KGE Calibrator: An Efficient Probability Calibration Method of Knowledge Graph Embedding Models for Trustworthy Link Prediction

Yang Yang and **Mohan Timilsina** and **Edward Curry**
Insight Centre for Data Analytics, Data Science Institute, University of Galway, Ireland
{yang.yang,mohan.timilsina,edward.curry}@insight-centre.org

Abstract

Knowledge graph embedding (KGE) models are designed for the task of link prediction, which aims to infer missing triples by learning representations for entities and relations. While KGE models excel at ranking-based link prediction, the critical issue of probability calibration has been largely overlooked, resulting in uncalibrated estimates that limit their adoption in high-stakes domains where trustworthy predictions are essential. Addressing this is challenging, as we demonstrate that existing calibration methods are ill-suited to KGEs, often significantly degrading the essential ranking performance they are meant to support. To overcome this, we introduce the KGE Calibrator (KGEC), the first probability calibration method tailored for KGE models to enhance the trustworthiness of their predictions. KGEC integrates three key techniques: a Jump Selection Strategy that improves efficiency by selecting the most informative instances while filtering out less significant ones; Multi-Binning Scaling, which models different confidence levels separately to increase capacity and flexibility; and a Wasserstein distance-based calibration loss that further boosts calibration performance. Extensive experiments across multiple datasets demonstrate that KGEC consistently outperforms existing calibration methods in terms of both effectiveness and efficiency, making it a promising solution for calibration in KGE models[1].

1 Introduction

Knowledge graphs (KGs) are essential resources for a wide range of knowledge-driven tasks, including semantic search (Xiong et al., 2017), knowledge reasoning (Liu et al., 2021), question answering (Shen et al., 2019; Ye et al., 2023), and Neuro-Symbolic AI (Yang and Curry, 2025). Prominent large-scale KGs such as YAGO (Suchanek et al., 2007), DBpedia (Lehmann et al., 2015), and Freebase (Bollacker et al., 2008) encompass millions of entities and hundreds of millions of relational facts, which are typically structured as sets of *<head entity, relation, tail entity>* triples.

Query: (Greece, _member_of_domain_region, ?)	
True answer: sibyl	
Ranked candidate entities	Uncalibrated scores
Greece	-0.1873
Holy_See	-0.2946
sibyl	-0.5992
Colosseum	-0.8017
Sistine_Chapel	-0.8683

Figure 1: Uncalibrated scores for a query from WN18RR (Dettmers et al., 2018) produced by the TransE model (Bordes et al., 2013). Although the correct entity ("sibyl") is highly ranked, the uncalibrated scores lack probabilistic interpretability, highlighting the need for calibration.

However, most KGs are incomplete due to extraction errors and limited input resources. This makes link prediction, also known as knowledge graph completion, crucial for inferring missing links and improving KG quality. To this end, knowledge graph embedding (KGE) models such as TransE (Bordes et al., 2013) and ComplEx (Trouillon et al., 2016) tackle this problem by learning latent representations of entities and relations to score the plausibility of candidate triples. Beyond link prediction, KGE models have demonstrated remarkable success across diverse applications, including entity alignment (Sun et al., 2018) and canonicalization (Shen et al., 2022).

While the ranking accuracy of KGE models has seen significant advancements, the critical issue of probability calibration remains largely overlooked. Specifically, KGE models should output calibrated probabilities alongside their predictions. How-

[1]Codes available at https://github.com/Yang233666/KGE-Calibrator

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 29964–29987
November 4-9, 2025

ever, they typically produce uncalibrated scores (Pezeshkpour et al., 2020; Tabacof and Costabello, 2020). This stems from link prediction being framed as a ranking task, where metrics like Mean Rank (MR) and HITS@N prioritize relative ordering while ignoring the reliability of output scores. As a result, KGE models can assign implausible scores to correct entities yet still perform well, as shown in Figure 1. This shortcoming limits their applications in high-stakes domains such as drug and protein target discovery (Zeng et al., 2022; Mohamed et al., 2020), where calibrated estimates are essential.

To address this critical issue, increasing attention has been directed toward the probability calibration task of KGE models, which aims to convert the uncalibrated scores assigned to candidate triples into well-calibrated probability estimates. As a post-processing technique, calibration improves the trustworthiness of link prediction results, making them more reliable for downstream applications. However, probability calibration in KGE poses unique challenges compared to traditional classification. Image classification datasets like CIFAR-100 (Krizhevsky et al., 2009) or document classification datasets like SST (Socher et al., 2013) involve tens or hundreds of classes. In contrast, KGE tasks treat each entity as a distinct class. This creates a massive class space, a challenge present even in standard benchmarks (e.g., FB15K and WN18RR contain 14,951 and 40,943 entities, respectively). This high cardinality yields extremely small per-class probabilities and makes calibration highly sensitive. On WN18RR, for instance, we empirically observe that 99.1% of uncalibrated scores produced by TransE fall below 10^{-4}, highlighting the dominance of near-zero values. Even small perturbations in such distributions can distort the original ranking and negatively affect link prediction performance. Therefore, preserving the original ranking quality becomes a critical requirement, posing a distinctive challenge for probability calibration in the KGE setting.

Despite its importance and unique challenges, probability calibration in KGE remains largely underexplored. Prior studies (Tabacof and Costabello, 2020; Pezeshkpour et al., 2020) have shown that popular KGE models produce poorly calibrated scores, resulting in unreliable probability estimates. Several off-the-shelf calibration methods, such as Platt Scaling, Isotonic Regression, and Temperature Scaling, have been evaluated (Safavi et al., 2020; Zhu et al., 2022), but these methods are designed for standard classifiers and are not well-suited to the scale and ranking-sensitive nature of KGE. A few works have explored calibration in specific tasks, including triple classification (Tabacof and Costabello, 2020), relation prediction (Safavi et al., 2020), and low-dimensional entity expit transformations (Wang et al., 2021). However, no existing approach offers a calibration method explicitly tailored to the probabilistic characteristics of KGE models. This leaves a critical gap in improving the trustworthiness of KGE-based link prediction.

To fill this gap, we propose KGE Calibrator (KGEC), the first probability calibration method tailored specifically for KGE models. To enhance training efficiency under the large-scale class space characteristic of KGE, we introduce the Jump Selection Strategy, which selects the most informative instances while discarding less significant ones. To increase model expressiveness and better capture the ranking-sensitive nature of KGE predictions, we propose Multi-Binning Scaling, which models different probability levels separately, thereby increasing model capacity and flexibility. Additionally, we propose a Wasserstein distance-based loss function to further boost calibration performance. To the best of our knowledge, this is the first use of the Wasserstein distance for probability calibration.

Contributions. Our major contributions can be summarized as follows:

• We demonstrate that five of nine widely-used post-hoc calibration methods degrade link prediction performance for KGE entity prediction, indicating they are unsuitable in this setting.

• We propose KGEC, the first probability calibration method specifically designed for KGE models, which addresses the challenge of large class space in calibration while preserving the original ranking performance.

• A thorough experimental study over four datasets demonstrates that KGEC consistently outperforms existing calibration methods in both performance and efficiency.

2 Related Work

Probability Calibration in KGE Models. Several studies have highlighted that KGE models produce poorly calibrated probability estimates. Early work by (Tabacof and Costabello, 2020) and (Pezeshkpour et al., 2020) showed that widely used KGE models are poorly calibrated in triple

classification tasks. To address this, (Tabacof and Costabello, 2020) applied Platt Scaling (Platt et al., 1999) and Isotonic Regression (Zadrozny and Elkan, 2002), while (Safavi et al., 2020) explored Matrix Scaling and Vector Scaling (Guo et al., 2017) in relation prediction. A broader evaluation by (Zhu et al., 2022) tested additional off-the-shelf calibration techniques, including Histogram Binning (Zadrozny and Elkan, 2001), Beta Calibration (Kull et al., 2017), and Temperature Scaling (Guo et al., 2017) for triple classification. Furthermore, (Rao, 2021) examined calibration under both closed-world and open-world assumptions. While these works shed light on the calibration issue in KGE, they all rely on existing techniques originally designed for traditional classification problems. None propose a calibration method specifically tailored for KGE models, leaving a critical gap in the literature. For completeness, we provide a summary of calibration methods explored in prior KGE studies in Table 4 (Appendix A.1).

Expit Transformations. Expit transformations aim to convert uncalibrated scores into probabilities using functions such as the Sigmoid (Nickel et al., 2015; Tabacof and Costabello, 2020; Zhu et al., 2022) and Softmax (Pezeshkpour et al., 2020). Other approaches include neighborhood intervention consistency (NIC) (Wang et al., 2021) and min-max scaling (Rao, 2021). However, recent research (Zhu et al., 2022) has shown that even when expit-transformed scores can be interpreted as probabilities, they are still uncalibrated and unreliable. As a result, these expit transformations are generally viewed as a preliminary step, typically followed by a dedicated calibration method such as Platt Scaling or Isotonic Regression. In fact, (Zhu et al., 2022) concluded that expit transformations are ineffective in most cases and suggested probability calibration as a better approach. Following this direction, our work focuses exclusively on probability calibration and does not include expit transformations as part of our method design.

3 Preliminaries

Notations. We use calligraphic font for sets (e.g., $\mathcal{E}$). Matrices are denoted by bold uppercase (e.g., $\mathbf{P} \in \mathbb{R}^{n \times m}$). Row and column vectors extracted from a matrix are bold lowercase (e.g., the i-th row $\mathbf{p}_i \in \mathbb{R}^{1 \times m}$ and the j-th column $\mathbf{p}^j \in \mathbb{R}^{n \times 1}$). Standalone vectors (e.g., an embedding $\mathbf{h}$ or a vector of probability estimates $\mathbf{p}$) are also bold lowercase. The vector of calibrated estimates is distinguished with a hat (e.g., $\hat{\mathbf{p}}$). Scalars are denoted by plain italic letters (e.g., n, m).

Knowledge Graph. A knowledge graph (KG) $\mathcal{G}$ consists of a set of triples (h, r, t), where each triple includes a head entity $h \in \mathcal{E}$, a tail entity $t \in \mathcal{E}$, and a relation $r \in \mathcal{R}$ connecting head and tail. Here, $\mathcal{E}$ and $\mathcal{R}$ refer to the sets of entities and relations of $\mathcal{G}$ respectively, and $m = |\mathcal{E}|$ denotes the total number of entities.

Knowledge Graph Embeddings. Knowledge graph embedding (KGE) models aim to represent each head entity h, relation r, and tail entity t from a KG $\mathcal{G}$ as d-dimensional continuous embeddings $\mathbf{h}$, $\mathbf{r}$, and $\mathbf{t} \in \mathbb{R}^d$. A core component of the KGE model is its score function ψ, which evaluates the plausibility of a triple (h, r, t) by computing a compatibility score $\psi(\mathbf{h}, \mathbf{r}, \mathbf{t})$ from the corresponding embeddings. Table 10 in Appendix F lists the score functions of the most widely used KGE models.

Link Prediction. Link prediction, the primary downstream task for KGE models, encompasses both entity prediction and relation prediction. Entity prediction is generally more challenging due to the large number of candidate entities. For example, the widely used WN18RR (Dettmers et al., 2018) dataset contains 40,943 entities but only 11 relations. In this paper, we focus on the more challenging entity prediction task, which includes both head and tail prediction.

For head prediction, given a query $(?, r, t)$, each entity $e_i \in \mathcal{E}$ is treated as a candidate for the missing head entity. The trained KGE model assigns a score $\psi(\mathbf{e}_i, \mathbf{r}, \mathbf{t})$ to each candidate triple (e_i, r, t), where e_i is a candidate head entity, and r and t are the given relation and tail entity. These scores are then ranked, with higher-ranked triples considered more plausible, indicating that the corresponding entity e_i is a likely answer to the query $(?, r, t)$. The task of tail prediction is defined analogously for queries of the form $(h, r, ?)$.

KGE Probability Calibration. Given a head-entity query $(?, r, t)$, a KGE model with score function ψ first produces an uncalibrated score vector over all m entities:

$$\mathbf{s} = [\psi(\mathbf{e}_1, \mathbf{r}, \mathbf{t}), \ldots, \psi(\mathbf{e}_m, \mathbf{r}, \mathbf{t})]^\top \in \mathbb{R}^m, \quad (1)$$

where $\mathbf{s}$ is the uncalibrated score vector. These scores are typically converted into an initial vector of uncalibrated probability estimates $\mathbf{p}$ using an expit transformation such as the Softmax function (σ_{SM}), where $\mathbf{p} = \sigma_{\text{SM}}(\mathbf{s})$. However, these

Table 1: Effect of calibration on TransE's ranking performance (FB15K). Lower MR indicates better performance; higher MRR and HITS@K are better. ↓ denotes performance degradation compared to the uncalibrated (Uncal) baseline. Methods shown are Platt Scaling (PS), Histogram Binning (HB), Isotonic Regression (IR), Bayesian Binning into Quantiles (BBQ), Vector Scaling (VS), Matrix Scaling (MS), Temperature Scaling (TS), Meta-Cal, Parametrized Temperature Scaling (PTS), and our proposed KGEC.

Method	*MR*	*MRR*	*HITS@1*	*HITS@3*	*HITS@10*
		FB15K			
Uncal	40	0.731	0.646	0.793	0.865
PS	40	0.731	0.646	0.793	0.865
HB	2275 ↓	0.570 ↓	0.510 ↓	0.614 ↓	0.670 ↓
IR	982 ↓	0.615 ↓	0.530 ↓	0.675 ↓	0.761 ↓
BBQ	1275 ↓	0.589 ↓	0.509 ↓	0.646 ↓	0.726 ↓
VS	41 ↓	0.730 ↓	0.646	0.791 ↓	0.862 ↓
MS	3687 ↓	0.038 ↓	0.024 ↓	0.039 ↓	0.061 ↓
TS	40	0.731	0.646	0.793	0.865
Meta-Cal	1149 ↓	0.677 ↓	0.604 ↓	0.735 ↓	0.787 ↓
PTS	40	0.731	0.646	0.793	0.865
KGEC	40	0.731	0.646	0.793	0.865

estimates often provide poor reflections of the true likelihoods (Zhu et al., 2022).

The goal of probability calibration is to learn a mapping that transforms the uncalibrated estimates $\mathbf{p}$ into calibrated estimates $\hat{\mathbf{p}}$, such that each element $\hat{p}_i$ more faithfully reflects the likelihood of correctness for the i-th candidate. Addressing this challenge for KGE models is the primary focus of this work. After calibration, the predicted answer $\hat{y}$ to the query and its associated calibrated confidence are obtained as:

$$\hat{y} = \arg\max(\hat{\mathbf{p}}), \quad \hat{p} = \max(\hat{\mathbf{p}}), \qquad (2)$$

where $\hat{y}$ denotes the most likely entity and $\hat{p}$ quantifies the calibrated confidence in this prediction.

Calibration Method Evaluation. To motivate our work, we first demonstrate that preserving ranking performance is a non-trivial requirement that many standard calibration methods fail to meet. We evaluate a set of widely used post-processing methods[2] in the context of entity prediction. Specifically, we consider nine representative methods and examine whether they preserve or degrade the ranking performance of KGE models after calibration.

