Fact Ranking over Large-Scale Knowledge Graphs with Reasoning Embedding Models

Hongyu Ren†,[†] Ali Mousavi*, Anil Pacaci*, Shihabur R. Chowdhury*, Jason Mohoney‡*,
Ihab F. Ilyas*, Yunyao Li*, Theodoros Rekatsinas*
*Apple
†Stanford University, hyren@stanford.edu
‡University of Wisconsin-Madison, mohoney2@wisc.edu
*amousavi, apacaci, shihab, iilyas, yunyaoli, thodrek@apple.com

Abstract

Knowledge graphs (KGs) serve as the backbone of many applications such as recommendation systems and question answering. All these applications require reasoning about the relevance of facts in a KG to downstream applications. In this work, we describe our efforts in building a solution to reason about the importance of facts over continuously updated industry-scale KGs. We focus on the problem of fact ranking and evaluate to what extent modern knowledge graph embedding (KGE) models provide a representation for addressing this problem. To this end, we discuss unique challenges associated with solving this task in industrial settings and evaluate how accurately different KGE models and text-based embedding models can solve the problem of fact ranking.

1 Introduction

Knowledge graphs (KGs) are the backbone of applications such as question answering in virtual assistants and recommendation systems in. These applications require a broad range of knowledge that is continuously updated with recent facts from disparate data sources [13, 20]. Reasoning about the importance of facts in an industry-scale KG with billions of entities and facts across diverse domains is a challenging problem. An automated and scalable solution scalable across entity types and domains in a KG has obvious benefits.

Here, we focus on the problem of fact ranking. Fact ranking provides an importance-based ranking over facts for a given real-world entity. For example, given the question "What is the occupation of LeBron James?", the answer "basketball player" should be ranked higher than "television actor" or "screenwriter" despite the fact that these two are also LeBron James's occupations. Fact ranking generalizes the problem of recommendation generation [3] over KGs. We are interested in facts that cannot be ranked using a simple importance or popularity score. For example, ranking occupation of entities as described earlier. Another example is to generate recommendation for entities that are relevant within users' search context. For example, for the user query "How tall is LeBron James", we want to recommend a ranked list of top KG entities that are related to the query entity "LeBron James" and are aligned with users' search intent, i.e., they are "Person" entities with a "height" attribute. Fact ranking is important during rendering these enriching entity-centric experiences in intelligent assistants.

In this paper, we propose a solution to fact ranking based on modern Knowledge Graph Embedding (KGE) models and present an experimental evaluation in large-scale settings. Our solution adopts state-of-the-art multi-hop reasoning models. Specifically, we build on the recent Query2Box model [24] and demonstrate how

[†] work done during an internship at Apple.

the embeddings obtained by this model can address fact ranking over large-scale KGs. A major challenge in employing KGE models for fact ranking in real-world applications is to reason about the importance score and the rank of a facts obtained by an embedding model. We address this challenge by proposing a new metric for measuring the stability of embedding models across different rounds, namely, an adaptive version of Kendall's Tau that also takes into account the importance scores obtained by the embedding models. In this way, we can better measure the effects of the learned embeddings on the downstream use cases. Our approach is in contrast to using the standard forms of Kendall's Tau or Rank-based Overlap metrics, which measure the consistency across two ranked lists by considering only the number of discordant pairs/swaps between the two lists. We demonstrate that the reasoning-based Query2Box model leads to significantly more stable embeddings compared to one-hop embedding models such as DistMult [37]. We also propose a new indexing scheme and apply multi-query optimization for efficient search over the generated embedding vectors for supporting use-cases such as vector similarity-based related entity search.

Finally, we compare Query2Box against modern generative natural language (NL) models [18] and demonstrate that NL models require significant fine-tuning of the prompt to obtain similar fact ranking results as the Query2Box model.

2 Preliminaries

We now review background relevant to our study. Our discussion focuses on knowledge graph representation models and aims to highlight the differences between shallow KG embedding models such as the popular DistMult [37], TransE [2], and RotatE [27] models and more recent reasoning-based embedding models [24, 25].

2.1 Shallow Knowledge Graph Embeddings

A knowledge graph (KG) $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ consists of a set of nodes $\mathcal{V}$, a set of edges $\mathcal{E}$. $\mathcal{G}$ also defines a set of relations $\mathcal{R}$, and each edge $e \in \mathcal{E}$ represents a triple (v_s, r, v_o) where $r \in \mathcal{R}$ and $v_s, v_o \in \mathcal{V}$. Here, v_s corresponds to the vector representation of the `subject` of the fact that corresponds to the edge and v_o to the `object` of the fact. Finally, r corresponds to the vector representation of the `predicate` associated with the triple.

Shallow KG embeddings [2, 37, 27, 31] learn an embedding function f_θ that maps all the entities and relations on the graph to latent space in order to preserve the structure of graph. Most KG embeddings implement the embedding function f_θ as a matrix lookup. Specifically, the parameters include an entity embedding matrix $\mathbf{V}_\theta \in \mathbb{R}^{|\mathcal{V}|\times d}$ and a relation embedding matrix $\mathbf{R}_\theta \in \mathbb{R}^{|\mathcal{R}|\times d}$, where d is the latent space dimension.

Training Shallow KG Embeddings: In order to train the two embedding matrices, these methods optimize a contrastive objective, which is to minimize a predefined distance function `Dist` of existing edges $e = (v_s, r, v_o) \in \mathcal{E}$ while maximize that of non-existing edges $e' = (v_s, r, v'_o) \notin \mathcal{E}$. Different shallow KG embeddings have different definitions of the distance function `Dist`, the detail is listed in Table 1. In previous KG embedding works [27, 39], the loss function in the contrastive objective is defined as:

$$\mathcal{L} = -\log\sigma\left(\gamma - \texttt{Dist}(v_s, r, v_o)\right) - \sum_{j=1}^{k}\frac{1}{k}\log\sigma\left(\texttt{Dist}(v_s, r, v'_{o_j}) - \gamma\right), \qquad (46)$$

where σ is the sigmoid function, γ is the margin. We optimize over the loss function using stochastic training for several iterations. In each iteration, these methods sample a batch of existing edges from the graph and construct non-existing edges by keeping the subject v_s and the type of the edge r fixed while perturbing the object v_o.

