Aggregate Queries on Knowledge Graphs: Fast Approximation with Semantic-aware Sampling

Yuxiang Wang[1], Arijit Khan[2], Xiaoliang Xu[1], Jiahui Jin[3], Qifan Hong[1], Tao Fu[1]
[1] *Hangzhou Dianzi University, China* [2] *Nanyang Technological University, Singapore* [3] *Southeast University, China*
{lsswyx,xxl,qfhong,taof}@hdu.edu.cn, arijit.khan@ntu.edu.sg, jjin@seu.edu.cn

***Abstract*—A knowledge graph (KG) manages large-scale and real-world facts as a big graph in a schema-flexible manner. Aggregate query is a fundamental query over KGs, e.g., *"what is the average price of cars produced in Germany?"*. Despite its importance, answering aggregate queries on KGs has received little attention in the literature. Aggregate queries can be supported based on factoid queries, e.g., *"find all cars produced in Germany"*, by applying an additional aggregate operation on factoid queries' answers. However, this straightforward method is challenging because both the accuracy and efficiency of factoid query processing will seriously impact the performance of aggregate queries. In this paper, we propose a "sampling-estimation" model to answer aggregate queries over KGs, which is the first work to provide an approximate aggregate result with an effective accuracy guarantee, and without relying on factoid queries. Specifically, we first present a semantic-aware sampling to collect a high-quality random sample through a random walk based on knowledge graph embedding. Then, we propose unbiased estimators for COUNT, SUM, and a consistent estimator for AVG to compute the approximate aggregate results based on the random sample, with an accuracy guarantee in the form of confidence interval. We extend our approach to support iterative improvement of accuracy, and more complex queries with filter, GROUP-BY, and different graph shapes, e.g., chain, cycle, star, flower. Extensive experiments over real-world KGs demonstrate the effectiveness and efficiency of our approach.**

I. Introduction

Knowledge graphs (KGs) are popular in managing large-scale and real-world facts [1], [2], such as DBpedia [3], YAGO [4], Freebase [5], and NELL [6], where a node represents an entity with attributes, and an edge denotes a relationship between two entities. Querying KGs is critical for a wide range of applications, e.g., question answering and semantic search [7]. However, it is challenging due to the KG's "*schema-flexible*" nature [8]–[13]: *The same kind of information can be represented as diverse substructures* [12], [13]. This schema-flexible nature should be carefully considered in the study of KG querying, especially for the following two important query forms: *factoid query* and *aggregate query* [14].

Factoid query. The answers to a factoid query are defined as an enumeration of noun phrases [15], e.g., "*Find all cars produced in Germany*" (Q117 from QALD-4 benchmark [16]). Given the KG in Figure 1(a), we expect answers as all entities having type *Automobile* that satisfy the semantic relation *product* to the specific entity *Germany*, e.g., Audi_TT (u_{10}), BMW_320 (u_6), etc. Notice that these correct answers are linked with *Germany* in structurally different ways in Figure 1(a), for instance, u_{10}: Audi_TT-*assembly*-Volkswagen-*country*-Germany; u_6: BMW_320-*assembly*-Germany. This reflects the "*schema-flexible*" nature of a KG and we expect to find all the semantically similar answers for factoid queries.

Aggregate query. A simple aggregate query is used to explore the statistical result of a set of entities given a specific entity and a semantic relation. For example, "*what is the average price of cars produced in Germany?*" is an aggregate query to achieve AVG(price) of all the *Automobiles* that satisfy the semantic relation *product* to the specific entity *Germany*. We find that 31% queries from the real query log *LinkedGeoData13* and 30% queries from the manually curated query set *WikiData17* are aggregate queries [17].

One frequently used technology to answer factoid queries is graph query [13], [18]–[22], which we adopt in this work: A user constructs a query graph Q to describe her query intention, and identifies the exact or approximate matches of Q in a KG G. We can also reduce other query forms, such as keywords and natural languages [20], to graph queries by translating input text to a query graph [23], [24]. In contrast, answering aggregate queries on KGs has been mostly ignored in the literature. Aggregate queries can be extended from factoid queries, by applying an additional aggregation on factoid queries' answers to obtain the statistical result of interest [22], [25] (as Figure 1(b) shows). However, this straightforward method is problematic due to following reasons.

Effectiveness issue. If aggregate queries are answered via factoid queries, its effectiveness would depend on the quality of factoid queries' returned answers. For example, subgraph isomorphism [22], [26] only returns answers that *exactly* match with the given query graph Q (e.g., only u_5 is returned for Q in Figure 1), while other semantically similar but structurally different answers are ignored (e.g., u_6, u_7, and u_{10}). Analogously, a relational or SPARQL query finds answers matching *exactly* the schema of the input query, and other valid answers with different schemas will be ignored (see [8], [13], [27] and also our experimental results). In addition to exact matching, several other works [12], [18], [20], [28] return similar answers to Q. However, it is difficult for them to return 100% accurate answers (the notion of "accurate" answers could very well depend on the user's query intension, or may even be vague [29], [30]). Calculating the aggregate result over answers with low quality leads to significant errors. Worse still, we lack an effective way to quantify the result's quality.

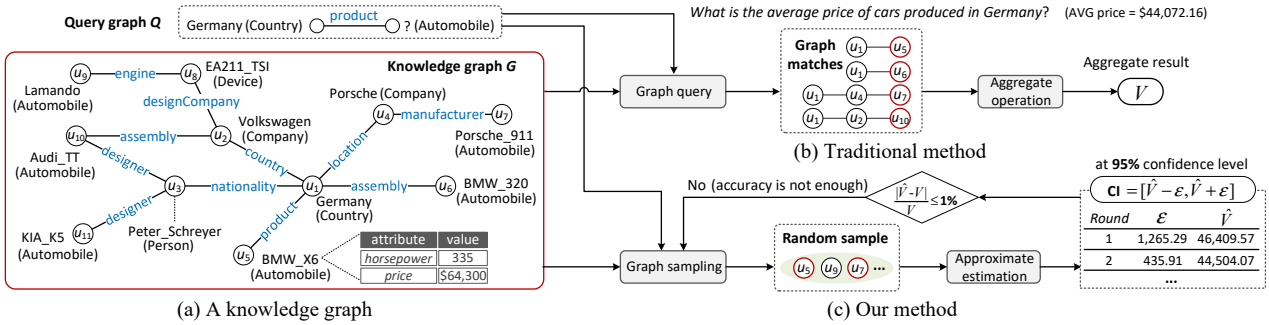


Fig. 1: (a) Each entity of this knowledge graph with type *Automobile* has many numerical attributes, including *horsepower*, *price*, etc. (b) The traditional method computes an aggregate result based on the graph matches via graph query. (c) Our method computes an approximate aggregate result with an accuracy guarantee in the form of confidence interval (CI) through a "sampling-estimation" model.

Efficiency issue. Since an additional aggregate operation is applied on the factoid queries' answers to obtain the aggregate result, factoid queries' efficiency substantially affects aggregate queries' efficiency. Finding answers to a given query graph Q, however, is computationally expensive (e.g., tens of seconds are required in [28]). Even the top-k graph query models still need hundreds of milliseconds to tens of seconds to respond [12], [13], [20], [22], [31].

In practice, aggregate queries may not need a tardy exact result. It is more desirable if a query engine first quickly returns an approximate aggregate result with some accuracy guarantee (e.g., a confidence interval), while improving the accuracy as more time is spent [32], [33]. In this way, we can early terminate the query once the approximate result is acceptable. This improves the user's experience and saves computing resources [13], [34]–[36].

Our solution. Due to the "*schema-flexible*" nature in KGs, we adopt the "*semantic similarity*" [13] (defined in §III) to measure how semantically similar a candidate answer is to a query graph. We then propose an iterative and approximate approach to efficiently answer aggregate queries over KGs, having an accuracy guarantee, but without requiring factoid query evaluations. As Figure 1(c) shows, we first collect answers that are semantically similar to a query graph Q as a random sample from a KG G (**§IV-A**). Next, we estimate an unbiased (or consistent) approximate aggregate result $\hat{V}$ based on the random sample (**§IV-B**), and provide an accuracy guarantee for $\hat{V}$ by iteratively computing a tight enough confidence interval CI $= [\hat{V} - \varepsilon, \hat{V} + \varepsilon]$ at a confidence level $1-\alpha$, where ε is the half-width of a CI (also called the Margin of Error). A CI states that the ground truth V is covered by an interval $\hat{V} \pm \varepsilon$ with probability $1 - \alpha$. We terminate the query when a tight CI (with a small enough ε) is obtained, and ensure that the relative error of $\hat{V}$ is bounded by a user-specific error bound e_b (**§IV-C**). To the best of our knowledge, *we are the first to use a "sampling-estimation" model to answer aggregate queries on KGs with an accuracy guarantee, together with iterative improvements in error bounds.*

Given a query graph Q, it is *non-trivial to collect semantically similar answers to Q as a random sample from a KG G*. First, we cannot directly apply the sampling approaches for relational datasets [34], [37]–[41], because a KG's structure differs from that of a relational data. Though we can model graphs as relations, it adds overhead in the query: We need expensive joins to generate intermediate views and then sample and aggregate over them. Furthermore, *relational or SPARQL query would not find valid answers having different schemas from Q* [8], [13], [27]. Second, existing graph sampling approaches, e.g., CNARW [42] and Node2Vec [43], *only consider topology information for sampling, which ignores semantic information in a KG*, so an answer in a random sample could probably have a low semantic similarity to Q.

To this end, we leverage an offline KG embedding model [44]–[46] to represent predicates as d-dimensional vectors that can well capture their semantic meanings and measure the predicate similarity. On top of this, we design a semantic-aware sampling algorithm via a random walk on G. As a result, answers that are more semantically similar to Q would be sampled with higher probabilities than others with lower semantic similarities. We formally prove that the random walk converges, and all answers in a random sample are independent and identically distributed (i.i.d.) random variables.

Contributions. Our key contributions include **(1)** *designing of a random walk following predicate similarity via KG embedding* to collect a high-quality sample of answers which are semantically similar to the query graph, **(2)** *theoretical characterization of our random walk and proposed estimators* that answer aggregate queries over KGs with accuracy guarantees and iterative improvements, **(3)** *extending our solution to support complex queries* with filter, GROUP-BY, and different graph shapes, e.g., chain, cycle, star, and flower [17] (**§V**), and **(4)** *thorough experiments over three diverse real-world KGs* showing accuracy and efficiency improvements against state-of-the-art methods, and our approach's effectiveness when a user varies the error bound interactively (**§VII**).

As the first step, we mainly focus on non-extreme aggregates $\{\mathsf{COUNT}, \mathsf{SUM}, \mathsf{AVG}\}$ with accuracy guarantees. Notice that our solution can also support extreme functions, e.g., MAX, MIN without accuracy guarantees (see §VII), while in future we will study their theoretical accuracy guarantees. Related work is discussed in **§VI**, while in **§VIII** we conclude.

II. Preliminaries

Definition 1: **Knowledge graph (KG)**. A KG is defined as $G = (V_G, E_G, L_G, A_G)$, where V_G is a finite set of nodes and $E_G \subseteq V_G \times V_G$ is a set of edges. (1) Each node $u \in$

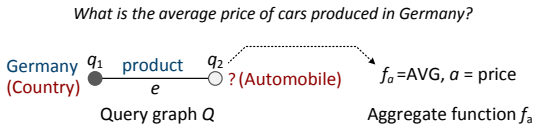


Fig. 2: An example of aggregate query $AQ_G = (Q, f_a)$

V_G represents an entity and each edge $e \in E_G$ denotes a relationship between two entities. (2) A label function L_G assigns a name and various types on each node $u \in V_G$, and a predicate on each edge $e \in E_G$. (3) Each node $u \in V_G$ has a set of numerical attributes, denoted by $A_G(u) = \{a_1, ..., a_n\}$, and $u.a_i$ indicates the value of attribute a_i of u.

We assume that each node u in a KG G has at least one type and a unique name [12], [47]. If the node type is unknown, we employ a probabilistic model-based entity typing method to assign a type on it [48]. For example, $L_G(u).\text{type} = \{\text{Automobile}, \text{MeanOfTransportation}\}$; $L_G(u).\text{name} = \text{BMW\_X6}$. For each edge e, it has a predicate such as $L_G(e) = \text{assembly}$. Moreover, each node u has a set of numerical attributes, e.g., $A_G(u) = \{\text{horsepower}, \text{price}, ...\}$, such as $u.\text{horsepower} = 335$ for BMW_X6.

Definition 2: **Aggregate query over** G. An aggregate query over G is defined as $AQ_G = (Q, f_a)$, where Q is a query graph for searching candidate answers from G and f_a is an aggregate function on the numerical attribute a of the answers to Q. In this paper, we primarily consider three widely-used non-extreme aggregate functions $\{\text{COUNT}, \text{SUM}, \text{AVG}\}$.

We start with the query graph Q for simple questions — one of the most common questions [49], [50], involving a single specific entity and a single predicate, taking the target entities as the answers [1]. Given our proposed framework to answer $AQ_G = (Q, f_a)$ for simple questions, we use it as a building block to support more general cases (discussed in §V).

