# Approximate and Interactive Processing of Aggregate Queries on Knowledge Graphs: A Demonstration


Yuxiang Wang
lsswyx@hdu.edu.cn
Hangzhou Dianzi University
Hangzhou, China

Arijit Khan
arijitk@cs.aau.dk
Aalborg University
Aalborg, Denmark

Xiaoliang Xu
xxl@hdu.edu.cn
Hangzhou Dianzi University
Hangzhou, China

Shuzhan Ye
yeshuzhan123@hdu.edu.cn
Hangzhou Dianzi University
Hangzhou, China

Shihuang Pan
shpan@hdu.edu.cn
Hangzhou Dianzi University
Hangzhou, China

Yuhan Zhou
yhzhou@hdu.edu.cn
Hangzhou Dianzi University
Hangzhou, China



## ABSTRACT

This paper demonstrates AGQ [26] — our system for approximate and interactive processing of **ag**gregate **q**ueries on knowledge graphs (KGs), e.g., "*what is the average price of cars produced in Germany?*" One can support aggregate queries based on factoid queries, e.g., "*find all cars produced in Germany*", by applying an aggregate operation on factoid queries' answers. However, this straightforward method is problematic since both the accuracy and efficiency of factoid query processing would impact the performance of aggregate queries. Moreover, returning a one-time, exact result might add computation overhead and hinder users' engagement and interactivity.

To this end, we design a system, called AGQ which employs a "sampling-estimation" model to answer aggregate queries over KGs. This is the first work to provide an approximate aggregate result with effective and interactive accuracy guarantees, and without relying on factoid queries. Our demonstration highlights **(1)** a novel semantic-aware sampling to collect a high-quality random sample through a random walk based on KG embedding, followed by our unbiased (or, consistent) estimators for {COUNT, SUM, AVG} to compute the approximate aggregate results using the random sample, with a confidence interval-based accuracy guarantee. **(2)** AGQ supports interactive improvements of accuracy, complex queries with filter, GROUP-BY, MAX/MIN, and different graph shapes, e.g., chain, cycle, star, flower. **(3)** Its GUI helps users compare simple and complex aggregate queries, intermediate results as the queries progress, confidence intervals, relative errors, and various schemas for different valid answers in a user-friendly and interactive manner. Additionally, our system permits users to input queries in natural languages, keywords, or to select from a set of example graph queries.


## CCS CONCEPTS

**• Information systems → Database query processing**; **• Mathematics of computing → Approximation algorithms**.


Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.


*CIKM '22, October 17–21, 2022, Atlanta, GA, USA*


© 2022 Association for Computing Machinery.


ACM ISBN 978-1-4503-9236-5/22/10…$15.00
https://doi.org/10.1145/3511808.3557158



## KEYWORDS

Knowledge graph, Approximate aggregate query, Random walk

**ACM Reference Format:**
Yuxiang Wang, Arijit Khan, Xiaoliang Xu, Shuzhan Ye, Shihuang Pan, and Yuhan Zhou. 2022. Approximate and Interactive Processing of Aggregate Queries on Knowledge Graphs: A Demonstration. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22), October 17–21, 2022, Atlanta, GA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3511808.3557158


## 1 INTRODUCTION

Knowledge graphs (KGs) such as DBpedia [19], YAGO [13], Freebase [5], and NELL [23] have been built to manage large-scale, real-world facts in a schema-flexible manner, where a node denotes an entity with attributes, and an edge is a relationship between two entities [10, 28]. We focus on aggregate queries over KGs, e.g., "*what is the average price of cars produced in Germany?*" — 31% queries from the real query log *LinkedGeoData13* and 30% queries from the manually-curated query set *WikiData17* are aggregate queries [6].