Table 1: The distance function of shallow KG embeddings and KG reasoning embeddings.

Model	Embedding Space	Distance
TransE [2]	$f_\theta(v_s), f_\theta(v_o) \in \mathbb{R}^d$, $f_\theta(r) \in \mathbb{R}^d$	$\|f_\theta(v_s) + f_\theta(r) - f_\theta(v_o)\|$
RotatE [27]	$f_\theta(v_s), f_\theta(v_o) \in \mathbb{C}^d$,$f_\theta(r) \in \mathbb{C}^d$	$\|f_\theta(v_s) \circ f_\theta(r) - f_\theta(v_o)\|$
DistMult [37]	$f_\theta(v_s), f_\theta(v_o) \in \mathbb{R}^d$,$f_\theta(r) \in \mathbb{R}^d$	$- < f_\theta(v_s), f_\theta(r), f_\theta(v_o) >$
ComplEx [31]	$f_\theta(v_s), f_\theta(v_o) \in \mathbb{C}^d$, $f_\theta(r) \in \mathbb{C}^d$	$-\text{Re}(< f_\theta(v_s), f_\theta(r), \overline{f_\theta(v_o)} >)$
Q2B [24]	$f_\theta(v_s), f_\theta(v_o) \in \mathbb{R}^d$, $f_\theta(r) \in \mathbb{R}^{2d}$	$\texttt{Dist}_{\text{out}} + \alpha\texttt{Dist}_{\text{in}}$

2.2 KG Reasoning Embeddings

KG reasoning embedding methods generalize the shallow KG embeddings to more complex reasoning tasks. KG reasoning embeddings also consider multi-hop reasoning over the KG, i.e., answering complex logical queries with logical/set operators including conjunction, disjunction and negation, e.g., "Predict drugs that might target proteins that are associated with a given disease, and do not have a given side effect" [10]. In order to answer such complex queries, one may need to perform multiple reasoning steps and graph traversal – first find all the proteins associated with the disease, and predict drugs $\mathcal{D}_1 \subset \mathcal{V}$ that bind with the proteins, at the same time find the drugs $\mathcal{D}_2 \subset \mathcal{V}$ that have the side effect and take complement of the set $\overline{\mathcal{D}_2}$, and finally take the intersection of the two sets $\mathcal{D}_1 \cap \overline{\mathcal{D}_2}$ to achieve the answers to the query. One of the main challenges of the above graph traversal method is that it suffers from missing and noisy information on the graph. The key insight of KG reasoning embedding methods is to embed these complex queries in the same latent space as the entity embeddings so that all the reasoning steps can be done in the embedding space instead of symbolic graph traversal. In detail, we follow the logical queries defined in **(author?)** [25].

Definition 2.1 (First-order logic queries) *A first-order logic query q consists of a non-variable anchor entity set $\mathcal{V}_q \subseteq \mathcal{V}$, existentially quantified bound variables $V_1, \ldots, V_k$ and a single target variable $V_?$, which provides the query answer. The disjunctive normal form of a logical query q is a disjunction of one or more conjunctions.*

$$q[V_?] = V_? \, . \, \exists V_1, \ldots, V_k : c_1 \vee c_2 \vee ... \vee c_n$$

1. *Each c represents a conjunctive query with one or more literals e. $c_i = e_{i1} \wedge e_{i2} \wedge \cdots \wedge e_{im}$.*
2. *e represents an atomic formula or its negation. $e_{ij} = r(v_a, V)$ or $\neg\, r(v_a, V)$ or $r(V', V)$ or $\neg\, r(V', V)$, where $v_a \in \mathcal{V}_q$, $V \in \{V_?, V_1, \ldots, V_k\}$, $V' \in \{V_1, \ldots, V_k\}$, $V \neq V'$, $r \in \mathcal{R}$.*

In order to reason over and embed such queries, one needs to consider the following operations. KG reasoning methods design neural logical operators that simulate their real counterparts. We refer the readers to [24] for more details.

1. **Relation Projection:** Given a set of entities $S \subseteq \mathcal{V}$ and relation type $r \in \mathcal{R}$, compute adjacent entities $\cup_{v \in S} A_r(v)$ related to S via r: $A_r(v) \equiv \{v' \in \mathcal{V} : (v, r, v') \in \mathcal{E}\}$.
2. **Intersection:** Given sets of entities $\{S_1, S_2, \ldots, S_n\}$, compute their intersection $\cap_{i=1}^n S_i$.
3. **Complement/Negation:** Given a set of entities $S \subseteq \mathcal{V}$, compute its complement $\overline{S} \equiv \mathcal{V} \setminus S$.
4. **Union:** Given sets of entities $\{S_1, S_2, \ldots, S_n\}$, compute their union $\cup_{i=1}^n S_i$.

Capturing Relational Context: While KG reasoning queries have been traditionally proposed to answer complex queries in the presence of incomplete KGs, here, we utilize them to learn vector representations of the entities

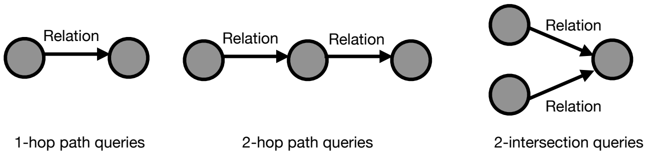

Figure 1: Types of queries we consider to train Query2box.

that are not biased towards one-hop relationships but take into account a richer relational context. We use a set of query templates (see Figure 1) to generate a sample workload of queries that can be answered over the input KG and use that payload to learn robust entity representations as we discuss next. We experimentally show (see Section 6) that this relational bias in the training process leads to entity representations that are more robust and lead to more stable representations (see Section 6).