Table 1 presents the results of applying these calibration methods to the TransE model on the FB15K dataset. The results reveal the following observations: (1) HB, IR, BBQ, MS, and Meta-Cal substantially degrade performance, making them unsuitable as calibrators for KGE models in the entity prediction task; (2) PS, TS, and KGEC successfully preserve the original ranking performance, demonstrating their suitability for this task; (3) VS slightly degrades performance on FB15K, but given that the decline is minor, so its overall impact remains acceptable.

These findings suggest that not all well-known calibration methods are compatible with KGE-based entity prediction. For a more comprehensive view, the detailed results for additional KGE models across multiple datasets are reported in Tables 5–8 in Appendix B.

[2]Brief descriptions of these calibration methods are provided in Appendix A.2.

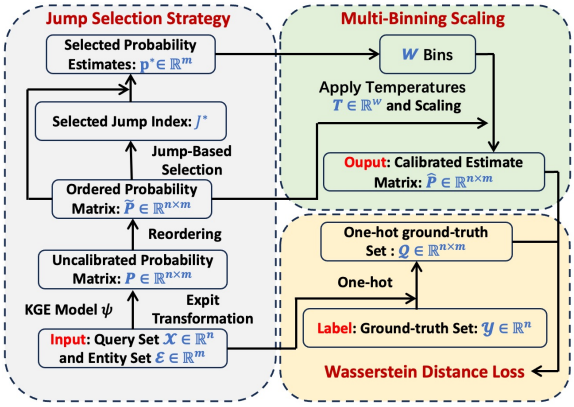

Figure 2: An illustration of the proposed KGEC method.

4 KGE Calibrator

Figure 2 presents an overview of our proposed KGE Calibrator (KGEC). We begin by describing the Jump Selection Strategy and Multi-Binning Scaling, followed by the Wasserstein distance-based loss function.

4.1 Jump Selection Strategy

Calibrating KGE models is challenging due to their extremely large class spaces: each query involves thousands of candidate entities, resulting in long-tailed probability estimates dominated by near-zero values. Directly using all candidates for calibration training is both computationally prohibitive and highly redundant. Our intuition is that calibration for KGE models requires selecting a small but informative subset of candidates for training. Building on this idea and inspired by the Log-Jump algorithm (Shen et al., 2022), we propose the **Jump Selection Strategy (JSS)**, which retains only the most informative instance per query identified via the *Jump measure* (Sugar and James, 2003), while discarding others. This reduces training size dra-

Algorithm 1 Jump Selection Strategy (JSS)

Input: Query set $\mathcal{X} = \{x_i = (?, r_i, t_i)\}_{i=1}^{n}$, candidate entities $\mathcal{E} = \{e_j\}_{j=1}^{m}$, trained KGE model ψ.

Output: Most informative column index J^* and vector of probability estimates $\mathbf{p}^* \in \mathbb{R}^n$.

1: **for** $i = 1, \ldots, n$ **do**
2: Compute scores: $\mathbf{s}_i \leftarrow [\psi(\mathbf{e}_j, \mathbf{r}_i, \mathbf{t}_i)]_{j=1}^{m}$.
3: Convert to probabilities: $\mathbf{p}_i \leftarrow \sigma_{\mathrm{SM}}(\mathbf{s}_i)$.
4: **end for**
5: Stack row vectors to form probability matrix $\mathbf{P} \leftarrow [\mathbf{p}_1, \ldots, \mathbf{p}_n]^\top \in \mathbb{R}^{n \times m}$.
6: Sort each row of $\mathbf{P}$ in descending order to obtain $\tilde{\mathbf{P}}$.
7: **for** $j = 1, \ldots, m-1$ **do**
8: $J_j \leftarrow D_{\mathrm{KL}}(\tilde{\mathbf{p}}^j \parallel \tilde{\mathbf{p}}^{j+1})$.
9: **end for**
10: $J^* \leftarrow \arg\max_j J_j$.
11: $\mathbf{p}^* \leftarrow \tilde{\mathbf{p}}^{J^*}$.

matically without sacrificing essential informativeness.

Principle of Informativeness. The core idea is that the most informative instance lies at the *transition point* between highly informative and less informative candidates. When candidate scores are ranked, the resulting distribution typically exhibits a steep drop from a few high-probability candidates to a long tail of near-zero values. Selecting the instance at this sharp transition ensures that the retained example captures the highest degree of informativeness. We formalize this intuition using the *Jump measure* to quantify informativeness.

Method. The JSS procedure is detailed in Algorithm 1. Given a set of queries $\mathcal{X} = \{x_i\}_{i=1}^{n}$ and candidate entities $\mathcal{E} = \{e_j\}_{j=1}^{m}$, the procedure begins by iterating through each query. For each query x_i, the trained KGE model ψ first produces a vector of uncalibrated scores $\mathbf{s}_i \in \mathbb{R}^m$ over all candidate entities (line 2). This score vector is then converted into a vector of probability estimates $\mathbf{p}_i$ using the Softmax function σ_{SM} (line 3).

After processing all queries, the resulting probability vectors $\{\mathbf{p}_i\}$ are stacked as rows to form a single matrix $\mathbf{P} \in \mathbb{R}^{n \times m}$ (line 5). To make confidence transitions explicit, each row of $\mathbf{P}$ is then sorted in descending order to yield a new matrix $\tilde{\mathbf{P}}$ (line 6). This reordering does not affect downstream performance. Link prediction metrics depend only on relative ranks, and the following calibration operates on the full vector of probability estimates, regardless of order.

To quantify informativeness and detect the most significant transition point, we compute the Kullback–Leibler (KL) divergence between consecutive columns of $\tilde{\mathbf{P}}$, i.e., $\tilde{\mathbf{p}}^j$ and $\tilde{\mathbf{p}}^{j+1}$, as the *Jump measure* J_j (lines 7–9). KL divergence is employed here not as a loss function, but as a measure of the informativeness shift between ranked adjacent columns. Finally, the column index J^* that maximizes this *Jump measure* is selected, and its corresponding vector of probability estimates $\mathbf{p}^* \in \mathbb{R}^n$ is retained as the most informative vector for subsequent calibration training (lines 10–11).

By transforming the training instances from a full probability matrix $\mathbf{P} \in \mathbb{R}^{n \times m}$ into a single informative column vector $\mathbf{p}^* \in \mathbb{R}^n$, JSS reduces the number of training instances by a factor of m without discarding critical information. A detailed discussion and theoretical analysis of potential information loss are provided in Appendix C.

4.2 Multi-Binning Scaling

An effective post-hoc calibrator for KGE models must satisfy two properties: it should be expressive enough to correct complex miscalibration patterns while strictly preserving the model's original ranking order, as metrics like MRR and HITS@K are paramount. Temperature Scaling (TS) (Guo et al., 2017), a widely used method, perfectly satisfies the second property by applying a single scalar temperature $T > 0$ to the logits. However, its simplicity comes at the cost of limited expressiveness: TS applies the same transformation to all probability estimates regardless of magnitude (e.g., scaling probabilities of 0.1 and 0.9 identically), making it inadequate for calibrating the highly non-uniform confidence distributions typical of KGE models. A single global parameter is often too restrictive to capture the nuanced calibration required across different confidence levels.

To address this limitation, we introduce **Multi-Binning Scaling (MBS)**, a highly expressive and flexible approach that inherits the rank-preserving benefit of TS while improving calibration quality. Inspired by histogram binning (Zadrozny and Elkan, 2001), the core idea of MBS is to partition the confidence space into multiple segments and learn a separate temperature for each, allowing the model to apply different transformations to different confidence levels.

Concretely, we partition the interval $[0, 1]$ into

W disjoint, equal-width bins, $B_1, \ldots, B_W$, and associate each bin with an independent trainable temperature parameter $T_w > 0$. For a given query i, we use the single most informative probability, p_i^*, identified by our Jump Selection Strategy (Section 4.1), to select the appropriate bin. The associated temperature is squared and inverted to yield a bin-specific scaling factor $1/T_w^2$, which then multiplies the entire vector of probability estimates $\tilde{\mathbf{p}}_i$, uniformly rescaling estimates across all candidate entities. For example, with $W = 10$ bins, a probability of $p_i^* = 0.75$ falls into the bin $B_8 = (0.7, 0.8]$, and its corresponding temperature T_8 determines the scaling factor applied to all candidate entities for that query.

This simple multiplicative rescaling is crucial: it guarantees ranking preservation while adaptively modulating confidence levels across bins. Squaring the temperature stabilizes optimization by smoothing gradients, ensuring positive scaling, and preventing excessively sharp updates. While we adopt equal-width bins for simplicity, more advanced strategies, such as adaptive or data-driven binning, represent a promising extension. Ultimately, MBS is highly efficient, as its complexity depends only on the number of bins, requiring the training of just W scalar temperature parameters. By combining the rank-preserving property of TS with the enhanced expressiveness of a bin-based transformation, MBS offers a principled and scalable solution for KGE model calibration.

4.3 Optimization

While KL divergence is a commonly used loss function in deep learning, it poses notable limitations for calibration in KGE models, such as gradient instability and explosion[3]. To address these issues, we propose using the Wasserstein distance as the loss function for KGEC. Unlike KL divergence, the Wasserstein distance provides a more stable and geometrically meaningful way to compare confidence distributions by considering the minimum cost of transforming one distribution into another. This perspective is especially valuable in calibration, where we aim to align the calibrated estimates with ground-truth probability distributions while preserving their structure.

The Wasserstein distance models calibration as an optimal transport (OT) problem. As the loss function requires a probability distribution, we first apply a Softmax transformation to the rescaled estimates from Multi-Binning Scaling (Section 4.2), yielding the calibrated probability distribution $\hat{\mathbf{p}}_i$. The goal of OT problem is to find the most efficient way to move mass from this calibrated probability distribution $\hat{\mathbf{p}}_i \in \mathbb{R}^m$ to the ground-truth one-hot[4] distribution $\mathbf{q}_i \in \{0, 1\}^m$. The feasible set of transport plans is defined by the transportation polytope $U(\hat{\mathbf{p}}_i, \mathbf{q}_i)$, which contains all nonnegative transport matrices $\mathbf{P} \in \mathbb{R}_+^{m \times m}$:

$$U(\hat{\mathbf{p}}_i, \mathbf{q}_i) = \{\mathbf{P} \in \mathbb{R}_+^{m \times m} | \mathbf{P}\mathbf{1}_m = \hat{\mathbf{p}}_i, \mathbf{P}^\top \mathbf{1}_m = \mathbf{q}_i\}, \tag{3}$$

where $\mathbf{1}_m \in \mathbb{R}^m$ is the vector of ones.

Given a cost matrix $\mathbf{M} \in \mathbb{R}^{m \times m}$, the Wasserstein distance is defined as the minimum transport cost required to map $\hat{\mathbf{p}}_i$ to $\mathbf{q}_i$ using the transport matrix $\mathbf{P}$.

$$\begin{aligned} D_{WD}(\hat{\mathbf{p}}_i, \mathbf{q}_i) &= \min_{\mathbf{P} \in U(\hat{\mathbf{p}}_i, \mathbf{q}_i)} \langle \mathbf{P}, \mathbf{M} \rangle \\ &= \sum_j^m \sum_l^m \mathbf{P}_j^l \mathbf{M}_j^l, \end{aligned} \tag{4}$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius dot-product and $\mathbf{M}_j^l = \left| \hat{\mathbf{p}}_i^j - \mathbf{q}_i^l \right|$ represents the absolute difference between the j-th and l-th elements of $\hat{\mathbf{p}}_i$ and $\mathbf{q}_i$.

To improve computational efficiency, we use the Sinkhorn distance (Cuturi, 2013), which provides a fast approximation to the constrained Wasserstein distance by introducing entropy regularization. Given the OT plan $\mathbf{P}^\lambda$ and cost matrix $\mathbf{M}$, the Sinkhorn distance D_{SD} is defined as follows:

$$D_{SD}(\hat{\mathbf{p}}_i, \mathbf{q}_i) = \left\langle \mathbf{P}^\lambda, \mathbf{M} \right\rangle, \tag{5}$$

where $\lambda > 0$ is the weight for entropy regularization. The OT plan $\mathbf{P}^\lambda$ is obtained by solving:

$$\mathbf{P}^\lambda = \underset{\mathbf{P} \in U(\hat{\mathbf{p}}_i, \mathbf{q}_i)}{\arg\min} \langle \mathbf{P}, \mathbf{M} \rangle - \frac{1}{\lambda} h(\mathbf{P}), \tag{6}$$

where $h(\mathbf{P})$ is the entropy of $\mathbf{P}$. The solution $\mathbf{P}^\lambda$ is computed iteratively via Sinkhorn normalization (Cuturi, 2013) as follows:

$$\begin{aligned} \mathbf{u}^{(t)} &= \hat{\mathbf{p}}_i \oslash \left(\mathbf{K}^\top \mathbf{v}^{(t-1)}\right), \\ \mathbf{v}^{(t)} &= \mathbf{q}_i \oslash \left(\mathbf{K}\, \mathbf{u}^{(t)}\right), \end{aligned} \tag{7}$$

where $\oslash$ indicates element-wise division, (t) denotes the iteration time, and $\mathbf{K} = \exp(-\frac{\mathbf{M}}{\lambda})$ is the

[3] A detailed analysis is provided in Appendix D.

[4] Here, $\mathbf{q}_i$ is a one-hot vector where $\mathbf{q}_i^j = 1$ for the correct entity j, and $\mathbf{q}_i^l = 0$ for all $l \neq j$.

kernel matrix with entropy regularization weight λ. Finally, the optimal transport plan $\mathbf{P}^{\lambda}$ is given by:

$$\mathbf{P}^{\lambda} = \mathrm{diag}\big(\mathbf{v}^{(t)}\big)\,\mathbf{K}\,\mathrm{diag}\big(\mathbf{u}^{(t)}\big). \tag{8}$$

This Sinkhorn-regularized Wasserstein loss enables more stable optimization and improves calibration performance, particularly in the large class-space settings typical of KGE tasks.

5 Experiments

We structure our experimental study to answer three key research questions (RQs): **RQ1**: Can KGEC outperform existing calibration methods? **RQ2**: Is KGEC efficient in terms of training time and memory usage? **RQ3**: What is the contribution of each of its components? We first detail our experimental setting (Section 5.1), then address each RQ in turn (Sections 5.2–5.4), and conclude with a sensitivity analysis and case study.

5.1 Experimental Setting

5.1.1 Datasets

We evaluate our proposed model on four popular datasets, which are commonly used to evaluate link prediction, where FB15K (Bordes et al., 2013) and FB15K-237 (Toutanova and Chen, 2015) were extracted from Freebase (Bollacker et al., 2008), WN18 (Bordes et al., 2013) and WN18RR (Dettmers et al., 2018) were extracted from WordNet (Miller, 1995). Note that FB15K-237 and WN18RR are subsets of FB15K and WN18, respectively, in which near-same and near-reverse relations have been removed. These datasets are publicly available, and already partitioned into training, validation and testing splits. The statistics of them are summarized into Table 9 in Appendix E.

