Evaluating Knowledge Graph Accuracy Powered by Optimized Human-machine Collaboration

Yifan Qi
School of Data Science, Fudan University
yfqi21@m.fudan.edu.cn

Weiguo Zheng*
School of Data Science, Fudan University
zhengweiguo@fudan.edu.cn

Liang Hong
School of Information Management,
Big Data Institute, Wuhan University
hong@whu.edu.cn

Lei Zou
Wangxuan Institute of Computer Technology,
Institute for Artificial Intelligence, Peking University
zoulei@pku.edu.cn

ABSTRACT

Estimating the accuracy of an automatically constructed knowledge graph (KG) becomes a challenging task as the KG often contains a large number of entities and triples. Generally, two major components information extraction (IE) and entity linking (EL) are involved in KG construction. However, the existing approaches just focus on evaluating the triple accuracy that indicates the IE quality, completely ignoring the entity accuracy. Motivated by the fact that the major advance of machines is the strong computing power while humans are skilled in correctness verification, we propose an efficient interactive method to reduce the overall cost for evaluating the KG quality, which produces accuracy estimates with a statistical guarantee for both triples and entities. Instead of annotating triples and entities separately, we design a general annotation cost that blends triples and entities generated from the identical source text. During human verification, the machine can pre-compute and infer triples to be annotated in the next round by speculating human feedback. The human-machine collaborative mechanism is optimized by formulating an order selection problem of triples which is NP-hard. Thus, a Monte Carlo Tree Search is proposed to guide the annotation process by finding an approximate solution. Extensive experiments demonstrate that our method takes less annotation cost while yielding higher accuracy estimation quality compared to the state-of-the-art approaches.

CCS CONCEPTS

• **Information systems → Data cleaning**; • **Human-centered computing → Human computer interaction (HCI)**.

KEYWORDS

Knowledge graph; Accuracy estimation; Human-machine collaboration; Cost optimization

*Weiguo Zheng is the corresponding author.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539233

ACM Reference Format:
Yifan Qi, Weiguo Zheng, Liang Hong, and Lei Zou. 2022. Evaluating Knowledge Graph Accuracy Powered by Optimized Human-machine Collaboration. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539233

1 INTRODUCTION

Knowledge graphs (KGs), also known as knowledge bases, have become important repositories for structured knowledge that are automatically extracted from web contents or text documents. An increasing number of large-scale knowledge graphs are constantly emerging, e.g., NELL (Never-Ending Language Learner) [21], DeepDive [22], Knowledge-Vault [7], Social-Impact Funding [18], and E-commerce [20]. Typically, automatic KG construction involves two critical tasks, i.e., information extraction (IE) and entity linking (EL). The task IE conducts named entity recognition and relation extraction to extract entities and triples, representing relations between entity mentions, from textual corpus or semi-structured data. To integrate these triples and build a structured knowledge graph, it is necessary to identify mentions which refer to the same entity in the real world (i.e., a node in the KG), which is the task EL. Although lots of efforts have been made to build such knowledge graphs from a large corpus automatically, the quality of knowledge graphs may be still far from perfect. Due to the notorious variability of natural language, neither of the two tasks above is guaranteed to never make mistakes. The mistakes made by the machine are hard to be identified by the machine itself, reducing the utility of the constructed knowledge graph in the downstream applications such as question answering [6] and personalized recommendation [14]. However, how to evaluate the accuracy (including both triple accuracy and entity linking accuracy) efficiently and accurately has not yet received much attention and research.

1.1 Previous Approaches and Limitations

Considering the large scale of automatically constructed KGs, it is prohibitively expensive to conduct manual evaluation over the whole KG. Hence, a practical approach is to randomly sample a small set of triples and check them manually. Then the accuracy can be estimated based on these annotations. Instead of randomly sampling triples, two representative methods KGEval [23] and TWCS [11] have been proposed recently. KGEval estimates the correctness of as many triples as possible by exploiting dependencies among triples. Meanwhile, it suffers from expensive time cost to achieve

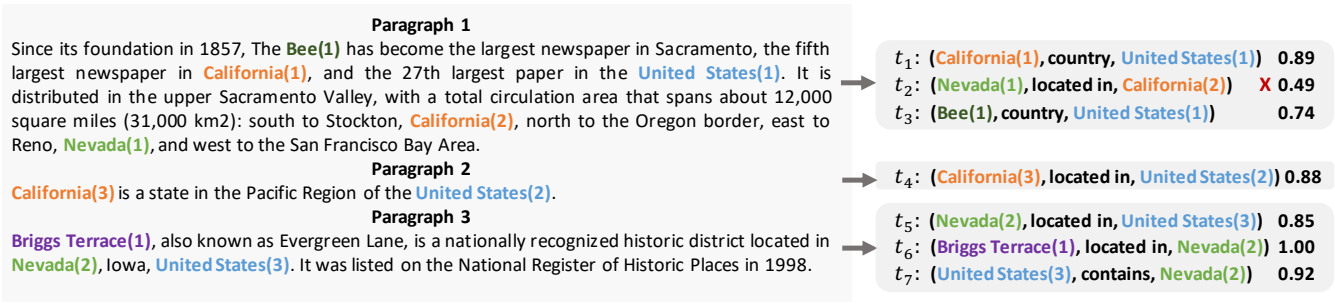


Figure 1: Example of an automatic constructed KG with its source texts. Seven triples are extracted from three paragraphs, where each triple is assigned a confidence score. Mentions with the same color are linked as an entity.

convergence of accuracy estimates. Moreover, it does not provide any statistical guarantee. TWCS applies cluster sampling with unequal probability theory that enables efficient manual evaluations [11]. Nonetheless, the potential computing power of machines is not fully exerted, thereby degrading the performance of the system.

Both KGEval and TWCS work under the assumption that the entity linking is infallible, which is impossible for the current KG construction process. Thus, both of the two methods fail to provide reliable and accurate KG quality estimation. Specifically, as KGEval makes inferences on the basis of the dependencies among triples, neglecting mistakes from entity linking will result in false estimation inevitably. Beyond that, incorrect linking of head entities may lead to false clusters in TWCS, misleading the annotators. Unreliable inference and annotation make the triple accuracy estimate deviate from the true value, resulting in an immeasurable bias.

Since most triples are automatically extracted from text, annotating triples or mentions from the same text in succession can further reduce the overall annotation cost because the user does not need to read the same text repeatedly. Furthermore, a triple is often assigned a score showing the confidence of the system. Both KGEval and TWCS neglect the associated text sources and triple confidence which can be used to reduce the annotation cost.

1.2 Motivating Example and Contributions

Example 1.1. Figure 1 presents a set of triples which are extracted from three paragraphs from DocRed [29], where all the triples but t_2 are correct. The numbers following mentions are used to distinguish entity mentions with the same string surface. In this example, we assume that the entity linking process does not make any mistakes, meaning that entity mentions with the same color refer to the same entity indeed in the real world. For each triple, the annotator reads the corresponding text to make a judgement. When we sample t_1, t_2 and t_4, there are several possible orders to annotate them. If these three triples are annotated in an order of t_1, t_2 and t_4, the user can conclude the correctness of t_2 quickly after annotating t_1 because the source text of t_1 and t_2 is the same one (i.e., the paragraph 1), and the user just reads this paragraph when annotating t_1. In contrast, if the user annotates them in an order of t_1, t_4 and t_2, the user may be less familiar with paragraph 1 when annotating t_2, which desires more time to re-read this paragraph and make a judgement. Therefore, considering the associated source texts of triples in annotation order contribute to annotation cost reduction.