Definition 3: **Query graph**. A query graph is defined as a graph $Q = (V_Q, E_Q, L_Q)$, with query node set V_Q, edge set E_Q, and label function L_Q. For a simple question, $V_Q = \{q^s, q^t\}$ contains two nodes, where q^s is a specific node and q^t is a target node. For q^s, both the types and name are known, while for q^t, only the types are known. Moreover, E_Q has one edge $e = q^s q^t$ with a predicate $L_Q(e)$.

Example 1: Given a simple question "*what is the average price of cars produced in Germany?*", we formulate $AQ_G = (Q, f_a)$ in Figure 2. The query graph Q contains a specific node $q^s = q_1$ (type: {Country}, name: Germany), a target node $q^t = q_2$ (type: {Automobile}), an edge $e = q_1 q_2$ (predicate: product), and $f_a = \text{AVG}$ on the attribute $a = \text{price}$.

Definition 4: **Candidate answers to** Q. Given a query graph Q and a KG G, candidate answers $\mathcal{A} = \{u_1^t, \cdots, u_n^t\}$ are certain nodes from G: (1) Each u_i^t must have at least one common type as the target node q^t from Q (i.e., $L_G(u_i^t).\text{type} \cap L_Q(q^t).\text{type} \neq \emptyset$). (2) Each u_i^t has semantic similarity $s_i \in [0, 1]$ (defined in §III), indicating how semantically similar the best subgraph match containing u_i^t is to Q.

Due to the schema-flexible nature of KGs, the same kind of information can be represented as different substructures [12], [13]. So we find many semantically similar but structurally different subgraph matches to a given query graph Q. This motivates us to adopt the semantic similarity to measure how semantically similar a candidate answer is to Q. We introduce a tunable parameter τ as the threshold and view those candidate answers having $s_i \geq \tau$ as the *correct answers* to Q, denoted by $\mathcal{A}^+ = \{u_i^t \in \mathcal{A} : s_i \geq \tau\}$.

TABLE I: Frequently used notations

Notations	Descriptions
G	A knowledge graph
$AQ_G = (Q, f_a)$	An aggregate query over G with a query graph Q and an aggregate function f_a
$\mathcal{A}$	A set of candidate answers to Q; $\mathcal{A} = \{u_1^t, \cdots, u_n^t\}$
$S_\mathcal{A}$	A random sample of answers collected from $\mathcal{A}$
s_i	The semantic similarity of an answer $u_i^t \in \mathcal{A}$ to Q
τ	A user-input semantic similarity threshold
$\mathcal{A}^+ \subseteq \mathcal{A}$	A set of correct answers to Q; $\mathcal{A}^+ = \{u_i^t \in \mathcal{A} : s_i \geq \tau\}$
V	The ground truth of AQ_G; V=$f_a(\mathcal{A}^+)$
$\hat{f}_a$	An unbiased (or consistent) estimator of f_a
$\hat{V}$	The estimated approximate result of AQ_G; $\hat{V}$=$\hat{f}_a(S_\mathcal{A})$
e_b	A user-input error bound
$1-\alpha$	A user-input confidence level
$\hat{V} \pm \varepsilon$	The confidence interval (CI) at $1-\alpha$ confidence level
ε	The half width of CI, called the Margin of Error (MoE)

TABLE II: Some candidate answers to the query (Q) in Figure 2

Subgraph matches to Q	Answers	s_i
BMW_X6–*product*–Germany	BMW_X6	1.0
BMW_320–*assembly*–Germany	BMW_320	0.98
Audi_TT–*assembly*–Volkswagen–*country*–Germany	Audi_TT	0.89
KIA_K5–*designer*–Peter_schreyer–*nationality*–Germany	KIA_K5	0.82

correct answers

Example 2: Table II shows four candidate answers to the query graph Q in Figure 2. Each answer (second column) is an entity from the subgraph match to Q (first column) that has a semantic similarity to Q (third column), e.g., BMW_X6 (first row) has a semantic similarity of 1.0 because its subgraph match is exactly the same as Q. Also, KIA_K5 (last row) has a semantic similarity of 0.82 because its subgraph match is semantically quite different from Q. By setting $\tau = 0.85$, KIA_K5 can be eliminated from the correct answers.

A domain expert can tune τ appropriately according to her experience and based on available human annotation (an example is given in §VII). So, we resort to semantic similarity-based ground truth (or τ-relevant ground truth, abbreviated as τ-GT) for evaluation, in addition to human annotation-based ground truth (abbreviated as HA-GT) when available. We obtain τ-GT V of AQ_G=(Q, f_a) by applying f_a on all those τ-relevant correct answers $\mathcal{A}^+$ to Q. That is, V=$f_a(\mathcal{A}^+)$. Given above definitions, we are interested in the following problem. Frequently used notations are given in Table I.

Approx-AQ$_G$. Given an aggregate query $AQ_G = (Q, f_a)$, a KG G, an input error bound e_b, and a confidence level $1-\alpha$, we aim to: (1) design a sampling algorithm $\mathcal{D}$ to collect a random sample $S_\mathcal{A} = \mathcal{D}(\mathcal{A})$ of the candidate answers $\mathcal{A}$ from G, (2) estimate the approximate result $\hat{V}$ based on $S_\mathcal{A}$ with a confidence interval $\hat{V} \pm \varepsilon$ at $1-\alpha$ confidence level, and (3) ensure that the relative error of $\hat{V}$ is bounded by e_b.

$$\hat{V} = \hat{f}_a(S_\mathcal{A})$$
$$s.t. \quad \Pr[\hat{V} - \varepsilon \leq V \leq \hat{V} + \varepsilon] = 1 - \alpha \quad \text{and} \quad |\hat{V} - V|/V \leq e_b \quad (1)$$

In Eq. 1, $\hat{f}_a$ is an unbiased (or consistent) estimator of the given aggregate function f_a. The approximate result $\hat{V}$

Algorithm 1: Semantic Similarity-based Baseline (SSB)

Data: knowledge graph G, aggregate query $AQ_G = (Q, f_a)$, threshold τ, and subgraph boundary n
Result: aggregate result V of the τ-relevant correct answers
// n-bounded subgraph construction
1 u^s = getMappingNode($Q.q^s$);
2 G' = getBoundedGraph(G, u^s, n);
// τ-relevant correct answers enumeration
3 **for** $\forall u_i^t \in \mathcal{A} \subseteq G'$ **do**
4 $\quad s_i$ = getSimilarity(u_i^t,Q)$\geq \tau$: $\mathcal{A}^+$.add(u_i^t) ? continue;
5 **return** $V = f_a(\mathcal{A}^+)$;

is a point estimator to the ground truth V and a $1-\alpha$ level confidence interval CI $= [\hat{V}-\varepsilon, \hat{V}+\varepsilon]$ is computed to quantify the point estimator's quality, which states that V is covered by an estimated range $\hat{V} \pm \varepsilon$ with probability $1-\alpha$. The half width of CI, denoted by ε, is called the Margin of Error (MoE). Generally, the smaller MoE shows the higher quality of $\hat{V}$, i.e., $\hat{V}$ is much closer to V. In §IV-C, we prove that the accuracy guarantee $|\hat{V}-V|/V \leq e_b$ is ensured if the MoE is small enough to satisfy $\varepsilon \leq \hat{V} \cdot e_b/(1+e_b)$ (Theorem 2). Otherwise, we enlarge the sample $S_{\mathcal{A}} = S_{\mathcal{A}} \cup \Delta S_{\mathcal{A}}$ and repeat Eq. 1 to continuously refine the CI $= \hat{V} \pm \varepsilon$ until $\varepsilon \leq \hat{V} \cdot e_b/(1+e_b)$.

III. The Semantic Similarity Baseline

Before discussing our approximate solution, we introduce a simple, but costly enumeration method, called ***Semantic Similarity-based Baseline*** (SSB), to answer an aggregate query AQ_G=(Q, f_a). We apply SSB to get semantic similarity-based ground truth (τ-GT) for effectiveness evaluation (§VII).

The basic idea of SSB (Algorithm 1) is to enumerate all candidate answers $\mathcal{A}$, find all correct answers $\mathcal{A}^+ \subseteq \mathcal{A}$ having semantic similarities $s_i \geq \tau$ (τ is a predefined threshold), and then compute the aggregate result over $\mathcal{A}^+$. Considering all candidate answers, however, is unnecessary, because graph queries exhibit strong access locality [51], thus most correct answers could be found in an n-bounded space of the specific node [13] (in §VII, we empirically find that n=3 can retrieve 99% of all correct answers). Hence, it is reasonable to limit the search space of SSB in an n-bounded subgraph G' of G.

n-bounded subgraph construction. Given a query graph Q and a KG G, we get the mapping node u^s from G for the specific node q^s from Q that satisfies: $L_G(u^s).\mathsf{name} = L_Q(q^s).\mathsf{name}$, $L_G(u^s).\mathsf{type} \cap L_Q(q^s).\mathsf{type} \neq \emptyset$ (Line 1). Then we conduct a BFS starting from u^s to construct the n-bounded subgraph G', of which each entity u from G' is within n-hops from u^s (Line 2). Since a KG adopts some entity disambiguation methods [52]–[54] to ensure that each node has a unique name, for simplicity we assume that q^s has a unique mapping node u^s. We next introduce how to measure the semantic similarity of each candidate answer from G' (Lines 3-4). This is the most expensive step in SSB.

Semantic similarity of an answer. We start with defining a subgraph match $M(u_i^t)$ of a candidate answer $u_i^t \in \mathcal{A}$.

Definition 5: **Subgraph match** [13]. Given a simple query graph Q and an n-bounded subgraph G', a subgraph match $M(u_i^t)$ to Q is defined as an edge-to-path mapping from the query edge $e = q^s q^t$ in Q to a path $\overline{u^s u_i^t}$ in G'. (1) For the specific node q^s, u^s is its mapping node satisfying $L_G(u^s).\mathsf{name} = L_Q(q^s).\mathsf{name}$ and $L_G(u^s).\mathsf{type} \cap L_Q(q^s).\mathsf{type} \neq \emptyset$. (2) For the target node q^t, u_i^t is a candidate answer that satisfies $L_G(u_i^t).\mathsf{type} \cap L_Q(q^t).\mathsf{type} \neq \emptyset$.

For example, in Table II, the subgraph match ⟨Audi_TT-*assembly*-Volkswagen-*country*-Germany⟩ contains an answer Audi_TT (i.e., u_i^t). Intuitively, a subgraph match $M(u_i^t)$ is more semantically similar to Q if each edge e' on the path $\overline{u^s u_i^t}$ is more semantically similar to the query edge e in Q. Following [13], we define the semantic similarity $s[M(u_i^t)]$ of $M(u_i^t)$ to Q as the geometric mean of the predicate similarities of all edges in $\overline{u^s u_i^t}$ (Eq. 2), where $sim(L_G(e'),L_Q(e))$ is the predicate similarity between e' and e; l is the length of $\overline{u^s u_i^t}$. If there are multiple subgraph matches of u_i^t, we compute the semantic similarity s_i of u_i^t as the maximum semantic similarity considering all its subgraph matches (Eq. 3).

$$s[M(u_i^t)] = \sqrt[l]{\prod_{e' \in \overline{u^s u_i^t}} sim(L_G(e'), L_Q(e))} \tag{2}$$

$$s_i = \max_{M(u_i^t)} s[M(u_i^t)] \tag{3}$$

We leverage an offline ***KG embedding*** model to obtain the predicate similarity between two edges e' and e. A KG embedding aims to represent each predicate and entity in a KG as a d-dimensional vector, it can preserve well the semantic meanings and relations using these learned semantic vectors [1]. We refer interested readers to [13], [44] for more details. The similarity between two predicates $sim(L_G(e'), L_Q(e))$ (e.g., sim(assembly, product)) can be computed by the cosine similarity between their predicate vectors $\boldsymbol{e}$ and $\boldsymbol{e'}$.

$$sim(L_G(e'), L_Q(e)) = \frac{\boldsymbol{e'} \cdot \boldsymbol{e}}{||\boldsymbol{e'}|| \times ||\boldsymbol{e}||} \tag{4}$$

Example 3: Consider the answer Audi_TT in Table II for the query graph in Figure 2. Its semantic similarity is $\sqrt[2]{0.98 \times 0.81}$=0.89, where sim(assembly, product)=0.98 and sim(country, product)=0.81, based on the TransE model [44].

In §VII, we apply SSB to get semantic similarity-based ground truth (i.e., τ-relevant ground truth, τ-GT) for effectiveness evaluation, besides human annotation-based ground truth (HA-GT). SSB can work with any KG embedding model. Ideally, if we have a high-quality KG embedding model, then we can distinguish the implicit semantics of predicates well by Eq. 4; hence, we can effectively represent the semantics of different paths and answers by Eq. 2-3. So, it is likely that both τ-GT and HA-GT would be similar for an appropriate τ. In §VII, we present the effectiveness of our approximate solution (§IV) w.r.t. both τ-GT and HA-GT. We also study the effect of KG embedding models on effectiveness.