5.1.2 KGE Models

We evaluate KGEC on four well-established KGE models: TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), and RotatE (Sun et al., 2019). Their score functions are listed in Table 10 in the Appendix F. It is noted that our proposed method KGEC is model-agnostic, as it can be applied to any KGE model that produces a score for each candidate triple. We therefore leave the evaluation of KGEC on other KGE architectures for future work.

5.1.3 Evaluation Measures

To evaluate calibration performance, we adopt three widely used evaluation metrics: Expected Calibration Error (ECE) (Naeini et al., 2015), Adaptive Calibration Error (ACE) (Nixon et al., 2019), and Negative Log-Likelihood (NLL). Each metric captures different aspects of calibration quality. Due to space constraints, we refer readers to (Naeini et al., 2015; Nixon et al., 2019) for detailed formulations. For an overall comparison, we report the **Average** performance by averaging each metric across all datasets and KGE models.

5.1.4 Setting Details

To ensure a fair comparison, all calibration baselines[5] and metrics we used are from third-party frameworks or their original implementations. Specifically, the code of PS, HB, IR, BBQ, and TS is from the net:cal library[6]. The code of MS and VS, as well as all calibration metrics, is provided by TorchUncertainty[7]. The code of Meta-Cal[8] and PTS[9] is from their official repositories. For the hyperparameter setting of KGEC, the number of bins is set to 10, the learning rate to 0.01, the batch size to 32, the initial temperature for each bin to 1.0, and the optimizer is AdamW (Loshchilov and Hutter, 2019). Except for VS, MS, and TS, which use the *Multiclass* setting, all other baselines use the *One-vs-All* setting to avoid prohibitive training time. We follow the closed world assumption in our experiments, since the open world assumption requires a label for each triple, which is not available in existing datasets. All reported results are averaged over 10 independent runs.

5.2 Effectiveness Study for RQ1

Table 2 presents the calibration performance of various methods across multiple KGE models and datasets. Notably, baselines such as HB, IR, BBQ, MS, and Meta-Cal are excluded, due to their detrimental impact on ranking performance, as evidenced in Table 1 (Section 3). Since preserving the original ranking order is essential in KGE settings, these calibration methods that degrade ranking performance are considered unsuitable for practical deployment and omitted from further evaluation.

Overall, KGEC consistently outperforms all competitive baselines, achieving the lowest average ECE, ACE, and NLL across all datasets and models.

[5] Due to space limitations, detailed descriptions of the calibration baselines are deferred to Appendix A.2.

[6] https://efs-opensource.github.io/calibration-framework/build/html/index.html

[7] https://torch-uncertainty.github.io

[8] https://github.com/maxc01/metacal/tree/master

[9] https://github.com/tochris/pts-uncertainty

Table 2: Effect of different calibration methods on the performance of various KGE models across multiple datasets. Best and second-ranked results are in bold and underlined, respectively. For ECE, ACE, and NLL, lower values indicate better calibration performance.

ECE	***TransE***				***ComplEx***				***DistMult***				***RotatE***				***Average***
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
Uncal	0.502	0.265	0.580	0.212	0.852	0.424	0.696	0.228	0.528	0.389	0.694	0.221	0.429	0.385	0.684	0.224	0.457
PS	0.634	0.031	0.530	0.218	0.854	0.427	0.701	0.229	0.529	0.394	0.700	0.222	0.876	0.425	0.722	0.235	0.483
VS	0.706	0.014	0.646	0.231	0.852	0.424	0.697	0.228	0.528	0.389	0.695	0.215	0.944	0.413	0.739	0.239	0.498
TS	0.634	0.031	0.680	0.203	0.852	0.424	0.701	0.228	0.528	0.389	0.700	0.221	0.687	0.384	0.722	0.223	0.475
PTS	0.523	0.013	0.530	0.231	0.854	0.430	0.060	0.214	0.456	0.393	0.526	0.778	0.337	0.425	0.221	0.365	0.397
KGEC	0.171	0.280	0.459	0.150	0.833	0.418	0.678	0.189	0.446	0.383	0.683	0.178	0.467	0.307	0.466	0.094	**0.388**

ACE	***TransE***				***ComplEx***				***DistMult***				***RotatE***				***Average***
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
Uncal	0.506	0.274	0.565	0.180	0.852	0.424	0.696	0.228	0.528	0.389	0.694	0.220	0.429	0.385	0.684	0.224	0.455
PS	0.628	0.033	0.530	0.217	0.854	0.427	0.701	0.229	0.529	0.394	0.700	0.222	0.876	0.425	0.722	0.235	0.483
VS	0.506	0.274	0.565	0.180	0.852	0.424	0.697	0.228	0.528	0.389	0.694	0.215	0.429	0.385	0.684	0.224	0.455
TS	0.628	0.033	3.312	0.154	0.852	0.423	0.701	0.228	0.528	0.389	0.700	0.220	0.687	0.384	0.722	0.222	0.636
PTS	0.516	0.013	0.530	0.231	0.854	0.424	0.060	0.207	0.446	0.391	0.522	0.778	0.337	0.418	0.221	0.363	0.394
KGEC	0.131	0.277	0.293	0.082	0.833	0.418	0.465	0.207	0.457	0.383	0.516	0.199	0.467	0.306	0.466	0.063	**0.348**

NLL	***TransE***				***ComplEx***				***DistMult***				***RotatE***				***Average***
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
Uncal	2.891	6.582	3.911	5.396	6.892	7.815	5.954	7.513	7.447	7.858	5.919	7.705	1.376	6.145	4.090	5.750	5.828
PS	3.839	7.304	3.829	5.836	8.831	8.974	7.093	8.438	9.117	9.065	7.257	8.621	3.350	7.364	4.799	6.271	6.874
VS	/	/	/	/	6.892	7.814	5.952	7.510	7.446	7.857	5.916	7.692	1.376	/	/	/	6.495
TS	3.839	7.304	1.285	4.909	6.892	7.802	7.093	7.513	7.447	7.856	7.257	7.704	2.069	6.121	4.799	5.617	5.969
PTS	/	9.181	3.829	9.448	9.314	9.171	1.906	5.714	/	9.496	4.847	/	/	/	/	/	6.990
KGEC	2.462	5.965	2.536	2.889	4.350	6.965	1.357	2.911	2.843	7.119	1.319	3.106	1.036	4.698	2.033	2.743	**3.396**

Table 3: Training time in seconds and memory usage in MBs taken to calibrate entity prediction using different calibration methods. Best and second-ranked results are in bold and underlined, respectively. For a fair comparison, these results are obtained using CPU only.

Method	Average Time	Average Memory
PS	40856.945	2542.715
VS	7.577	83.294
TS	8.649	2540.274
PTS	7035.177	8410.493
KGEC	**4.716**	**21.665**

A breakdown of these results reveals several key findings: (1) *Limited effectiveness of simple baselines*. PS, VS, and TS often perform worse than the uncalibrated models. Their poor performance is likely due to their low model capacity, which is insufficient to capture complex calibration patterns in high-cardinality KGE settings. (2) *Improved results with PTS*. PTS shows marked improvement over simple baselines by predicting temperature parameters adaptively using a neural network. This flexibility enables better handling of distributional variation, leading to improved performance. (3) *Superior performance of KGEC*. KGEC achieves the best overall results across all metrics and datasets. Together, these findings confirm that KGEC effectively addresses the unique challenges of KGE calibration while preserving ranking quality.

5.3 Efficiency Study for RQ2

Table 3 reports the average training time and memory usage of different calibration methods across multiple KGE models and datasets. To ensure a fair comparison, all methods are evaluated on CPU-only environments. Detailed experimental results for each calibration method on individual datasets and KGE models are presented in Table 11 (Appendix G).

Key Observations from Table 3: (1) KGEC is the most efficient model in both training time and memory usage, consistently outperforming all baseline methods. (2) VS and TS exhibit comparable efficiency, with slightly longer training times than KGEC, which can be attributed to their simple parametric structures. (3) PTS incurs significantly higher computational costs, both in time and memory, despite its strong calibration performance. This high overhead may limit its applicability in large-scale or resource-constrained scenarios. (4) PS is the slowest method, largely due to the immense number of classes in KGE settings, which makes binary logistic regression computationally expensive.

5.4 Ablation Study for RQ3

To assess the individual contribution of each component in KGEC, we perform a comprehensive ablation study across five key metrics: ECE, ACE, NLL, training time, and memory usage. We evaluate the following four variants: (1) KGEC: The full model, incorporating all components: Jump Selection Strategy (JSS), Multi-Binning Scaling (MBS), and the Wasserstein distance-based loss. (2) KGEC-loss: Replaces the Wasserstein loss with KL divergence while retaining JSS and MBS. (3) KGEC-loss-MBS: Further removes MBS, retaining

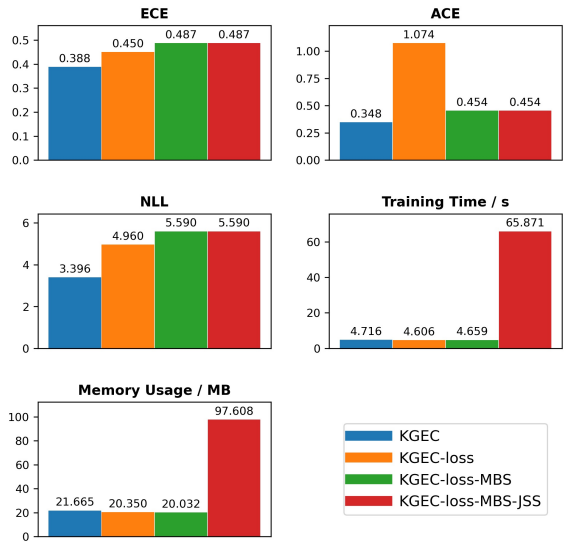

Figure 3: Ablation study of KGEC components across five evaluation metrics: ECE, ACE, NLL, training time (seconds), and memory usage (MB). Lower values indicate better performance.

only JSS and KL divergence. (4) KGEC-loss-MBS-JSS: The base version using only KL divergence, without any of the proposed enhancements. Figure 3 reports the average performance across all datasets and KGE models, providing an overall comparison of model variants. Detailed experimental results for each component on individual datasets and KGE models are presented in Table 12 (Appendix H).

Key Observations: (1) *Full Model Superiority.* KGEC achieves the best performance across all five metrics. It yields the lowest calibration errors (ECE = 0.388, ACE = 0.348, NLL = 3.396) while maintaining high efficiency (training time = 4.716s, memory usage = 21.665MB). (2) *Impact of Wasserstein Loss.* Comparing KGEC to KGEC-loss reveals substantial calibration improvements, validating the advantage of using Wasserstein distance over KL divergence in high-cardinality, ranking-sensitive KGE settings. This supports our hypothesis that the Wasserstein-based objective is better suited to the probability distribution landscape of KGE. (3) *Effect of MBS.* Removing MBS (KGEC-loss vs. KGEC-loss-MBS) degrades ECE (from 0.450 to 0.487) and NLL (from 4.960 to 5.590), indicating that MBS enhances calibration by modeling probability intervals more effectively. Interestingly, ACE improves after removing MBS. This anomaly may arise because the KL divergence used in KGEC-loss amplifies ACE more than expected, suggesting ACE is especially sensitive to the choice of loss function. (4) *Efficiency Gain from JSS.* While KGEC-loss-MBS and KGEC-loss-MBS-JSS exhibit similar calibration performance, the inclusion of JSS dramatically reduces training time (from 65.871s to 4.659s) and memory usage (from 97.608MB to 20.032MB), confirming JSS's effectiveness in improving computational efficiency. Furthermore, a direct comparison against a random sampling baseline shows JSS is not only more efficient but also significantly more effective at preserving calibration quality (see Appendix H).

Overall, all three components are essential for balancing calibration performance and computational cost. MBS and Wasserstein loss enhance calibration performance, while JSS ensures efficiency and stability. The full KGEC model thus delivers the strongest and most balanced performance.

5.5 Sensitivity Analysis and Case Study

To further assess the robustness and practical utility of KGEC, we conducted a sensitivity analysis on its key hyperparameters and a qualitative case study. Our sensitivity analysis, which varies three hyperparameters: the number of bins, initial temperature, and learning rate, confirms that KGEC's performance is stable across a wide range of settings. Additionally, the case study provides concrete examples of how the method corrects miscalibrated predictions, addressing both over- and under-confidence from base models. A detailed presentation is shown in Appendix I and Appendix J.

6 Conclusion

In this paper, we propose KGEC, the first probability calibration method tailored to the unique challenges of KGE models. By integrating a novel Jump Selection Strategy for efficiency, a Multi-Binning Scaling module for expressiveness, and a Wasserstein distance-based loss for stable optimization, KGEC effectively calibrates KGE predictions while strictly preserving their ranking performance. Comprehensive experiments across multiple KGE models and datasets demonstrate that KGEC significantly outperforms existing calibration baselines in both calibration performance and computational efficiency. Our work establishes a strong foundation for trustworthy link prediction, and future work may explore extensions to dynamic KGs or integration with uncertainty-aware reasoning systems.

Limitations

While KGEC achieves strong performance, we identify several promising directions for future work based on its current limitations:

(1) Alternative Expit Transformations. In this work, we adopt the Softmax function as the expit transformation, as our primary focus is on the calibration method itself. However, alternative approaches, such as NIC (Wang et al., 2021) and min-max normalization (Rao, 2021), may further improve performance and merit exploration in future work.

(2) Task-Specific Calibration Considerations. KGEC is optimized for static entity prediction tasks in knowledge graphs. Its effectiveness in other KGE-based applications, such as multi-hop reasoning, fact verification, or temporal/dynamic KG settings, remains untested. These tasks may require adaptation or redesign of the calibration strategy to accommodate different data characteristics and evaluation protocols.

(3) Limited Evaluation Across Advanced KGE Architectures. While KGEC has been extensively evaluated on several representative KGE models (e.g., TransE, DistMult, ComplEx, and RotatE), its generalization to more complex architectures, such as hyperbolic embeddings, graph neural networks, or transformer-based KGE models, has not yet been studied. Extending KGEC to these settings poses challenges in modeling and scalability, and is an important direction for future work.

Acknowledgments

This publication has emanated from research conducted with the financial support of Taighde Éireann – Research Ireland under Grant number SFI/12/RC/2289 P2.

References

Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. 2019. Uncertainty-based continual learning with adaptive regularization. In *NeurIPS*, volume 32.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *EMNLP*, pages 615–620.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NeurIPS*, pages 2787–2795.

Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, volume 26.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *AAAI*, volume 32.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, pages 1050–1059. PMLR.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *ICML*, pages 1321–1330. PMLR.

Alex Krizhevsky et al. 2009. Learning multiple layers of features from tiny images.

Meelis Kull, Telmo Silva Filho, and Peter Flach. 2017. Beta calibration: a well-founded and easily implemented improvement on logistic calibration for binary classifiers. In *Artificial intelligence and statistics*, pages 623–631. PMLR.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, volume 30.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Chris Bizer. 2015. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6(2):167–195.