In this example, the triple accuracy is 85.7%. Notice that the correctness distributions differ between two confidence ranges if we divide these triples into two groups according to triple confidence. Triples with confidence ranging from 0.4 to 0.7 are considered as group 1, and those from 0.7 to 1.0 form group 2. Assume we sample three triples t_1, t_2 and t_4 with manual annotations true, false and true, respectively. The estimated accuracy is $\frac{1}{3} \times (1+0+1) = 66.7\%$ by simply computing the mean of the labels. If triples in group 1 and group 2 are assigned distinct weights $\frac{1}{7}$ and $\frac{6}{7}$, respectively, we can compute an estimate as a weighted mean of triple labels as $\frac{1}{7} \times 0 + \frac{6}{7} \times 1 = 85.7\%$. Furthermore, the sample variance of t_1, t_2, and t_3 is 0. As a result, taking confidence into consideration helps in improving the quality estimation.

For the task of evaluating KG accuracy, *an ideal system minimizes the total cost of sampling and human annotation to produce accuracy estimates with statistical guarantees.* The major strength of the computer is its extraordinary computing power, while humans perform better than machines on tasks like the semantic interpretation of natural language and knowledge reasoning. It is desired to take full advantage of both strengths of computers and humans rather than just optimizing one of them. Therefore, we propose an interactive framework that interleaves triple sampling and human annotation. In other words, the machine performs sampling and pre-computation during human annotation.

Specifically, we build inference graphs (IGs) by integrating triples, entity linking results (i.e., whether two mentions refer to the same entity), and dependency rules together. To estimate the triple accuracy and entity linking accuracy, we just need to sample a small set of IGs rather than annotating all triples and entity linkings in the KG. Considering the benefit of annotating triples and mentions from the same source text in succession, we define a general annotation cost function. We formalize an optimization problem, namely finding the optimal order of annotating triples (and linkings) to minimize the cost of annotations, which is NP-hard as proved in the paper. Thus we exploit a Monte Carlo Tree Search (MCTS) to determine the "best" triples (or linking) for the annotator each time. Instead of being idle during human verification, the computer can pre-compute and infer triples and linkings to be annotated in the next round by considering possible feedback from annotators. The computer can immediately deliver the triple or linking which needs to be further annotated once it receives feedback from annotators. It is clear that the overall time cost will be reduced because of the overlap of machine computation time and human annotation time. The system can produce triple accuracy and linking accuracy estimates with statistical guarantees after each round of annotation.

To summarize, we make the following contributions in the paper.

- To the best of our knowledge, we are the first to evaluate KG quality comprehensively by estimating both triple accuracy and entity linking accuracy;
- By optimizing the human-machine collaborative mechanism, we propose an efficient interactive method to reduce the overall cost for evaluating the KG accuracy, which produces accuracy estimates with statistical guarantees;
- We integrate triples, entity linking results, and dependency rules to facilitate triple and linking sampling, which can reduce the burden of annotators;
- We formalize an optimization problem of order selection for triple and linking annotations and prove its NP-hardness. MCTS is applied to find an approximate solution and guide the annotation process;
- Extensive experiments have been conducted over both real and synthetic datasets. The empirical results confirm that the proposed method outperforms the state-of-the-art approaches significantly by taking less annotation cost while yielding higher accuracy estimate quality.

2 PROBLEM DEFINITION AND FRAMEWORK

We formulate the problem and present the framework of our approach. Table 5 (Appendix A) lists the frequently used notations.

2.1 Problem Formulation

As discussed above, automatic KG construction involves two tasks, i.e., information extraction (IE) and entity linking (EL). The result of IE is represented as a quadruple (T, S, Φ, ρ), in which T is a set of subject-predicate-object (SPO) triples $T = \{t_1, t_2, ..., t_n\}$, S is a set of source text $S = \{s_1, s_2, ..., s_m\}$, and Φ and ρ map a triple $t_i \in T$ to its source text $s_j \in S$ and a confidence score in $[0, 1]$, respectively. Each triple t_i is in the form of (e_{i1}, r_i, e_{i2}), where e_{i1} and e_{i2} are entity mentions (specially, e_{i2} can also be a literal), and r_i is the relation between the two mentions. Since several triples may be extracted from the same text, we have $m \leq n$ in general.

All head and tail entity mentions from T compose the set $\mathcal{E}$. Entity linking is a process that given any two entity mentions $e_i, e_j \in \mathcal{E}, i \neq j$, determining whether these two mentions refer to the same entity. Formally, entity linking is defined as a mapping $\mathcal{L} : \mathcal{E} \times \mathcal{E} \rightarrow \{0, 1\}$, where 0 means two entity mentions refer to different entities and 1 represents that two mentions are linked.

Definition 2.1. (Mention Cluster). A mention cluster is a set of mentions that refer to the same entity.

For each pair of mentions from $\mathcal{E}$, we conduct entity linking and obtain the result $\mathscr{E}_s = \{\{e_{11}, e_{12}, ..., e_{1n_1}\}, ..., \{e_{k1}, e_{k2}, ..., e_{kn_k}\}\}$. Each element $\{e_{i1}, e_{i2}, ..., e_{in_i}\}$ in $\mathscr{E}_s$ is a mention cluster. For ease of presentation, let $\mathscr{E}$ represent all the real-world entities that mentions in $\mathcal{E}$ may refer to. A mapping from entity mentions to real-world entities is defined as $L : \mathcal{E} \rightarrow \mathscr{E}$. Let T_L denote the set of triples after performing entity linking, where $t_{Li} = (L(e_{i1}), r_i, L(e_{i2})) \in T_L$. The knowledge graph we finally obtain is $G = (T_L, S, \Phi_L, \rho_L)$. If entity linking is not required in the construction, we have $T_L = T$, leading to $G = (T, S, \Phi, \rho)$.

Let $\mu_1(G)$ and $\mu_2(G)$ denote the real triple accuracy and entity linking accuracy of G, respectively. Formally, $\mu_1(G) = \frac{\tau}{M}$ and $\mu_2(G) = \frac{\gamma}{H}$, where M and H represent the total number of triples and linked entities, respectively, τ and γ represent the number of correct triples and linked entities, respectively. Accuracy estimate is calculated by a set of sampled triples and their correctness. Let us denote a sample of G as $G' = \mathcal{S}(G) = (T'_L, S', \Phi'_L, \rho'_L)$ with $T'_L \subseteq T_L$ and $S' \subseteq S$. $\mathcal{S}$ is a method designed to select triples to annotate. As associated to G', we have a set of entity mentions $\mathcal{E}'$ and a set of linked entities $\mathscr{E}'_s$.

Cost Function. The overall cost is comprised of machine time and human cost (i.e. annotation time), which have not been well investigated in prior research. Since machines and humans have complementary advantages as we discussed in Section 1, the human-machine collaborative mechanism enables an annotator and a computer to work together, and thus reduces the overall time cost. As a result, in view of possible overlap between machine time and annotation time, the overall cost function is defined as

$$Cost(G') = Cost_m(G') + Cost_h(G') - Cost_o(G'), \tag{1}$$

where $Cost_m$, $Cost_h$, and $Cost_o$ represent the machine cost, human annotation cost, and overlap part between the first two, respectively. $Cost_m(G')$ is composed of the pre-computation cost and inference cost, which will be introduced in detail in Section 5. We define a general annotation cost function considering entity linking verification and the benefit of conducting annotations based on the associated source text in Equation (6).

The paper aims to produce $\hat{\mu}_1(\mathcal{S}(G))$ and $\hat{\mu}_2(\mathcal{S}(G))$ (abbreviated as $\hat{\mu}_1$ and $\hat{\mu}_2$ for simplicity), estimates of $\mu_1(G)$ and $\mu_2(G)$ according to correctness of the sampled triples T'_L and linked entities $\mathscr{E}'_s$. The half width of a confidence interval, also called Margin of Error (MoE), is used to represent the precision of estimate.