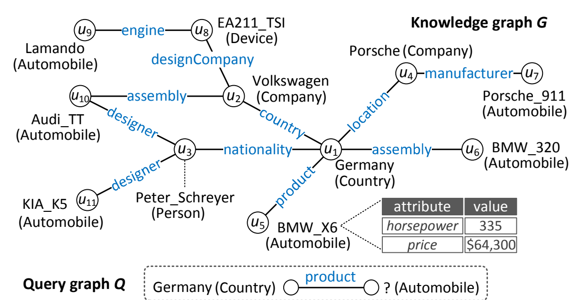


**Figure 1: A KG $G$ and a query graph $Q$. Each entity of $G$ with type *Automobile* has many numerical attributes, e.g., *horsepower*, *price*.**

Question answering (both aggregate and factoid queries) over KGs is challenging due to KG's "*schema-flexible*" nature [25, 30, 31, 33]: *The same kind of information can be represented as diverse substructures*. Consider the factoid query: "*Find all cars produced in Germany*" (Q117 from QALD-4 benchmark [1]) over the KG in Figure 1, 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. These correct answers are linked to *Germany* in structurally different ways in Figure 1, for instance, $u_{10}$: Audi_TT-*assembly*-Volkswagen-*country*-Germany; $u_6$:

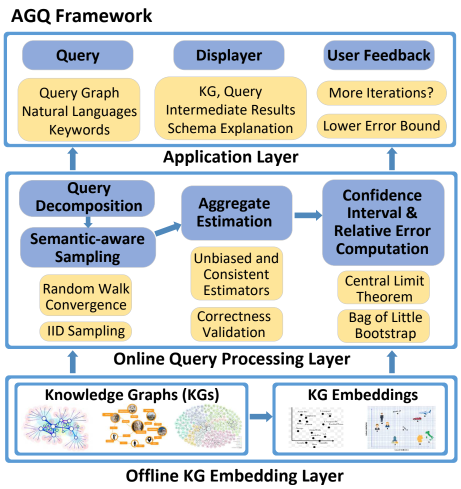


**Figure 2: The architecture of the AGQ system**

BMW_320-*assembly*-Germany. This reflects the "*schema-flexible*" nature of a KG. We aim at finding all semantically similar answers.

A commonly used technology for factoid queries is graph querying [17, 25, 29, 34, 35], which we consider in this work: A user forms a query graph $Q$ to describe her query intention, and identifies the exact or approximate matches of $Q$ in a KG $G$. Subgraph isomorphism [35] only returns answers that *exactly* match with the given $Q$ (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 [25, 26, 30]. Other works [17, 33] return approximate matches to $Q$, but it might be difficult for them to return 100% accurate answers (the notion of "accurate" could depend on the user's query intension, or may even be vague [22]). So, when one processes aggregate queries by an aggregate operation on factoid queries' answers, calculating the aggregate result over answers with low quality leads to significant errors. We also lack an effective way to quantify the result's quality. Finally, factoid queries' efficiency affects aggregate queries' efficiency.

Practically, an aggregate query often does not need an exact result. An early accurate-enough approximate result is more valuable to users, while we can improve the accuracy as more time is given [9], thus enhancing users' experience and saving resources [4, 25].

**Our solution**. We propose the "*semantic similarity*" [25, 26] to measure how semantically similar an answer is to a query graph $Q$. We then develop an iterative and approximate approach to efficiently answer aggregate queries over KGs with an accuracy guarantee, but without requiring factoid queries. We first collect semantically similar answers to $Q$ as a random sample $S$ from a KG. Next, we estimate an unbiased (or consistent) approximate result $\hat{V}$ based on $S$, and provide an accuracy guarantee by iteratively computing a confidence interval CI $= \hat{V} \pm \varepsilon$ at a confidence level $1 - \alpha$. We terminate the query when a tight CI with a small $\varepsilon$ is obtained, and ensure that the relative error of $\hat{V}$ is bounded by a user-specific error bound $e$. To the best of our knowledge, *we are the first to use a "sampling-estimation" model to answer aggregate queries on KGs, together with interactive improvements in error bounds*. The demonstration of our system, AGQ is also the first demonstration proposal on *fast, approximate, and interactive processing of* **ag**gregate **q**ueries over KGs.