Remarks. (1) Different from structure-based similarity which assumes that a shorter path has higher similarity [18], [31], [55], our semantic similarity captures the implicit semantics of a path by KG embedding. The path length usually does not reflect the semantics of a path, and a longer path might have a higher semantic similarity than a shorter one to a given query graph. E.g., a longer path (BMW_Z4-*assembly*-

Regensburg-*federalState*-Bavaria-*country*-Germany) may be semantically more similar than a shorter path (KIA_K5-*designer*-Peter_schreyer-*nationality*-Germay) w.r.t. the query graph in Figure 2. **(2)** As semantic similarity of a path is non-monotonic w.r.t. its length (Eq. 2), Dijkstra-like algorithm is inadequate to directly find the path from u^s to u_i^t with the highest semantic similarity. Instead, one needs to enumerate all paths from u^s to u_i^t, compute their semantic similarities, and find the best one (Eq. 3). **(3)** SSB's time complexity is: $O(|\mathcal{A}| \cdot m^n)$, where m is the average degree of an entity in a KG and m^n is the search space of path enumeration for each candidate answer in $\mathcal{A}$. In our implementation, we do not explicitly construct the n-bounded subgraph G', rather all paths up to length n from u^s are considered for each candidate answer in $\mathcal{A}$. Due to the inefficiency of SSB, we introduce a lightweight "sampling-estimation" solution in §IV.

IV. Sampling-Estimation Solution

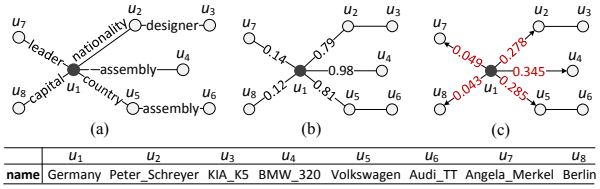
We first provide a high-level introduction to our "sampling-estimation" solution, then we drill down into the details (§IV-A-IV-C) and show the entire algorithm in §IV-D. Given a KG G and an aggregate query $AQ_G = (Q, f_a)$, we do the followings. (1) *Semantic-aware sampling on KGs* (§IV-A): We collect answers with higher semantic similarities to Q as a random sample $S_{\mathcal{A}}$, via a semantic-aware random walk sampling over G. (2) *Approximate result estimation* (§IV-B): We consider the semantic similarity to design unbiased estimators for {COUNT, SUM} and a consistent estimator for AVG, then apply them on $S_{\mathcal{A}}$ to estimate the approximate result $\hat{V}$. (3) *Accuracy guarantee* (§IV-C): We derive an accuracy guarantee based on the *Central Limit Theorem* (CLT) in the form of confidence interval CI $= \hat{V} \pm \varepsilon$ and iteratively refine the CI until an acceptable accuracy (Theorem 2) is attained.

A. Semantic-aware Sampling on KGs

1) Classic Random Walk on Graphs: Random walk is the mainstream technique for graph sampling due to its scalability and simplicity of implementation [56]. Random walk sampling on a KG G is modeled as a finite Markov Chain [42], having the following steps. A walker starts from a randomly selected node $u_0 \in V_G$, then randomly chooses u_0's one neighbor and moves to it with the transition probability defined in the transition matrix $\boldsymbol{P} = |V_G| \times |V_G|$. This walker continues to walk until a stationary distribution $\boldsymbol{\pi} = \{\pi_1, ..., \pi_{|V_G|}\}$ is reached, where $\sum \pi_i = 1$ and π_i is the stationary visiting probability of each node $u_i \in V_G$ when random walk converges. The walker keeps walking after $\boldsymbol{\pi}$ is reached and collects all visited nodes as a random sample of V_G. Each collected node u_i can be viewed as being sampled with its visiting probability π_i.

The transition matrix $\boldsymbol{P}$ is the key to the random walk sampling. Different $\boldsymbol{P}$ are required for different downstream applications. Unlike previous topology-aware graph sampling works [42], [43], [56], *we are the first to develop a semantic-aware graph sampling on KGs for aggregate queries*.

2) Semantic-aware Random Walk Sampling: Similar to SSB, we expect to run our semantic-aware random walk sampling on the n-bounded subgraph G' rather than on the entire G, to improve the efficiency by reducing the scope of random walk. To achieve this, in our implementation, we limit the random walk on G within n-hops from the mapping node u^s. For simplicity of understanding, we still use G' (the n-bounded subgraph) to indicate the scope of our random walk, this is the induced graph formed by those nodes within n-hops from u^s. We next introduce our *design of transition matrix* $\boldsymbol{P}$ by considering the semantic similarity. Then, we discuss two phases of our semantic-aware sampling based on $\boldsymbol{P}$: *random walk until convergence* and *continuous sampling*.



	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
name	Germany	Peter_Schreyer	KIA_K5	BMW_320	Volkswagen	Audi_TT	Angela_Merkel	Berlin

Fig. 3: (a) A KG with a table of the entities' names. (b) The predicate similarities of u_1's adjacency edges to the predicate product. (c) The transition probabilities p_{1j} for $u_j \in N(u_1)$.

(1) Transition matrix. We design a $|V_{G'}| \times |V_{G'}|$ transition matrix $\boldsymbol{P} = [p_{ij}]$ for an n-bounded subgraph G' ($V_{G'}$ is its node set). We aim to collect answers having higher semantic similarities as a random sample $S_{\mathcal{A}}$. So, we ensure that such an answer has a greater visiting probability π_i when the random walk converges to a stationary distribution $\boldsymbol{\pi}$. With greater π_i, there is a higher probability of the answer being sampled.

Transition probability. Intuitively, if we can design a transition matrix $\boldsymbol{P}$ to guide a walker towards an answer u_i^t with the greatest s_i as much as possible, then this u_i^t is more likely to have a greater visiting probability π_i at convergence. So, we assign a greater transition probability p_{ij} on each edge e' from G' having a higher predicate similarity $sim(L_G(e'), L_Q(e))$ to the query edge e from Q. Given a node u_i from G', its neighbors are denoted as $N(u_i)$. We use $e' = u_i u_j$ to indicate the adjacency edge between u_i and its neighbor $u_j \in N(u_i)$. We can easily compute the predicate similarity $sim(L_G(e'), L_Q(e))$ by Eq. 4 and assign a transition probability p_{ij} on e' that is proportional to $sim(L_G(e'), L_Q(e))$ by Eq. 5, where Z is the normalization constant and the total probability of moving from u_i to other nodes in $N(u_i)$ should equal to one according to the property of Markov Chain.

$$p_{ij} = Z \cdot sim(L_G(e'), L_Q(e)) \qquad (5)$$
$$s.t. \quad \sum_{u_j \in N(u_i)} p_{ij} = 1$$

Example 4: Figure 3 illustrates an example of transition probability calculation for the query AQ_G in Figure 2. Figure 3(a) shows a KG with nodes in 2-hops from u_1 (Germany). In Figure 3(b), each adjacency edge of u_1 has a predicate similarity to the predicate product. The transition probability p_{1j} for $u_j \in N(u_1)$, such as $p_{14}=\frac{0.98}{0.98+0.81+0.79+0.14+0.12}=0.345$, is given in Figure 3(c). A walker starting from u_1 will likely select u_4, u_5, or u_2 as next step (they are more relevant to reach automobile nodes), rather than other irrelevant nodes (i.e., u_7 and u_8). Moreover, the automobile u_4 has a higher probability to be sampled than u_3 and u_6, because $p_{14} > \{p_{15}, p_{12}\}$.

Analysis of the convergence. We must ensure that the random walk using the transition matrix $\boldsymbol{P}$ initialized by Eq. 5 can converge to a stationary distribution. A finite Markov Chain (MC) can converge if it is *irreducible* and *aperiodic* [57]. An MC is irreducible if any two nodes are reachable in finite steps.

Lemma 1: Our semantic-aware random walk is irreducible.

We prove this in our extended version [58]. Intuitively, this holds because each edge has a non-zero transition probability.

In an MC, each node u_i has period k if any return to u_i occurs in multiples of k steps, and an MC is aperiodic if it has at least one node having period one [57]. To satisfy the aperiodic property, we change the structure of G' with a small modification: We add an additional self-loop edge on the mapping node u^s with a small predicate similarity p_{ss} on it (0.001 in this work). A walker starting from u^s tends to walk outward rather than be stuck at u^s due to this small p_{ss}, so it has little effect on the convergence time. It is easy to verify that our random walk is aperiodic after this modification.

Lemma 2: Our semantic-aware random walk is aperiodic after addition of self-loop edge on the mapping node u^s.

Based on Lemma 1-2, we conclude that our semantic-aware random walk can converge to a stationary distribution $\boldsymbol{\pi}$.

(2) Random walk until convergence. We start the random walk from the mapping node u^s. In each walk step, we use a walking-with-rejection policy [42] to determine the next node. Specifically, suppose a walker is currently at node u_i, we select a neighbor $u_j \in N(u_i)$ randomly and accept it with probability p_{ij}. If u_j is rejected, then we repeat it until one node is accepted. After moving from u_i to u_j, we update the stationary probability π_j by Eq. 6. We initialize the stationary distribution as $\boldsymbol{\pi} = \{1, 0, ..., 0\}$, where the mapping node u^s has $\pi_s = 1$ because we start the random walk from u^s.

$$\pi_j = \sum_{u_i \in N(u_j)} \pi_i \cdot p_{ij} \tag{6}$$

Eq. 6 is a standard way to update $\boldsymbol{\pi}$ in Markov Chain, which guarantees $\sum \pi_i = 1$. Random walk converges if $\boldsymbol{\pi}$ is no longer changing (i.e., $\boldsymbol{\pi} \cdot \boldsymbol{P} = \boldsymbol{\pi}$), then we obtain the stationary distribution $\boldsymbol{\pi} = \{\pi_1, ..., \pi_{|V_{G'}|}\}$ for all nodes $V_{G'} \in G'$.

(3) Continuous sampling. As mentioned in §IV-A1, the original continuous sampling returns a random sample of node set $V_{G'}$ by continuous walking on G' after convergence. In our case, we expect to collect a random sample $S_{\mathcal{A}}$ of the candidate answers $\mathcal{A}$ instead of a sample of $V_{G'}$. So, we make the following changes to the original continuous sampling. First, we extract the stationary distribution $\boldsymbol{\pi}_{\mathcal{A}} = \{\pi'_1, \cdots, \pi'_{|\mathcal{A}|}\}$ of $\mathcal{A}$ from the stationary distribution $\boldsymbol{\pi} = \{\pi_1, \cdots, \pi_{|V_{G'}|}\}$ of $V_{G'}$. For each answer $u_i^t \in \mathcal{A}$, we compute its new visiting probability $\pi'_i = \pi_i / \sum \pi_i$ where $\pi_i \in \boldsymbol{\pi}$ is the original visiting probability of u_i^t, so that the cumulative probability $\sum \pi'_i = 1$ holds for $\boldsymbol{\pi}_{\mathcal{A}}$. Second, we conduct the continuous sampling to collect each visited answer $u_i^t \in \mathcal{A}$ to $S_{\mathcal{A}}$ with its probability of $\pi'_i \in \boldsymbol{\pi}_{\mathcal{A}}$, and ignore all the non-answer nodes.

Theorem 1: All the answers in $S_{\mathcal{A}}$ are independent and identically distributed (i.i.d.) random variables.

Proof: In continuous sampling, the acceptance decision for each $u_i^t \in S_{\mathcal{A}}$ is made independently, whether u_i^t is sampled depends only on its visiting probability π'_i and is irrelevant to the fact that other answers are sampled or not. Moreover, each visiting probability π'_i comes from the identical distribution $\boldsymbol{\pi}_{\mathcal{A}}$. So, all the answers in $S_{\mathcal{A}}$ are i.i.d random variables. □

Remarks. We stop sampling after collecting enough answers. A large or small $|S_{\mathcal{A}}|$ may lead to over- or under-sampling. We discuss how to configure $|S_{\mathcal{A}}|$ appropriately in §IV-C.

B. Approximate Result Estimation

We present unbiased estimators for {SUM, COUNT} and a consistent estimator for AVG.

1) Aggregation Estimators: We collect a random sample $S_{\mathcal{A}}$ from a nonuniform stationary distribution $\boldsymbol{\pi}_{\mathcal{A}}$. Since each answer in $S_{\mathcal{A}}$ is sampled with different probability, different answers should contribute differently to estimation. So, we cannot apply estimators designed for a uniform sample, e.g., [34], [37], [38], because they are unbiased only when each tuple is sampled with the same probability. Alternatively, we provide estimators $\hat{f}_a$ based on the *Horvitz-Thompson estimators* [59] as follows, where $\pi'_i \in \boldsymbol{\pi}_{\mathcal{A}}$ is the visiting probability of each answer $u_i^t \in S_{\mathcal{A}}$, and $S_{\mathcal{A}}^+ = \{u_i^t \in S_{\mathcal{A}} : s_i \geq \tau\}$ is the set of answers in $S_{\mathcal{A}}$ having semantic similarity $\geq \tau$.

$$\hat{f}_a^{\text{sum}}(S_{\mathcal{A}}) = \frac{1}{|S_{\mathcal{A}}^+|} \sum_{u_i^t \in S_{\mathcal{A}}^+} \frac{u_i^t.a}{\pi'_i} \tag{7}$$

$$\hat{f}_a^{\text{count}}(S_{\mathcal{A}}) = \frac{1}{|S_{\mathcal{A}}^+|} \sum_{u_i^t \in S_{\mathcal{A}}^+} \frac{1}{\pi'_i} \tag{8}$$

$$\hat{f}_a^{\text{avg}}(S_{\mathcal{A}}) = \frac{\hat{f}_a^{\text{sum}}(S_{\mathcal{A}})}{\hat{f}_a^{\text{count}}(S_{\mathcal{A}})} = \frac{\sum_{u_i^t \in S_{\mathcal{A}}^+} u_i^t.a/\pi'_i}{\sum_{u_i^t \in S_{\mathcal{A}}^+} 1/\pi'_i} \tag{9}$$

Lemma 3: The estimator for SUM is unbiased. $E[\hat{f}_a^{\text{sum}}] = \sum_{u_i^t \in \mathcal{A}^+} u_i^t.a$, here $\mathcal{A}^+ \subseteq \mathcal{A}$ is the set of correct answers.