Problem Statement 1. *Given a knowledge graph G, the goal is formulated as:*

$$\begin{aligned} \min_{\mathcal{S}} \quad & Cost(\mathcal{S}(G)) \\ s.t. \quad & \mathbb{E}[\hat{\mu}_1] = \mu_1(G), \;\; MoE(\hat{\mu}_1, \alpha) \leq \epsilon_1, \\ & \mathbb{E}[\hat{\mu}_2] = \mu_2(G), \;\; MoE(\hat{\mu}_2, \alpha) \leq \epsilon_2. \end{aligned}$$

where ϵ_1 and ϵ_2 are thresholds of MoE at confidence level 1-α. The parameters ϵ_1, ϵ_2, and α can be specified by the user on demand.

2.2 Framework of Our Approach

As shown in Figure 2, we develop a systematic framework for KG accuracy estimation. It consists of four components as follows.

- **IG Constructor** combines triples, linked entities, and dependencies together to build Inference Graphs (IGs);
- **Sampler** draws a subset of IGs by utilizing the sampling method $\mathcal{S}$;
- **Annotation Helper** assists the annotator by pre-computing proper triples and mentions to annotate and performing inference after receiving feedback from the annotator;
- **Collector** collects all the annotated IGs, calculates two estimates with their confidence intervals (CIs) using the annotated IGs, and determines when to stop the process.

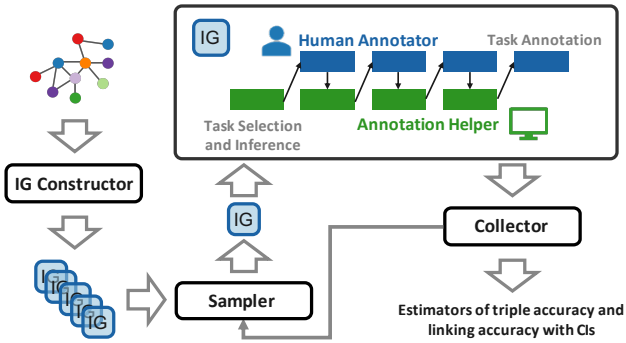


Figure 2: Accuracy evaluation framework. On the right side, rectangles colored with blue and green represent text annotation sequence and machine pre-computation respectively.

Before the whole accuracy estimation process, we provide IG Constructor with T_L, the set of triples after performing entity linking. Then a set of IGs are generated to facilitate the annotation. The Sampler draws a subset of IGs and delivers them to the annotator. Receiving an IG, the Annotation Helper exploits MCTS to select the first triple or mention and provides it together with source text to the annotator. The annotator reads the source text and checks the correctness of the triples or linked mentions. Instead of being idle, the Annotator Helper pre-computes the next triple or mention to annotate by considering possible feedback from annotators. As soon as the annotator accomplishes the annotation, a new annotation task (i.e., a triple or a mention) to annotate will be offered to the annotator immediately. When an IG has been annotated, the Collector uses the annotated IG to calculate the estimates and their MoEs. The whole process terminates if the MoEs meet the user's demand. Otherwise, the component Sampler draws a new IG again and the steps above are repeated.

3 INFERENCE GRAPH CONSTRUCTION

Before entering the sampling-annotation loop, we employ inference graphs (IGs) to facilitate human-machine collaboration later.

3.1 Triple Inference Graph

To reduce the number of triples to annotate, an effective approach is to automatically infer correctness from annotated triples by employing the dependencies (also called rules) among triples' correctness. A rule used in the paper can be defined using a Horn Clause, whose premises and conclusions are all triples. For example, when a A is the spouse of B, B must be the spouse of A, represented with Horn Clause as $(A, \textit{spouse of}, B) \rightarrow (B, \textit{spouse of}, A)$. It is clear that such a rule can facilitate the correctness verification of triples. There have been some researches about automatic construction of rules, such as PRA [17] and AMIE [10]. To guarantee the validity of inference, the automatically extracted rules need to be refined by humans.

We design the triple inference graph, a heterogeneous directed graph combining scattered triples together with rules. Notice that a rule can be applied to several sets of triples, so multiple rule instances are built to distinguish each other.

Definition 3.1. (Triple Inference Graph, shorted as TIG). A triple inference graph $G_I = (V_I, E_I)$ is a directed graph, where $V_I = V_{I_1} \cup V_{I_2}$ and E_I are the node and edge sets, respectively. V_{I_1} is a set of nodes corresponding triples (called triple nodes) and V_{I_2} is a set of rule nodes representing rule instances. For one rule node v, its premise triple nodes have edges pointing to v, and v points to its conclusion node. Each rule node v takes v's in-neighbors as its premises, and takes v's out-neighbors as its conclusion.

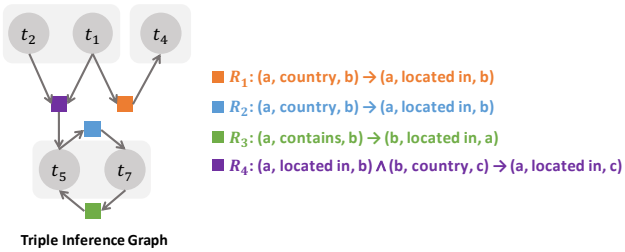


Figure 3: An example of TIG for the knowledge graph in Figure 1, where triples $t_1, \cdots, t_7$ are defined in Figure 1. Circles represent triple nodes, and squares represent rule nodes. For simplicity, only a subset of rule instances are presented.

A TIG has two characteristics: (1) edges only exist between triple nodes and rule nodes; (2) the out-degree of each rule node is one, which means that one rule only has one conclusion. This constraint does not limit the variety of rules because rules with multiple conclusions can be split into several single-conclusion rules.

Example 3.2. A TIG for triples in Figure 1 is presented in Figure 3, where 4 dependency rules (R_1, R_2, R_3, and R_4) are defined among triples. If an annotator labels triple t_1 as "true", the correctness of triple t_4 is easily inferred as "true" based on rule R_1.

3.2 Inference Graph

One entity in a KG is often linked by several entity mentions in corpus. Take the rule $(A, \textit{spouse of}, B) \rightarrow (B, \textit{spouse of}, A)$ as an example, we assume that the head entity A is linked by mentions A_1 and A_2, and the tail entity B is linked by mentions B_1 and B_2. If A_1 and A_2 are linked incorrectly, i.e., they refer to different entities, this inference rule will not work anymore. As a result, this rule works under the premise that $(A_1 = A_2) \wedge (B_1 = B_2) \wedge (A, \textit{spouse of}, B) \rightarrow (B, \textit{spouse of}, A)$, where $(A_1 = A_2)$ and $(B_1 = B_2)$ mean that mentions A_1 and A_2, B_1 and B_2 refer to the same entity, respectively.

We extend TIGs by connecting entity linking results with rule nodes as premises for inference. Besides, some new rules can be also derived from existing rules to strengthen the inference ability of TIG. Specifically, when a set of triples whose correctness contradicts a rule, there must exist some incorrectly linked entities.

Not only the correctness of triple-triple and triple-entity linking can infer each other, but also entity linkings can use inference techniques. Thus, we build an entity linking graph for each group of entity mentions which are linked together to assist inference.

Definition 3.3. (Entity Linking Graph). Given a set of entity mentions $V_e = \{e_i\}$, an entity linking graph, shorted as ELG, is an undirected complete graph $g_e = (V_e, E_e)$, where each two nodes in V_e has an edge in E_e, meaning that two entity mentions are linked.

ELG is designed to infer linkings among any three mentions. The straightforward rule is that when two pairs of mentions are checked and at least one of them is truly linked, the correctness of

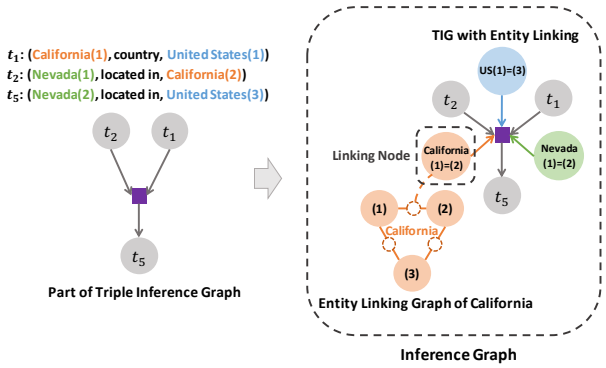


Figure 4: An example of IG. For simplicity, only a part of TIG in Figure 3 is shown in this example.

the remaining mention pair is determined. For example, in the ELG part of Figure 4, if the user finds that the mention *California(1)* and California(2) refer to one state of the United States, and *California(1)*, *California(3)* are also linked truly, it can be inferred that the linking between *California(2)* and *California(3)* is also correct.