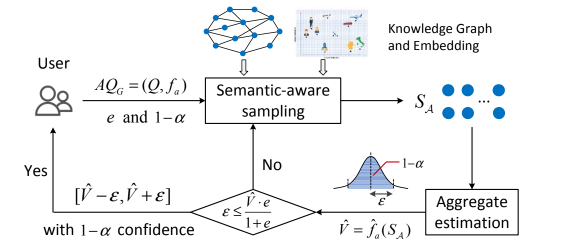


**Figure 3: Workflow of AGQ**

## 2 FRAMEWORK OVERVIEW

Figure 2 shows AGQ's architecture with three layers: *KG embedding* layer, *online query processing* layer, and *application* layer.

• Offline KG Embedding Layer. A KG embedding learns representations of entities and relations in a low-dimensional vector space, it can preserve well the semantic meanings and relations using these learned vectors [7, 14, 25]. We leverage an offline KG embedding model to obtain the predicate similarity between two edges from the KG and query graph, and subsequently the semantic similarity of a path from the KG with a query edge. AGQ can work with any KG embedding. By employing a high-quality embedding, we can distinguish the implicit semantics of predicates and different paths, this is critical for finding all semantically similar answers to the query graph. In the full version [26], we demonstrated that translation-based models (TransE [7], TransD [15], TransH [27]) perform better than tensor factorization-based models, e.g., RESCAL [24] and relation-specific projection-based models, e.g., SE [8].

• Online Query Processing Layer. Given complex queries with different shapes, e.g., chain, star, cycle, and flower, we adopt the "decomposition-assembly" framework [25]: We decompose a complex query into a set of simple or chain-shaped queries that share the same target entity. In particular, a simple query consists of a single edge, connecting a specific entity with the target entity, returning possible matches to target entity as query answers [14]. Each simple query is processed by **(a)** collecting a random sample of candidate answers from the KG that are semantically similar to the query graph, and **(b)** estimating an unbiased (or consistent) approximate aggregate result based on the random sample, together with **(c)** an accuracy guarantee in the form of confidence interval.

• Application Layer: User Feedback and Displayer. If the estimated error is higher than the user-input error bound, we enlarge the random sample with additional candidate answers and repeat the estimation till an acceptable accuracy is attained. During runtime, users can interactively reduce the error bound to achieve more accurate results. AGQ contains a user-friendly *Displayer* module (developed with the D3.js library) for interactive visualization. Users can interact with the KG, intermediate results, confidence intervals at every round, and various schemas for different valid answers. Finally, users can input queries in multiple forms: (a) select from a given set of natural language and graph queries, and (b) input ad-hoc queries in natural languages and keywords. AGQ extracts entities and predicates from text and converts these query forms to graph queries [12, 21, 32].

## 3 ALGORITHMS AND PERFORMANCE

The workflow of our AGQ framework is presented in Figure 3.

**Aggregate Query.** An aggregate query over KG $G$ is denoted as $AQ_G = (Q, f_a)$, where $Q$ is a query graph for finding candidate

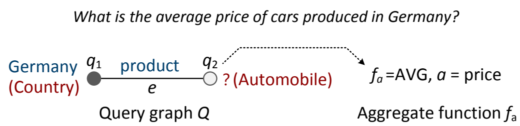


**Figure 4: An aggregate query** $AQ_G = (Q, f_a)$

answers from $G$ and $f_a$ is an aggregate function on the numerical attribute $a$ of the answers to $Q$. Figure 4 shows the $AQ_G$ of a simple query "*what is the average price of cars produced in Germany?*". 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 $e = q_1q_2$ (predicate: product), and $f_a$ = AVG on the attribute $a$ = price. Given our framework for simple questions, we later extend it for more complex queries and shapes.