Lemma 4: The estimator for COUNT is unbiased: $E[\hat{f}_a^{\text{count}}] = |\mathcal{A}^+|$, here $\mathcal{A}^+ \subseteq \mathcal{A}$ is the set of correct answers.

We provide unbiasedness proofs in extended version [58].

Lemma 5: The estimator for AVG is consistent.

Proof: We prove that $\hat{f}_a^{\text{avg}}$ is consistent through two steps by the *Strong Law of Large Numbers* (SLLN) [42], [60], [61].
Step 1. We first transform $\hat{f}_a^{\text{avg}}$ according to the *importance sampling framework* [62] by setting a weight $w_i = U_i/\pi'_i$ for each $u_i^t \in S_{\mathcal{A}}^+$, where $U_i = \frac{1}{|S_{\mathcal{A}}^+|}$ (U is a uniform distribution).

$$\hat{f}_a^{\text{avg}} = \frac{\sum_{u_i^t \in S_{\mathcal{A}}^+} \frac{u_i^t.a}{\pi'_i}}{\sum_{u_i^t \in S_{\mathcal{A}}^+} \frac{1}{\pi'_i}} = \frac{\frac{1}{|S_{\mathcal{A}}^+|} \sum_{u_i^t \in S_{\mathcal{A}}^+} \frac{U_i}{\pi'_i} \cdot u_i^t.a}{\frac{1}{|S_{\mathcal{A}}^+|} \sum_{u_i^t \in S_{\mathcal{A}}^+} \frac{U_i}{\pi'_i}} = \frac{\mu(w_i \cdot u_i^t.a)}{\mu(w_i)} .$$

Step 2. According to SLLN, the mean (μ) of any function of samples collected from a stationary distribution approximates to the expectation of this function over the stationary distribution [42], that is, $\mu(g) \to E_\pi[g]$. So we have the following:

$$\frac{\mu(w_i \cdot u_i^t.a)}{\mu(w_i)} \to \frac{E_{\boldsymbol{\pi}_{\mathcal{A}}}[w_i \cdot u_i^t.a]}{E_{\boldsymbol{\pi}_{\mathcal{A}}}[w_i]} = \frac{E_U[u_i^t.a]}{1} = \frac{\sum_{u_i^t \in \mathcal{A}^+} u_i^t.a}{|\mathcal{A}^+|} .$$

Thus, AVG's estimator is consistent, because its expectation converges almost surely to the true mean of all $u_i^t \in \mathcal{A}^+$. □

Remarks. The estimators in Eq. 7-9 are mean-like expressions, so we can provide CLT-based accuracy guarantee for them (§IV-C). Since it is hard to form a mean-like estimator for extreme functions (e.g., MAX, MIN), we should seek other theories, e.g., Extreme Value Theory (EVT) [63] for accuracy guarantee. If we can control the sample's distribution to follow Generalized Extreme Value (GEV) or Generalized Pareto distribution (GPD), then we may be able to do the EVT-based estimation. We keep this as an open problem.

2) Correctness Validation for the Sample: Although our semantic-aware sampling is designed to collect answers with greater semantic similarities as the sample $S_{\mathcal{A}}$, it is still likely to involve a few answers with lower semantic similarity due to the randomness of sampling. For the query in Figure 2, we find that 12% sampled answers have semantic similarity $< \tau$ on average when $\tau = 0.85$. This would affect the accuracy if we directly regard all sampled answers as correct and apply Eq. 7-9 for estimation. We need a method to quickly validate each answer's correctness before estimation. This is exactly what we do in Eq. 7-9: We first compute $S^+_{\mathcal{A}} = \{u^t_i \in S_{\mathcal{A}} : s_i \geq \tau\}$. In §VII-C, we show that our method accounts for 27% of query time, but offers 9X effectiveness improvement on average.

Basic idea. A straightforward way of correctness validation for an answer $u^t_i \in S_{\mathcal{A}}$ is to find the subgraph match $M(u^t_i) = \overline{u^s u^t_i}$ having the greatest semantic similarity to the query graph. Enumerating all subgraph matches, however, is computationally expensive. Instead, we present a heuristic method to find a subgraph match with a higher likelihood of having the greatest semantic similarity and check its correctness, thus pruning other matches to improve the efficiency.

Greedy search heuristic. We start search from the mapping node u^s by considering it as the current node. We also initialize a candidate set with u^s's neighbors. Every time, we select the node u from the candidate set that has the highest visiting probability π, and make it the current node. We update the candidate set by adding u's neighbors that were not current nodes in the past and remove u from it. We continue to search and record all the possible paths from u^s till an answer u^t_i becomes the current node. The found path $\overline{u^s u^t_i}$ is used as the (heuristically) best subgraph match for correctness checking.

Effectiveness analysis. In the context of our greedy search heuristic, a false positive indicates that an incorrect answer u^t_i is reported as correct. False positive would strongly impact the estimation accuracy. Fortunately, no false positive would happen in our method. Consider an incorrect answer u^t_i, all its subgraph matches must have a semantic similarity $< \tau$. So, we never include it in $S^+_{\mathcal{A}}$, no matter what subgraph match is found by our method. On the other hand, a false negative indicates that a correct answer u^t_i is reported as incorrect. This happens when the subgraph match found so far is not the optimal one. To reduce the chances of false negative, we introduce *repeat factor* r in our correctness validation. Specifically, greedy search continues searching until r paths from u^s to u^t_i are found and returns the correctness of the one with the greatest semantic similarity. The larger r is, the lower is the probability of false negative, but this increases validation time. In §VII-D, we show that a balance is achieved when $r = 3$.

C. Accuracy Guarantee

Given an approximate result $\hat{V}$ (Eq. 7-9) and a user-specific error bound e_b, we ensure that the relative error $|\hat{V} - V|/V \leq e_b$, where V is the τ-relevant ground truth. Specifically, we compute a confidence interval CI $= \hat{V} \pm \varepsilon$ (at a user-specific confidence level $1 - \alpha$) to quantify $\hat{V}$'s quality. This CI states that V is covered by an estimated range $\hat{V} \pm \varepsilon$ with probability $1 - \alpha$. The half width ε is the Margin of Error (MoE). We first prove that $|\hat{V} - V|/V \leq e_b$ holds with probability $1 - \alpha$ if $\varepsilon \leq \frac{\hat{V} \cdot e_b}{1+e_b}$. Then, we show how to compute ε efficiently.

Theorem 2: If the MoE ε of the CI satisfies $\varepsilon \leq \frac{\hat{V} \cdot e_b}{1+e_b}$, then the relative error of the approximate result is upper bounded by e_b, denoted as $|\hat{V} - V|/V \leq e_b$, with probability $1 - \alpha$.

Proof: Given the CI $= [\hat{V} - \varepsilon, \hat{V} + \varepsilon]$ at the confidence level $1 - \alpha$, we prove this theorem in two steps.

Step 1. Suppose that V is located in the CI's right half-width, i.e., $\hat{V} \leq V \leq \hat{V} + \varepsilon$, then we have the following derivation and $(V - \hat{V})/V \leq e_b$ holds if $\varepsilon/\hat{V} \leq e_b$ (i.e., $\varepsilon \leq \hat{V} \cdot e_b$).

$$(V - \hat{V})/V \leq (V - \hat{V})/\hat{V} \leq \varepsilon/\hat{V}$$

Step 2. Suppose that V is located in CI's left half-width, i.e., $\hat{V} - \varepsilon \leq V \leq \hat{V}$. Then we say that $(\hat{V} - V)/V \leq e_b$ holds if $\varepsilon/(\hat{V} - \varepsilon) \leq e_b$ (i.e., $\varepsilon \leq \frac{\hat{V} \cdot e_b}{1+e_b}$) by the following derivation.

$$(\hat{V} - V)/V \leq \varepsilon/V \leq \varepsilon/(\hat{V} - \varepsilon)$$

In summary, $\frac{|\hat{V} - V|}{V} \leq e_b$ holds if $\varepsilon \leq \frac{\hat{V} \cdot e_b}{1+e_b}$, because $\frac{\hat{V} \cdot e_b}{1+e_b}$ is a tighter bound ($\leq \hat{V} \cdot e_b$). Since our CI of $\hat{V}$ has a confidence level $1 - \alpha$, the above holds with probability $1 - \alpha$. □

Confidence interval calculation. The *Central Limit Theorem* (CLT) is applied to compute the confidence interval [37]. We take SUM as an example to show the basic idea. Consider $V_i = u^t_i.a/\pi'_i$ as a random variable w.r.t. each answer $u^t_i \in S^+_{\mathcal{A}}$. The approximate result $\hat{V}$ can be viewed as the mean of a set of random variables: $\{V_i | u^t_i \in S^+_{\mathcal{A}}\}$ according to Eq. 7. From CLT, we know that if a point estimator takes the form of the mean of i.i.d. random variables, then it follows a normal distribution [2]. Hence, we have $\hat{V} \sim N(\mu_{\hat{V}}, \sigma^2_{\hat{V}})$, and the MoE ε of the confidence interval $\hat{V} \pm \varepsilon$ at a confidence level $1 - \alpha$ can be calculated based on CLT as follows.

$$\varepsilon = z_{\alpha/2} \cdot \sigma_{\hat{V}} \tag{10}$$

Here, $z_{\alpha/2}$ is the normal critical value with right-tail probability $\alpha/2$, that can be obtained from a standard normal table. We use *Bag of Little Bootstrap* (BLB) [64] to estimate $\sigma_{\hat{V}}$.

Bag of little bootstrap. We initialize the desired sample size $N = \lambda \cdot |\mathcal{A}|$ as a fraction of the candidate answer set $\mathcal{A}$, where λ is the desired sample ratio. (1) BLB collects t small samples $\{S_i | i = 1, \cdots, t\}$ from $\mathcal{A}$ to form the random sample $S_{\mathcal{A}} = \bigcup_{i=1}^{t} S_i$, each S_i has a size $|S_i| = N^m$, so we have $|S_{\mathcal{A}}| = t \cdot N^m$. The scale factor $m \in [0.5, 1]$ is used in [64] to satisfy $t \cdot N^m < N$. (2) For each small sample S_i, BLB estimates $\sigma_{\hat{V}}$ by a standard bootstrap [65] (given below) and computes

an MoE ε_i by Eq. 10. (3) Given a set of MoE $\{\varepsilon_1 \cdots \varepsilon_t\}$ for t small samples, BLB computes the final $\varepsilon = \sum \varepsilon_i / t$.

Bootstrap. (1) We collect B resamples from $S_{\mathcal{A}}$ with replacement, each resample contains $|S_{\mathcal{A}}|$ answers. (2) We compute the approximate result for each resample as $\{\hat{V}_1^* ... \hat{V}_B^*\}$. (3) Bootstrap takes the empirical distribution of $\{\hat{V}_1^* ... \hat{V}_B^*\}$ as an approximation to $N(\mu_{\hat{V}}, \sigma_{\hat{V}}^2)$, so we estimate $\sigma_{\hat{V}}$ by Eq. 11.

$$\mu_{\hat{V}} = \sum \hat{V}_i^* / B \ , \quad \sigma_{\hat{V}}^2 = \sum (\hat{V}_i^* - \mu_{\hat{V}})^2 / (B-1) \tag{11}$$

Configuration of $|\Delta S_{\mathcal{A}}|$. We terminate the query when the MoE $\varepsilon \leq \frac{\hat{V} \cdot e_b}{1+e_b}$ (Theorem 2). Otherwise, we update $S_{\mathcal{A}}$ by additional answers, i.e., $S_{\mathcal{A}} = S_{\mathcal{A}} \cup \Delta S_{\mathcal{A}}$, and continue the estimation until we obtain a small enough ε. Intuitively, we need a large $|\Delta S_{\mathcal{A}}|$ when ε is large. Otherwise, a small $|\Delta S_{\mathcal{A}}|$ would be sufficient. To achieve the right balance, we present an error-based method that automatically configures $|\Delta S_{\mathcal{A}}|$.