We add an auxiliary node (the dotted circles as shown in Figure 4) on each edge of an ELG, whose label represents the correctness of EL between the two end nodes. The auxiliary nodes are used to make connections with the TIG, leading to the inference graph.

Definition 3.4. (Inference Graph). Given a TIG $G_I = (V_I, E_I)$, an inference graph G_C, shorted by IG, is defined as the graph obtained by connecting each rule node $u \in V_I$ with auxiliary nodes of the ELGs $\{g_e\}$ through newly added nodes, also called linking nodes, where $\{g_e\}$ is the set of ELGs built for all the entities in the triples adjacent to u.

Example 3.5. The right part of Figure 4 is an example of IG which is built based on Figure 3. We just present the result of adding linking nodes as the premise for rule node R_4 and one entity linking graph for the entity *California* for simplification.

After obtaining an IG, all the correctness of triple nodes and auxiliary nodes are waiting for annotation. The annotation process is assisted by the proposed annotation helper to reduce human annotation cost. Details of annotation are shown in Section 5.

4 SELF-ADJUSTED STRATIFIED SAMPLING

Stratified sampling [4] is often used to reduce the variance of estimates. The key idea of this sample approach is splitting the population into several homogeneous subpopulations (called stratum). Motivated by the example in Section 1, confidence scores of triples can be used to guide the stratification because triples with similar confidences are more likely to have a similar probability of being correct. To generalize the concept of confidence to IG, we compute the median of all triples' confidences in an IG as its confidence.

In each stratum we use an estimate from weighted sampling $\hat{\mu}_{wi,h}$. The weight of the h-th stratum is W_h. The unbiased estimate of triple accuracy μ_1 with $1-\alpha$ CI is defined as follows:

$$\hat{\mu}_{s1} = \Sigma_h W_h \hat{\mu}_{w1,h} \tag{2}$$

$$\hat{\mu}_{s1} \pm z_{\alpha/2}\sqrt{\frac{1}{n(n-1)}\Sigma_h W_h^2 Var(\hat{\mu}_{w1,h})} \tag{3}$$

The unbiased estimate of entity linking accuracy μ_2 with $1-\alpha$ CI is defined as follows:

$$\hat{\mu}_{s2} = \Sigma_h W_h \hat{\mu}_{w2,h} \tag{4}$$

$$\hat{\mu}_{s2} \pm z_{\alpha/2}\sqrt{\frac{1}{n(n-1)}\Sigma_h W_h^2 Var(\hat{\mu}_{w2,h})} \tag{5}$$

Generally, it is difficult for the user to determine how to split the confidence range [0,1] into several sub-intervals. Thus, we propose to adjust stratum automatically as follows.

(1) Initialization: Confidence range is divided into k equal sub-intervals, where k is a user-specified parameter.
(2) Combination Simulation: After some TIGs have been annotated and the MoE of the estimate is recomputed, we try to combine neighborhood strata as a new stratification and calculate the new MoE under these new strata. If the new MoE is smaller than the previous MoE, we believe the new stratification is better than the unadjusted one.
(3) Adjustment: If there exist combinations that can result in better stratification, we choose the one which minimizes MoE to adjust stratification.
(4) Stable State: If an adjustment cannot be made for more than h times or the number of strata is less than t, the stable state of stratification is achieved and no more adjustment will be made later. Both h and t can be determined by the user.

5 OPTIMIZING INTERACTIVE ANNOTATION

5.1 Annotation Cost

For one entity (i.e., a set of entity mentions referring to the same entity), instead of checking each mention pair one by one, we ask the annotator to identify each mention individually. When identifying a mention e_i from the mention clusters F, the user is asked to review mentions in F which have been identified. If e_i refers to the same entity with one of them called e_j, the user just needs to classify e_i and e_j as the same class. This design reduces the time for recording the identification result of e_i. Therefore, annotating a set of mentions (i.e. checking if any pairs of them refer to the same entity) is divided into two situations:

- Entity classification: if current mention refers to the same entity with any mention which has been identified, classify them as the same class;
- Entity creation: if current mention refers to a new entity, record current identification result.

As discussed in Section 1, annotating triples or mentions from the same source text in succession can reduce time cost. We define the annotation cost next:

$$Cost_h(G') = c_1 \cdot v + (c_2 \cdot \alpha + c_3 \cdot \beta) - \gamma \cdot s, \tag{6}$$

where v is the number of triples checked by the user, and α and β are the number of entity classification and entity creation, respectively. s is the number of situations that the adjacent annotation tasks share the same source text. c_1, c_2, c_3, and γ are set by the user.

5.2 Annotation Order Problem

Annotating an IG is a process that a human labels correctness of triples and identifies entity mentions according to the source texts

with the help of Annotation Helper. As one IG may contain several triples and mentions, the order in which the triples and mentions are annotated may affect the annotation cost.

Example 5.1. As shown in Figure 3, if t_1 is annotated as "true" before t_4, the correctness of t_4 can be inferred as "true" using R_1 without extra human efforts. As a result, it is required to select the optimal order for triple annotation.

To minimize the human annotation cost in Equation (6), we can define the annotation order problem next.

Definition 5.2. (Annotation Order Problem, shorted by AOP). Given an inference graph G_C, the goal of AOP is to find the optimal order O^* such that

$$O^* = \arg\min_{O} \ Cost_h(O(V_e \cup V_{I_1})),$$

where V_{I_1} is a set of triples waiting for verification, V_e is the set of entity mentions needed to be identified by the user, and O is an order of annotation.

Theorem 5.3. *The Annotation Order Problem is NP-Hard.*

Proof. The proof can be achieved by reducing from the NP-hard Information Maximization problem [15]. For more details, please refer to Appendix B. □

Because the annotation process is indeterminate (for example, one triple can be true or false), we use expectation of $Cost_h(O(V_e \cup V_{I_1}))$ as the measure of human efforts. The most simple idea is to enumerate all possible orders and calculate their expectations, however, the time cost is prohibitively expensive. To utilize the computation power of machines, we use a simulation-based method to obtain the approximate solution.

5.3 Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a simulation-based method to find a better decision [3], becoming more and more widely used after its success in AlphaGo [9]. It is employed to find an optimal policy (i.e., a function maps each state to action) of large state-space Markov Decision Process (MDP). The MDP models sequential decision problems in fully observable environments with state space S, action space A, transition probability $P_a(s, s')$, reward $R_a(s, s')$, and a terminal state.

We can take the IG annotation as an MDP and greedily choose the near-optimal text to verify dynamically based on MCTS. Specifically, our IG annotation process is also a sequential decision problem, by considering IG's annotation condition as states and unannotated triples and unidentified mentions as actions. Choosing one triple to verify or one entity mention to identify is the same as choosing an action. Annotating one triple as true or false or linking one mention to one entity makes the state of MDP change. The terminal state is an annotation condition that all the triples in IG (i.e., V_I) and all entity linkings i.e., $\{V_a\}$) have been annotated. $P_a(s, s')$ is the probability of transferring from state s to s' when taking action a. When the action is a triple, the translation probability is estimated by the triple accuracy estimate $\bar{\mu}_1$ and $1 - \bar{\mu}_1$. When the action is an entity mention, the probability of entity classification and creation can be estimated by frequency during the annotation process.