**Semantic Similarity.** We employ an offline *KG embedding* to compute the predicate similarity between two edges $e'$ and $e$, from $G$ and $Q$, respectively. The similarity between their predicates, denoted as $sim(L_G(e'), L_Q(e))$, is computed by the cosine similarity between their predicate vectors $\boldsymbol{e}$ and $\boldsymbol{e'}$, obtained via KG embedding.

$$sim(L_G(e'), L_Q(e)) = \frac{\boldsymbol{e'} \cdot \boldsymbol{e}}{||\boldsymbol{e'}|| \times ||\boldsymbol{e}||} \tag{1}$$

Given a simple query graph $Q$ and a KG $G$, 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) The specific node $q^s$ is mapped to $u^s$, having similar name and type. (2) The target node $q^t$ is mapped to $u_i^t$ having similar type. 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 $l$ is the length of $\overline{u^s u_i^t}$ [25]. 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).

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

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

Due to the schema-flexible nature of KGs, there can be many semantically similar, yet structurally different subgraph matches to a given $Q$. We set a threshold $\tau$ and view those candidate answers having semantic similarity $s_i \geq \tau$ as the *correct answers* to $Q$, denoted by $\mathcal{A}^+ = \{u_i^t \in \mathcal{A} : s_i \geq \tau\}$. The ground truth of $AQ_G$ is $V = f_a(\mathcal{A}^+)$.

**Approximate Answers.** Given an aggregate query $AQ_G = (Q, f_a)$, a KG $G$, an input error bound $e$, and a confidence level $1-\alpha$, our system AGQ (1) designs 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) estimates 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 $e$. $\hat{f}_a$ is an unbiased (or consistent) estimator of $f_a$.

$$\hat{V} = \hat{f}_a(S_{\mathcal{A}}) \quad \text{s.t.} \quad \Pr[\hat{V} - \varepsilon \leq V \leq \hat{V} + \varepsilon] = 1 - \alpha, \quad |\hat{V} - V|/V \leq e \tag{4}$$

**Semantic-aware Sampling.** We collect answers with higher semantic similarities to $Q$ as a random sample $S_{\mathcal{A}}$, via our novel semantic-aware random walk sampling over $G$. We design a transition matrix to guide a walker towards an answer $u_i^t$ with the greatest $s_i$ as much as possible, so this $u_i^t$ is more likely to have a greater visiting probability at convergence. Hence, we assign a greater transition probability on each edge $e'$ from $G$ having a higher predicate similarity $sim(L_G(e'), L_Q(e))$ to the query edge $e$ from $Q$. We ensure that our random walk converges to a stationary distribution, and all the answers in $S_{\mathcal{A}}$ are i.i.d. random variables. We depicted in [26] that our semantic-aware sampling is more effective in finding correct answers compared to topology-based sampling, e.g., Node2Vec [11].

**Table 1: Accuracy (considering human-annotated ground truth) and efficiency for various aggregate queries (*DBpedia*)**

| Method | Relative error (%) | | | Efficiency (sec) | | |
|---|---|---|---|---|---|---|
| | Simple | Filter | GROUP-BY | Simple | Filter | GROUP-BY |
| **AGQ** | **0.99** | **0.71** | **1.13** | **0.37** | **0.43** | **31.67** |
| EAQ | 21.14 | - | - | 4.53 | - | - |
| GraB | 7.31 | 20.94 | - | 9.23 | 1.10 | - |
| QGA | 19.01 | 46.95 | - | 1.51 | 1.41 | - |
| SGQ | 9.97 | 17.71 | - | 0.82 | 0.73 | - |
| JENA | 17.62 | 48.98 | 16.30 | 1.20 | 0.70 | 95.76 |
| Virtuoso | 17.62 | 48.98 | 16.30 | 1.21 | 0.72 | 94.67 |

**Correctness Validation.** Due to randomness of sampling, a few answers with lower semantic similarity might still be collected in $S_{\mathcal{A}}$. We design an efficient heuristic algorithm to remove sampled answers with semantic similarity $< \tau$, thus improving the accuracy.