Lihui Liu, Boxin Du, Yi Ren Fung, Heng Ji, Jiejun Xu, and Hanghang Tong. 2021. Kompare: a knowledge graph comparative reasoning system. In *SIGKDD*, pages 3308–3318.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Xingchen Ma and Matthew B Blaschko. 2021. Meta-cal: Well-controlled post-hoc calibration by ranking. In *ICML*, pages 7235–7245. PMLR.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Sameh K Mohamed, Vít Nováček, and Aayah Nounu. 2020. Discovering protein drug targets using knowledge graph embeddings. *Bioinformatics*, 36(2):603–610.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, volume 29.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Jeremy Nixon, Michael W Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. 2019. Measuring calibration in deep learning. In *CVPR workshops*, volume 2.

Pouya Pezeshkpour, Yifan Tian, and Sameer Singh. 2020. Revisiting evaluation of knowledge base completion models. In *AKBC*.

John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

ZAishwarya Rao. 2021. Calibrating knowledge graphs. In *Rochester Institute of Technology*.

Tara Safavi, Danai Koutra, and Edgar Meij. 2020. Evaluating the calibration of knowledge graph embeddings for trustworthy link prediction. In *EMNLP*, pages 8308–8321.

Tao Shen, Xiubo Geng, Tao Qin, Daya Guo, Duyu Tang, Nan Duan, Guodong Long, and Daxin Jiang. 2019. Multi-task learning for conversational question answering over a large-scale knowledge base. In *EMNLP-IJCNLP*, pages 2442–2451.

Wei Shen, Yang Yang, and Yinan Liu. 2022. Multi-view clustering for open knowledge base canonicalization. In *SIGKDD*, pages 1578–1588.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*, pages 697–706.

Catherine A Sugar and Gareth M James. 2003. Finding the number of clusters in a dataset: An information-theoretic approach. *Journal of the American Statistical Association*, 98(463):750–763.

Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. 2018. Bootstrapping entity alignment with knowledge graph embedding. In *IJCAI*, volume 18.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *ICLR*.

Pedro Tabacof and Luca Costabello. 2020. Probability calibration for knowledge graph embedding models. In *ICLR*.

Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. 2019. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *NeurIPS*, volume 32.

Christian Tomani, Daniel Cremers, and Florian Buettner. 2022. Parameterized temperature scaling for boosting the expressive power in post-hoc uncertainty calibration. In *ECCV*, pages 555–569. Springer.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*, pages 2071–2080.

Kai Wang, Yu Liu, and Quan Z Sheng. 2021. Neighborhood intervention consistency: Measuring confidence for knowledge graph link prediction. In *IJCAI*, pages 2090–2096.

Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *WWW*, pages 1271–1279.

Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

Yang Yang and Edward Curry. 2024. Open knowledge base canonicalization: Techniques and challenges. In *Text2KG@ESWC*.

Yang Yang and Edward Curry. 2025. Neuro-symbolic techniques in open knowledge graph canonicalization. In *Handbook on Neurosymbolic AI and Knowledge Graphs*, pages 280–299. IOS Press.

Yang Yang, Wei Shen, Junfeng Shu, Yinan Liu, Edward Curry, and Guoliang Li. 2025. Cmvc+: a multi-view clustering framework for open knowledge base canonicalization via contrastive learning. *IEEE Transactions on Knowledge and Data Engineering*.

Qichen Ye, Bowen Cao, Nuo Chen, Weiyuan Xu, and Yuexian Zou. 2023. Fits: Fine-grained two-stage training for knowledge-aware question answering. In *AAAI*, volume 37, pages 13914–13922.

Bianca Zadrozny and Charles Elkan. 2001. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*, pages 609–616.

Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *SIGKDD*, pages 694–699.

Xiangxiang Zeng, Xinqi Tu, Yuansheng Liu, Xiangzheng Fu, and Yansen Su. 2022. Toward better drug discovery with knowledge graph. *Current opinion in structural biology*, 72:114–126.

Ruiqi Zhu, Fangrong Wang, Alan Bundy, Xue Li, Kwabena Nuamah, Lei Xu, Stefano Mauceri, and Jeff Z Pan. 2022. A closer look at probability calibration of knowledge graph embedding. In *IJCKG*, pages 104–109.

A Calibration Baselines

A.1 Calibration Techniques Explored in Existing KGE Literature

For completeness, Table 4 summarizes which calibration methods have been employed in prior studies of KGE calibration. This overview highlights that existing works all exclusively adapt off-the-shelf methods originally designed for standard classification tasks, underscoring the need for calibration techniques tailored specifically to the KGE setting.

A.2 Calibration Baselines Evaluated in Our Experiments

We summarize the post-hoc probability calibration baselines considered in this work:

• **Platt Scaling (PS)** (Platt et al., 1999) is a parametric calibration method that transforms the non-probabilistic outputs of a binary classifier into calibrated confidence scores.

• **Histogram Binning (HB)** (Zadrozny and Elkan, 2001) is a simple non-parametric approach that partitions predictions into mutually exclusive bins, assigning each bin a calibrated score.

• **Isotonic Regression (IR)** (Zadrozny and Elkan, 2002) generalizes HB by jointly optimizing both bin boundaries and predictions through a monotonic regression function.

• **Bayesian Binning into Quantiles (BBQ)** (Naeini et al., 2015) extends HB by applying Bayesian model averaging over multiple binning models.

• **Matrix Scaling (MS)** and **Vector Scaling (VS)** (Guo et al., 2017) are multi-class extensions of PS, using matrix and vector transformations, respectively.

• **Temperature Scaling (TS)** (Guo et al., 2017) is the simplest extension of PS, employing a single scalar temperature parameter $T > 0$ shared across all predictions.

• **Meta-Cal** (Ma and Blaschko, 2021) combines a bipartite-ranking model with selective classification to construct a more flexible calibration mapping.

• **Parametrized Temperature Scaling (PTS)** (Tomani et al., 2022) generalizes TS by computing prediction-specific temperatures, parameterized by a neural network.

In this study, we restrict our attention to *post-hoc probability calibration* methods, which adjust model outputs without altering the underlying KGE training process. This ensures that the original ranking of entities is preserved. Training-modifying approaches such as regularization (Ahn et al., 2019), ensembles (Lakshminarayanan et al., 2017), MC-dropout (Gal and Ghahramani, 2016), and mixup (Thulasidasan et al., 2019) fall outside our scope, as they fundamentally alter the embedding training procedure.

We also exclude **Beta Calibration** (Kull et al., 2017) due to its prohibitive computational cost. For example, even on the smallest dataset (WN18RR), it required over 60 hours to complete, making it infeasible for our large-scale experiments. Finally, we emphasize that this work focuses strictly on *probability calibration*. Expit transformation alternatives such as replacing the Softmax function with Sigmoid or NIC (Wang et al., 2021) are conceptually distinct and therefore not considered here.

B Effect of Different Calibration Methods Across Datasets

In this section, we systematically evaluate nine widely used post-hoc calibration methods on the entity prediction task across four benchmark datasets. The goal is to assess whether these calibration techniques can improve probabilistic reliability while preserving the ranking quality essential for knowledge graph completion.

We report results using standard link prediction metrics. Specifically, lower *Mean Rank (MR)* indicates better performance, while higher values of *Mean Reciprocal Rank (MRR)*, *HITS@1*, *HITS@3*, and *HITS@10* correspond to better ranking quality.

As shown in Table 5–8, several calibration methods, including HB, IR, BBQ, MS, and Meta-Cal, substantially degrade entity ranking performance. This suggests that these approaches disrupt the original link prediction scores after calibration and are therefore unsuitable for KGE-based entity prediction.

Table 4: Overview of calibration methods employed in prior KGE calibration studies. Each method is marked as parametric or non-parametric, along with the corresponding references.

Calibration Method	Parametric	Used in Works
Isotonic Regression (Zadrozny and Elkan, 2002)	No	(Tabacof and Costabello, 2020), (Wang et al., 2021), (Zhu et al., 2022)
Histogram Binning (Zadrozny and Elkan, 2001)	No	(Zhu et al., 2022)
Beta Calibration (Kull et al., 2017)	Yes	(Zhu et al., 2022)
Platt Scaling (Platt et al., 1999)	Yes	(Tabacof and Costabello, 2020), (Wang et al., 2021), (Zhu et al., 2022)
Matrix Scaling (Guo et al., 2017)	Yes	(Safavi et al., 2020)
Vector Scaling (Guo et al., 2017)	Yes	(Safavi et al., 2020)
Temperature Scaling (Guo et al., 2017)	Yes	(Zhu et al., 2022)

C Detailed Description of the Jump Selection Strategy (JSS)

In the main paper (Section 4.1), we introduced the **Jump Selection Strategy** (JSS) as a principled method to reduce the size of calibration training data while preserving the essential informativeness of the retained samples. Here, we provide a more detailed exposition of the motivation, intuition, and theoretical considerations behind JSS.

C.1 Challenge of Large Class Spaces.

A core difficulty in calibrating KGE models lies in the extremely large candidate space. Each query may involve thousands of possible entities, resulting in probability distributions dominated by a long tail of near-zero values. Using all candidates for calibration training is both computationally prohibitive and information-redundant: the majority of entries contribute little to overall informativeness. Thus, an effective selection mechanism is needed to retain only the most informative instance per query.

C.2 Principle of Informativeness.

The informativeness of a candidate is defined as its ability to characterize the transition from highly informative predictions to less informative ones. Empirically, when candidate probabilities are ranked, the distribution exhibits a steep drop from a small number of dominant candidates to a long tail of negligible ones. The most informative instance lies at this transition point, where the contrast between candidates is strongest. Selecting this instance ensures that calibration focuses on the sample carrying the greatest informativeness.

C.3 Reordering Step.

To detect this transition consistently, we first reorder each query's probability vector into descending order. This guarantees a monotonic sequence from most informative to least informative candidates, making the transition explicit. Although this step perturbs the raw candidate alignment, it does not alter downstream objectives: (i) Link prediction is ranking-based, not order-sensitive. Thus, perturbing candidate order within a query does not affect evaluation metrics such as MRR or HITS@K. (ii) As demonstrated in Multi-Binning Scaling (Section 4.2), for query i, all of the M elements in the probability vector $\mathbf{P_i}$ are transformed using the same temperature parameter T_w^2. MBS is applied uniformly across the entire vector $\mathbf{P_i}$, regardless of its internal order. Therefore, the calibrated output preserves the original ranking. Reordering is therefore a benign preprocessing step that enables consistent identification of the most informative instance.

C.4 Illustrative Toy Example.

To make the procedure concrete, consider the following probability matrix with three queries (rows) and three candidates (columns):

$$\mathbf{P} = \begin{bmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.6 & 0.2 \\ 0.5 & 0.1 & 0.4 \end{bmatrix}. \tag{9}$$

After reordering each row in descending order, we obtain

$$\tilde{\mathbf{P}} = \begin{bmatrix} 0.3 & 0.2 & 0.1 \\ 0.6 & 0.4 & 0.2 \\ 0.5 & 0.4 & 0.1 \end{bmatrix}. \tag{10}$$

We then compute KL-based Jump Measures between adjacent columns of $\tilde{\mathbf{P}}$, which yields

$$J = [0.0039,\ 0.0541], \tag{11}$$

indicating that the maximum jump occurs at the second column ($J^* = 1$ in zero-based indexing).

Table 5: Effect of different calibration methods on the performance of the TransE model across various datasets. ↑ indicates an improvement, while ↓ indicates a decline compared to the original uncalibrated results.

Method	*MR*	*MRR*	*HITS@1*	*HITS@3*	*HITS@10*
		WN18			
Uncal	263	0.772	0.706	0.807	0.920
PS	260 ↑	0.772	0.706	0.807	0.920
HB	15299 ↓	0.225 ↓	0.212 ↓	0.236 ↓	0.240 ↓
IR	14590 ↓	0.251 ↓	0.232 ↓	0.267 ↓	0.279 ↓
BBQ	15178 ↓	0.218 ↓	0.200 ↓	0.233 ↓	0.244 ↓
VS	258 ↑	0.772	0.706	0.807	0.920
MS	16483 ↓	0.013 ↓	0.005 ↓	0.013 ↓	0.029 ↓
TS	260 ↑	0.772	0.706	0.807	0.920
Meta-Cal	1784 ↓	0.718 ↓	0.657 ↓	0.749 ↓	0.856 ↓
PTS	2116 ↓	0.751 ↓	0.706	0.775 ↓	0.849 ↓
KGEC	263	0.772	0.706	0.807	0.920
		WN18RR			
Uncal	3437	0.223	0.014	0.401	0.528
PS	3437	0.223	0.014	0.401	0.528
HB	19455 ↓	0.071 ↓	0.053 ↑	0.087 ↓	0.099 ↓
IR	18143 ↓	0.102 ↓	0.080 ↑	0.119 ↓	0.139 ↓
BBQ	18196 ↓	0.071 ↓	0.050 ↑	0.085 ↓	0.105 ↓
VS	3421 ↑	0.224 ↑	0.014	0.401	0.529 ↑
MS	18178 ↓	0.009 ↓	0.003 ↓	0.008 ↓	0.020 ↓
TS	3437	0.223	0.014	0.401	0.528
Meta-Cal	3437	0.223	0.014	0.401	0.528
PTS	3437	0.223	0.014	0.401	0.528
KGEC	3437	0.223	0.014	0.401	0.528
		FB15K			
Uncal	40	0.731	0.646	0.793	0.865
PS	40	0.731	0.646	0.793	0.865
HB	2275 ↓	0.570 ↓	0.510 ↓	0.614 ↓	0.670 ↓
IR	982 ↓	0.615 ↓	0.530 ↓	0.675 ↓	0.761 ↓
BBQ	1275 ↓	0.589 ↓	0.509 ↓	0.646 ↓	0.726 ↓
VS	41 ↓	0.730 ↓	0.646	0.791 ↓	0.862 ↓
MS	3687 ↓	0.038 ↓	0.024 ↓	0.039 ↓	0.061 ↓
TS	40	0.731	0.646	0.793	0.865
Meta-Cal	1149 ↓	0.677 ↓	0.604 ↓	0.735 ↓	0.787 ↓
PTS	40	0.731	0.646	0.793	0.865
KGEC	40	0.731	0.646	0.793	0.865
		FB15K-237			
Uncal	173	0.330	0.231	0.368	0.527
PS	173	0.330	0.231	0.368	0.527
HB	3497 ↓	0.289 ↓	0.224 ↓	0.321 ↓	0.416 ↓
IR	2141 ↓	0.309 ↓	0.234 ↑	0.343 ↓	0.455 ↓
BBQ	2335 ↓	0.280 ↓	0.209 ↓	0.310 ↓	0.422 ↓
VS	173	0.330	0.231	0.368	0.527
MS	3704 ↓	0.033 ↓	0.014 ↓	0.032 ↓	0.070 ↓
TS	173	0.330	0.231	0.368	0.527
Meta-Cal	1231 ↓	0.308 ↓	0.218 ↓	0.344 ↓	0.490 ↓
PTS	173	0.330	0.231	0.368	0.527
KGEC	173	0.330	0.231	0.368	0.527

Table 6: Effect of different calibration methods on the performance of the ComplEx model across various datasets. ↑ indicates an improvement, while ↓ indicates a decline compared to the original uncalibrated results.