The reward $R_a(s, s')$ is defined according to the annotation cost Equation (6). When a is a triple, the reward is $-c_1$. When a is an entity mention, rewards are $-c_2$ and $-c_3$ for entity classification and entity creation, respectively. If the current action's source text is the same as the last action, current reward is added with β. A policy is a mapping from states to actions. Given a policy π, a utility score of a state s is defined as follows:

$$U^{\pi}(s) = E_{P([s_0, s_1, \ldots] | s_0 = s, \pi)}[\Sigma_{t=0}^{T} R(s_t)]$$

Therefore, an optimal policy is given by $\pi_s^* = U^{\pi}(s) = \pi^*$. If from state s, all actions are selected according to π^*, a utility score under the optimal policy is $U(s) = U^{\pi^*}(s)$. In this case, the optimal policy of a state s is given by

$$\pi^*(s) =_a \Sigma_{s'} P(s'|s, a) U(s')$$

It is clear that the optimal policy of s is the action with the maximal expected accumulated utilities, called *expected reward of an action*. This expectation can be estimated using simulation, which is the key idea of MCTS.

UCB is a common metric used by MCTS which balances the probability of choosing the action with a good accumulated reward and choosing the action with fewer simulations [1].

$$UCB(a_i) = \bar{R}_i + C\sqrt{\frac{2ln(n)}{n(a_i)}},$$

where $n(a_i)$ is the time of simulations beginning with a_i and n is the total number of simulations.

To choose this policy of a given state, in other words, the best text in a given annotation condition for the annotator, MCTS is performed. Monte Carlo Tree (MCT) is built with the current state as the root node. Each possible action (i.e., unverified triple or unidentified entity mention) will become one child of the root node. Given a state, the MCTS works as follows:

(1) Expansion: If there are still actions not been added into MCT, randomly choose one and the root node expands a child node for it with an initial UCB score of 0.
(2) Selection: Choose the child of the root node with the largest UCB score for simulation.
(3) Simulation: Simulate the iteration of choosing an action, annotating a triple or identifying a mention, making inferences until the terminal state is reached. During simulation, choices of the subsequent actions are random. The probability of annotating a triple as "true" equals the accuracy estimator $\hat{\mu}_1$ we have, and entity classification and entity identification are simulated with probabilities estimated from frequencies of all the annotated IGs. The reward of this action is achieved by adding rewards of states during the simulation.
(4) Update: Update the UCB score of the chosen child node using the reward of it from simulation.
(5) Termination: Repeat steps 1-4 iteratively until the termination condition, e.g., limit of elapsed time and limit of simulation times, is reached. The child node with the maximum $\bar{R}_i$ is chosen as the next triple or mention to annotate.

The optimality of MCTS is discussed in Appendix C.

5.4 Interactive Mechanism

If we want to choose the optimal annotation task in each round for the annotator, plenty of simulations are needed to guarantee the quality of MCTS, which will consume much time. To relieve such a problem, an effective interactive mechanism is desired.

Let us recall the cost function defined in Equation (1), the total cost can be reduced by reducing $Cost_o$, the overlapping of time shared by the machine and annotator. Notice that reading a text to verify a set of triples usually takes tens of seconds, which is time-consuming as well. The principle of the interactive mechanism is interleaving both human annotation and order selection. Instead of being idle, the Annotation Helper progressively performs two tasks, such that the system can immediately deliver the next task to annotate upon receiving the label returned by the annotator.

(1) It pre-computes the next annotation task by taking into consideration the possible feedback from the annotator. Suppose the user is verifying a triple, the label of the triple will be "true" or "false", leading to two possible states (i.e. the annotation condition of the IG). The annotation helper begins MCTS processes with states as root nodes and chooses the next annotation task separately.

(2) For each pre-computed annotation task, it pre-computes the inference results based on possible responses from the user.

Suppose the user is identifying an entity mention. For a mention, possible actions include entity classification and entity creation. Because if there are several mentions which have been identified, the entity classification may have many possible results. For example, there is an entity A in the IG which is linked by mentions a_1, a_2, a_3, and a_4. Assume that the first three mentions have been identified that they refer to three different entities. Therefore, possible results of mention a_4 are creating a new entity or classifying it as a_1, a_2 or a_3, 4 types in total. Because $Cost_o$ has an upper bound of $Cost_h$, we want a more instructive pre-computation result when the number of feedback is large in a limited time. In other words, for more likely feedback, we want a more precise pre-computation, for MCTS, a longer simulation. Therefore, we estimate the probabilities of all the possible feedback and use these probabilities as weights to arrange the pre-computation time, which leads to that in most cases the real feedback from the user has been simulated with enough time and the choice of the next task is instructive.

6 EXPERIMENTS

6.1 Experiment Setup

Datasets. As listed in Table 6 (see Appendix D) , three real-world datasets including NELL [23], YAGO [23], and OPIEC [12] and three synthetic datasets SYN-IGs with different sizes are used to evaluate the performance of our framework.

Cost Function. To reduce the uncertainty of measuring time for each experiment, we estimate the annotation time factors c_1, c_2 and c_3 and cost reduction factor γ to calculate the cost by the cost function 6 instead of measuring time cost directly. Parameters are estimated by measuring real human annotation time and the result is presented in Table 1. When just considering triple accuracy in *YAGO* and *NELL*, we use the same cost function as TWCS [11] (introduced in detail in Appendix D.2) to make a fair comparison with it. They divide the annotation process into entity identification and relationship validation, leading to the cost function $Cost_h(G') = 45|E'| + 25|G'|$, where $|E'|$ and $|G'|$ represent the number of entity identification and the number of relationship validation, respectively. In *OPIEC*, we use the cost function 6 for both TWCS and our method, setting c_2, c_3 as 0. When estimating triple accuracy and entity linking accuracy at the same time, we use the complete version of our cost function.

Table 1: Parameters of annotation cost function.

Triple Verification Cost c_1	5.7s
Entity Creation Cost c_2	11.0s
Entity Classification Cost c_3	10.0s
Cost Reduction γ	1.2s

Implementations. We repeat triple accuracy estimation in YAGO and NELL for 1,000 times and report the mean estimate with standard deviation. We set the minimum number of annotated triples to 20 because the accuracies of these two datasets are high. Otherwise, it is at serious risk of drawing one IG whose triples are all correct, getting an estimate of 1 and 0 MoE. Parameters of the cost function are estimated by measuring real human annotation time and the result is presented in Table 1.

Both of upper bounds of MoEs ϵ_1, ϵ_2 are set as 5%, and α is set as 5%. We implemented our method in Python3 and all experiments were performed on Ubuntu 16.04.7 using Intel Xeon E5-2678 v3 @ 2.50GHz processor with 220 GB of memory.

Evaluation Metric. The number of IGs and the number of triples drawn by the sampler are used to reflect the efficiency of sampling methods. Besides, we call the ratio of the number of annotated triples and the number of triples annotated by humans as the **triple inference rate**, measuring the inference ability of our framework. Similarly, we can also define the entity inference rate.

6.2 Evaluation on Datasets

6.2.1 Triple Evaluation. An efficiency improvement to TWCS can be seen in Table 2 using our framework. Compared with TWCS, our method reduces 46.88%, 52.35%, and 42.11% number of manual annotations in YAGO, NELL, and OPIEC, respectively. The reduction of the overall time cost is 25.00%, 25.4% and 37%. OPIEC has the greatest improvement on overall time because this dataset provides source texts and our framework can use them to reduce annotation time, i.e., $Cost_h$. Our method has a tiny $Cost_m - Cost_o$, i.e., extra time of the machine, for the need of MCTS to select the first triple to annotate, and this burden does not have a negative effect on $Cost$, while the use of MCTS boosts inference mechanism as Section 6.3.2. The effect of inference is presented by the triple inference rates 43.33%, 12.35%, and 31.19%.

6.2.2 Entity Linking Evaluation. Table 3 presents the result of estimating triple accuracy and entity linking accuracy simultaneously in OPIEC and SYN-IG-10m. It can be found that for large graphs with 100,000 and 10,000,000 triples, our method can find estimates satisfying the user's demand within 1 hour. We can find that only estimating triple accuracy of NELL in Table 2 takes more time than getting triple and entity linking accuracies of OPIEC, meaning that our framework is suitable for giving a complete accuracy evaluation of an automatically constructed KG from texts with crowdsourcing.