Training KG Reasoning Embeddings: For a KG, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$, and a query q, we need to learn an embedding function f_θ that maps from a computation graph of a query to its embedding with the parameterized neural logical operators. Together with the entity and relation embedding matrices (same as the shallow KG embeddings), f_θ also embeds all the nodes on the graph by embedding lookup. In order to measure the similarity/distance between a query q and an entity $v \in \mathcal{V}$, a distance function $\texttt{Dist}(\cdot,\cdot)$ is defined that takes as input the query embedding $f_\theta(q)$ and the entity embedding $f_\theta(v)$ and outputs the distance. The distance function $\texttt{Dist}(\cdot,\cdot)$ is tailored to different embedding space and model design f_θ as in Table 1.

During training, we are given a data sampler $\mathcal{D}$, each sample in $\mathcal{D}$ is a tuple $(q, \mathcal{A}_q, \mathcal{N}_q)$, which represents a query q, its answers $\mathcal{A}_q \subseteq \mathcal{V}$ and the negative samples $\mathcal{N}_q \subseteq \overline{\mathcal{A}_q}$. The training objective is to minimize the distance between the query embedding and its answers $\texttt{Dist}(q, v), v \in \mathcal{A}_q$ while maximizing the distance between the query embedding and the negative samples $\texttt{Dist}(q, v'), v' \in \mathcal{N}_q$, optimizing a contrastive loss term similar to the shallow KG embeddings. As used in most previous KG reasoning embedding works [25, 24], the loss is defined as:

$$\mathcal{L} = -\log \sigma\left(\gamma - \texttt{Dist}(q, v)\right) - \sum_{j=1}^{k} \frac{1}{k} \log \sigma\left(\texttt{Dist}(q, v'_j) - \gamma\right), \tag{47}$$

where γ is a margin hyperparameter and σ is the sigmoid function.

3 Fact Ranking

Here we introduce the task of fact ranking and its corresponding applications that power critical user experiences over KGs. We define a fact on KG as a triple (v_s, r, v_o), where v_s and v_o are entities and r is a relation type from the KG. User queries we target (such as the "What is the occupation of LeBron James?") correspond to queries of the form $q = (v_s, r, ?)$.

3.1 Problem Description

Intelligent assistants rely on entity-centric experiences to answer user queries. For many of these experiences, facts or answers to queries are of different importance/accuracy/uncertainty to users. Besides providing all the answers, one key aspect of service quality is to display relevant facts in a sorted way according to the importance or uncertainty score. For example, the answer to a query "What is the occupation of Selena Gomez?" includes singer and child actor, but for the majority of users, we know the answer singer should rank higher than child actor. The goal of fact ranking is to rank all the answers to a given set of queries that aligns well with user expectation. Fact ranking is ubiquitous in intelligent assistants and the key to improving the user experience.

We define fact ranking as follows: Given a query $q = (v_s, r, ?)$ for a subject v_s and a predicate r we see to find answers $\mathcal{A}_q$ (achieved by graph traversal as discussed in Section 2) for the missing object. The goal of fact ranking is to find a function $\mathsf{Rank}(a)$ which can be used to obtain an importance ranking for each answer $a \in \mathcal{A}_q$ and generate the ranking list $[\mathsf{Rank}(a_1), \ldots, \mathsf{Rank}(a_n)],\ a_1, \ldots, a_n \in \mathcal{A}_q$. We focus on queries that target facts that cannot be ranked using simple popularity scores, e.g., occupations, genres etc., (an occupation is not necessarily more important than the other).

Unsupervised machine learning (ML) mechanisms are needed to learn a function $\mathsf{Rank}(\cdot)$. It is typical that KGs do not associate any importance scores and weights to their edges, which applies to many large KGs including FreeBase [1] and WikiData [32]. Consequently, it is not possible to use simple traversal mechanisms to implement the ranking functions for fact ranking and different mechanisms need to be considered. Such a mechanism may correspond to PageRank-based algorithms which can be used to assign an importance score for each answer a (or fact (q, a) with personalized PageRank), however, they are often not effective on such large-scale heterogeneous graphs where multiple edge types exist [12]. To alleviate these challenges, we consider a setting (see Section 3.2) where unsupervised representation learning is used to learn Function $\mathsf{Rank}(\cdot)$.

3.2 A Solution with KG Embeddings

We obtain a solution to the fact ranking problem by leveraging graph embedding models. This solution applies to both shallow KG embeddings and KG reasoning embeddings. The idea is to first train the entity embeddings, relation embeddings, and neural logical operators on the KG using standard training protocols [37, 24] (see Section 2.2). Then, given a fact (v_s, r, v_o), we can use the pre-trained embeddings to efficiently calculate the distance $\mathsf{Dist}(v_s, r, v_o)$. The distance plays a crucial role for solving the fact ranking problem since it represents a proxy of plausibility of a fact. This solution is inspired by our prior work on error detection, missing value imputation, and data repairs which showed that all problems correspond to inference tasks over a pre-trained model that learns how to reconstruct the input data [6]. Nonetheless, using the distance obtained by different KG embedding models raises two critical considerations for industrial settings.

For fact ranking, the distance obtained by the KG embedding model can be applied to rank the candidate objects for a specific subject and object configuration. However, different models can learn significantly different geometries and in most cases the distances in these spaces can lead to significant variations in the obtained rankings. In the settings we consider, it is critical that the rankings obtained are stable (i.e., we do not have significant variations in the order of different objects) across training iterations of the embedding model. To this end, we use a post-processing step that verifies the stability of KG embedding models before deployment. This post-processing step utilizes a consistency metric that extends standard ranking comparison methods such as Kendall's Tau to also consider the distance value associated with each query (see Section 4).

4 Consistency in Fact Ranking

We consider different ways to measure the stability of ranking across different training runs. Given a query $(v_s, r, ?)$ and its answers $\mathcal{A}_q = \{a_1, \ldots, a_n\}$, we calculate: $d_i = \mathsf{Dist}(v_s, r, a_i), \forall i = 1, \ldots, n$, and create a distance list $\mathsf{DistList} = [d_1, \ldots, d_n], d_i \in \mathbb{R}$ and a ranking list of the answers by the distance $\mathsf{RankList} = [\mathsf{Rank}(a_1), \ldots, \mathsf{Rank}(a_n)], \mathsf{Rank}(a_i) \in \{1, 2, \ldots, n\}$. We consider training KG embeddings multiple times, and obtain multiple $\mathsf{DistList}$ and $\mathsf{RankList}$. Our goal is to measure the stability/consistency of the lists across different runs. We assume the KG stays unchanged across different runs/training of the embedding models, hence the items in the list of a query also remain the same.