**Aggregate Estimation.** We develop unbiased estimators for {SUM, COUNT} and a consistent estimator for AVG, that is, it converges almost surely to the true expectation. We derive an approximate result $\hat{V}$ using these estimators.

**Accuracy Guarantee, Additional Sampling, and Interactive Refinement.** Given a user-input confidence level $1 - \alpha$, we compute a confidence interval CI = $\hat{V} \pm \varepsilon$ to quantify $\hat{V}$'s quality using the *Central Limit Theorem* and the *bag of little bootstrap* method [18]. We show that when $\varepsilon$ is small enough to satisfy $\leq \hat{V} \cdot e/(1+e)$, the relative error is bounded by a user-input error bound $e$. Otherwise, we update $S_{\mathcal{A}}$ with additional $\Delta S_{\mathcal{A}}$ answers. To avoid over- and under-sampling, we propose an error-based method that automatically configures $|\Delta S_{\mathcal{A}}|$. Moreover, AGQ permits interactive refinements in error bound $e$ during runtime. It quickly obtains a new approximate result with a small additional overhead, because the error-based method can sense the variation of $e$ and updates $|\Delta S_{\mathcal{A}}|$ accordingly.

**Filters, GROUP-BY, MAX/MIN, and Complex Shapes.** We support filter operations during correctness validation. For GROUP-BY, we divide the collected sample into different groups, then estimate the approximate result for each group. For MAX/MIN, we cannot provide accuracy guarantees, though we can support them by returning the MAX/MIN answers from the sample. For queries with complex shapes, e.g., chain, star, cycle, flower, we first decompose them into a set of simple or chain-shaped queries that share the same target entity. We omit details due to lack of space and refer to [26].

**System Performance.** We run experiments on a 2.1GHZ, 64GB memory AMD-6272 server, and compare our system AGQ with recent works on KG search: EAQ [20], SGQ [25], GraB [16], and QGA [12] (Table 1). Since SGQ, GraB, and QGA process factoid queries, we extend them by adding an additional aggregate operation after obtaining factoid query answers. We also compare with one RDF store, JENA [3] and a SPARQL endpoint Virtuoso, both supporting SPARQL queries. The ground truth is obtained based on crowdsourcing-based human annotations [26]. AGQ *reduces the relative error by two orders of magnitude, than existing methods; and requires up to an order of magnitude less response time than other approaches*. Our **code and all datasets** are available at [2].