6.2.3 Effect of Entity Linking. In the knowledge graph construction, entity linking is impossible to be perfect, leading to wrong clusters

Table 2: Performance of triple evaluation.

	YAGO		NELL		OPIEC	
	TWCS	**Ours**	**TWCS**	**Ours**	**TWCS**	**Ours**
#Annotated triples	32 ± 5	26 ± 16	149 ± 47	75 ± 31	171	146
#Manually annotated triples	32 ± 5	**16 ± 10**	149 ± 47	**60 ± 24**	171	**99**
Triple inference rate	0	38.46%	0	20.00%	0	32.19%
$Cost_h$ (second)	1584 ± 252	1116 ± 684	6660 ± 2160	4212 ± 1692	972	540
$Cost_m - Cost_o$ (second)	< 0.1	36 ± 24	< 0.1	102 ± 45	< 0.1	64.30
Overall Cost (second)	1584 ± 252	**1152 ± 744**	6660 ± 2160	**4314 ± 1737**	972	**612**
Triple accuracy estimate	99.20%(96.70% − 100%)	99.61%(98.63% − 100%)	91.63%(89.33% − 93.93%)	95.62%(88.8% − 100%)	92.17%	94.85%

Table 3: Performance of entity linking evaluation.

	OPIEC	**SYN-IG-10m**
#Annotated triples	181	140
#Manually annotated triples	169	122
#Triple inference rate	6.63%	12.86%
#Annotated mentions	260	155
#Manually annotated mentions	238	138
#Entity inference rate	8.46%	10.97%
$Cost_h$ (second)	3420	1944
$Cost_m - Cost_o$ (second)	164	130
Cost (second)	3584	2074
Triple accuracy estimate	93.24%	58.71%
Entity linking accuracy estimate	11.12%	23.78%

Table 4: Accuracy evaluation of SYN-IG-100k.

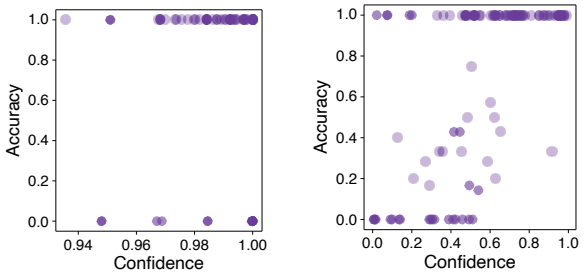
	TWCS	**Ours**
Triple accuracy estimate	39.91% ± 4.99%	**59.51% ± 4.21%**
Entity linking accuracy estimate	-	**24.25% ± 4.99%**

of triples for TWCS. Entity identification is hard when several head entity mentions are linked together falsely, and the correctness of triple verification is not guaranteed, making the triple accuracy estimate deviate from the true value. In our experiment, we simulate the process of annotating triple clusters considering the mistakes made by entity linking in our synthetic datasets. When we get a cluster, we identify the entity as the one most head entity mentions refer to according to the real entity linking results we generate artificially. After entity identification, those triples whose head entity mentions are wrongly linked are annotated as false directly.

As shown in Table 4, we use our method in SYN-IG-100k and compare the triple accuracy estimates given by TWCS and our method, where the true value is 61.49%. Our unbiased triple accuracy estimate is much closer to the true accuracy than TWCS. Ignoring entity linking leads to TWCS's underestimate of accuracy. Besides, our method also provides the user with an entity linking accuracy which is close to the true value 28.34%, making the evaluation result more complete.

6.3 Ablation Experiments

6.3.1 Sampler. To compare the efficiency of different sampling methods sufficiently, we run weighted sampling (WS) and stratified sampling (SS) 20 times in four graphs and record the number of IGs and triples in Figure 7. Because confidence score is a key standard for stratified sampling, we visualize the relation between confidence and accuracy of IGs in two datasets in Figure 5. For clarity, only 100 IGs are chosen randomly and demonstrated for each dataset.



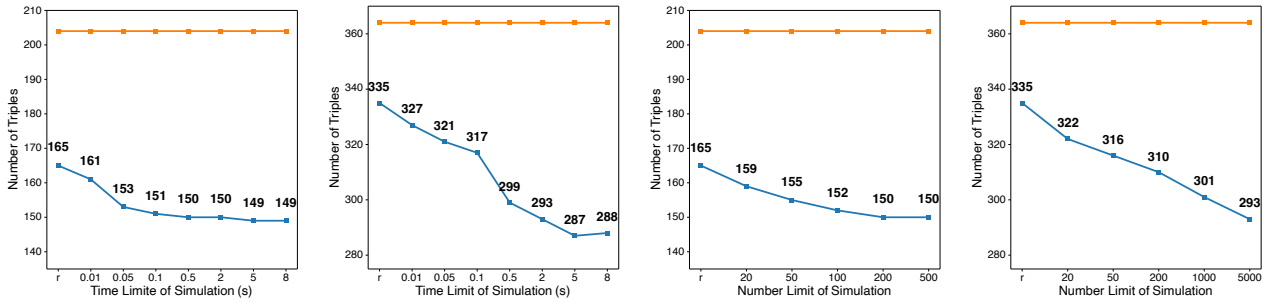
(a) NELL (b) SYN-IG-100k

Figure 5: Relation between IG confidence and accuracy.

We make a comparison between the weighted sampling (Appendix D.3) and the stratified sampling introduced in Section 4. To verify the effect of stratified sampling adequately, we run our framework with both sampling methods on a real graph NELL and SYN-IGs with three different sizes each for 20 times and record the number of IGs and triples sampled by the sampler in Figure 7. The difference between the effects of the two sampling methods is obvious. Under the premise of obtaining an estimate of the same accuracy, stratified sampling needs much fewer triples, which proves that the confidence score of a triple is related to its correctness as we mentioned above, and it can be used to guide the stratification.

Based on sampling methods, our framework has the ability to deal with large datasets without unbearable human effort and time cost. We can find in Figure 7 that the number of triples and IGs, i.e., the efficiency, of our sampler (either using WS or SS) barely changes when the graph size changes from 10k to 10m.

6.3.2 Helper. Inference ability is a key feature of our framework which can reduce human annotation cost significantly, and MCTS we use is an important technique to exert machine's computing power when human is annotating triples or mentions and select a good order of them for humans to make the most use of inference. Our helper use inference and MCTS to assist the annotation process of an IG. To verify the effect of helper, we select IGs constructed from YAGO whose size (i.e. the number of triples) is 3. Besides, we generate a set of IGs whose sizes are 5, 6, 7 (ignoring entity linking) to verify that our framework is also suitable for larger IGs. The



(a) YAGO: Time Limit (b) SYN-IG: Time Limit (c) YAGO: Number Limit (d) SYN-IG: Number Limit

Figure 6: The number of triples annotated by human in the IG set from YAGO and synthetic IG set under different setting of choosing the next triple. The orange line shows the total number of triples waiting for annotation.

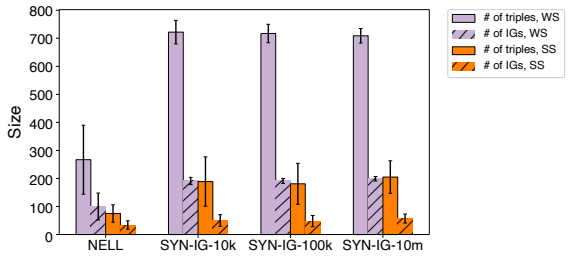


Figure 7: Weighted sampling vs. stratified sampling.

number of triples annotated by humans of IGs from YAGO and synthetic IGs is reported in Figure 6.