To measure the stability of ranking, i.e., compare whether two $\mathsf{RankList}$ from two runs are consistent, we consider several metrics, including 1) the Kendall rank correlation coefficient, 2) a weighted version of Kendall's Tau, 3) set-based overlap, and 4) rank-biased overlap. Given two distance lists $\mathsf{DistList}_1 = [x_1, \ldots, x_n]$ and

Algorithm 1: AdaptiveCluster

```
Input  : A list of distance of answers DistList = $[x_1, \ldots, x_n]$, a scalar threshold $\delta'$ (hyperparameter).
Output : A list of cluster IDs ClusterList.
DiffList = [];
for $i \leftarrow 1$ to $n$ do
    DiffList.append($x_i - x_{i-1}$);
$\mu$ = DiffList.mean(), $\sigma$ = DiffList.std();
Threshold $\delta = \min(\mu - 0.2\sigma,\ \delta')$;
ClusterList = [0], $clusterid = 0$;
for $i \leftarrow 0$ to $n - 1$ do
    if DiffList[$i$] $> \delta$ then
        $clusterid++$;
    ClusterList.append($clusterid$);
return ClusterList;
```

Algorithm 2: Adaptive Tau

```
Input  : Two lists of distance of answers DistList$_1$, DistList$_2$, a scalar threshold $\delta'$ (hyperparameter).
Output : Kendall's Tau coefficient.
ClusterList$_1$ = AdaptiveCluster(DistList$_1$, $\delta'$);
ClusterList$_2$ = AdaptiveCluster(DistList$_2$, $\delta'$);
return KendallTau(ClusterList$_1$, ClusterList$_2$);
```

$\mathtt{DistList}_2 = [y_1, \ldots, y_n]$, the four metrics are calculated as:

1. Kendall's Tau: $\frac{m_c - m_d}{\binom{m}{2}}$, where m_c is the number of concordant pairs between $\mathtt{DistList}_1$ and $\mathtt{DistList}_2$, and m_d is the number of discordant pairs. A pair of (i, j) is concordant if the sort order of (x_i, x_j) and (y_i, y_j) is the same, otherwise the pair is discordant. Kendall's Tau ranges from -1 to 1.

2. Weighted Tau: It is an extension of Kendall's Tau where each pair also has a weight that is inverse-proportional to the rank, i.e., low ranking objects are not as important as the top ranking objects.

3. Rank-biased overlap (RBO): $(1 - p)\sum_{i=1}^{n} p^{i-1} \cdot A_i$, where i is the depth of the ranking being examined. With $ArgSort$ function, let $\mathtt{ASList} = ArgSort(\mathtt{DistList})$, we define $A_i = \frac{|\mathtt{ASList}_1[:i] \cap \mathtt{ASList}_2[:i]|}{i}$. The idea of RBO is to compare the overlap of the two rankings at incrementally increasing depths. It is a weighted metric, which means that the top rank items get higher weights.

However, the downside of the above metrics is that they do not explicitly consider the absolute value of items in DistList. One observation is that when two answers have similar distance with the query embedding, a swap in the ranking of the two answers from two runs should not matter as much as a swap in the ranking when the two answers have different distance to the query embedding. Consider the following two scenarios, assume in both scenarios, the length of the DistList is 3. In Scenario #1 we have $\mathtt{DistList}_1 : [0.20, 0.30, 0.33]$ $\mathtt{DistList}_2 : [0.45, 0.61, 0.60]$ and in Scenario #2 we have $\mathtt{DistList}_1 : [0.20, 0.30, 0.63]$ $\mathtt{DistList}_2 : [0.45, 0.61, 0.50]$.

Although in both scenarios, there exists one discordant pair (the second and third item), yet in Scenario #1, the two items have extremely close distance compared with Scenario #2. So an ideal metric would output a higher consistency score for Scenario #1 than Scenario #2. However, all above metrics give the same results.

To address the above shortcoming, we use an evaluation metric that adaptively considers the margins of different items when measuring the consistency of two DistList. In order to identify the items with close

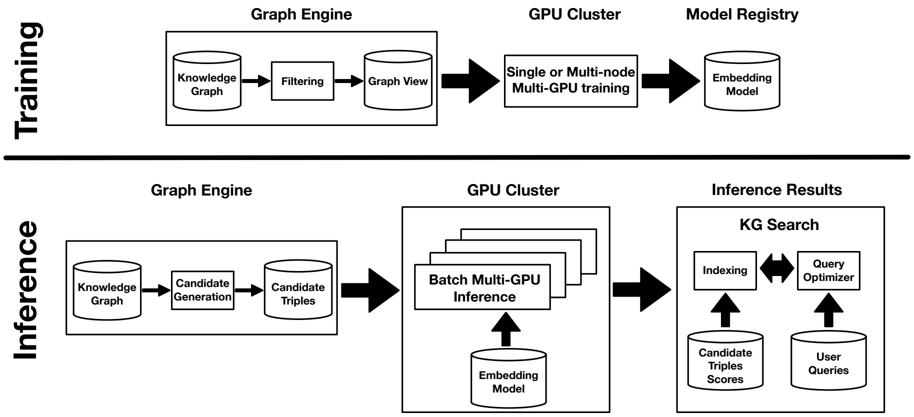

Figure 2: An overview of using Knowledge Graph Embedding models for large-scale fact ranking.

values, we sort the `DistList`, calculate the difference between neighbor items, and measure the average and variance, which we use to set as a threshold. Then, we loop over all the items and aim to cluster the item by checking whether the difference between the current item and the previous item is larger than the threshold. For items in the same cluster, we assign the same value to them such that they will have the same ranking. Finally, we run Kendall's Tau metric over this updated list. The details are shown in Algorithms 1 and 2. We refer to this method as Adaptive Tau since it considers the absolute value of the discordant pairs using clustering with an adaptive threshold. An experimental analysis of the different metrics is shown in Section 6. We find that Adaptive Tau provides a more precise description of the stability and utility of the rankings obtained by embedding models.