Consider an MoE $\varepsilon > \frac{\hat{V} \cdot e_b}{1+e_b}$. We use $\varepsilon / \frac{\hat{V} \cdot e_b}{1+e_b}$ to denote how far ε is away from the desired value $\frac{\hat{V} \cdot e_b}{1+e_b}$. The larger $\varepsilon / \frac{\hat{V} \cdot e_b}{1+e_b}$ is, the more answers that $\Delta S_{\mathcal{A}}$ requires. Ideally, if we can reduce ε to a new ε' by at least $\varepsilon / \frac{\hat{V} \cdot e_b}{1+e_b}$ times, we can satisfy $\varepsilon' \leq \frac{\hat{V} \cdot e_b}{1+e_b}$. According to Eq. 10, reducing ε by $\varepsilon / \frac{\hat{V} \cdot e_b}{1+e_b}$ times is equivalent to reducing $\sigma_{\hat{V}}$ by $\varepsilon / \frac{\hat{V} \cdot e_b}{1+e_b}$ times. Since $\sigma_{\hat{V}} = \frac{\sigma_V}{\sqrt{N}}$ according to CLT, where σ_V is the standard deviation of the population, we say that reducing $\sigma_{\hat{V}}$ by $\varepsilon / \frac{\hat{V} \cdot e_b}{1+e_b}$ times is equivalent to increasing N by $(\varepsilon / \frac{\hat{V} \cdot e_b}{1+e_b})^2$ times. In summary, we can increase N by $(\varepsilon / \frac{\hat{V} \cdot e_b}{1+e_b})^2$ times to reduce ε by $\varepsilon / \frac{\hat{V} \cdot e_b}{1+e_b}$ times. Hence, we derive $|\Delta S_{\mathcal{A}}|$ as follows.

$$\begin{aligned} |\Delta S_{\mathcal{A}}| &= t \cdot [N \cdot (\varepsilon / \frac{\hat{V} \cdot e_b}{1+e_b})^2]^m - t \cdot N^m \\ &= |S_{\mathcal{A}}| \cdot [(\varepsilon / \frac{\hat{V} \cdot e_b}{1+e_b})^{2m} - 1] \end{aligned} \tag{12}$$

Example 5: Suppose we have a CI $= \hat{V} \pm \varepsilon$ with $\hat{V}$ = 578 and ε = 6.5, by a sample of size $|S_{\mathcal{A}}|$ = 100. If we set the scale factor m = 0.6 and error bound e_b = 0.01, then we need $|\Delta S_{\mathcal{A}}| = 100 \cdot ((6.5 / \frac{578 \cdot 0.01}{1.01})^{2 \cdot 0.6} - 1) \approx 16$ additional answers, which is much smaller than $|S_{\mathcal{A}}|$.

Interactive refinement of e_b. In an interactive scenario, a user may vary e_b in runtime, that is, she gradually reduces e_b to achieve more accurate results. We can quickly obtain a new approximate result with a small additional overhead, because our sample size configuration method can sense the variation of e_b and update $S_{\mathcal{A}}$ appropriately (Eq. 12). This can be very beneficial to reduce overhead caused by oversampling.

Remarks. The empirical results (§VII-D) show that the desired sample ratio $\lambda = 0.3$ is enough to achieve a good result for $t \geq 3$, m=0.6, and $B \geq 50$ as recommended in [64].

D. Putting All Together

We present the entire algorithm in Algorithm 2, including an offline KG embedding phase to generate a predicate vector space $\boldsymbol{E}$ (Line 1) and an online "sampling-estimation" phase to obtain an approximate result with an accuracy guarantee (Lines 2-14). In the online phase, given an aggregate query $AQ_G = (Q, f_a)$, we do the following (Lines 2-3): (1) Conduct a semantic-aware random walk on G (within n-hops) to reach a stationary distribution $\boldsymbol{\pi}$ by using $\boldsymbol{E}$, and (2) collect a random sample $S_{\mathcal{A}}$ according to $\boldsymbol{\pi}$. Next, we validate the correctness of each answer in $S_{\mathcal{A}}$ to obtain correct answers $S_{\mathcal{A}}^+ \subseteq S_{\mathcal{A}}$ (Line 5) and estimate the approximate result $\hat{V}$ by applying the proposed unbiased (or consistent) estimators on $S_{\mathcal{A}}^+$ (Line 6). After that, we compute the MoE ε of a confidence interval CI $= \hat{V} \pm \varepsilon$ at the confidence level $1-\alpha$ (Line 7). Finally, we check the termination condition (Theorem 2) and return $\hat{V} \pm \varepsilon$ when $\varepsilon \leq \frac{\hat{V} \cdot e_b}{1+e_b}$ (Lines 8-9 & 14). Otherwise, we enlarge $S_{\mathcal{A}}$ with an appropriate sample size $|\Delta S_{\mathcal{A}}|$, and repeat above steps to continuously refine $\hat{V} \pm \varepsilon$ (Lines 11-13).

Algorithm 2: Approximate Aggregate Query on KGs

Data: A KG G, an aggregate query $AQ_G = (Q, f_a)$, a user-input error bound e_b and a confidence level $1-\alpha$
Result: approximate aggregate result with a CI $= \hat{V} \pm \varepsilon$

1 $\boldsymbol{E}$ = KG_Embedding_Model(G);
2 $\boldsymbol{\pi}$ = randomWalk(G, $\boldsymbol{E}$); /* §IV-A2(2) */
3 $S_{\mathcal{A}}$ = collectSample(G, $\boldsymbol{\pi}$, $Q.q^t$); /* §IV-A2(3) */
4 **while** true **do**
5 $S_{\mathcal{A}}^+$ = correctnessValidate(G, $S_{\mathcal{A}}$, $\boldsymbol{\pi}$, τ); /* §IV-B2 */
6 $\hat{V}$ = estimate($S_{\mathcal{A}}^+$); /* §IV-B1 */
7 ε = getMoE($S_{\mathcal{A}}^+$, $1-\alpha$); /* §IV-C */
8 **if** $\varepsilon \leq \hat{V} \cdot e_b / (1+e_b)$ **then**
9 **break**; /* §IV-C: Theorem 2 */
10 **else**
11 $|\Delta S_{\mathcal{A}}|$ = configSampleSize(ε,$\hat{V}$, e_b); /* §IV-C */
12 $\Delta S_{\mathcal{A}}$ = collectSample(G, $\boldsymbol{\pi}$, $Q.q^t$);
13 $S_{\mathcal{A}} = S_{\mathcal{A}} \cup \Delta S_{\mathcal{A}}$;
14 **return** CI $= \hat{V} \pm \varepsilon$;

Remarks. The total time of our method consists of sampling time (T_s) and estimation time (T_e). We get $T_s = O(|E_{G'}| + N_{ws} + |S_{\mathcal{A}}|)$, where $|E_{G'}|$ is the average number of edges in the scope of our random walk and we compute the transition probability for each such edge, N_{ws} is the average walk steps required for random walk convergence ($N_{ws} \leq 500$ in practice), and $|S_{\mathcal{A}}|$ is the initial sample size. Next, we get $T_e = O(N_e \cdot (|S_{\mathcal{A}}| + |S_{\mathcal{A}}^+| + |\Delta S_{\mathcal{A}}|))$, where N_e is the average iterations till termination condition (Theorem 2) is satisfied ($N_e \leq 10$ in practice). In each iteration, we validate the correctness of $|S_{\mathcal{A}}|$ answers (Line 5), estimate based on $|S_{\mathcal{A}}^+|$ correct answers (Lines 6-7), and include $|\Delta S_{\mathcal{A}}|$ additional answers to refine the approximate result (Lines 11-13).

V. Extensions

A. Aggregate Queries with Filters and GROUP-BY

Queries with filters. We define queries with filters as follows.

Definition 6: AQ_G **with filters**. An aggregate query with filters is defined as $AQ_G^F = (Q, \bigcup_{i=1}^n L_i \leq b_i \leq U_i, f_a)$, where (1) $\bigcup_{i=1}^n L_i \leq b_i \leq U_i$ is a set of filters (or ranges), and (2) for each filter, the attribute b_i of each answer to Q must be between a lower (L_i) and an upper (U_i) bound.

Example 6: Given a query: "*Find the average price of cars produced in Germany with a fuel_economy between 25 and 30 MPG*", we can form the AQ_G^F of this query by adding a filter 25$\leq$ fuel_economy $\leq$30 on the aggregate query in Figure 2.

TABLE III: Statistics of our datasets

Datasets	DBpedia	Freebase	YAGO2
#Nodes	4,521,912	5,706,539	7,308,072
# Edges	15,045,801	48,724,743	36,624,106
# Node-Types	359	11,666	6,543
# Edge-Predicates	676	5118	101

We support AQ_G^F by adding a filter operation in the correctness validation (§IV-B2): only the answer u_i^t satisfying the filter condition and its semantic similarity $s_i \geq \tau$ is a correct answer, i.e., $c(u_i^t) = (L \leq u_i^t.b \leq U \ \&\& \ s_i \geq \tau)?1 : 0$.

Queries with GROUP-BY. We can also support queries with GROUP-BY to return answers in groups. For example, "*How many Spanish soccer players of each age group are there?*". When GROUP-BY is applied on the query node that serves as a target node, we only need to divide the collected sample of soccer players into different groups based on their ages, then we estimate and return the approximate result for each group.

B. Aggregate Queries with Different Shapes

Users generally form complex queries with different shaped query graphs, e.g., chain, star, cycle, and flower shapes [17]. Due to the page limitations, we refer to our full version [58] for answering chain-shaped queries. Next, we adopt "*decomposition-assembly*" framework [13], [20] to support queries with other shapes: (1) We decompose a complex query into a set of simple or chain-shaped queries $\{Q_1, ..., Q_n\}$ (Def. 3) that share the same target entity, (2) we collect a random sample $S_{\mathcal{A}}^i$ for each Q_i, (3) we use the intersection $S_{\mathcal{A}} = \bigcap_{i=1}^n S_{\mathcal{A}}^i$ as the random sample for the original complex query, and (4) we iteratively estimate the approximate result based on $S_{\mathcal{A}}$ until we obtain an accurate enough result.

VI. Related Work

Online aggregation. Online aggregation (OLA) was first proposed in [38], which is a sampling-based technology to return approximate aggregate results on relational data. Much follow-up work has continued over the years, including (1) OLA over joins, group-by [34], [66]–[71], (2) OLA for distributed environments [72]–[77], and (3) multi queries optimization for OLA [37], [78]. None of them can be deployed directly to answer aggregate queries on KGs, because a KG's *schema-flexible* nature is different from relational data. So, *we design a new semantic-aware sampling suitable for KGs, and unbiased (or consistent) estimators for the nonuniform sample*.

Graph query on knowledge graphs. Graph matching is widely used for querying KGs [8], [12], [13], [18], [20]–[22], [26]–[28], [79]–[81]. They do not consider aggregate queries, except [22], [25], [82]–[84]. Since [22], [25], [82]–[84] answer aggregate queries based on the factoid queries (often via SPARQL aggregate queries), they suffer from the issues mentioned in §I. *We propose a novel "sampling-estimation" model to answer aggregate queries more effectively and efficiently*.

Aggregate query on knowledge graphs. Recently, [85] considers aggregate queries on knowledge graphs: It collects candidate entities via link prediction, and computes the aggregate result based on them. Unlike ours, *it does not support the edge-to-path mapping based on semantic similarity, and thus reports lower-quality answers* (§VII). In [85], the user cannot specify the relative error bound and confidence level, and can neither update them interactively. Finally, unlike our extensions, [85] performs aggregation only for simple queries.

VII. Experimental Results

We evaluate (1) effectiveness and efficiency for simple and complex queries (§VII-B), (2) effect of each step (§VII-C), and (3) interactive performance and parameter sensitivity (§VII-D). Our code [86] were implemented in Java1.8 and run on a 2.1GHZ, 64GB memory AMD-6272 server (CentOS Linux).

A. Environment Setup

Datasets. We used three real-world datasets (Table III). (1) *DBpedia* [3] is an open-domain KG constructed from Wikipedia. (2) *Freebase* [5] is a KG harvested from many individual, user-submitted wiki contributions. Since we assume that each entity has a name, we used a Freebase-Wikipedia mapping file [87] to filter 5.7M entities with names from Wikipedia. (3) *YAGO2* [4] is a KG with information from the Wikipedia, WordNet, and GeoNames. We used the CORE portion of YAGO as our dataset. Since we focused on aggregate queries, we complemented the numerical attributes of entities via web crawling. For example, we added attributes, e.g., price, horsepower, etc. (from edmunds.com) to automobiles, added age, transfer_value, etc. (from sofifa.com) to soccer players, and added ratings, box_office, etc. (from IMDB) to movies.