Method	*MR*	*MRR*	*HITS@1*	*HITS@3*	*HITS@10*
		WN18			
Uncal	311	0.893	0.854	0.925	0.953
PS	311	0.893	0.854	0.925	0.953
HB	14328 ↓	0.274 ↓	0.262 ↓	0.285 ↓	0.289 ↓
IR	14094 ↓	0.290 ↓	0.280 ↓	0.298 ↓	0.304 ↓
BBQ	13657 ↓	0.236 ↓	0.194 ↓	0.271 ↓	0.306 ↓
VS	305 ↑	0.893	0.854	0.925	0.953
MS	16825 ↓	0.011 ↓	0.004 ↓	0.012 ↓	0.022 ↓
TS	311	0.893	0.854	0.925	0.953
Meta-Cal	1260 ↓	0.851 ↓	0.813 ↓	0.880 ↓	0.908 ↓
PTS	311	0.893	0.854	0.925	0.953
KGEC	311	0.893	0.854	0.925	0.953
		WN18RR			
Uncal	5469	0.469	0.428	0.486	0.552
PS	5469	0.469	0.428	0.486	0.552
HB	18836 ↓	0.107 ↓	0.100 ↓	0.112 ↓	0.118 ↓
IR	18244 ↓	0.103 ↓	0.090 ↓	0.110 ↓	0.124 ↓
BBQ	18200 ↓	0.087 ↓	0.076 ↓	0.097 ↓	0.105 ↓
VS	5447 ↑	0.469	0.428	0.486	0.552
MS	18191 ↓	0.009 ↓	0.003 ↓	0.009 ↓	0.020 ↓
TS	5469	0.469	0.428	0.486	0.552
Meta-Cal	6416 ↓	0.445 ↓	0.407 ↓	0.459 ↓	0.522
PTS	5469	0.469	0.428	0.486	0.552
KGEC	5469	0.469	0.428	0.486	0.552
		FB15K			
Uncal	45	0.770	0.703	0.816	0.885
PS	45	0.770	0.703	0.816	0.885
HB	1747 ↓	0.610 ↓	0.543 ↓	0.661 ↓	0.724 ↓
IR	970 ↓	0.652 ↓	0.579 ↓	0.704 ↓	0.780 ↓
BBQ	797 ↓	0.597 ↓	0.509 ↓	0.656 ↓	0.757 ↓
VS	43 ↑	0.770	0.703	0.816	0.886 ↑
MS	3693 ↓	0.025 ↓	0.010 ↓	0.024 ↓	0.055 ↓
TS	45	0.770	0.703	0.816	0.885
Meta-Cal	484 ↓	0.715 ↓	0.651 ↓	0.759 ↓	0.826 ↓
PTS	45	0.770	0.703	0.816	0.885
KGEC	45	0.770	0.703	0.816	0.885
		FB15K-237			
Uncal	166	0.322	0.230	0.352	0.511
PS	166	0.322	0.230	0.352	0.511
HB	2882 ↓	0.274 ↓	0.201 ↓	0.305 ↓	0.420 ↓
IR	2185 ↓	0.296 ↓	0.220 ↑	0.328 ↓	0.449 ↓
BBQ	1661 ↓	0.249 ↓	0.176 ↓	0.273 ↓	0.399 ↓
VS	166	0.322	0.230	0.352	0.512 ↑
MS	3704 ↓	0.033 ↓	0.014 ↓	0.032 ↓	0.070 ↓
TS	166	0.322	0.230	0.352	0.511
Meta-Cal	267 ↓	0.310 ↓	0.218 ↓	0.339 ↓	0.498 ↓
PTS	166	0.322	0.230	0.352	0.511
KGEC	166	0.322	0.230	0.352	0.511

Thus, the second column of $\tilde{\mathbf{P}}$ is selected as the most informative instance per query:

$$\mathbf{p}^* = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \end{bmatrix}. \tag{12}$$

For each query (row), suppose the temperatures selected by MBS based on the binned values in $\mathbf{p}^*$ are

$$\mathbf{T} = \begin{bmatrix} 0.7 \\ 1.5 \\ 1.0 \end{bmatrix}. \tag{13}$$

Applying temperature scaling yields the calibrated confidence matrix

$$\hat{\mathbf{P}} = \begin{bmatrix} \frac{0.1}{0.7^2} & \frac{0.2}{0.7^2} & \frac{0.3}{0.7^2} \\ \frac{0.4}{1.5^2} & \frac{0.6}{1.5^2} & \frac{0.2}{1.5^2} \\ \frac{0.5}{1.0^2} & \frac{0.1}{1.0^2} & \frac{0.4}{1.0^2} \end{bmatrix} = \begin{bmatrix} 0.20 & 0.40 & 0.60 \\ 0.18 & 0.27 & 0.09 \\ 0.50 & 0.10 & 0.40 \end{bmatrix} \tag{14}$$

As observed, this rescaling preserves the rela-

Table 7: Effect of different calibration methods on the performance of the DistMult model across various datasets. ↑ indicates an improvement, while ↓ indicates a decline compared to the original uncalibrated results.

Method	*MR*	*MRR*	*HITS@1*	*HITS@3*	*HITS@10*
			WN18		
Uncal	227	0.685	0.529	0.829	0.933
PS	227	0.685	0.529	0.829	0.933
HB	14718 ↓	0.240 ↓	0.216 ↓	0.262 ↓	0.271 ↓
IR	14271 ↓	0.260 ↓	0.237 ↓	0.279 ↓	0.294 ↓
BBQ	13614 ↓	0.201 ↓	0.154 ↓	0.232 ↓	0.293 ↓
VS	224 ↑	0.685	0.529	0.829	0.933
MS	16984 ↓	0.011 ↓	0.004 ↓	0.012 ↓	0.022 ↓
TS	227	0.685	0.529	0.829	0.933
Meta-Cal	770 ↓	0.663 ↓	0.508 ↓	0.805 ↓	0.908 ↓
PTS	240 ↓	0.685	0.529	0.829	0.932 ↓
KGEC	227	0.685	0.529	0.829	0.933
			WN18RR		
Uncal	4912	0.439	0.394	0.453	0.532
PS	4909 ↑	0.439	0.394	0.453	0.532
HB	19006 ↓	0.100 ↓	0.090 ↑	0.108 ↓	0.117 ↓
IR	18174 ↓	0.099 ↓	0.083 ↑	0.109 ↓	0.124 ↓
BBQ	18192 ↓	0.088 ↓	0.073 ↑	0.100 ↓	0.109 ↓
VS	4888 ↑	0.439	0.394	0.453	0.532
MS	18172 ↓	0.009 ↓	0.003 ↓	0.009 ↓	0.020 ↓
TS	4909 ↑	0.439	0.394	0.453	0.532
Meta-Cal	6157 ↓	0.406 ↓	0.366 ↓	0.419 ↓	0.493 ↓
PTS	4909 ↑	0.439	0.394	0.453	0.532
KGEC	4909 ↑	0.439	0.394	0.453	0.532
			FB15K		
Uncal	41	0.768	0.701	0.813	0.884
PS	41	0.768	0.701	0.813	0.884
HB	1528 ↓	0.630 ↓	0.562 ↓	0.679 ↓	0.748 ↓
IR	952 ↓	0.667 ↓	0.599 ↓	0.713 ↓	0.787 ↓
BBQ	692 ↓	0.603 ↓	0.512 ↓	0.659 ↓	0.775 ↓
VS	39 ↑	0.768	0.701	0.814 ↑	0.885 ↑
MS	3693 ↓	0.025 ↓	0.010 ↓	0.024 ↓	0.055 ↓
TS	41	0.768	0.701	0.813	0.884
Meta-Cal	202 ↓	0.746 ↓	0.680 ↓	0.790 ↓	0.861 ↓
PTS	41	0.768	0.701	0.813	0.884
KGEC	41	0.768	0.701	0.813	0.884
			FB15K-237		
Uncal	174	0.309	0.222	0.337	0.484
PS	174	0.309	0.222	0.337	0.484
HB	2695 ↓	0.256 ↓	0.184 ↓	0.286 ↓	0.401 ↓
IR	2156 ↓	0.280 ↓	0.205 ↑	0.311 ↓	0.427 ↓
BBQ	1562 ↓	0.235 ↓	0.163 ↓	0.259 ↓	0.378 ↓
VS	172 ↑	0.305 ↓	0.216 ↓	0.333 ↓	0.484
MS	3704 ↓	0.033 ↓	0.014 ↓	0.032 ↓	0.070 ↓
TS	174	0.309	0.222	0.337	0.484
Meta-Cal	259 ↓	0.300 ↓	0.213 ↓	0.327 ↓	0.474 ↓
PTS	5659 ↓	0.222 ↓	0.222	0.222 ↓	0.223 ↓
KGEC	174	0.309	0.222	0.337	0.484

Table 8: Effect of different calibration methods on the performance of the RotatE model across various datasets. ↑ indicates an improvement, while ↓ indicates a decline compared to the original uncalibrated results.

Method	*MR*	*MRR*	*HITS@1*	*HITS@3*	*HITS@10*
			WN18		
Uncal	270	0.950	0.944	0.952	0.960
PS	270	0.950	0.944	0.952	0.960
HB	13910 ↓	0.279 ↓	0.263 ↓	0.294 ↓	0.299 ↓
IR	13962 ↓	0.297 ↓	0.286 ↓	0.308 ↓	0.313 ↓
BBQ	13801 ↓	0.271 ↓	0.253 ↓	0.286 ↓	0.297 ↓
VS	270	0.950	0.944	0.952	0.960
MS	16626 ↓	0.013 ↓	0.005 ↓	0.013 ↓	0.027 ↓
TS	270	0.950	0.944	0.952	0.960
Meta-Cal	1917 ↓	0.905 ↓	0.904 ↓	0.905 ↓	0.905 ↓
PTS	474 ↓	0.949 ↓	0.944	0.951 ↓	0.958 ↓
KGEC	270	0.950	0.944	0.952	0.960
			WN18RR		
Uncal	3421	0.476	0.429	0.496	0.570
PS	3421	0.476	0.429	0.497 ↑	0.570
HB	18719 ↓	0.114 ↓	0.104 ↓	0.122 ↓	0.127 ↓
IR	18047 ↓	0.118 ↓	0.103 ↓	0.128 ↓	0.143 ↓
BBQ	18189 ↓	0.086 ↓	0.073 ↓	0.095 ↓	0.105 ↓
VS	3422 ↓	0.476	0.429	0.497 ↑	0.570
MS	18195 ↓	0.009 ↓	0.003 ↓	0.008 ↓	0.020 ↓
TS	3421	0.476	0.429	0.497 ↑	0.570
Meta-Cal	6168 ↓	0.448 ↓	0.409 ↓	0.464 ↓	0.523 ↓
PTS	3776 ↓	0.474 ↓	0.429	0.493 ↓	0.564 ↓
KGEC	3421	0.476	0.429	0.497 ↑	0.570
			FB15K		
Uncal	41	0.791	0.739	0.825	0.881
PS	41	0.791	0.739	0.825	0.881
HB	1843 ↓	0.642 ↓	0.588 ↓	0.682 ↓	0.731 ↓
IR	961 ↓	0.696 ↓	0.635 ↓	0.741 ↓	0.799 ↓
BBQ	1027 ↓	0.662 ↓	0.599 ↓	0.709 ↓	0.768 ↓
VS	42 ↓	0.791	0.739	0.825	0.880 ↓
MS	3693 ↓	0.025 ↓	0.010 ↓	0.024 ↓	0.055 ↓
TS	41	0.791	0.739	0.825	0.881
Meta-Cal	457 ↓	0.750 ↓	0.700 ↓	0.783 ↓	0.835 ↓
PTS	1122 ↓	0.763 ↓	0.739	0.782 ↓	0.801 ↓
KGEC	41	0.791	0.739	0.825	0.881
			FB15K-237		
Uncal	178	0.336	0.239	0.374	0.530
PS	178	0.336	0.239	0.374	0.530
HB	3458 ↓	0.285 ↓	0.221 ↓	0.317 ↓	0.412 ↓
IR	2131 ↓	0.307 ↓	0.232 ↓	0.340 ↓	0.455 ↓
BBQ	2292 ↓	0.275 ↓	0.204 ↓	0.305 ↓	0.415 ↓
VS	179 ↓	0.336	0.239	0.374	0.530
MS	3704 ↓	0.033 ↓	0.014 ↓	0.032 ↓	0.070 ↓
TS	178	0.336	0.239	0.374	0.530
Meta-Cal	246 ↓	0.328 ↓	0.232 ↓	0.365 ↓	0.522 ↓
PTS	179 ↓	0.336	0.239	0.374	0.530
KGEC	178	0.336	0.239	0.374	0.530

tive ordering of candidates within each row, and thus ranking-based metrics (e.g., MRR, HITS@K) remain unaffected.

C.5 KL Divergence as Jump Measure.

Given the reordered probability matrix $\tilde{\mathbf{P}}$, we compute the *Jump measure* J_j as the KL divergence between adjacent columns. KL is not used here as a loss function but as a relative difference metric, quantifying how informativeness changes between consecutive ranked positions. A large KL value indicates a sharp change in informativeness, corresponding to the transition point. Unlike its unstable behavior in one-hot settings (where zero entries occur), KL is well-defined here because all compared vectors are soft probability distributions with non-zero entries.

Formally, the column index J^* that maximizes J_j is selected, and the corresponding column vector $\mathbf{p}^* \in \mathbb{R}^n$ is retained as the most informative sample per query.

C.6 Information Loss Quantification via Shannon Entropy

To analyze the effect of JSS on information preservation, we quantify the potential information loss incurred when reducing a probability matrix $\mathbf{P} \in \mathbb{R}^{N \times M}$ to a single informative column vector $\mathbf{p}^* \in$

$\mathbb{R}^N$. We measure information content using Shannon entropy.

Let $\mathbf{P} \in \mathbb{R}^{N \times M}$ be a real-valued matrix with N rows and M columns. To quantify the amount of information contained in $\mathbf{P}$, we apply Shannon entropy to the empirical distribution of its elements. Suppose the entries are discretized into a finite alphabet $\mathcal{X}$ (e.g., via binning or quantization). Then the entropy of the matrix $\mathbf{P}$ is defined as:

$$H(\mathbf{P}) = -\sum_{x \in \mathcal{X}} p_{\mathbf{P}}(x) \log p_{\mathbf{P}}(x), \tag{15}$$

where $p_{\mathbf{P}}(x)$ denotes the empirical probability mass function over the elements of $\mathbf{P}$.