5 Scaling to Large KGs

We discuss systems considerations when using KG reasoning embedding models over large-scale KGs. Beyond scalable training, we also require that inference over Query2Box models is scalable and incremental. This requirement is important to enable practical deployments over dynamic billion-scale KGs. We next discuss the main components of the architecture we adopt (see Figure 2).

The multi-hop nature of embedding models such as Query2Box poses unique challenges when training these models over billion-scale KGs. In the case of shallow KG embedding models, graph partitioning is a common method for scaling training [40, 17]. Unfortunately, these methods are not applicable in the case of reasoning-based embeddings. When using Query2Box it is important that we can generate training samples by performing multi-hop traversals of the graph. Such traversals can span multiple partitions. At the same time, it is not always practical to pre-compute such samples in advance. To alleviate this issue, we opt for a single-machine multi-GPU deployment during training and leverage the recently introduced SMORE engine [23] to perform training. SMORE provides a mixed GPU-CPU solution that leverages both the main memory and GPU memory to scale training. In addition, training examples are generated on the fly thus avoiding unnecessary pre-compution. Indicative throughput measurements and scaling of SMORE is shown in Figure 3. A requirement here is that the machine have sufficient main and on-device memory to store the entire graph and thus avoid partitioning. While this requirement is satisfied by modern hardware configurations it is a cost-hungry option. We believe disk-based or distributed training of reasoning-based KG embeddings is an exciting research direction. Once training is complete, the embedding models are archived and then used for inference.

At inference time, we opt for a batch inference setting. We first compute a series of candidate queries that correspond to the set of facts that we want to enable ranking over. We leverage a computation graph engine to materialize all candidate queries (i.e., `(subject, predicate, object` triples) and use the learned Query2Box model to obtain a score for each query. The number of candidate facts can exceed the size of the

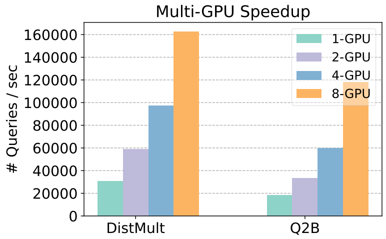

Figure 3: Multi-GPU scaling of SMORE for training the DistMult and Query2box embedding models.

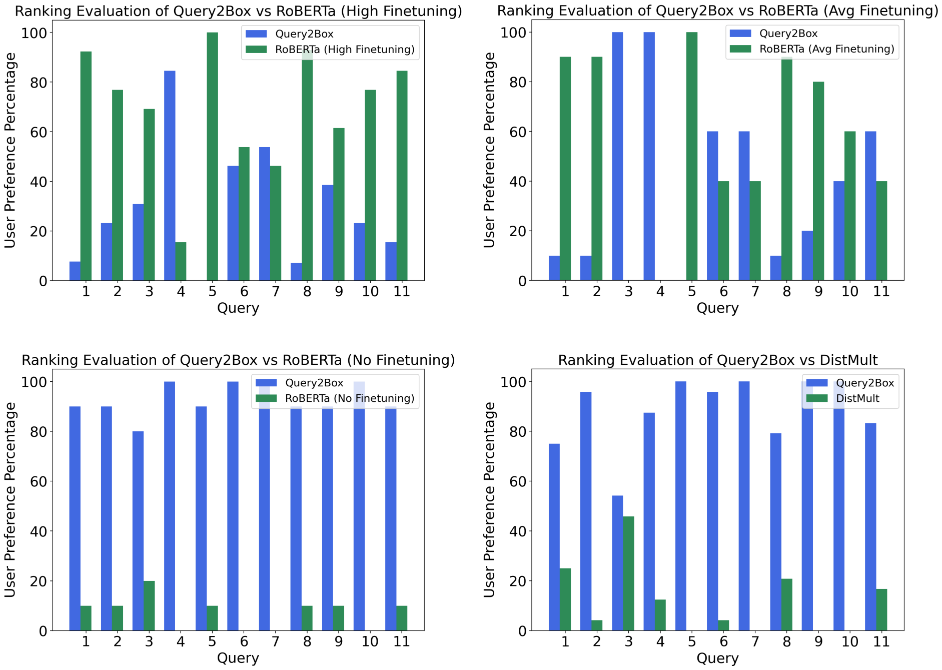

Figure 4: Fact ranking using Query2Box vs. RoBERTa and Query2Box vs. DistMult.

original KG as we consider multiple subject, object configurations that may not appear in the original graph. To deal with the volume of generated queries we opt for a scatter gather-based multi-GPU inference across multiple machines in a GPU cluster. The trained model is loaded in the executor allocated to each machine and the candidate facts are partitioned across machines. The corresponding inference results are gathered into a single relational store and then used for downstream processing. Given the stability of the Query2Box representations (see Section 6), we maintain the fact ranking results via periodic retraining of the Query2Box models followed by batch inference. Inference results for fact ranking are versioned across different training and batch inference runs.

6 Experiments

We evaluate reasoning-based KG embeddings for fact ranking. We focus on fact ranking tasks that align with use cases in industrial deployments. We evaluate the following aspects of our proposed framework: (1) the utility to end users when using KG embeddings for fact ranking, (2) the stability of KG embedding models and hence to what extent they satisfy deployment requirements.

Table 2: Average user preference (based on results in Figures 4) for Q2Box vs other methods for different tasks. For RoBERTa models, (H): high finetuning, (A): average finetuning, (N): no finetuning.