Query workload. We tested both *benchmark* and *synthetic queries*. Table IV shows 10 examples out of 400 queries. (1) *QALD-4* [16] is a benchmark for factoid queries on *DBpedia*. We selected factoid queries from QALD-4 as seeds to form COUNT, AVG, and SUM queries through a simple modification (e.g., Q1-Q2). For instance, we changed "*Find all cars produced in Germany.*" to "*How many cars are produced in Germany?*" (Q1). We added filters and GROUP-BY to these queries to form more queries (e.g., Q3-Q4). (2) *WebQuestions* [88] is a benchmark for *Freebase*. We applied the same method on it to form different queries (e.g., Q5-Q6). (3) *Synthetic queries* were generated to evaluate our approach on *YAGO2* (e.g., Q7-Q8). Moreover, we generated complex queries with different shapes (e.g., Q9-Q10). From [17], most of real-world queries have small number of edges, e.g., 73.8% and 81.2% of queries for DBpedia have edges ≤ 2 and 4, respectively. So, we formed 2-hop chain-shaped queries (e.g., Q10) and used them together with simple queries to form queries with other shapes. In Table IV, we show each query's selectivity, defined as the percentage of correct answers over all candidate answers. The average selectivity of all queries is 6.39%, whereas the largest selectivity of a query is around 70%. *Thus, our approach can support both high and low-selectivity queries.*

Metrics. We run each query for 5 times, and used *average relative error* and *average response time* as the metrics for effectiveness and efficiency, respectively. Initially, 38.5% (154/400) of our queries, which were formed from seed queries of *QALD-4* and *WebQuestions* (e.g., Q1-Q3, Q5-Q6 in Table IV), already had human-annotated ground truth. For the remaining

TABLE IV: Ten query examples out of all queries (10/400) that we used in §VII

QID	Queries	Category & selectivity (%)	Aggregate function f_a
Q1	*How many cars are produced in Germany?*	Simple, 3.01%	COUNT(*)
Q2	*What's the average price of cars that are produced in Germany?*	Simple, 3.01%	AVG(price)
Q3	*What's the average price of cars produced in Germany with a fuel_economy between 25MPG and 30MPG?*	Simple+Filter, 0.51%	AVG(price)
Q4	*How many football clubs are there in each country?*	Simple+GROUP-BY	COUNT(*)
Q5	*How many languages are spoken in Nigeria?*	Simple, 69.59%	COUNT(*)
Q6	*What's the total box office of the movies that were directed by Steven Spielberg?*	Simple, 0.05%	SUM(box_office)
Q7	*How many museums are there in England?*	Simple, 3.23%	COUNT(*)
Q8	*What's the average population of China's cities?*	Simple, 2.91%	AVG(population)
Q9	*How many soccer players were born in Spain and played for Barcelona FC?*	Star, 0.12%	COUNT(*)
Q10	*How many cars are designed by German designers?*	Chain, 2.91%	COUNT(*)

TABLE V: Average Jaccard similarity (AJS) between the human-annotated and τ-relevant correct answers and its variance (Var)

Threshold τ	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
DBpedia-AJS	0.64	0.71	0.74	0.78	0.88	**0.95**	0.83	0.64
DBpedia-Var	0.084	0.080	0.079	0.043	0.023	**0.011**	0.121	0.184
Freebase-AJS	0.68	0.74	0.80	0.85	**0.96**	0.91	0.82	0.70
Freebase-Var	0.227	0.215	0.208	0.117	0.061	**0.029**	0.308	0.502
Yago2-AJS	0.52	0.63	0.70	0.82	**0.93**	0.88	0.77	0.59
Yago2-Var	0.042	0.046	0.045	0.025	**0.013**	0.015	0.66	0.108

246 queries, we conducted crowdsourcing with the *Baidu Data Crowdsoursing Platform* (https://zhongbao.baidu.com) to obtain human-annotated ground truth. For details, see our extended version [58]. *In summary, we have human-annotated ground truth for all our queries. In our experiments, we measured the effectiveness in two ways for all our queries: based on human-annotated ground truth (HA-GT) and semantic similarity-based ground truth (τ-GT).*

Comparing methods. We compared our approach with recent works on KG search: (1) EAQ [85] is state-of-the-art work that supports aggregate queries on KGs: It collects candidate entities via link prediction and computes aggregate results *only for simple queries*. (2) SGQ [13] finds the top-k semantically similar answers to a query graph. SGQ supports incremental query processing: when k is increased, additional answers can be retrieved incrementally, without finding all the top-k answers from ground. To find all correct answers, we initialize k=50 and increase k in steps of 50 till all correct answers are included. Notice that in this process, some incorrect answers can also get included in the last step. (3) GraB [21] is an index-free graph query based on structural similarity. (4) QGA [24] is a keyword-based KG search method. We also compared with one RDF store (5) JENA [89] and a graph DB (6) Neo4j [90], both supporting SPARQL queries. We also compared with (7) SSB provided in §III. Since (2)-(4) process factoid queries, we extended them by adding an additional aggregate operation after achieving the factoid query answers. Moreover, (7) serves as a baseline method, we demonstrate our accuracy vs. efficiency trade-offs by comparing with (7), thereby showing the usefulness of "sampling-estimation".

Parameters. The default configuration is: error bound e_b = 1%, confidence level $1-\alpha$ = 95%, repeat factor r = 3, desired sample ratio λ = 0.3, n = 3 of n-bounded subgraph, similarity threshold τ from Table V, and TransE [44] for KG embedding.

Remarks. (1) In real applications, HA-GT may not be available for all queries. Alternately, a domain expert can set τ according to her experience and based on available human-annotations for a limited number of queries, and then use our solution to retrieve a good approximation to both τ-GT (Table VI) and HA-GT (Table VII) over all queries. The key is whether we can find an appropriate τ that makes the human-annotated correct answers and τ-relevant correct answers highly consistent. **(2)** Table V shows the average Jaccard similarity (AJS) between the τ-relevant (with different τ) and human-annotated correct answers and the variance (Var) of Jaccard similarity, considering 35% of all queries over three datasets (with TransE embedding). Though we obtained HA-GT for *all* queries with additional crowdsourcing, in Table V we use HA-GT from 35% of all queries and find the appropriate τ for each dataset. This is to demonstrate that the optimal τ obtained based on available human-annotations for *a limited number of queries* on a dataset, is still sufficient to retrieve a good approximation to both τ-GT and HA-GT for many other queries on that dataset. Intuitively, if we have a high-quality KG embedding model, then we can distinguish the implicit semantics of predicates well; hence, we can also accurately represent the semantics of different paths. So, it is very likely that the ground truths HA-GT and τ-GT would be similar for an appropriate value of τ. **(3)** The optimal τ for different datasets are different; though roughly in the range of 0.8-0.85, we get a relatively higher AJS and smaller Var. For example, in DBpedia, AJS is the highest (0.95) and Var is the smallest (0.011) when $\tau = 0.85$, implying that the difference of each query's τ-relevant correct answers to its human annotated correct answers is small. In this case, our solution can get a good approximation to HA-GT (Table VII), because HA-GT and τ-GT are similar when $\tau = 0.85$, and our solution can achieve a good approximation to τ-GT (Table VI).

B. Effectiveness and Efficiency Evaluation

Effectiveness. Table VI and VII show the relative error based on τ-GT (with the optimized configuration of τ from Table V) and HA-GT, respectively, for *all queries* having different graph shapes. Our solution achieves a good approximation to τ-GT (Table VI). In Table VI, *for all datasets, our solution has two orders of magnitude less relative error on average than other methods*. The reasons are: **(1)** We collected the random sample $S_{\mathcal{A}}$ by considering the semantic similarity via "edge-to-path" mapping in the KG, so that we could find more correct answers than other graph query methods (EAQ, GraB, QGA) that do not consider semantic similarity. **(2)** *Since we provided a specific SPARQL expression as input to both JENA and Neo4j, they only found those correct answers matching* exactly *with the graph schema of the input SPARQL query, and other correct answers having different schemas were ignored.* **(3)** Our approach continues to refine the approximate result by increasing $S_{\mathcal{A}}$ until the relative error is bounded by the user-specified input error bound e_b. We show a case study for Q1, Q2, and Q6 in Table IX, where the approximate results are refined iteratively until all relative errors are bounded by $e_b = 1\%$. **(4)** For the top-k based incremental method SGQ,

TABLE VI: Effectiveness: relative error (%) for different query shapes over all datasets and all queries (τ-relevant ground truth)

Method	DBpedia					Freebase					Yago2				
	Simple	Chain	Star	Cycle	Flower	Simple	Chain	Star	Cycle	Flower	Simple	Chain	Star	Cycle	Flower
Ours	**0.84**	**0.33**	**1.65**	**0.72**	**0.95**	**0.86**	**0.62**	**0.91**	**0.71**	**0.98**	**0.54**	**0.85**	**0.30**	**0.75**	**0.92**
EAQ	20.02	-	-	-	-	17.74	-	-	-	-	14.72	-	-	-	-
GraB	8.08	29.10	18.69	19.24	16.38	12.84	18.47	18.91	23.13	18.87	8.57	19.23	17.53	24.61	12.87
QGA	17.94	31.37	38.49	24.17	31.01	34.63	29.68	34.12	21.76	20.51	19.46	22.38	32.21	42.19	32.54
SGQ	10.67	12.08	16.13	9.98	12.11	6.96	5.54	14.74	13.15	15.61	7.97	14.03	10.01	15.77	8.52
JENA	16.60	29.37	42.03	23.09	21.32	28.17	29.68	34.77	22.76	21.24	11.30	26.38	27.38	41.03	39.47
Virtuoso	16.60	29.37	42.03	23.09	21.32	28.17	29.68	34.77	22.76	21.24	11.30	26.38	27.38	41.03	39.47
SSB	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TABLE VII: Effectiveness: relative error (%) for different query shapes over all datasets and all queries (human-annotated ground truth)

Method	DBpedia					Freebase					Yago2				
	Simple	Chain	Star	Cycle	Flower	Simple	Chain	Star	Cycle	Flower	Simple	Chain	Star	Cycle	Flower
Ours	**0.99**	**0.84**	**1.10**	**1.19**	**0.80**	**0.96**	**1.38**	**1.02**	**0.98**	**1.60**	**0.57**	**1.30**	**0.86**	**0.92**	**1.04**
EAQ	21.14	-	-	-	-	17.74	-	-	-	-	14.72	-	-	-	-
GraB	7.31	29.38	18.29	18.72	16.06	11.41	17.74	18.52	22.97	20.60	7.62	20.56	17.16	24.58	14.02
QGA	19.01	32.90	40.52	25.58	32.45	34.65	29.65	34.68	21.53	22.27	18.94	23.81	32.70	43.50	34.35
SGQ	9.97	12.93	17.16	9.08	11.64	5.30	4.33	14.16	12.54	17.28	7.01	13.19	9.35	15.31	9.63
JENA	17.62	30.79	44.31	22.77	21.22	27.66	29.65	35.39	22.58	23.02	12.40	26.19	27.54	42.23	41.69
Virtuoso	17.62	30.79	44.31	22.77	21.22	27.66	29.65	35.39	22.58	23.02	12.40	26.19	27.54	42.23	41.69
SSB	0.91	0.81	0.92	1.10	0.73	1.83	1.32	0.93	0.91	1.70	1.11	1.21	0.83	0.85	1.13

TABLE VIII: Efficiency: average response time (ms) for different query shapes over all datasets and all queries

Method	DBpedia					Freebase					Yago2				
	Simple	Chain	Star	Cycle	Flower	Simple	Chain	Star	Cycle	Flower	Simple	Chain	Star	Cycle	Flower
Ours	**368.0**	**590.63**	**735.1**	**841.9**	**1518.3**	**419.4**	**673.1**	**905.8**	**1037.7**	**1871.1**	**562.5**	**902.8**	**1154.4**	**1322.4**	**2384.6**
EAQ	4525.3	-	-	-	-	2711.0	-	-	-	-	4267.4	-	-	-	-
GraB	922.6	1541.3	1918.1	1727.4	3114.9	607.2	1014.5	1471.5	1325.2	2389.6	1449.0	2420.9	3511.5	3162.4	5702.5
QGA	1511.0	2120.7	2239.1	3023.3	5451.6	1088.5	1527.8	2173.3	2489.7	4489.4	2369.8	3326.1	2703.1	3096.6	5583.8
SGQ	817.4	1033.2	1285.7	1157.9	2087.9	621.9	786.1	1174.5	1057.8	1907.4	1392.5	1760.1	2630.2	2368.5	4271.1
JENA	1197.9	1681.4	2546.2	2916.9	5259.8	1051.4	1475.7	1606.1	1839.9	3317.7	1505.7	2113.3	2702.2	3095.5	5581.9
Virtuoso	1212.2	1701.4	2550.9	2922.2	5269.4	1074.8	1508.5	1634.6	1872.5	3376.6	1545.9	2169.7	2894.4	3315.7	5979.1
SSB	6675.8	7547.9	9577.6	10972.3	20578.6	2264.0	4766.5	10003.1	11459.7	20663.5	5231.1	6627.6	9624.5	10471.2	19278.6

its relative error > 0. This is because we increased k in steps of 50 till all correct answers were included, in this process, some incorrect answers were included in the last step.

Table VII shows the effectiveness results w.r.t. HA-GT for *all queries*. By setting an appropriate τ, our solution can get a good approximation to HA-GT, because HA-GT and τ-GT are similar for this τ. However, more results have slightly larger relative error than the predefined error bound ($e_b = 1\%$). This is reasonable, since we provide an accuracy guarantee for τ-GT, and not directly for HA-GT; though HA-GT and τ-GT are similar for a specific τ, they may not be exactly same. Moreover, if we do not select τ appropriately, the relative error w.r.t. HA-GT could be affected. We show the effect of τ w.r.t. HA-GT in §VII-D (Figure 5(c) (right)).