Similarly, consider selecting a single column vector $\mathbf{v} \in \mathbb{R}^N$ from $\mathbf{P}$, i.e., $\mathbf{v} = \mathbf{P}_{:,j}$ for some $j \in \{1, \ldots, M\}$, its entropy is:

$$H(\mathbf{v}) = -\sum_{x \in \mathcal{X}} p_{\mathbf{v}}(x) \log p_{\mathbf{v}}(x), \tag{16}$$

The loss in information due to column selection is thus:

$$\Delta H = H(\mathbf{P}) - H(\mathbf{v}). \tag{17}$$

We discuss three representative cases:

Case 1: Independent and Identically Distributed (i.i.d.) Columns. If each column of $\mathbf{P}$ is drawn independently from the same distribution (i.i.d.), the matrix entropy decomposes additively:

$$H(\mathbf{P}) = \sum_{j=1}^{M} H(\mathbf{v}_j) = M \cdot H(\mathbf{v}), \tag{18}$$

where $H(\mathbf{v}_j) = H(\mathbf{v})$ for all j. Therefore, the information loss becomes:

$$\Delta H = (M - 1) \cdot H(\mathbf{v}), \tag{19}$$

indicating that $\mathbf{P}$ stores M times more information than any single column under the i.i.d. assumption.

Case 2: Correlated Columns. If the columns are not independent, entropy is subadditive due to redundancy:

$$H(\mathbf{P}) < \sum_{j=1}^{M} H(\mathbf{v}_j), \tag{20}$$

and the total information is reduced by mutual dependencies. Formally,

$$H(\mathbf{P}) = \sum_{j=1}^{M} H(\mathbf{v}_j) - \text{Redundancy}, \tag{21}$$

where *Redundancy* quantifies mutual information shared between columns.

Case 3: Identical Columns. In the extreme case where all columns are identical,

$$H(\mathbf{P}) = H(\mathbf{v}), \quad \Delta H = 0, \tag{22}$$

meaning no information is lost by reducing $\mathbf{P}$ to one column.

Discussion. These cases illustrate that although selecting a single column inevitably reduces entropy in the i.i.d. case, real KGE outputs exhibit strong redundancy across candidate entities. JSS exploits this redundancy by identifying the column with the largest information jump, thereby retaining the most informative subset of probabilities while dramatically reducing computational cost.

C.7 Outcome and Comparison.

JSS reduces the training size by a factor of m (the number of candidates per query), transforming the full probability matrix $\mathbf{P} \in \mathbb{R}^{n \times m}$ into a single informative column vector $\mathbf{p}^* \in \mathbb{R}^n$. This compression preserves the informativeness required for calibration while eliminating redundancy. Unlike naive random sampling, which risks discarding boundary instances with high informativeness, JSS consistently identifies the most valuable instance. Empirical results and theoretical analysis confirm that JSS improves both efficiency and reliability in calibration training.

D Limitations of KL Divergence in High-Cardinality Calibration

Kullback–Leibler (KL) divergence is one of the most widely used loss functions in deep learning and probability calibration. However, when applied to high-cardinality tasks such as entity prediction in KGE models, KL divergence exhibits critical limitations that undermine its effectiveness. Each query in KGE involves tens of thousands of candidate entities, yielding probability distributions with extremely sparse support and long tails of near-zero values. In such regimes, KL divergence is prone to gradient vanishing and gradient explosion, leading to instability during optimization.

Failure modes. Two issues are particularly problematic: (i) when the true label probability q_i is nonzero but the predicted probability $p_i \to 0$, the KL term becomes negligible, suppressing the

contribution of informative but low-probability instances; (ii) when $p_i > 0$ but $q_i = 0$, the divergence becomes infinite, yielding unstable or divergent gradients. Both behaviors compromise the robustness of calibration in large-scale KGE settings.

Formal definition. Let p and q be two discrete probability distributions over a finite set $\mathcal{X}$. The KL divergence from q to p is defined as:

$$D_{\mathrm{KL}}(p \parallel q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}. \quad (23)$$

While this expression is well-defined when both $p(x) > 0$ and $q(x) > 0$, edge cases involving zero probabilities require special attention. Below, we analyze two important cases.

Case 1: $p(x) = 0$

If $p(x) = 0$, the corresponding term is:

$$0 \cdot \log \frac{0}{q(x)}. \quad (24)$$

Although $\log 0$ is undefined, this product is conventionally set to 0, justified by the limit:

$$\lim_{u \to 0^+} u \log \frac{u}{q(x)} = 0. \quad (25)$$

Hence, for both analytical and numerical purposes:

$$p(x) \log \frac{p(x)}{q(x)} = 0 \quad \text{when } p(x) = 0. \quad (26)$$

Case 2: $q(x) = 0$ **and** $p(x) > 0$

If $p(x) > 0$ but $q(x) = 0$, the ratio inside the logarithm diverges:

$$\log \frac{p(x)}{q(x)} = +\infty, \quad (27)$$

and thus the corresponding term is:

$$D_{\mathrm{KL}}(p \parallel q) = +\infty \quad \text{if } \exists x \in \mathcal{X} \text{ such that } p(x) > 0 \text{ and } q(x) = 0. \quad (28)$$

Summary

Each term $p(x) \log \frac{p(x)}{q(x)}$ in the KL divergence admits the following interpretation:

- If $p(x) = 0$, the contribution is defined as 0 (by convention via limiting argument).
- If $p(x) > 0$ and $q(x) = 0$, the contribution is $+\infty$, making the divergence infinite.

Thus, KL divergence is finite if and only if the support of p is a subset of the support of q:

$$D_{\mathrm{KL}}(p \parallel q) = \begin{cases} \sum_x p(x) \log \frac{p(x)}{q(x)}, & \text{if } \mathrm{supp}(p) \subseteq \mathrm{supp}(q), \\ +\infty, & \text{otherwise.} \end{cases} \quad (29)$$

Implication for KGE calibration. This support mismatch arises frequently in knowledge graph entity prediction, where sparse distributions and zero-valued targets dominate. In practice, it yields vanishing gradients for informative low-probability entities and exploding gradients when mismatched supports occur. These issues render KL divergence unstable and unreliable as a calibration loss, motivating our adoption of the Wasserstein distance in Section 4.3, which remains finite and geometrically meaningful even under sparse distributions.

E Dataset Statistics

Table 9: Statistics of the used KGE datasets.

dataset	*#Entity*	*#Relation*	*#Training*	*#Validation*	*#Testing*
WN18	40,943	18	141,442	5,000	5,000
WN18RR	40,943	11	86,835	3,034	3,134
FB15K	14,951	1,345	483,142	50,000	59,071
FB15K-237	14,541	237	272,115	17,535	20,466

Table 9 summarizes the key statistics of the benchmark KGE datasets used in our experiments. All four datasets are widely adopted in the link prediction literature, with FB15K (Bordes et al., 2013) and WN18 (Bordes et al., 2013) serving as the original benchmarks. However, both contain a substantial number of inverse or redundant relations, which can cause information leakage across training, validation, and testing splits, leading to overly optimistic results. To address this issue, FB15K-237 was introduced by Toutanova and Chen (2015) as a cleaned version of FB15K, obtained by removing near-duplicate and inverse relations so that test triples cannot be trivially inferred from training data. Similarly, WN18RR (Dettmers et al., 2018) was constructed from WN18 by excluding inverse relations, thereby providing a more challenging and realistic evaluation benchmark. Each dataset is publicly available and comes pre-partitioned into training, validation, and testing splits, which we use without modification.

These datasets differ substantially in both scale and relational complexity. For example, FB15K

Table 10: Score functions of popular KGE models. Here, $\|\cdot\|$ denotes the L_1 norm, $\langle\cdot\rangle$ denotes the generalized dot product, $\mathbf{t}^*$ is the complex conjugate of $\mathbf{t}$, $\mathrm{Re}(\cdot)$ extracts the real part of a complex number, and $\circ$ denotes the Hadamard (element-wise) product.

KGE Model	Score Function
TransE (Bordes et al., 2013)	$-\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$
DistMult (Yang et al., 2015)	$\langle\mathbf{r},\mathbf{h},\mathbf{t}\rangle$
ComplEx (Trouillon et al., 2016)	$\mathrm{Re}(\langle\mathbf{r},\mathbf{h},\mathbf{t}^*\rangle)$
RotatE (Sun et al., 2019)	$-\|\mathbf{h}\circ\mathbf{r}-\mathbf{t}\|$

contains over 1,300 relation types, while FB15K-237 reduces this number to 237 to mitigate redundancy. WN18 and WN18RR share the same set of entities (40,943) but differ in their relation sets, with WN18RR offering a more robust evaluation by removing symmetric and inverse patterns. Together, these datasets span diverse characteristics of real-world knowledge graphs, covering both lexical (WordNet-based) and factual (Freebase-based) domains. Notably, the large entity space in FB15K and FB15K-237 introduces a long-tailed distribution of candidate entities, which poses significant challenges for probability calibration: most classes receive extremely low predicted probabilities, amplifying the issues of sparsity and miscalibration that our proposed method is designed to address.

F Score Functions of Popular KGE Models

We summarize the score functions of several widely used KGE models in Table 10. These definitions provide the basis for the experiments discussed in the Section 5. Beyond link prediction, KGE models have also been successfully applied to a broad range of tasks, such as entity alignment (Sun et al., 2018), canonicalization (Yang and Curry, 2024; Yang et al., 2025), and question answering (Bordes et al., 2014), highlighting their versatility and impact across diverse knowledge-intensive applications.

G Detailed Experimental Results on Efficiency Study for RQ2

To complement the aggregated efficiency results in Table 3 in Section 5.3, we present a fine-grained breakdown of calibration costs across different datasets and KGE models in Table 11. Specifically, the table reports both the training time (in seconds) and peak memory usage (in MBs) required to calibrate entity prediction under CPU-only environments, ensuring a fair comparison across all methods.

This detailed analysis provides two important insights. First, it reveals the scalability challenges of certain methods: for example, Platt Scaling (PS) and Parametrized Temperature Scaling (PTS) incur prohibitive computational overhead on larger datasets such as FB15K, making them impractical for large-scale applications. In contrast, lightweight approaches such as Vector Scaling (VS), Temperature Scaling (TS), and our proposed KGEC method exhibit consistently low resource consumption. Second, the memory profiles highlight significant disparities: PTS can require over 8 GB of memory on FB15K, whereas KGEC achieves state-of-the-art calibration accuracy with average memory usage of only $\sim$22 MB.

Overall, Table 11 demonstrates that KGEC not only achieves superior calibration performance but also remains the most resource-efficient approach across all benchmarks. These results further support the conclusions in Section 5.3, where we argued that efficiency is essential for deploying calibration in knowledge-intensive systems.

H Detailed Experimental Results on Ablation Study for RQ3

In this section, we provide the complete experimental results for our ablation study.

Table 12 reports the detailed performance of each component in KGEC across individual datasets and KGE models, covering all evaluation metrics (ECE, ACE, NLL, training time, and memory usage). These results complement the averaged findings presented in Section 5.4 by illustrating the effect of each component in a fine-grained manner.

Effectiveness of JSS vs. Random Sampling. Beyond the above comparisons, we further analyze the effectiveness of JSS against a Random Sampling baseline to directly validate its contribution. JSS consistently retains the most informative sample per query, thereby guiding the calibration process more effectively. In contrast, Random Sampling often discards informative instances and introduces instability in large class spaces, leading to degraded calibration. As shown in Figure 4, JSS achieves markedly better calibration across all metrics (ECE: 0.388 vs. 0.467, ACE: 0.348 vs. 0.495, NLL: 3.396 vs. 5.725). These results highlight that JSS simultaneously enhances efficiency and preserves calibration quality, whereas Random

Table 11: Training time in seconds and memory usage in MBs taken to calibrate entity prediction using different calibration methods. Best and second-ranked results are in bold and underlined, respectively. For a fair comparison, these results are obtained using CPU only.

Time	*TransE*				*ComplEx*				*DistMult*				*RotatE*				*Average*
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
PS	50551.471	32130.612	66566.552	22756.968	44484.280	27740.023	66631.859	20060.975	48902.412	31739.057	58074.230	21682.032	46162.422	30198.810	65506.688	20522.725	40856.945
VS	2.857	1.893	25.357	3.493	2.661	1.620	16.228	3.218	4.114	1.914	20.779	3.456	2.656	1.706	25.995	3.277	<u>7.577</u>
TS	5.235	3.207	20.037	6.475	5.063	3.121	18.825	6.276	5.180	3.204	19.734	6.412	5.456	3.171	20.646	6.345	8.649
PTS	3452.440	2123.849	16769.166	5856.000	3432.436	2122.273	16510.019	5764.345	3450.331	2120.555	16898.528	5868.468	3425.148	2113.001	16802.984	5853.287	7035.177
KGEC	2.727	1.776	10.873	3.602	2.698	1.727	10.560	3.624	2.741	1.696	10.645	3.705	2.662	1.658	10.758	4.003	**4.716**

Memory	*TransE*				*ComplEx*				*DistMult*				*RotatE*				*Average*
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
PS	1564.336	950.762	5706.102	1948.508	1566.598	950.270	5706.832	1948.664	1565.820	949.633	5705.828	1947.574	1566.477	950.793	5706.875	1948.371	2542.715
VS	84.477	84.383	86.098	84.348	82.059	83.152	86.918	80.770	83.609	83.883	80.883	81.320	80.570	83.145	86.152	80.941	<u>83.294</u>
TS	1562.625	948.453	5703.750	1947.629	1562.984	949.285	5703.047	1945.566	1562.340	948.504	5704.801	1945.828	1562.914	948.566	5703.359	1944.730	2540.274
PTS	6655.574	7017.359	11154.340	9554.723	6804.816	7022.313	10185.500	9629.871	6957.012	6696.055	10180.105	9407.988	7047.270	7074.051	10521.520	8659.395	8410.493
KGEC	30.484	28.289	7.570	15.273	26.652	32.176	9.535	15.285	34.316	32.047	10.531	13.492	34.320	32.191	7.551	16.930	**21.665**

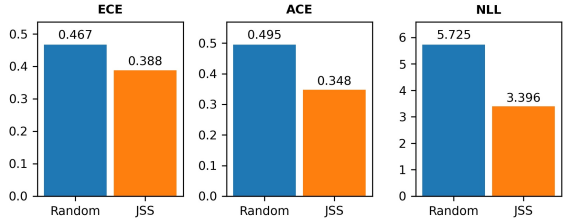

Figure 4: Overall comparison between Random sample and JSS in KGEC, showing average performance across all datasets and KGE models. Lower values of ECE, ACE, and NLL indicate better performance.

Sampling fails to achieve this balance.

In addition, Table 13 presents a focused comparison between JSS and a Random Sampling baseline across four representative KGE models and multiple datasets. Results are reported on the three calibration metrics (ECE, ACE, and NLL), where lower values indicate better performance. This table provides the full results underlying the summary trends shown in Figure 4, further demonstrating the effectiveness and stability of JSS over Random Sampling.

I Sensitivity Analysis

To assess the robustness and stability of our proposed KGEC method, we conduct a comprehensive sensitivity analysis by varying three critical hyperparameters: the number of bins, the initial temperature, and the learning rate. We evaluate the impact of each parameter on three calibration metrics, i.e., ECE, ACE, and NLL, across all KGE models and datasets. Results are summarized in Tables 14, 15, and 16.