We compare MCTS with the simplest baseline: choose the next annotation task randomly. At the same time, we explore how the end condition (i.e. the limit of simulation time and simulation number) of MCTS influences the inference ability. In each condition, we run the annotation process 20 times and record the mean number of triples annotated by humans. In Figure 6, we find that in spite of the specific method of selecting the next triple, inference leads to at least 19.12% and 8.00% reduction of triples annotation cost. The MCTS can lead to cost reduction even with just 0.01 seconds of simulation. By increasing the limit of time and number of simulations in MCTS, the helper tends to find better triples to reduce triples to annotate.

7 RELATED WORK

Several automatic methods are designed to evaluate KGs in different quality dimensions [5], using both of semantic and structural information [28] and external information like other knowledge graphs [19]. To relieve the quality issue, refinement methods focusing on correctness and completion of KG have been proposed [26]. Some researchers use techniques like entity linking, common sense reasoning, and linguistic analysis to find missing facts in KGs [24], and use the multi-task model to remove inaccurate relations [25]. How KG quality influences other tasks, e.g., KG embedding, is also investigated [27]. In addition to automatic evaluation methods, two recent approaches (KGEval [23] and TWCS [11]) import crowdsourcing into accuracy estimation of KG. Both the two methods focus on designing effective strategies to select triples for annotators. KGEval [23], points out that the labels (i.e. the correctness of triples) in a KG can propagate. Dependencies like type consistency and Horn-clause constraints between triples and their labels make it possible to use Probabilistic Soft Logic [2] to achieve new labels from labeled triples. However, it is time-consuming for large graphs and does not have a statistical guarantee.

TWCS [11] studied several sampling methods, especially Two Stage Weighted Cluster Sampling (TWCS). The authors divide the time cost of annotating one triple into the time of entity identification and relationship validation. They find that providing triples about the same head entity (i.e. a cluster) to the annotator can save the entity identification cost and design an unbiased estimator. However, TWCS just focuses on optimizing the annotation cost of a triple set, without optimizing the computing power of machines.

Both of these two methods above ignore the mistakes introduced in the entity linking process. Lack of entity linking accuracy not only leads to incompletion of KG quality evaluation but also influences the triple accuracy estimation negatively. In contrast, we aim to evaluate both triple and linking accuracy and reduce the overall cost by optimizing human-machine collaboration in this paper.

8 CONCLUSION

In this paper, we propose an interactive knowledge graph accuracy evaluation framework considering IE and EL at the same time. To estimate the triple accuracy and entity linking accuracy with statistical guarantee efficiently, we introduce a human-machine collaborative mechanism that utilizes the strong computing power of computers and the correctness verification skills of humans. We formulate an order selection problem which is NP-hard. Techniques including inference, stratified sampling, and MCTS are proposed to assist the human annotation process. Experiments in both real and synthetic knowledge graphs confirm the efficiency, effectiveness, and scalability of our method.

ACKNOWLEDGMENTS

This work was supported by the Scientific and Technological Innovation 2030 - New Generation of Artificial Intelligence Major Project (Grant No. 2020AAA0108505), National Natural Science Foundation of China (Grant No. 61902074), and Science and Technology Committee Shanghai Municipality (Grant No. 19ZR1404900).

REFERENCES

[1] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2 (2002), 235–256.

[2] Matthias Brocheler, Lilyana Mihalkova, and Lise Getoor. 2012. Probabilistic similarity logic. *arXiv preprint arXiv:1203.3469* (2012).

[3] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4, 1 (2012), 1–43.

[4] George Casella and Roger L Berger. 2002. *Statistical inference*. Vol. 2. Duxbury Pacific Grove, CA.

[5] Haihua Chen, Gaohui Cao, Jiangping Chen, and Junhua Ding. 2019. A Practical Framework for Evaluating the Quality of Knowledge Graph. In *China Conference on Knowledge Graph and Semantic Computing*. Springer, 111–122.

[6] Jose Ortiz Costa and Anagha Kulkarni. 2018. Leveraging Knowledge Graph for Open-domain Question Answering. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, 389–394.

[7] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *SIGKDD*. 601–610.

[8] MS Fabian, Kasneci Gjergji, WEIKUM Gerhard, et al. 2007. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In *WWW*. 697–706.

[9] Michael C Fu. 2016. AlphaGo and Monte Carlo tree search: the simulation optimization perspective. In *Winter Simulation Conference (WSC)*. 659–670.

[10] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*. 413–422.

[11] Junyang Gao, Xian Li, Yifan Ethan Xu, Bunyamin Sisman, Xin Luna Dong, and Jun Yang. 2019. Efficient Knowledge Graph Accuracy Evaluation. *Proc. VLDB Endow.* 12, 11 (2019), 1679–1691.

[12] Kiril Gashteovski, Sebastian Wanner, Sven Hertling, Samuel Broscheit, and Rainer Gemulla. 2019. OPIEC: An Open Information Extraction Corpus. In *AKBC*.

[13] Morris H Hansen and William N Hurwitz. 1943. On the theory of sampling from finite populations. *The Annals of Mathematical Statistics* 14, 4 (1943), 333–362.

[14] Steven Haussmann, Oshani Seneviratne, Yu Chen, Yarden Ne'eman, James Codella, Ching-Hua Chen, Deborah L McGuinness, and Mohammed J Zaki. 2019. FoodKG: a semantics-driven knowledge graph for food recommendation. In *International Semantic Web Conference*. Springer, 146–162.

[15] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *SIGKDD*. 137–146.

[16] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*. Springer, 282–293.

[17] Ni Lao, Tom Mitchell, and William Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *EMNLP*. 529–539.

[18] Ying Li, Vitalii Zakhozhyi, Daniel Zhu, and Luis J Salazar. 2020. Domain Specific Knowledge Graphs as a Service to the Public: Powering Social-Impact Funding in the US. In *SIGKDD*. 2793–2801.

[19] Shuangyan Liu, Mathieu d'Aquin, and Enrico Motta. 2017. Measuring accuracy of triples in knowledge graphs. In *International Conference on Language, Data and Knowledge*. Springer, 343–357.

[20] Xusheng Luo, Le Bo, Jinhang Wu, Lin Li, Zhiy Luo, Yonghua Yang, and Keping Yang. 2021. AliCoCo2: Commonsense Knowledge Extraction, Representation and Application in E-commerce. In *SIGKDD*. 3385–3393.

[21] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhanava Dalvi, Matt Gardner, Bryan Kisiel, et al. 2018. Never-ending learning. *Commun. ACM* 61, 5 (2018), 103–115.

[22] Feng Niu, Che Zhang, Christopher Ré, and Jude W Shavlik. 2012. DeepDive: Web-scale Knowledge-base Construction using Statistical Learning and Inference. *VLDS* 12 (2012), 25–28.

[23] Prakhar Ojha and Partha Talukdar. 2017. KGEval: Accuracy estimation of automatically constructed knowledge graphs. In *EMNLP*. 1741–1750.

[24] Ankur Padia et al. 2017. Cleaning noisy knowledge graphs. In *Proceedings of the Doctoral Consortium at the 16th International Semantic Web Conference*, Vol. 1962.

[25] Ankur Padia, Frank Ferraro, and Tim Finin. 2018. KGCleaner: Identifying and correcting errors produced by information extraction systems. *arXiv preprint arXiv:1808.04816* (2018).

[26] Heiko Paulheim. 2017. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web* 8, 3 (2017), 489–508.

[27] Jay Pujara, Eriq Augustine, and Lise Getoor. 2017. Sparsity and noise: Where knowledge graph embeddings fall short. In *EMNLP*. 1751–1756.

[28] Jan Rörden, Artem Revenko, Bernhard Haslhofer, and Andreas Blumauer. 2017. Network-based Knowledge Graph Assessment.. In *SEMANTICS Posters&Demos*.

[29] Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A Large-Scale Document-Level Relation Extraction Dataset. In *ACL*. 764–777.

A NOTATIONS

Table 5: Frequently used notations.