Table XI shows the effectiveness results w.r.t. HA-GT and τ-GT for queries with filters, GROUP-BY, and MAX/MIN. For filters and GROUP-BY, we achieve *two orders of magnitude less error on average than others*, because we offer a good accuracy guarantee with confidence interval. For MAX/MIN, though we cannot provide accuracy guarantees, we can support them by returning the MAX/MIN answers of the collected sample. As more sample is collected, the result gets closer to the exact one. We configured a fix sample size (5% of the candidate answers) and found that *the exact result can be included in the sample after 8 rounds on average*. We show the result after 4 rounds, which is already better than others.

Efficiency. The runtime of our method is independent of which ground truth is used. Specifically, when the query terminates, we use the approximate result to compute the relative error w.r.t. HA-GT and τ-GT to obtain different effectiveness results, while the query time is the same. Hence, we report the efficiency results in Tables VIII and X, and find that *our method requires up to an order of magnitude less response time than others*, because we apply the "sampling-estimation" model that does not rely on the factoid query. Hence, we do not need to wait a long time for the graph query results, thereby improving the efficiency. The query time increases as the graph shape gets more complex. This is because we use the "decomposition-assembly" framework to answer complex queries – with more sub-queries, more time is required for sampling and correctness validation. We show the detailed efficiency results of our three steps on DBpedia in Table XII: *semantic-aware sampling* (S1), *approximate estimation* (S2), and *accuracy guarantee* (S3). S1 is the most time-consuming step and S3 is the fastest step, because we must wait for the more time-consuming random walk to converge before collecting the sample. In S2, we validate the correctness of the collected sample, this costs more time than that for S3.

TABLE IX: Case study (τ-GT for Q1, Q2, and Q6 are 596, $44,072, and $7.56B)

QID	Approximate result			
	RD	V	**MoE** ε	error %
Q1	1	578.55	8.51	2.93
	2	**599.97**	**5.99**	**0.67**
Q2	1	46,409	1,265	5.30
	2	**44,504**	**435**	**0.98**
Q6	1	7.07	0.59	6.48
	2	7.89	0.13	4.37
	3	**7.53**	**0.071**	**0.40**

TABLE X: Efficiency for various operators (*DBpedia*)

Method	Efficiency (sec)		
	Filter	GROUP-BY	MAX/MIN
Ours	**0.43**	**31.67**	**0.47**
EAQ	-	-	2.68
GraB	1.10	-	0.80
QGA	1.41	-	1.28
SGQ	0.73	-	0.64
JENA	0.70	95.76	1.07
Virtuoso	0.72	94.67	0.97
SSB	0.78	54.75	8.17

TABLE XI: Effectiveness for various operators (*DBpedia*)

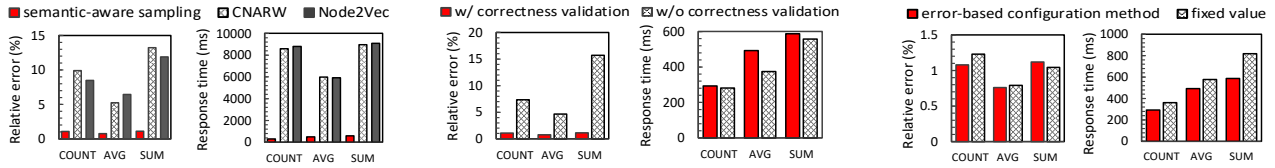
Method	Relative error (%) w.r.t. τ-GT			Relative error (%) w.r.t. HA-GT		
	Filter	GROUP-BY	MAX/MIN	Filter	GROUP-BY	MAX/MIN
Ours	**0.58**	**0.75**	**5.68**	**0.71**	**1.13**	**6.33**
EAQ	-	-	10.29	-	-	10.92
GraB	21.49	-	10.02	20.94	-	10.65
QGA	45.55	-	11.23	46.95	-	11.85
SGQ	18.42	-	6.14	17.71	-	6.79
JENA	46.18	16.75	12.59	48.98	16.30	13.21
Virtuoso	46.18	16.75	12.59	48.98	16.30	13.21
SSB	0	0	0	1.29	1.06	1.14

TABLE XII: Detailed efficiency results (ms) over (*DBpedia, simple*)

Operator	S1	S2	S3
COUNT	246.0	19.6	6.2
AVG	248.2	104.1	42.4
SUM	277.1	108.9	51.5

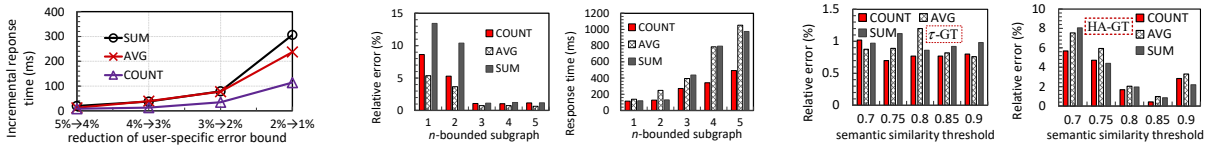
TABLE XIII: Effect of KG embedding models (*DBpedia*, simple, HA-GT)

Model	Embed time (h)	Mem (GB)	Relative error(%)
TransE	**6.63**	**8.8**	**0.99**
TransD	10.06	9.73	**0.83**
TransH	7.82	9.35	1.07
RESCAL	≈1day	50	5.46
SE	≈1day	55	3.38



(a) Effect of S1 on effectiveness, efficiency (b) Effect of S2 on effectiveness, efficiency (c) Effect of S3 on effectiveness, efficiency

Fig. 4: Effect of each step (S1-S3) on effectiveness and efficiency (*DBpedia*, simple, HA-GT)



(a) Interactive performance (b) Effect of n-bounded subgraph (c) Effect of semantic similarity threshold τ

Fig. 5: (a) Interactive performance; (b)-(c) Effect of different parameters on effectiveness and efficiency (*DBpedia*, simple, HA-GT)

C. Effect of Each Step on the Performance

Semantic-aware sampling (S1). We implemented other two versions of S1: both topology-aware samplings (CNARW [42] and Node2Vec [43]). Figure 4(a) shows that the relative errors are reduced by at least 8X, 6X, and 10X for COUNT, AVG, and SUM through our semantic-aware sampling. Our method is also more efficient, because it converges faster than others.

Approximate estimation (S2). We used a correctness validation method in S2 to identify each sampled answer's correctness. We show the effectiveness result w/ or w/o correctness validation in Figure 4(b) (left). The relative errors are reduced by 7X, 6X, and 14X for COUNT, AVG, and SUM. However it increases the response time (Figure 4(b) (right)), though is acceptable considering the improvement in accuracy.

Accuracy guarantee (S3). We applied an error-based method to automatically configure $|\Delta S_{\mathcal{A}}|$ in S3. Different to this, approximate query processing methods on relations increase sample size at a fixed rate, e.g., 50. In Figure 4(c), the relative errors are similar for both, but ours offers better efficiency (improved by 1.3X on average), as we can automatically configure $|\Delta S_{\mathcal{A}}|$, avoiding oversampling or undersampling.

D. Interactive Performance and Parameter Sensitivity

We studied our interactive approach's runtime by varying the error bound e_b during query processing. We initialize $e_b = 5\%$. Instead of terminating the query when $\varepsilon \leq \frac{\hat{V} \cdot e_b}{1+e_b}$, we decrease e_b by 1% to continue processing and report the incremental response time in Figure 5(a). We require less than 100 ms of additional time to meet a new e_b till $e_b \geq 2\%$, which is efficient. For a more stringent error bound, e.g., $e_b = 1\%$, we need 100∼300 ms to re-satisfy the termination condition.

KG embedding. We measure accuracy based on HA-GT due to different KG embedding (Table XIII): translation-based models (TransE [44], TransD [45], TransH [46]), tensor factorization-based model RESCAL [91], and relation-specific projection-based model SE [92]. Unlike RESCAL and SE, translation-based models preserve many important properties such as antisymmetry, inversion, and composition [93], which are critical for the datasets and queries that we tested, so these models can well-represent the semantics of predicates and paths, and we find a good τ which makes τ-GT obtained from these models similar to HA-GT. Therefore, translation-based models perform better than others. We find that different translation-based models (TransE, TransD, TransH) have small differences in accuracy. TransE is also more efficient in embedding time and memory (Table XIII). These results demonstrate that a high-quality KG embedding model is important to our solution when we use HA-GT.

n-bounded subgraph search. The relative error decreases as n increases in Figure 5(b). As most correct answers belong to 3-bounded subgraph, the reduction of relative error gets stable when $n \geq 3$. The run time increases as n increases, since random walks need more time to converge on a larger graph.

Semantic similarity threshold τ. We achieve a good approximate result (error $< 1.5\%$) w.r.t. τ-GT (Fig. 5(c)(left)), as we sample answers with higher semantic similarity (§IV-A), validate answers' correctness (§IV-B2), and iteratively refine the approximate result until we get an accurate enough result (§IV-C). For HA-GT, we should carefully select τ, we achieve the smallest relative error when $\tau = 0.85$ (Fig. 5(c)(right)), as τ-GT and HA-GT are quite similar with this τ (Table V).

Sensitivity results w.r.t. confidence level, repeat factor, and the desired sample ratio are given in our full version [58].

VIII. Conclusions

We proposed a "sampling-estimation" model to answer aggregate queries on KGs effectively and efficiently. We first presented a semantic-aware sampling to collect a high-quality sample from a KG. Then, we proposed two unbiased estimators for COUNT, SUM, and a consistent estimator for AVG. An effective accuracy guarantee was provided through a tight confidence interval and user-input error bound. We extended our solution for iterative improvement of accuracy, complex queries with filters, GROUP-BY, and different shapes. Experimental results on real datasets confirm the effectiveness and efficiency of our approach. In future, we shall derive accuracy guarantees for extreme functions (e.g., MAX, MIN).

Acknowledgment

This work was supported by National NSF of China (62072149, 62072099), Primary R&D Plan of Zhejiang (2021C03156 and 2021C02004), and Center-initiated Research Project of Zhejiang Lab (2021KG0AC01). We thank to the Key Laboratory of Brain Machine Collaborative Intelligence of Zhejiang (2020E10010).

REFERENCES

[1] X. Huang, J. Zhang, D. Li, and P. Li, "Knowledge Graph Embedding Based Question Answering," in *WSDM*, 2019.

[2] J. Gao, X. Li, Y. E. Xu, B. Sisman, X. L. Dong, and J. Yang, "Efficient Knowledge Graph Accuracy Evaluation," *PVLDB*, vol. 12, no. 11, pp. 1679–1691, 2019.

[3] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "DBpedia - A Large-Scale, Multilingual Knowledge Base Extracted from Wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[4] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum, "YAGO2: A Spatially and Temporally Enhanced Knowledge Base from Wikipedia," *Artificial Intelligence*, vol. 194, pp. 28–61, 2013.

[5] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge," in *SIGMOD*, 2008.

[6] T. M. Mitchell, W. W. Cohen, E. R. H. Jr., P. P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. D. Mishra, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. A. Platanios, A. Ritter, M. Samadi, B. Settles, R. C. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling, "Never-Ending Learning," *Commun. ACM*, vol. 61, no. 5, pp. 103–115, 2018.

[7] R. V. Guha, R. McCool, and E. Miller, "Semantic Search," in *WWW*, 2003.

[8] S. Yang, Y. Wu, H. Sun, and X. Yan, "Schemaless and Structureless Graph Querying," *PVLDB*, vol. 7, no. 7, pp. 565–576, 2014.

[9] W. Zheng, H. Cheng, J. X. Yu, L. Zou, and K. Zhao, "Interactive Natural Language Question Answering over Knowledge Graphs," *Information Sciences*, vol. 481, pp. 141–159, 2019.

[10] W. Zheng, H. Cheng, L. Zou, J. X. Yu, and K. Zhao, "Natural Language Question/Answering: Let Users Talk with the Knowledge graph," in *CIKM*, 2017, pp. 217–226.

[11] V. Janev, D. Graux, H. Jabeen, and E. Sallinger, *Knowledge Graphs and Big Data Processing*. Springer Nature, 2020.

[12] W. Zheng, L. Zou, W. Peng, X. Yan, S. Song, and D. Zhao, "Semantic SPARQL Similarity Search over RDF Knowledge Graphs," *PVLDB*, vol. 9, no. 11, pp. 840–851, 2016.

[13] Y. Wang, A. Khan, T. Wu, J. Jin, and H. Yan, "Semantic Guided and Response Times Bounded Top-k Similarity Search over Knowledge Graphs," in *ICDE*, 2020.

[14] W. Cui, Y. Xiao, H. Wang, Y. Song, S.-w. Hwang, and W. Wang, "KBQA: Learning Question Answering over QA Corpora and Knowledge Bases," *PVLDB*, vol. 10, no. 5, pp. 565–576, 2017.

[15] E. Agichtein, S. Cucerzan, and E. Brill, "Analysis of Factoid Questions for Effective Relation Extraction," in *SIGIR*, 2005.

[16] "QALD-4," http://qald.aksw.org/index.php?x=challenge&q=4, 2014.