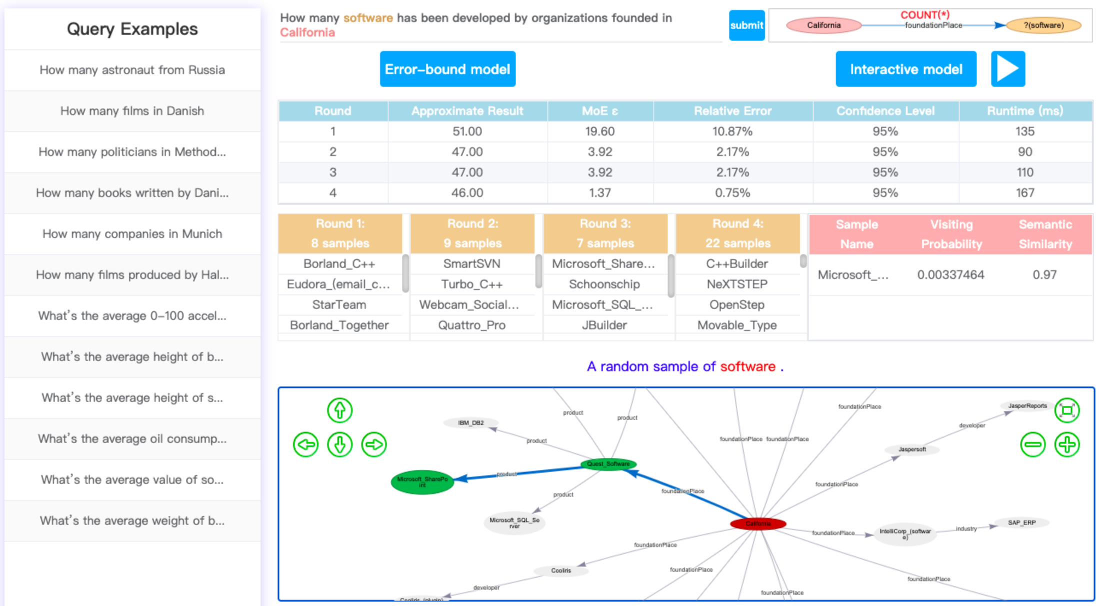


**Figure 5: User interface of** AGQ**: natural language query form with interactive query model**

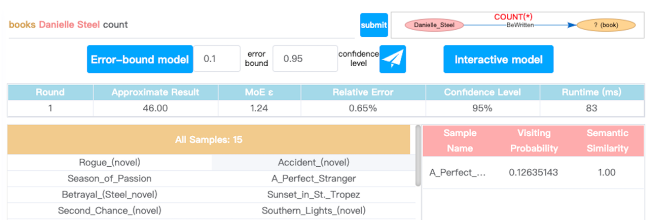


**Figure 6: Keyword query forms with error-bound query model**

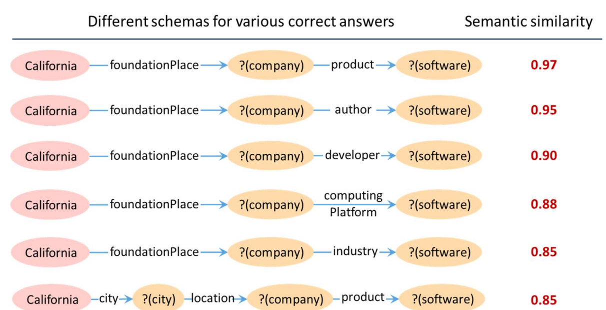


**Figure 7:** AGQ **finds different schemas for various correct answers**

# 4 DEMOSTRATION

**Demonstration and Audience.** We present a web app for demonstration. A video is available at the **YouTube - https://www.youtube.com/watch?v=FEJeIIxDizQ**. It is intended for researchers, system designers, data scientists, practitioners, and enthusiasts in the broad area of data management, querying, knowledge graphs, complex networks, Web science, graph embedding, machine learning and deep learning.

**User Interface and Interactive-ness.** Figure 5 presents the user interface of AGQ. When we open this web app for the first time, it shows a snapshot of the input KG on the right bottom. A set of sample aggregate queries are provided in the left panel. Alternatively, users can input ad-hoc natural language and keyword queries by typing them on the top panel, as shown in Figure 5 and 6, respectively. We extract various entities and predicates from input text following [12, 21, 32], and convert them to graph queries. When the user inputs one query, its equivalent query graph is shown at the top of the monitor window, and a partial knowledge graph containing the specific entity is presented on the right bottom.

When the user submits a query, AGQ provides two options to process it. **First,** in the *interactive mode* (see Figure 5), AGQ shows the first round's estimated result with confidence interval, relative error, and running time in a blue table on the top; all collected samples for estimation are presented in another orange table below it. When the user selects one sample from the orange table, our interface shows its detailed information in a red table on the top right, and also highlights the corresponding path from the specific entity in the partial KG. Finally, the user may proceed to subsequent rounds by pressing the "continue" button, and more accurate results are provided round-by-round. **Second,** in the *error-bound mode*, the user inputs the desired error bound and confidence level. AGQ computes approximate result and directly reports the final result with confidence interval, relative error, and running time (see Figure 6).