Query2Box	**DistMult**	**RoBERTa (N)**	**RoBERTa (A)**	**RoBERTa (H)**
1- TV Actor	1- Film Actor	1- Singer	1- Film Producer	1- Actor
2- Film Actor	2- Singer	2- Film Producer	2- Film Director	2- Singer
3- Film Director	3- Film Director	3- Film Director	3- Film Actor	3- Film Producer
4- Actor	4- Film Producer	4- Film Actor	4- TV Actor	4- Film Director
5- Film producer	5- Actor	5- Actor	5- Actor	5- Film Actor
6- Singer	6- TV Actor	6- TV Actor	6- Singer	6- TV Actor

Table 3: Average user preference (based on results in Figure 4) for Q2Box vs other methods for different tasks. For RoBERTa models, (H): high finetuning, (A): average finetuning, (N): no finetuning.

Comparison Task	**Competitor**	**Competitor Avg. Percentage**	**Q2Box Avg. Percentage**
DistMult / Q2Box	DistMult	12%	**88%**
RoBERTa (H) / Q2Box	RoBERTa (H)	**70%**	30%
RoBERTa (A) / Q2Box	RoBERTa (A)	**57%**	43%
RoBERTa (N) / Q2Box	RoBERTa (N)	7%	**93%**

Table 4: Stability results of fact ranking. Query2box consistently achieves better performance than DistMult.

	Kendall	Weighted Kendall	Set-based Overlap	Rank-biased Overlap	AdaptiveTau $\delta' = 0.02$	AdaptiveTau $\delta' = 0.05$	AdaptiveTau $\delta' = 0.1$
DistMult	0.380	0.384	0.962	0.990	0.460	0.498	0.484
Query2Box	**0.854**	**0.868**	**0.990**	**0.997**	**0.877**	**0.917**	**0.943**

6.1 Experiment Setup

Queries and Facts We focus on queries of the format "What is the occupation of `[Celeb_Name]`?" which captures the ranking task. We rank possible object completions for the structured query $(v_s, \texttt{OccupationOf}, ?)$, where v_s is the entity of interest. Our dataset contains several million queries obtained by real intelligent assistant user queries.

Knowledge Graph We consider the entire Wikidata KG [32] to validate the proposed framework. The version of Wikidata that we use contains 1,754,058,566 facts defined over 91,900,599 entities and 35,446 relation types. We train our framework on this KG and obtain the answers to the aforementioned set of user queries by finding entities in this KG.

Baselines We evaluate a diverse array of methods, including KG reasoning embeddings *Query2box* [24], shallow KG embeddings *DistMult* [37], and masked language models (MLMs) *RoBERTa* [18].

For KG embedding based methods, both DistMult and Query2box fit in our unified framework. We adopt the standard training procedures for these models (see Section 2). For DistMult, we sample existing edges as positive samples and non-existing edges as negative samples to train the DistMult model with the objective defined in equation 46, where the distance and residue functions are defined in Section 2. For Query2box, similar to [24] and as discussed in Section 2.2, we sample multi-hop queries (Figure 1), and their answers and non-answers to optimize the contrastive objective in equation 47. Besides the entity and relation embedding matrices, we use the neural logic operators in Query2box to embed the complex queries and optimize the query embeddings such that they are close to the answer embedding and pushed far away from the embedding of the sampled non-answers. We use the distance function of the original Query2box model and design the residue function as described in Section 2.

For both DistMult and Query2Box, we use SMORE [23] for training. We train both for 100k iterations with the Adam optimizer [16]. We anneal the learning rate from 0.001 to 0.0001, and adopt a batch size of 8,192 queries with 1,024 negative answers for each query in the batch. We score each candidate answer using the distance functions defined for both models.

For MLMs, we use the RoBERTa model. Given a triple-format query $(v_s, \texttt{OccupationOf}, ?)$, we provide three templates and convert the query into a natural language question. These templates include (1) "v_o is a `[mask]`.", (2) "The occupation of v_o is `[mask]`.", and (3) prompting [34] in which the template is "Barack Obama is a politician, LeBron James is a basketball player, v_o is a `[mask]`.". For different query types, we need to provide different prompts in order to make predictions more effective. As we show later, fine-tuning of the prompt is necessary to obtain competitive results and hence, MLMs are not a universal solution to our task. Given v_o, we score the candidate answer by calculating the likelihood score of v_o in replacement of the `[mask]` in each of the three templates.

6.2 Fact Ranking Evaluation

Utility We first evaluate the utility of our framework on the fact ranking task. To assess the quality of Query2Box and compare it against other methods, we consider eleven celebrities and their occupations as listed in WikiData. We present our users with four different questionnaires. Each of these questionnaires compares Query2box ranking vs. another baseline ranking obtained using one of the methods we mentioned earlier. Each questionnaire contains eleven questions where each asks users to choose between two different rankings (Query2Box vs. a baseline) of a celebrity occupation.

Figure 4 shows the summary of user preferences between DistMult/RoBERTa and Query2box rankings. In Figure 4, Query2box has outperformed DistMult and Query No. 3 (occupations of Jennifer Lawrence) shows the tightest competition. Table 2 shows the ranking derived from these methods. Jennifer Lawrence is mostly known as a *Film Actor* which is correctly predicted by DistMult. Some users have focused on the first occupation and hence voted for DistMult while other users have considered other occupations and voted for Query2box. In addition, we can notice the flipping of user preferences based on the amount of fine-tuning for RoBERTa models (see also Table 3). As shown in Table 3 which represents the average user preference, Query2Box outperforms DistMult and RoBERTa requires significant fine-tuning of the prompt to outperform the Query2box model.

Stability We now measure the stability of different systems using the metrics we introduced in Section 3.2. We take all triples with `OccupationOf` as the relation type from the massive KG. Overall the dataset involves 6,566,224 queries of structure $(v_o, \texttt{OccupationOf}, ?)$ for which we measure the stability and consistency of rankings. Here we mainly consider two methods Query2box and DistMult, but not the MLMs due to their necessity of contexts for better ranking utility as discussed. We train both models 5 times and measure the stability of both models using the metrics introduced in Section 4. As shown in Table 4, we find Query2box is more stable and consistent than DistMult since Query2box is trained on more complex multi-hop queries, which better captures the neighborhood structure for each fact. For set-based overlap and rank-biased overlap, both methods achieve extremely high values. This is expected since the occupations of a celebrity are fixed across runs, and the overlap will always be 1 at the last step as we gradually compare the intersection of two sets starting from top-ranking items to the low-ranking ones. Among evaluation metrics, our adaptive method can better characterize a more meaningful measurement of ranking stability than the vanilla Kendall's Tau and rank-biased overlap. As shown in Table 4, Query2box achieves higher performance in AdaptiveTau than the other two metrics. We argue such an adaptive metric is crucial in evaluating ranking stability in production.

