang *et al.*, 2022] Shinhwan Kang, Kyuhan Lee, and Kijung Shin. Personalized graph summarization: formulation, scalable algorithms, and applications. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 2319–2332. IEEE, 2022.

[Khan *et al.*, 2026] Sohel Aman Khan, Raghava Mutharaju, and Supratim Shit. Corekg: Coreset-guided personalized summarization of knowledge graphs. *arXiv preprint arXiv:2605.14900*, 2026.

[Koutra, 2021] Danai Koutra. The power of summarization in graph mining and learning: smaller data, faster methods, more interpretability. *Proceedings of the VLDB Endowment*, 14(13):3416–3416, 2021.

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

[Li *et al.*, 2025] Zihao Li, Dongqi Fu, Mengting Ai, and Jingrui He. Apex2: Adaptive and extreme summarization for personalized knowledge graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 741–752, 2025.

[Malaviya *et al.*, 2024] Jayesh Malaviya, Anirban Dasgupta, and Rachit Chhaya. Simple weak coresets for non-decomposable classification measures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14289–14296, 2024.

[Mendes *et al.*, 2012] Pablo N Mendes, Max Jakob, and Christian Bizer. *DBpedia: A multilingual cross-domain knowledge base*. European Language Resources Association (ELRA), 2012.

[Moro *et al.*, 2023] Gianluca Moro, Luca Ragazzi, Lorenzo Valgimigli, et al. Graph-based abstractive summarization of extracted essential knowledge for low-resource scenarios. *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, 372:1747–1754, 2023.

[Niazmand *et al.*, 2022] Emetis Niazmand, Gezim Sejdiu, Damien Graux, and Maria-Esther Vidal. Efficient semantic summary graphs for querying large knowledge graphs. *International Journal of Information Management Data Insights*, 2(1):100082, 2022.

[Peng *et al.*, 2023] Ciyuan Peng, Feng Xia, Mehdi Naseriparsa, and Francesco Osborne. Knowledge graphs: Opportunities and challenges. *Artificial Intelligence Review*, 56(11):13071–13102, 2023.

[Safavi *et al.*, 2019] Tara Safavi, Caleb Belth, Lukas Faber, Davide Mottin, Emmanuel Müller, and Danai Koutra. Personalized knowledge graph summarization: From the cloud to your pocket. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 528–537. IEEE, 2019.

[Shabani *et al.*, 2024] Nasrin Shabani, Amin Beheshti, Jia Wu, Maryam Khanian Najafabadi, Jin Foo, and Alireza Jolfaei. Graphsum: Scalable graph summarization for efficient question answering. In *27th International Conference on Extending Database Technology, EDBT 2024*, pages 794–797. OpenProceedings. org, 2024.

[Shit *et al.*, 2022] Supratim Shit, Anirban Dasgupta, Rachit Chhaya, and Jayesh Choudhari. Online coresets for parameteric and non-parametric bregman clustering. *Transactions on Machine Learning Research*, 2022.

[Shit *et al.*, 2025] Supratim Shit, Gurmehak Kaur Chadha, Surendra Kumar, and Bapi Chatterjee. Improved coresets for vertical federated learning: Regularized linear and logistic regressions. In *International Conference on Machine Learning*, pages 55303–55324. PMLR, 2025.

[Song *et al.*, 2018] Qi Song, Yinghui Wu, Peng Lin, Luna Xin Dong, and Hui Sun. Mining summaries for knowledge graph search. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1887–1900, 2018.

[Stadler *et al.*, 2024] Claus Stadler, Muhammad Saleem, Qaiser Mehmood, Carlos Buil-Aranda, Michel Dumontier, Aidan Hogan, and Axel-Cyrille Ngonga Ngomo. Lsq 2.0: A linked dataset of sparql query logs. *Semantic Web*, 15(1):167–189, 2024.

[Vassiliou, 2023] G. et al. Vassiliou. isummary: Workload-based, personalized summaries for knowledge graphs. In *ESWC*, 2023.

[Wang and Cheng, 2024] Xiaxia Wang and Gong Cheng. A survey on extractive knowledge graph summarization: applications, approaches, evaluation, and future directions. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 8290–8298, 2024.

[Yih *et al.*, 2016] Wen-tau Yih, Matthew Richardson, Christopher Meek, Ming-Wei Chang, and Jina Suh. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, 2016.