Effect of the Number of Bins. We vary the number of bins from 1 to 20. Table 14 shows that using only one bin (equivalent to vanilla temperature scaling) results in poor performance across all metrics, highlighting its limited flexibility. As the number of bins increases, KGEC becomes more expressive and better calibrated. The best average performance is observed at 19 bins (ECE = 0.352, ACE = 0.343, NLL = 3.361), though results are stable within the 10–20 bin range. This confirms the importance of multi-binning for modeling diverse score distributions, while also indicating that KGEC is robust to bin selection within a reasonable interval.

Effect of Initial Temperature. We examine initial temperature values ranging from 0 to 2.0. As shown in Table 15, extreme initializations (e.g., 0.0 or 2.0) lead to degraded performance due to optimization instability. An initial temperature of 1.0 yields the best results (ECE = 0.388, ACE = 0.348, NLL = 3.396), aligning with standard practice in temperature scaling (Guo et al., 2017). The results indicate that KGEC is relatively insensitive to this hyperparameter, as long as it is initialized within a moderate range.

Effect of Learning Rate. Table 16 presents results under learning rates ranging from 0.001 to 0.1. We find that too small learning rates (e.g., 0.001) may underfit the calibration model, while overly large values (e.g., 0.1) can cause instability and degraded performance. The learning rate of 0.01 achieves the best overall calibration (ECE = 0.388, ACE = 0.348, NLL = 3.396), striking a balance between convergence speed and stability.

Summary. Across all experiments, KGEC demonstrates strong robustness to hyperparameter variations. The best performance is consistently achieved with moderate hyperparameter values: a bin count between 10 and 20, an initial temperature near 1.0, and a learning rate around 0.01. These findings suggest that KGEC is both stable and practical, requiring minimal hyperparameter tuning for optimal performance across diverse KGE models and datasets.

Table 12: Effect of each component in KGEC on the performance and efficiency of various KGE models across multiple datasets. For all the five metrics, the lower the better.

ECE	*TransE*				*ComplEx*				*DistMult*				*RotatE*				*Average*
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
KGEC-loss-MBS-JSS	0.642	0.195	0.637	0.213	0.852	0.423	0.691	0.228	0.528	0.389	0.689	0.220	0.805	0.383	0.671	0.222	0.487
KGEC-loss-MBS	0.634	0.196	0.637	0.213	0.852	0.423	0.691	0.228	0.528	0.389	0.688	0.220	0.821	0.383	0.672	0.222	0.487
KGEC-loss	0.611	0.196	0.408	0.199	0.824	0.377	0.689	0.161	0.501	0.388	0.683	0.165	0.813	0.327	0.642	0.215	0.450
KGEC	0.171	0.280	0.459	0.150	0.833	0.418	0.678	0.189	0.446	0.383	0.683	0.178	0.467	0.307	0.466	0.094	0.388

ACE	*TransE*				*ComplEx*				*DistMult*				*RotatE*				*Average*
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
KGEC-loss-MBS-JSS	0.517	0.285	0.636	0.168	0.852	0.423	0.691	0.227	0.527	0.389	0.688	0.220	0.405	0.383	0.636	0.220	0.454
KGEC-loss-MBS	0.516	0.285	0.630	0.168	0.852	0.423	0.690	0.227	0.527	0.389	0.688	0.220	0.402	0.383	0.636	0.220	0.454
KGEC-loss	0.510	0.283	7.651	0.943	0.823	0.350	0.670	0.161	0.501	0.388	0.666	0.163	0.401	0.278	3.092	0.308	1.074
KGEC	0.131	0.277	0.293	0.082	0.833	0.418	0.465	0.207	0.457	0.383	0.516	0.199	0.467	0.306	0.466	0.063	0.348

NLL	*TransE*				*ComplEx*				*DistMult*				*RotatE*				*Average*
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
KGEC-loss-MBS-JSS	2.827	6.544	3.270	5.177	6.830	7.777	5.329	7.294	7.384	7.820	5.294	7.485	1.313	6.107	3.465	5.531	5.590
KGEC-loss-MBS	2.834	6.544	3.310	5.189	6.831	7.778	5.311	7.300	7.384	7.812	5.265	7.479	1.304	6.107	3.470	5.521	5.590
KGEC-loss	2.834	6.330	0.687	4.093	4.856	7.636	6.732	3.811	5.407	7.772	6.444	3.950	1.309	6.327	5.014	6.156	4.960
KGEC	2.462	5.965	2.536	2.889	4.350	6.965	1.357	2.911	2.843	7.119	1.319	3.106	1.036	4.698	2.033	2.743	3.396

Training Time / s	*TransE*				*ComplEx*				*DistMult*				*RotatE*				*Average*
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
KGEC-loss-MBS-JSS	39.769	24.194	139.544	54.700	40.996	23.856	148.151	50.186	39.602	24.557	147.145	52.021	39.659	24.270	153.269	52.023	65.871
KGEC-loss-MBS	2.894	1.638	11.442	3.598	2.714	1.611	10.166	3.546	2.661	1.645	10.147	3.603	2.825	1.608	10.760	3.695	4.659
KGEC-loss	2.785	1.676	10.305	3.598	2.915	1.650	10.246	3.597	2.660	1.644	10.490	3.527	2.671	1.605	10.747	3.578	4.606
KGEC	2.727	1.776	10.873	3.602	2.698	1.727	10.560	3.624	2.741	1.696	10.645	3.705	2.662	1.658	10.758	4.003	4.716

Memory Usage / MB	*TransE*				*ComplEx*				*DistMult*				*RotatE*				*Average*
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
KGEC-loss-MBS-JSS	161.801	126.141	58.465	50.965	170.859	111.168	41.953	65.574	174.496	124.086	56.414	56.473	160.766	94.270	45.000	63.301	97.608
KGEC-loss-MBS	29.027	27.121	6.871	17.969	31.258	26.676	7.535	10.391	25.426	27.184	8.750	14.277	32.906	30.742	6.422	17.961	20.032
KGEC-loss	29.414	27.145	6.898	18.016	25.645	26.879	8.145	10.375	32.254	26.613	8.695	14.320	32.754	30.965	10.172	17.316	20.350
KGEC	30.484	28.289	7.570	15.273	26.652	32.176	9.535	15.285	34.316	32.047	10.531	13.492	34.320	32.191	7.551	16.930	21.665

Table 13: Ablation study on the effectiveness of the JSS component in KGEC. We compare KGEC with JSS against a Random baseline across four KGE models (TransE, ComplEx, DistMult, RotatE) on multiple datasets. Results are reported using ECE, ACE, and NLL, where lower values indicate better performance.

ECE	*TransE*				*ComplEx*				*DistMult*				*RotatE*				*Average*
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
Random	0.474	0.199	0.644	0.199	0.851	0.423	0.697	0.228	0.527	0.390	0.697	0.221	0.579	0.398	0.714	0.225	0.467
JSS	0.171	0.280	0.459	0.150	0.833	0.418	0.678	0.189	0.446	0.383	0.683	0.178	0.467	0.307	0.466	0.094	0.388

ACE	*TransE*				*ComplEx*				*DistMult*				*RotatE*				*Average*
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
Random	0.486	0.292	0.936	0.259	0.851	0.423	0.697	0.228	0.527	0.390	0.697	0.220	0.579	0.398	0.714	0.225	0.495
JSS	0.131	0.277	0.293	0.082	0.833	0.418	0.465	0.207	0.457	0.383	0.516	0.199	0.467	0.306	0.466	0.063	0.348

NLL	*TransE*				*ComplEx*				*DistMult*				*RotatE*				*Average*
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
Random	3.028	6.518	2.575	3.355	6.497	7.761	6.043	7.577	7.222	8.007	6.338	7.735	1.721	6.516	4.880	5.825	5.725
JSS	2.462	5.965	2.536	2.889	4.350	6.965	1.357	2.911	2.843	7.119	1.319	3.106	1.036	4.698	2.033	2.743	3.396

Query: (Greece, _member_of_domain_region, ?)

True answer: sibyl

Ranked candidate entities	Uncalibrated scores	Calibrated probabilities
Greece	-0.1873	0.0302
Holy_See	-0.2946	0.0272
sibyl	-0.5992	0.0200
Colosseum	-0.8017	0.0164
Sistine_Chapel	-0.8683	0.0153
Roman	-1.1427	0.0116
Italy	-1.1464	0.0116
Rome	-1.1873	0.0111
Seven_Hills_of_Rome	-1.3174	0.0098
augur	-1.3962	0.0090

Figure 5: Case 1 from the WN18RR dataset using the TransE model.

Query: ('North_Atlantic_Treaty_Organization, _member_meronym, ?)

True answer: Netherlands

Ranked candidate entities	Uncalibrated scores	Calibrated probabilities
'North_Atlantic_Treaty_Organization	1.9763	0.3756
Netherlands	1.6756	0.2781
European_Union	0.9763	0.1382
Benelux	0.9763	0.1382
Apeldoorn	-0.4998	0.0316
Leiden	-0.5236	0.0308
Frisian_Islands	-0.5844	0.0290
Friesland	-0.6578	0.0270
Netherlander	-0.6780	0.0264
British_Commonwealth	-0.7083	0.0256

Figure 6: Case 2 from the WN18RR dataset using the TransE model.

J Case Study

To illustrate the practical benefits of KGEC calibration, we present two representative case studies from the WN18RR dataset using the TransE model, as shown in Figure 5 and Figure 6. These examples highlight how calibrated probabilities offer more interpretable and informative confidence scores compared to raw, uncalibrated scores.

Case 1: (Greece, _member_of_domain_region, ?) The ground-truth answer for this query is *sibyl*, which is ranked third among the candidate entities based on the model's raw scores. However, the uncalibrated scores do not reflect a meaningful confidence distribution, with the top-ranked entity *Greece* receiving a score of -0.1873 and the correct answer *sibyl* receiving -0.5992, a difference that is difficult to interpret probabilistically.

After applying KGEC calibration, the corresponding estimates become more interpretable:

- *Greece*: 0.0302
- *Holy See*: 0.0272
- *sibyl* (true answer): 0.0200

These calibrated estimates clearly reflect the uncertainty inherent in the model's prediction. Although the correct answer is not ranked first, its estimate is close to that of the top candidates, suggesting it is still a plausible prediction. This shows that KGEC can better express confidence levels, especially in cases with closely competing candidates.

Case 2: (North_Atlantic_Treaty_Organization, _member_meronym, ?) In this case, the true answer is *Netherlands*, which is correctly ranked second. The raw score of the correct answer (1.6756) is only slightly lower than that of the top-ranked entity *North Atlantic Treaty Organization* (1.9763), but the significance of this difference is unclear without proper calibration.

With KGEC, the calibrated estimates provide a more informative picture:

- *North Atlantic Treaty Organization*: 0.3756
- *Netherlands* (true answer): 0.2781
- *European Union*: 0.1382

Here, although the true answer is not ranked first, its calibrated estimate is still relatively high, reflecting the model's uncertainty and partially shared semantics among top candidates. This enables downstream applications to interpret and potentially leverage multiple candidates rather than overcommitting to the top-1 prediction.

Insights. These case studies demonstrate that:

- KGEC enhances the interpretability of model outputs by transforming unnormalized scores into well-calibrated estimates.
- It allows more accurate reflection of confidence levels, particularly in ambiguous or competitive ranking situations.
- Even when the top-1 prediction is incorrect, KGEC highlights alternative candidates with meaningful confidence, which is valuable for applications such as knowledge graph reasoning, question answering, and downstream ensemble methods.

Overall, these cases exemplify the effectiveness of KGEC in improving the trustworthiness and usability of KGE models.

Table 14: Effect of different number of bins in KGEC on the performance of various KGE models across multiple datasets. For all the three metrics, the lower the better.

ECE	***TransE***				***ComplEx***				***DistMult***				***RotatE***				***Average***
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
Bin=1	0.702	0.196	0.586	0.198	0.851	0.422	0.642	0.227	0.527	0.389	0.687	0.221	0.904	0.382	0.663	0.222	0.489
Bin=2	0.305	0.316	0.581	0.184	0.850	0.422	0.677	0.190	0.521	0.387	0.683	0.219	0.476	0.305	0.671	0.105	0.431
Bin=3	0.214	0.238	0.498	0.183	0.848	0.422	0.677	0.190	0.515	0.385	0.683	0.180	0.467	0.293	0.653	0.101	0.409
Bin=4	0.245	0.249	0.491	0.180	0.848	0.421	0.677	0.190	0.447	0.385	0.682	0.179	0.486	0.286	0.646	0.098	0.407
Bin=5	0.235	0.262	0.479	0.182	0.848	0.420	0.677	0.189	0.447	0.385	0.682	0.179	0.470	0.297	0.622	0.102	0.405
Bin=6	0.211	0.260	0.514	0.170	0.848	0.419	0.677	0.189	0.447	0.384	0.682	0.179	0.487	0.290	0.584	0.112	0.403
Bin=7	0.159	0.273	0.457	0.147	0.848	0.418	0.678	0.189	0.447	0.384	0.682	0.179	0.451	0.304	0.593	0.104	0.394
Bin=8	0.194	0.269	0.460	0.161	0.848	0.418	0.678	0.189	0.446	0.384	0.682	0.179	0.464	0.307	0.529	0.126	0.396
Bin=9	0.181	0.276	0.444	0.160	0.841	0.418	0.678	0.189	0.446	0.384	0.683	0.178	0.464	0.305	0.498	0.157	0.394
Bin=10	0.171	0.280	0.459	0.150	0.833	0.418	0.678	0.189	0.446	0.383	0.683	0.178	0.467	0.307	0.466	0.094	0.388
Bin=11	0.164	0.283	0.416	0.137	0.833	0.418	0.678	0.189	0.446	0.383	0.683	0.178	0.476	0.316	0.491	0.093	0.387
Bin=12	0.163	0.281	0.388	0.162	0.833	0.418	0.678	0.189	0.446	0.383	0.683	0.178	0.475	0.316	0.475	0.100	0.386
Bin=13	0.148	0.287	0.370	0.123	0.835	0.418	0.678	0.189	0.446	0.383	0.683	0.178	0.471	0.317	0.459	0.086	0.379
Bin=14	0.125	0.293	0.376	0.140	0.835	0.417	0.677	0.189	0.446	0.382	0.683	0.178	0.472	0.319	0.458	0.116	0.381
Bin=15	0.102	0.294	0.336	0.129	0.824	0.417	0.678	0.189	0.446	0.381	0.682	0.178	0.475	0.318	0.461	0.076	0.374
Bin=16	0.154	0.296	0.349	0.061	0.769	0.416	0.677	0.189	0.446	0.379	0.682	0.178	0.472	0.319	0.494	0.087	0.373
Bin=17	0.120	0.296	0.313	0.064	0.764	0.415	0.585	0.189	0.446	0.377	0.682	0.178	0.478	0.324	0.490	0.091	0.363
Bin=18	0.115	0.293	0.256	0.085	0.749	0.415	0.589	0.189	0.446	0.377	0.640	0.178	0.472	0.325	0.489	0.134	0.360
Bin=19	0.113	0.293	0.256	0.084	0.749	0.415	0.579	0.189	0.446	0.377	0.603	0.178	0.482	0.326	0.488	0.062	0.352
Bin=20	0.135	0.298	0.261	0.073	0.753	0.416	0.580	0.189	0.445	0.376	0.577	0.178	0.478	0.319	0.493	0.046	0.351