**Demonstration with the *DBPedia* Dataset.** *DBpedia* is an open-domain KG constructed from Wikipedia, with about 4.5M nodes, 15M edges, 359 distinct node types, and 676 distinct edge predicates. We shall demonstrate our system AGQ with *DBpedia*. Figure 7 shows six different schemas that are used in answering the query: "*How many software has been developed by organizations founded in California?*", which are all identified by the AGQ system.

# ACKNOWLEDGMENTS

This work was supported by the National NSF of China (62072149), the Novo Nordisk Foundation (NNF22OC0072415), the Primary Research & Development Plan of Zhejiang Province (2021C03156 and 2021C02004), and the Fundamental Research Funds for the Provincial Universities of Zhejiang (GK219909299001-006). We thank the Key Laboratory of Brain Machine Collaborative Intelligence of Zhejiang (2020E10010).

# REFERENCES


[1] 2014. QALD-4. http://qald.aksw.org/index.php?x=challenge&q=4.

[2] 2021. Our Code and Data. https://anonymous.4open.science/r/60487db8-a5a4-46d6-bfeb-cbb326df0c4d/.

[3] 2022. Apache Jena. https://jena.apache.org/.

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

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

[6] A. Bonifati, W. Martens, and T. Timm. 2017. An Analytical Study of Large SPARQL Query Logs. *PVLDB* 11, 2 (2017), 149–161.

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

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

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

[10] J. Gao, X. Li, Y. E. Xu, B. Sisman, X. L. Dong, and J. Yang. 2019. Efficient Knowledge Graph Accuracy Evaluation. *PVLDB* 12, 11 (2019), 1679–1691.

[11] A. Grover and J. Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks. In *KDD*.

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

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

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

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

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

[17] A. Khan, Y. Wu, C. C. Aggarwal, and X. Yan. 2013. NeMa: Fast Graph Search with Label Similarity. *PVLDB* 6, 3 (2013), 181–192.

[18] A. Kleiner, A. Talwalkar, P. Sarkar, and M. I Jordan. 2014. A Scalable Bootstrap for Massive Data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 76, 4 (2014), 795–816.

[19] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. v. Kleef, S. Auer, and C. Bizer. 2015. DBpedia - A Large-Scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195.

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

[21] S. Liang, K. Stockinger, T. M. de Farias, M. Anisimova, and M. Gil. 2021. Querying Knowledge Graphs in Natural Language. *J. Big Data* 8, 1 (2021), 3.

[22] M. Lissandrini, T. B. Pedersen, K. Hose, and D. Mottin. 2020. Knowledge Graph Exploration: Where Are We and Where Are We Going? *ACM SIGWEB Newsletter* 4 (2020).

[23] T. M. Mitchell, W. W. Cohen, E. R. Hruschka 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. 2018. Never-Ending Learning. *Commun. ACM* 61, 5 (2018), 103–115.

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

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

[26] Y. Wang, A. Khan, X. Xu, J. Jin, Q. Hong, and T. Fu. 2022. Aggregate Queries on Knowledge Graphs: Fast Approximation with Semantic-aware Sampling. In *ICDE*.

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

[28] G. Weikum, X. L. Dong, S. Razniewski, and F. M. Suchanek. 2021. Machine Knowledge: Creation and Curation of Comprehensive Knowledge Bases. *Found. Trends Databases* 10, 2-4 (2021), 108–490.

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

[30] S. Yang, Y. Wu, H. Sun, and X. Yan. 2014. Schemaless and Structureless Graph Querying. *PVLDB* 7, 7 (2014), 565–576.

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

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

[33] W. Zheng, L. Zou, W. Peng, X. Yan, S. Song, and D. Zhao. 2016. Semantic SPARQL Similarity Search over RDF Knowledge Graphs. *PVLDB* 9, 11 (2016), 840–851.

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

[35] L. Zou, M. T. Özsu, L. Chen, X. Shen, R. Huang, and D. Zhao. 2014. gStore: A Graph-based SPARQL Query Engine. *VLDB J.* 23, 4 (2014), 565–590.