7 Use-case: Ranking for Related Entity Search

Recommendation generation is a key component of question answering in entity-centric user experiences, and the task of providing a ranked list of KG entities related to that of users' query can be performed via fact ranking with KGE models. Specifically, given a user query $q = (v_s, r, ?)$, the goal of related entity search is to find a ranking function $\texttt{Rank}(v_r)$ over a subset KG of entities $v_r \in \mathcal{R} \subseteq \mathcal{V}$ such that the query $q_r = (v_r, r, ?)$ is relevant to the original query, and $\texttt{Rank}(v_r)$ provides a ranking of each entity v_r based on relatedness to the original query. We leverage the KGE models described in the previous sections for embedding a KG into a vector space and define relatedness between two entities in a KG to be the similarity between their vector representations. Thus, we can use similarity search over KG embeddings to find related entities for a given KG entity. Depending on the specific application of related entity search, we can use different embedding models. As an example, if we are interested in relatedness in the ontology space of a KG, Poincaré embeddings [41] is a suitable model. Otherwise, if we care about relatedness in the whole graph we can use either a shallow (e.g., DistMult [37]) or reasoning-based embedding model.

In addition to the use of KGE-based fact ranking for similarity based relatedness, the task of finding a ranked list of related KG entities for a query requires evaluating additional constraints for aligning answers with users' search intent. For instance, for the query "How tall is LeBron James", the goal is to find other "Person" entities that are related to "Lebron James" and have the corresponding fact for the same predicate "height". Consequently, related entity search use-case goes beyond the traditional vector similarity search and requires batch processing of hybrid queries [42]. Hybrid queries are two part queries consisting of: (i) vector similarity search for retrieving the most similar entities in the embedding space; and (ii) evaluation of conjunction of relational constraints for ensuring the returned results are relevant to search context (e.g., only include "Person" entities). In addition to hybrid query processing, the task of related entity search exhibits following characteristics: (i) hybrid queries are evaluated in a **batch setting** over past user queries, (ii) and relational predicates in industrial KG workloads exhibit filter commonality and filter stability [28], allowing us to customize the system design based on **available prior workload** characteristics. To this end, we employ HQI [42] hybrid vector similarity search system for batch inference over KG embeddings and adopt the following suite of optimizations for *high-throughput batch processing of hybrid queries*:

Workload-aware vector index: Specialized vector indexes that either partition the data or form multi-level indexes over centroids are commonly used in vector databases to speed-up vector similarity search [33]. HQI utilizes the past workload information to guide the partitioning of the vectors in the underlying index in a way that hybrid queries can be answered by accessing as few partitions as possible. By extending the concept of query-data routing trees (qd-trees) [36] to vector databases, HQI considers both vectors and relational predicates from a hybrid query workload when generating physical data layout at data loading time. The resulting data layout partitions the vectors using the distribution of the attributes associated with vectors, the attribute constraints, and similarity of vectors present in the hybrid query workload. We then use the resulting partitioning scheme to generate an index layout that enables processing a batch workload of hybrid queries by accessing vectors from as few partitions as possible.

Batch query optimization: Second, we use HQI's a multi-query optimization technique that (i) batches queries with similar attribute and vector similarity constraints; and (ii) performs batch vector distance computation against a posting list of vectors obtained from a clustering-based index over the vectors. This optimization is motivated by the fact that the set of candidate queries are computed from past user queries and evaluated in a batch setting, which enables computation sharing across queries. Note that this optimization is orthogonal to the workload-aware vector index and is applicable to any clustering-based vector index.

We evaluate the performance improvements of these optimizations for related entity search over KG. We use a subset of KG entity embedding vectors, and we focus on ranking related entities for queries of the format "What is the `[Predicate]` of `[Entity_Name]`?", similar to Section 6. Table 5 compares the performance of our solution against available existing hybrid query processing strategies (see [42] for more details) using a randomly

Table 5: Slowdown for related entity search compared to HQI @ Recall >= .8

	HQI	PreFilter	PostFilter	Range
Slowdown	1×	31×	136×	NA

sampled and aggregated query workload from anonymized, historical queries. HQI and its optimizations provide orders of magnitude performance improvements over best performing baselines for the related entity search task.

8 Conclusion

In this work, we studied fact ranking over large-scale knowledge graphs. We evaluated to what extent modern knowledge graph embedding (KGE) models provide a solution for addressing the problem of fact ranking. We highlighted unique challenges associated with solving this task in industrial settings and evaluated different KGE and text-based embedding models. Our work demonstrated that, in contrast to neural language models or shallow KGE models, multi-hop reasoning models such as Query2Box can better meet user satisfaction.

References

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. SIGMOD, 2008.

[2] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko. Translating Embeddings for Modeling Multi-Relational Data. Neural Information Processing Systems, 2787—2795, 2013.

[3] S. Bouraga, I. Jurerta, S. Faulkner, and C. Herssens. Knowledge-based recommendation systems: a survey. International Journal of Intelligent Information Technologies, 10(2):1–19, 2014.

[4] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Re, and K. Murphy. Machine learning on graphs: A model and comprehensive taxonomy. arXiv preprint, arXiv:2005.03675, 2020.

[5] W. Cohen. TensorLog: A Differentiable Deductive Database. arXiv preprint, arXiv:1605.06523, 2016.

[6] C. De Sa, I. Ilyas, B. Kimelfeld, C. Ré, and T. Rekatsinas. A formal framework for probabilistic unclean databases. arXiv preprint, arXiv:1801.06750, 2018.

[7] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 601–610, 2014.

[8] D. Xin, and T. Rekatsinas. Data Integration and Machine Learning: A Natural Synergy. ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019.

[9] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, K. Murphy, S. Sun, and W. Zhang. From Data Fusion to Knowledge Fusion. VLDB, 7(10):881–892, 2014.