[17] A. Bonifati, W. Martens, and T. Timm, "An Analytical Study of Large SPARQL Query Logs," *PVLDB*, vol. 11, no. 2, pp. 149–161, 2017.

[18] A. Khan, Y. Wu, C. C. Aggarwal, and X. Yan, "NeMa: Fast Graph Search with Label Similarity," *PVLDB*, vol. 6, no. 3, pp. 181–192, 2013.

[19] L. Zou, R. Huang, H. Wang, J. X. Yu, W. He, and D. Zhao, "Natural Language Question Answering over RDF: A Graph Driven Approach," in *SIGMOD*, 2014.

[20] S. Yang, F. Han, Y. Wu, and X. Yan, "Fast Top-k Search in Knowledge Graphs," in *ICDE*, 2016.

[21] J. Jin, S. Khemmarat, L. Gao, and J. Luo, "Querying Web-Scale Information Networks Through Bounding Matching Scores," in *WWW*, 2015.

[22] L. Zou, M. T. Özsu, L. Chen, X. Shen, R. Huang, and D. Zhao, "gStore: A Graph-based SPARQL Query Engine," *VLDB J.*, vol. 23, no. 4, pp. 565–590, 2014.

[23] W. Zheng, L. Zou, X. Lian, J. X. Yu, S. Song, and D. Zhao, "How to Build Templates for RDF Question/Answering: An Uncertain Graph Similarity Join Approach," in *SIGMOD*, 2015.

[24] S. Han, L. Zou, J. X. Yu, and D. Zhao, "Keyword Search on RDF Graphs - A Query Graph Assembly Approach," in *CIKM*, 2017.

[25] X. Hu, D. Dang, Y. Yao, and L. Ye, "Natural Language Aggregate Query over RDF Data," *Inf. Sci.*, vol. 454-455, pp. 363–381, 2018.

[26] J. Cheng, J. X. Yu, B. Ding, S. Y. Philip, and H. Wang, "Fast Graph Pattern Matching," in *ICDE*, 2008.

[27] A. Khan, N. Li, X. Yan, Z. Guan, S. Chakraborty, and S. Tao, "Neighborhood Based Fast Graph Search in Large Networks," in *SIGMOD*, 2011.

[28] W. Fan, J. Li, S. Ma, H. Wang, and Y. Wu, "Graph Homomorphism Revisited for Graph Matching," *PVLDB*, vol. 3, no. 1-2, pp. 1161–1172, 2010.

[29] M. Lissandrini, T. B. Pedersen, K. Hose, and D. Mottin, "Knowledge Graph Exploration: Where Are We and Where Are We Going?" *ACM SIGWEB Newsletter*, no. 4, 2020.

[30] Y. Wu and A. Khan, "Graph Pattern Matching," in *Encyclopedia of Big Data Technologies*. Springer, 2019.

[31] J. Jin, S. Khemmarat, L. Gao, and J. Luo, "A Distributed Approach for Top-k Star Queries on Massive Information Networks," in *ICPADS*, 2014.

[32] N. Laptev, K. Zeng, and C. Zaniolo, "Early Accurate Results for Advanced Analytics on Mapreduce," *PVLDB*, vol. 5, no. 10, pp. 1028–1039, 2012.

[33] S. Chaudhuri, B. Ding, and S. Kandula, "Approximate Query Processing: No Silver Bullet," in *SIGMOD*, 2017.

[34] F. Li, B. Wu, K. Yi, and Z. Zhao, "Wander Join: Online Aggregation via Random Walks," in *SIGMOD*, 2016.

[35] S. S. Bhowmick, B. Choi, and S. Zhou, "VOGUE: Towards A Visual Interaction-aware Graph Query Processing Framework," in *CIDR*, 2013.

[36] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica, "BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data," in *Eurosys*, 2013.

[37] S. Wu, B. C. Ooi, and K.-L. Tan, "Continuous Sampling for Online Aggregation over Multiple Queries," in *SIGMOD*, 2010.

[38] J. M. Hellerstein, P. J. Haas, and H. J. Wang, "Online Aggregation," in *SIGMOD*, 1997.

[39] B. Mozafari and N. Niu, "A Handbook for Building an Approximate Query Engine." *IEEE Data Eng. Bull.*, vol. 38, no. 3, pp. 3–29, 2015.

[40] D. Huang, D. Y. Yoon, S. Pettie, and B. Mozafari, "Join on Samples: A Theoretical Guide for Practitioners," *PVLDB*, vol. 13, no. 4, pp. 547–560, 2019.

[41] S. Chaudhuri, G. Das, and V. R. Narasayya, "Optimized Stratified Sampling for Approximate Query Processing," *ACM Trans. Database Syst.*, vol. 32, no. 2, p. 9, 2007.

[42] Y. Li, Z. Wu, S. Lin, H. Xie, M. Lv, Y. Xu, and J. C. Lui, "Walking with Perception: Efficient Random Walk Sampling via Common Neighbor Awareness," in *ICDE*, 2019.

[43] A. Grover and J. Leskovec, "Node2vec: Scalable Feature Learning for Networks," in *KDD*, 2016.

[44] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating Embeddings for Modeling Multi-relational Data," in *NIPS*, 2013.

[45] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge Graph Embedding via Dynamic Mapping Matrix," in *ACL*, 2015.

[46] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge Graph Embedding by Translating on Hyperplanes," in *AAAI*, 2014.

[47] W. Zheng, J. X. Yu, L. Zou, and H. Cheng, "Question Answering over Knowledge Graphs: Question Understanding via Template Decomposition," *PVLDB*, vol. 11, no. 11, pp. 1373–1386, 2018.

[48] N. Nakashole, T. Tylenda, and G. Weikum, "Fine-Grained Semantic Typing of Emerging Entities," in *ACL*, 2013, pp. 1488–1497.

[49] J. Bao, N. Duan, Z. Yan, M. Zhou, and T. Zhao, "Constraint-based Question Answering with Knowledge Graph," in *COLING*, 2016.

[50] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale Simple Question Answering with Memory Networks," *arXiv:1506.02075*, 2015.

[51] S. Yang, X. Yan, B. Zong, and A. Khan, "Towards Effective Partition Management for Large Graphs," in *SIGMOD*, 2012.

[52] S. Cucerzan, "Large-scale Named Entity Disambiguation based on Wikipedia Data," in *EMNLP-CoNLL*, 2007.

[53] Y. Li, S. Tan, H. Sun, J. Han, D. Roth, and X. Yan, "Entity Disambiguation with Linkless Knowledge Bases," in *WWW*, 2016.

[54] L. Hu, J. Ding, C. Shi, C. Shao, and S. Li, "Graph Neural Entity Disambiguation," *Knowl. Based Syst.*, vol. 195, p. 105620, 2020.

[55] J. Jin, J. Luo, S. Khemmarat *et al.*, "Querying web-scale knowledge graphs through effective pruning of search space," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 8, pp. 2342–2356, 2017.

[56] J. Zhao, J. C. Lui, D. Towsley, P. Wang, and X. Guan, "A Tale of Three Graphs: Sampling Design on Hybrid Social-Affiliation Networks," in *ICDE*, 2015.

[57] S. M. Ross, *Introduction to Probability Models*. Academic press, 2014.

[58] Y. Wang, A. Khan, X. Xu, J. Jin, Q. Hong, and T. Fu, "Aggregate queries on knowledge graphs: Fast approximation with semantic-aware sampling," *arXiv:2203.03792*, 2022.

[59] D. G. Horvitz and D. J. Thompson, "A Generalization of Sampling without Replacement from a Finite Universe," *Journal of the American statistical Association*, vol. 47, no. 260, pp. 663–685, 1952.

[60] G. L. Jones, "On the Markov Chain Central Limit Theorem," *Probability Surveys*, vol. 1, no. 299-320, pp. 5–1, 2004.

[61] C.-H. Lee, X. Xu, and D. Y. Eun, "Beyond Random Walk and Metropolis-Hastings Samplers: Why You Should Not Backtrack for Unbiased Graph Sampling," in *SIGMETRICS*, 2012.

[62] M. H. Hansen and W. N. Hurwitz, "On the Theory of Sampling from Finite Populations," *The Annals of Mathematical Statistics*, vol. 14, no. 4, pp. 333–362, 1943.

[63] S. Coles, J. Bawa, L. Trenner, and P. Dorazio, *An Introduction to Statistical Modeling of Extreme Values*. Springer, 2001, vol. 208.

[64] A. Kleiner, A. Talwalkar, P. Sarkar, and M. I. Jordan, "A Scalable Bootstrap for Massive Data," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 76, no. 4, pp. 795–816, 2014.

[65] J. Shao and D. Tu, *The Jackknife and Bootstrap*. Springer Science & Business Media, 2012.

[66] P. J. Haas and J. M. Hellerstein, "Ripple Joins for Online Aggregation," in *SIGMOD*, 1999.

[67] C. M. Jermaine, S. Arumugam, A. Pol, and A. Dobra, "Scalable Approximate Query Processing with the DBO Engine," in *SIGMOD*, 2007.

[68] G. Luo, C. J. Ellmann, P. J. Haas, and J. F. Naughton, "A Scalable Hash Ripple Join Algorithm," in *SIGMOD*, 2002.

[69] Y. Park, B. Mozafari, J. Sorenson, and J. Wang, "VerdictDB: Universalizing Approximate Query Processing," in *SIGMOD*, 2018.

[70] S. Acharya, P. B. Gibbons, and V. Poosala, "Congressional Samples for Approximate Answering of Group-By Queries," in *SIGMOD*, 2000.

[71] B. Ding, S. Huang, S. Chaudhuri, K. Chakrabarti, and C. Wang, "Sample + Seek: Approximating Aggregates with Distribution Precision Guarantee," in *SIGMOD*, 2016.

[72] Y. Shi, X. Meng, F. Wang, and Y. Gan, "You Can Stop Early with COLA: Online Processing of Aggregate Queries in the Cloud," in *CIKM*, 2012.

[73] S. Wu, S. Jiang, B. C. Ooi, and K. Tan, "Distributed Online Aggregation," *PVLDB*, vol. 2, no. 1, pp. 443–454, 2009.

[74] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, J. Gerth, J. Talbot, K. Elmeleegy, and R. Sears, "Online Aggregation and Continuous Query Support in Mapreduce," in *SIGMOD*, 2010.

[75] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "Mapreduce Online," in *NSDI*, 2010.

[76] N. Pansare, V. R. Borkar, C. Jermaine, and T. Condie, "Online Aggregation for Large MapReduce Jobs," *PVLDB*, vol. 4, no. 11, pp. 1135–1145, 2011.

[77] K. Zeng, S. Agarwal, A. Dave, M. Armbrust, and I. Stoica, "G-OLA: Generalized On-Line Aggregation for Interactive Analysis on Big Data," in *SIGMOD*, 2015.

[78] Y. Wang, J. Luo, A. Song, and F. Dong, "OATS: Online Aggregation with Two-Level Sharing Strategy in Cloud," *Distributed Parallel Databases*, vol. 32, no. 4, pp. 467–505, 2014.

[79] W. Zheng, X. Lian, L. Zou, L. Hong, and D. Zhao, "Online Subgraph Skyline Analysis over Knowledge Graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1805–1819, 2016.

[80] S. Ma, Y. Cao, W. Fan, J. Huai, and T. Wo, "Strong Simulation: Capturing Topology in Graph Pattern Matching," *ACM TODS*, vol. 39, no. 1, p. 4, 2014.

[81] W. Zheng, L. Zou, X. Lian, D. Wang, and D. Zhao, "Graph Similarity Search with Edit Distance Constraint in Large Graph Databases," in *CIKM*, 2013.

[82] X. Hu, J. Duan, and D. Dang, "Scalable Aggregate Keyword Query over Knowledge Graph," *Future Gener. Comput. Syst.*, vol. 107, pp. 588–600, 2020.

[83] C. Unger, L. Bühmann, J. Lehmann, A. N. Ngomo, D. Gerber, and P. Cimiano, "Template-based Question Answering over RDF Data," in *WWW*, 2012.

[84] K. Höffner, J. Lehmann, and R. Usbeck, "CubeQA - Question Answering on RDF Data Cubes," in *ISWC*, 2016.

[85] Y. Li, T. Ge, and C. X. Chen, "Online Indices for Predictive Top-k Entity and Aggregate Queries on Knowledge Graphs," in *ICDE*, 2020.

[86] "Our Code and Data," https://anonymous.4open.science/r/60487db8-a5a4-46d6-bfeb-cbb326df0c4d/, 2021.

[87] "Freebase Links," http://downloads.dbpedia.org/2016-10/core-i18n/en/, 2016.

[88] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic Parsing on Freebase from Question-answer Pairs," in *EMNLP*, 2013.

[89] "Apache Jena," https://jena.apache.org/, 2021.

[90] "Neo4j," https://neo4j.com/.

[91] M. Nickel, V. Tresp, and H. Kriegel, "A Three-Way Model for Collective Learning on Multi-Relational Data," in *ICML*, 2011.

[92] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning Structured Embeddings of Knowledge Bases," in *AAAI*, 2011.

[93] Z. Sun, Z. Deng, J. Nie, and J. Tang, "RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space," in *ICLR*, 2019.