Notations	Descriptions
$\mathcal{E}$	The set of all entity mentions
$\mathscr{E}_s$	A set of mention clusters, i.e., sets of linked entities
N	Total number of IGs
M_i	Number of triples in the i-th IG, $i = 1, 2, ..., N$
$M = \Sigma_i^N M_i$	Number of triples in T_L
H_i	Number of entity linking pairs in the i-th IG, $i = 1, 2, ..., N$
$H = \Sigma_i^N H_i$	Total number of entity linking pairs
τ_i	Number of correct triples in the i-th IG
$\tau = \Sigma_i^N \tau_i$	Total number of correct triples
γ_i	Number of correct entity linking pairs in the i-th IG
$\gamma = \Sigma_i^N \gamma_i$	Total number of correct entity linking pairs
μ_{1i}	Triple accuracy of the i-th IG
μ_{2i}	Entity linking accuracy of the i-th IG
$M_{[h]}$	Number of triples in the h-th stratum
W_h	Weight of the h-th stratum
n	Total number of sampled IGs

B PROOF OF THEOREM 5.3

Proof. Consider an instance M of the NP-hard Information Maximization problem, defined by a linear threshold model with fixed thresholds $\theta_v = 1, \forall v$ and an integer k, each currently inactive node v becomes active if and only if the total weight of its active neighbors is at least θ_v[15]:

$$\sum_{w \to v, w \text{ active}} b_{v,w} \geq \theta_v,$$

where $b_{v,w} \in [0, 1]$ is the weight assigned to each incoming neighbor w and $\sum_{w \to v} b_{v,w} = 1$; we want to know the maximum number of nodes that can be totally activated by k initially active nodes. We show that this can be viewed as a special case of our *AOP* problem.

Assume that there is an IG without entity linking and all triples have different source texts. In this case, the annotation cost function $Cost_h$ is simplified to be in direct proportion to the number of triples annotated by humans. Since rule nodes are employed to produce the label of its successor node, and it doesn't hold any information about the label, neither affect the cost. Thus we can safely simplify the graph to obtain a graph G' by removing them and reconnecting its parent nodes with its successor node. Therefore, our target is selecting a set of annotating triples in G' such that the correctness of all triples can be checked thereafter.

Recall that we set threshold $\theta_v = 1, \forall v$ in M and $\sum_{w \to v} b_{v,w} = 1$, in which case a node v can be activated if and only if each of v's incoming neighbor w is active. In this sense, we say that a node is active when it is annotated initially and otherwise inactive, then a label of a node can be inferred when all of its incoming nodes are annotated (active). Then when we find an optimal solution S^* of our *AOP* problem, S^* is also the optima of M, that is, the solution of our *AOP* problem can solve the *IM* problem. However, obviously the *IM* problem is NP-hard, thus we can conclude that our *AOP*

Table 6: Dataset features.

	NELL	YAGO	OPIEC	SYN-IG-10k	SYN-IG-100k	SYN-IG-10m
#Triples	1,860	1,386	100,000	10,000	100,000	10,000,000
#Rules	119	27	169	28	28	28
Triple Accuracy	91.34%	99.21%	-	61.02%	61.49%	61.64%
#Mentions	-	-	193,235	12,953	129,351	12,892,741
#Linking Pairs	-	-	102,806	7,130	71,784	7,092,669
Entity Linking Accuracy	-	-	-	25.79%	28.34%	27.91%

problem is at least as hard as the *IM* problem, which proves the NP-hardness. □

C OPTIMALITY OF MCTS

Assume the reward of an action is R. As introduced before, the optimal action is the one with the maximum $\mathbb{E}(R)$. According to Chernoff Bound, the gap between $\bar{R}$ and real expectation $\mathbb{E}(R)$ for one action has an upper bound [16]:

$$P(|E(R) - \frac{1}{w}\Sigma_i R_i|) \geq \epsilon) \leq exp(-(\frac{\epsilon}{Z})^2 w),$$

where w is the simulation time of the action and Z is the maximum of $|R|$. This equation indicates the statement that if $w \geq (\frac{Z}{\epsilon})^2 ln\frac{1}{\delta}$, we have at least $1-\delta$ probability of having $|E(R) - \frac{1}{w}\Sigma_i r_i| \leq \epsilon$, meaning the action chosen by MCTS is the optimal action.

D EXPERIMENTAL SETTINGS

D.1 Dataset

NELL and **YAGO** are sampled from NELL-Sport [21] and YAGO2 [8] with human annotation collected by [23]. NELL is sport-related including relations like *athlete*, *league*, *stadium*, etc. YAGO contains relations such as *directed*, *actedIn* and *isMarriedTo*, etc. We also collect rules used for inference from [23], which are mined by PRA [17] (for NELL) and AMIE [10] (for YAGO). Some manually designed rules are added to improve the inference ability.

We choose 100,000 triples from an open information extraction dataset [12] with source text and confidence score, and unite some relations together to build **OPIEC**. OPIEC includes domestic relationships such as *be mother of* and *be father of*. A simple entity linking method that entity mentions with the same string are linked together is used. 169 rules are manually designed to construct IGs.

To verify the scalability of our framework, we generate three synthetic graphs **SYN-IG**s containing 10,000, 100,000, and 10,000,000 triples, respectively. To ensure the diversity of rules, we generate 28 kinds of rules, covering all possible label situations with one to three triples as the premise. For example, a synthetic rule with two triples as premise can be like two correct premise triples infer that the conclusion triple is also correct.

D.2 Baseline

We compare our framework with the SOTA method TWCS [11]. It finds that annotating a cluster of triples with the same head entity, i.e., entity cluster, takes less time than annotating a group of random triples. Therefore, it chooses triples for annotation by sampling entity clusters. Specifically, TWCS contains two stages:

(1) sample entity clusters using weighted cluster sampling;
(2) only a small number of triples are sampled without replacement from clusters from stage 1.

The unbiased estimator of triple accuracy μ_1 with $1-\alpha$ CI is defined as follows:

$$\hat{\mu}_{w,m} = \frac{1}{n}\Sigma_k \hat{\mu}_{I_k} \tag{7}$$

$$\hat{\mu}_{w,m} \pm z_{\alpha/2}\sqrt{\frac{1}{n(1-n)}\Sigma_k(\hat{\mu}_{I_k} - \hat{\mu}_{w,m})^2} \tag{8}$$

where $\hat{\mu}_{I_k}$ is the mean accuracy of the sampled triples in the k-th sampled entity cluster, and n is the number of sampled clusters.

D.3 Weighted Random Sampling

The size of IG can be used as a standard to guide the weighted sampling. In other words, the i-th IG is drawn with a probability of M_i/M. We use Hansen-Hurwitz estimator [13], a widely used unbiased estimator for weighted sampling, to give an estimate of triple accuracy μ_1 with $1-\alpha$ CI as follow:

$$\hat{\mu}_{w1} = \frac{1}{n}\Sigma_{k=1}^{n}\mu_{1I_k} \tag{9}$$

$$\hat{\mu}_{w1} \pm z_{\alpha/2}\sqrt{\frac{1}{n(n-1)}\Sigma_{k=1}^{n}(\mu_{1I_k} - \hat{\mu}_{w1})^2} \tag{10}$$

The unbiased estimate of entity linking accuracy μ_2 with $1-\alpha$ CI is defined as follows:

$$\hat{\mu}_{w2} = \frac{M}{nH}\Sigma_{k=1}^{n}\frac{H_{Ik}}{M_{Ik}}\mu_{2I_k} \tag{11}$$

$$\hat{\mu}_{w2} \pm z_{\alpha/2}\frac{M}{H}\sqrt{\frac{1}{n(n-1)}\Sigma_{k=1}^{n}(\mu_{2I_k} - \hat{\mu}_{w2})^2} \tag{12}$$

D.4 Reproducibility

The datasets used in the papar and source codes are available at: https://github.com/KGQualityEvaluation/KG_Accuracy_Evaluation