[10] W. Hamilton, P. Bajaj, M. Zitnik, D. Jurafsky, and J. Leskovec. Embedding Logical Queries on Knowledge Graphs. Neural Information Processing Systems, 2018.

[11] A. Heidari, G. Michalopoulos, S. Kushagra, I. Ilyas, and T. Rekatsinas. Record fusion: A learning approach. arXiv preprint, arXiv:2006.10208, 2020.

[12] H. Huang, L. Sun, B. Du, C. Liu, W. Lv, and H. Xiong. Representation Learning on Knowledge Graphs for Node Importance Estimation. ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 646-655, 2021.

[13] I. Ilyas, T. Rekatsinas, V. Konda, J. Pound, X. Qi, and M. Soliman. Saga: A Platform for Continuous Construction and Serving of Knowledge At Scale. ACM SIGMOD International Conference on Management of data, 2022.

[14] I. Ilyas, and X. Chu. Data Cleaning. Morgan & Claypool, 2019.

[15] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S.Y. Philip. A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Transactions on Neural Networks and Learning Systems, 2021.

[16] D. Kingma, and J. Ba. Adam: A method for stochastic optimization. International Conference on Learning Representations (ICLR), 2015.

[17] A. Lerer, L. Wu, J. Shen, T. Lacroix, L. Wehrstedt, A. Bose, and A. Peysakhovich. Pytorch-biggraph: A large-scale graph embedding system. Conference on Machine Learning and Systems (MLSys), 2019.

[18] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. arXiv preprint, arXiv:1907.11692, 2019.

[19] J. Mohoney, R. Waleffe, H. Xu, T. Rekatsinas, S. Venkataraman. Marius: Learning Massive Graph Embeddings on a Single Machine. USENIX Symposium on Operating Systems Design and Implementation (OSDI), 2021.

[20] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, and J. Taylor. Industry-Scale Knowledge Graphs: Lessons and Challenges. Queue, 17(2):48–75, 2019.

[21] J. Pujara, H. Miao, L. Getoor, and W. Cohen. Knowledge graph identification. International semantic web conference, 542-557, 2013.

[22] T. Rekatsinas, X. Chu, I. Ilyas, and C. Ré. HoloClean: Holistic Data Repairs with Probabilistic Inference. VLDB Endowment, 10(11):1190–1201, 2017.

[23] H. Ren, H. Dai, B. Dai, X. Chen, D. Zhou, J. Leskovec, and D. Schuurmans. SMORE: Knowledge Graph Completion and Multi-hop Reasoning in Massive Knowledge Graphs. arXiv preprint, arXiv:2110.14890, 2021.

[24] H. Ren, W. Hu, and J. Leskovec. Query2box: Reasoning over Knowledge Graphs in Vector Space using Box Embeddings. International Conference on Learning Representations (ICLR), 2020.

[25] H. Ren, and J. Leskovec. Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs. Neural Information Processing Systems (NeurIPS), 2020.

[26] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, P. Merialdo. Knowledge graph embedding for link prediction: A comparative analysis. ACM Transactions on Knowledge Discovery from Data, 15(2)1–49, 2021.

[27] Z. Sun, Z. Deng, J. Nie, and J. Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. International Conference on Learning Representations (ICLR), 2019.

[28] L. Sun, M.J. Franklin, S. Krishnan, and R.S. Xin. Fine-grained partitioning for aggressive data skipping, ACM SIGMOD International Conference on Management of Data, 1115-1126, 2014

[29] Z. Sun, S. Vashishth, S. Sanyal, P. Talukdar, and Y. Yang. A re-evaluation of knowledge graph completion methods. arXiv preprint, arXiv:1911.03903, 2019.

[30] P. Tabacof and L. Costabello. Probability Calibration for Knowledge Graph Embedding Models. International Conference on Learning Representations (ICLR), 2020.

[31] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, G. Bouchard. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML), 2016.

[32] D. Vrandečić, and M. Krötzsch. Wikidata: A Free Collaborative Knowledgebase. Communications of the ACM, 57(10):78–85, 2014.

[33] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu, and others. Milvus: A purpose-built vector data management system. ACM SIGMOD International Conference on Management of Data, 2614–2627, 2021.

[34] C. Wei, S.M. Xie, and T. Ma. Why Do Pretrained Language Models Help in Downstream Tasks? An Analysis of Head and Prompt Tuning. Advances in Neural Information Processing Systems, 2021.

[35] G. Weikum. Knowledge graphs 2021: a data odyssey. VLDB Endowment, 14(12):3233–3238, 2021.

[36] Z. Yang, B. Chandramouli, C. Wang, J. Gehrke, Y. Li, U.F. Minhas, P. Larson, D. Kossman, and R. Acharya. Qd-Tree: Learning Data Layouts for Big Data Analytics. ACM SIGMOD International Conference on Management of Data, 193–208, 2020.

[37] B. Yang, W. Yih, X. He, J. Gao, L. Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. International Conference on Learning Representations (ICLR), 2015.

[38] J. You, X. Ma, Y. Ding, M. Kochenderfer, and J. Leskovec. Handling missing data with graph representation learning. Advances in Neural Information Processing Systems (NeurIPS), 2020.

[39] S. Zhang, Y. Tay, L. Yao, Q. Liu. Quaternion knowledge graph embeddings. Advances in Neural Information Processing Systems (NeurIPS), 2019.

[40] Z. Zhu, S. Xu, J. Tang, and M. Qu. Graphvite: A high-performance cpu-gpu hybrid system for node embedding. The World Wide Web Conference (WWW), 2019.

[41] M. Nickel, and D. Kiela. Poincaré embeddings for learning hierarchical representations Advances in neural information processing systems (NeurIPS), 2017.

[42] J. Mohoney, A. Pacaci, S.R. Chowdhury, A. Mousavi, I. Ilyas, U.F. Minhas, J. POund, T. Rekatsinas High-Throughput Vector Similarity Search in Knowledge Graphs ACM SIGMOD/PODS International Conference on Management of

Data (SIGMOD), 2023.