ACE	***TransE***				***ComplEx***				***DistMult***				***RotatE***				***Average***
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
Bin=1	0.598	0.285	0.565	0.158	0.851	0.422	0.633	0.227	0.527	0.389	0.686	0.221	0.385	0.382	0.602	0.220	0.447
Bin=2	0.318	0.323	0.406	0.128	0.849	0.422	0.494	0.217	0.519	0.387	0.538	0.217	0.476	0.242	0.495	0.093	0.383
Bin=3	0.232	0.243	0.378	0.104	0.848	0.422	0.491	0.217	0.511	0.385	0.533	0.217	0.467	0.226	0.471	0.076	0.364
Bin=4	0.171	0.253	0.328	0.097	0.848	0.421	0.488	0.213	0.460	0.385	0.530	0.213	0.449	0.281	0.450	0.082	0.354
Bin=5	0.155	0.264	0.307	0.099	0.848	0.420	0.484	0.211	0.460	0.385	0.529	0.210	0.458	0.297	0.441	0.069	0.352
Bin=6	0.125	0.262	0.261	0.083	0.848	0.419	0.481	0.210	0.460	0.384	0.527	0.204	0.457	0.290	0.454	0.070	0.346
Bin=7	0.135	0.271	0.277	0.061	0.848	0.418	0.478	0.209	0.460	0.384	0.526	0.204	0.451	0.304	0.419	0.067	0.345
Bin=8	0.129	0.267	0.264	0.080	0.848	0.418	0.475	0.208	0.457	0.384	0.522	0.203	0.464	0.307	0.456	0.072	0.347
Bin=9	0.142	0.274	0.270	0.085	0.841	0.418	0.471	0.208	0.457	0.384	0.521	0.200	0.463	0.305	0.454	0.099	0.349
Bin=10	0.131	0.277	0.293	0.082	0.833	0.418	0.465	0.207	0.457	0.383	0.516	0.199	0.467	0.306	0.466	0.063	0.348
Bin=11	0.111	0.280	0.278	0.076	0.833	0.418	0.460	0.207	0.457	0.383	0.513	0.199	0.461	0.316	0.491	0.061	0.347
Bin=12	0.107	0.278	0.259	0.100	0.833	0.418	0.456	0.207	0.457	0.383	0.508	0.199	0.475	0.316	0.475	0.061	0.346
Bin=13	0.128	0.284	0.240	0.077	0.834	0.418	0.450	0.206	0.457	0.383	0.503	0.197	0.471	0.316	0.459	0.059	0.343
Bin=14	0.113	0.291	0.238	0.088	0.834	0.417	0.446	0.205	0.455	0.382	0.497	0.196	0.471	0.319	0.458	0.062	0.342
Bin=15	0.107	0.292	0.239	0.086	0.823	0.417	0.441	0.204	0.455	0.381	0.492	0.195	0.475	0.318	0.461	0.057	0.340
Bin=16	0.111	0.294	0.234	0.063	0.767	0.416	0.436	0.204	0.455	0.379	0.486	0.194	0.472	0.319	0.494	0.053	0.336
Bin=17	0.100	0.296	0.237	0.064	0.762	0.415	0.550	0.204	0.453	0.377	0.481	0.193	0.478	0.324	0.490	0.053	0.342
Bin=18	0.119	0.293	0.248	0.083	0.746	0.415	0.576	0.204	0.453	0.377	0.466	0.192	0.471	0.325	0.489	0.075	0.346
Bin=19	0.115	0.293	0.247	0.090	0.746	0.415	0.534	0.203	0.453	0.377	0.482	0.192	0.482	0.326	0.488	0.046	0.343
Bin=20	0.107	0.298	0.249	0.073	0.750	0.416	0.557	0.203	0.450	0.376	0.553	0.192	0.478	0.319	0.493	0.048	0.348

NLL	***TransE***				***ComplEx***				***DistMult***				***RotatE***				***Average***
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
Bin=1	2.544	6.543	3.910	4.774	6.524	7.496	3.589	7.208	6.944	7.947	5.152	8.395	1.165	6.078	3.184	5.515	5.436
Bin=2	2.865	6.212	3.076	3.365	5.963	7.513	1.350	2.908	5.477	7.578	1.314	6.650	1.281	4.606	1.989	3.056	4.075
Bin=3	2.712	6.265	2.986	3.290	5.633	7.513	1.350	2.908	4.913	7.338	1.315	3.101	1.184	4.472	2.042	2.774	3.737
Bin=4	2.605	6.185	2.845	3.191	5.633	7.299	1.351	2.909	2.843	7.338	1.315	3.101	1.100	4.745	1.974	2.911	3.584
Bin=5	2.537	6.109	2.766	3.061	5.633	7.202	1.351	2.909	2.843	7.338	1.315	3.102	1.088	4.821	2.011	2.713	3.550
Bin=6	2.523	6.085	2.642	3.062	5.633	7.123	1.352	2.909	2.843	7.200	1.316	3.104	1.062	4.716	2.083	2.643	3.518
Bin=7	2.515	6.023	2.641	3.112	5.633	7.009	1.352	2.910	2.843	7.200	1.316	3.104	1.038	4.787	1.939	2.672	3.506
Bin=8	2.499	6.024	2.607	2.956	5.656	7.001	1.353	2.910	2.844	7.200	1.317	3.104	1.051	4.772	2.039	2.580	3.495
Bin=9	2.493	5.987	2.602	2.895	4.835	6.965	1.355	2.910	2.843	7.200	1.317	3.105	1.035	4.711	2.011	2.473	3.421
Bin=10	2.462	5.965	2.536	2.889	4.350	6.965	1.357	2.911	2.843	7.119	1.319	3.106	1.036	4.698	2.033	2.743	3.396
Bin=11	2.466	5.944	2.532	2.898	4.350	6.965	1.358	2.911	2.843	7.119	1.319	3.106	1.013	4.754	2.123	2.743	3.403
Bin=12	2.448	5.943	2.529	2.798	4.350	6.929	1.359	2.911	2.843	7.114	1.321	3.106	1.043	4.731	2.037	2.666	3.383
Bin=13	2.447	5.919	2.486	2.871	4.415	6.933	1.360	2.911	2.843	7.114	1.323	3.107	1.022	4.714	1.959	2.772	3.387
Bin=14	2.437	5.889	2.482	2.809	4.415	6.867	1.363	2.912	2.845	7.027	1.326	3.108	1.017	4.714	1.941	2.593	3.359
Bin=15	2.438	5.880	2.498	2.803	3.987	6.867	1.363	2.912	2.845	6.935	1.329	3.108	1.024	4.689	1.942	2.813	3.340
Bin=16	2.449	5.870	2.463	2.883	2.931	6.773	1.366	2.912	2.845	6.813	1.331	3.109	1.014	4.684	2.060	2.669	3.261
Bin=17	2.434	5.859	2.487	2.855	2.873	6.726	2.678	2.913	2.846	6.722	1.333	3.109	1.023	4.722	2.030	2.638	3.328
Bin=18	2.449	5.865	2.513	2.779	2.711	6.724	2.860	2.913	2.846	6.722	1.860	3.110	1.010	4.715	2.012	2.509	3.350
Bin=19	2.445	5.862	2.498	2.745	2.711	6.733	2.569	2.913	2.846	6.682	2.235	3.110	1.029	4.706	1.996	2.702	3.361
Bin=20	2.438	5.842	2.489	2.791	2.749	6.734	2.710	2.913	2.849	6.635	2.743	3.110	1.014	4.611	2.010	2.818	3.404

Table 15: Effect of different initial temperature parameters in KGEC on the performance of various KGE models across multiple datasets. For all the three metrics, the lower the better.

ECE	TransE				ComplEx				DistMult				RotatE				Average
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
init=0	0.382	0.582	0.405	0.213	0.699	0.374	0.677	0.190	0.447	0.349	0.683	0.180	0.118	0.322	0.612	0.237	0.404
init=0.1	0.382	0.582	0.312	0.213	0.699	0.374	0.677	0.190	0.447	0.349	0.683	0.180	0.118	0.389	0.670	0.236	0.406
init=0.2	0.337	0.014	0.626	0.221	0.699	0.374	0.678	0.189	0.447	0.349	0.683	0.179	0.603	0.420	0.719	0.239	0.424
init=0.3	0.696	0.013	0.626	0.221	0.699	0.374	0.678	0.189	0.447	0.349	0.683	0.179	0.939	0.428	0.719	0.239	0.467
init=0.4	0.705	0.014	0.627	0.221	0.699	0.374	0.678	0.189	0.447	0.349	0.683	0.178	0.944	0.426	0.729	0.239	0.469
init=0.5	0.706	0.014	0.645	0.221	0.699	0.279	0.678	0.189	0.447	0.222	0.683	0.178	0.944	0.397	0.709	0.239	0.453
init=0.6	0.706	0.268	0.624	0.233	0.699	0.348	0.678	0.189	0.447	0.336	0.683	0.178	0.944	0.320	0.668	0.239	0.472
init=0.7	0.706	0.390	0.566	0.233	0.699	0.384	0.678	0.189	0.447	0.356	0.683	0.178	0.907	0.244	0.597	0.239	0.468
init=0.8	0.706	0.424	0.520	0.226	0.793	0.404	0.678	0.189	0.446	0.370	0.683	0.178	0.355	0.215	0.530	0.173	0.431
init=0.9	0.444	0.342	0.496	0.200	0.823	0.412	0.678	0.189	0.446	0.376	0.683	0.178	0.402	0.273	0.492	0.124	0.410
init=1.0	0.171	0.280	0.459	0.150	0.833	0.418	0.678	0.189	0.446	0.383	0.683	0.178	0.467	0.307	0.466	0.094	0.388
init=1.1	0.199	0.232	0.419	0.109	0.848	0.420	0.678	0.189	0.446	0.386	0.683	0.178	0.547	0.331	0.495	0.098	0.391
init=1.2	0.278	0.195	0.365	0.073	0.849	0.422	0.606	0.189	0.485	0.388	0.683	0.178	0.608	0.350	0.528	0.101	0.394
init=1.3	0.340	0.165	0.335	0.051	0.850	0.423	0.592	0.189	0.513	0.388	0.683	0.178	0.656	0.362	0.551	0.106	0.399
init=1.4	0.390	0.141	0.360	0.064	0.851	0.424	0.618	0.189	0.519	0.390	0.683	0.178	0.696	0.373	0.568	0.108	0.409
init=1.5	0.430	0.122	0.388	0.082	0.852	0.425	0.599	0.190	0.524	0.390	0.602	0.178	0.727	0.382	0.581	0.133	0.413
init=1.6	0.463	0.105	0.411	0.097	0.852	0.425	0.625	0.209	0.525	0.391	0.597	0.178	0.752	0.390	0.594	0.128	0.421
init=1.7	0.491	0.090	0.429	0.110	0.852	0.425	0.611	0.213	0.526	0.391	0.609	0.193	0.773	0.395	0.609	0.150	0.429
init=1.8	0.515	0.079	0.447	0.121	0.853	0.426	0.607	0.223	0.527	0.392	0.611	0.186	0.791	0.400	0.622	0.162	0.435
init=1.9	0.534	0.069	0.462	0.133	0.853	0.426	0.616	0.223	0.527	0.392	0.614	0.198	0.807	0.404	0.631	0.167	0.441
init=2.0	0.550	0.061	0.475	0.141	0.853	0.426	0.644	0.225	0.528	0.392	0.613	0.209	0.820	0.407	0.638	0.172	0.447

ACE	TransE				ComplEx				DistMult				RotatE				Average
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
init=0	55.708	27.896	26.189	4.896	0.679	0.163	0.515	0.226	0.471	0.240	0.547	0.228	50.620	3.966	4.828	1.306	11.155
init=0.1	37.388	18.451	12.026	4.896	0.679	0.163	0.515	0.226	0.471	0.240	0.547	0.228	27.979	3.158	4.275	1.278	7.032
init=0.2	8.518	4.825	2.672	1.805	0.688	0.160	0.467	0.210	0.458	0.238	0.517	0.201	8.266	1.911	2.159	0.995	2.131
init=0.3	4.237	2.449	2.671	1.805	0.688	0.160	0.466	0.208	0.458	0.238	0.517	0.200	3.453	1.006	2.154	0.995	1.357
init=0.4	2.585	1.508	2.646	1.814	0.688	0.160	0.465	0.208	0.457	0.238	0.516	0.200	1.404	0.483	0.288	0.995	0.916
init=0.5	1.535	0.994	0.311	1.795	0.688	0.265	0.465	0.207	0.457	0.208	0.516	0.200	0.726	0.195	0.298	0.995	0.616
init=0.6	0.947	0.721	0.249	0.392	0.688	0.347	0.465	0.207	0.457	0.335	0.516	0.200	0.314	0.119	0.327	0.994	0.455
init=0.7	0.573	0.542	0.242	0.290	0.689	0.384	0.465	0.207	0.457	0.355	0.516	0.199	0.105	0.152	0.362	0.994	0.408
init=0.8	0.307	0.422	0.257	0.209	0.792	0.404	0.465	0.207	0.457	0.369	0.516	0.199	0.230	0.213	0.402	0.113	0.348
init=0.9	0.144	0.339	0.275	0.141	0.822	0.412	0.465	0.207	0.457	0.376	0.516	0.199	0.366	0.273	0.427	0.070	0.343
init=1.0	0.131	0.277	0.293	0.082	0.833	0.418	0.465	0.207	0.457	0.383	0.516	0.199	0.467	0.306	0.466	0.063	0.348
init=1.1	0.200	0.230	0.309	0.050	0.847	0.420	0.465	0.207	0.457	0.386	0.516	0.199	0.547	0.331	0.495	0.064	0.358
init=1.2	0.278	0.195	0.331	0.042	0.848	0.422	0.582	0.207	0.478	0.388	0.516	0.199	0.608	0.350	0.528	0.088	0.379
init=1.3	0.340	0.167	0.359	0.048	0.850	0.423	0.493	0.207	0.509	0.388	0.516	0.199	0.656	0.362	0.551	0.094	0.385
init=1.4	0.390	0.145	0.385	0.059	0.851	0.424	0.607	0.207	0.516	0.390	0.517	0.199	0.696	0.373	0.568	0.095	0.401
init=1.5	0.430	0.126	0.409	0.074	0.851	0.425	0.563	0.197	0.523	0.390	0.501	0.199	0.727	0.382	0.581	0.122	0.406
init=1.6	0.463	0.110	0.429	0.094	0.852	0.425	0.618	0.209	0.525	0.391	0.509	0.199	0.752	0.390	0.594	0.116	0.417
init=1.7	0.491	0.096	0.445	0.109	0.852	0.425	0.593	0.211	0.526	0.391	0.597	0.199	0.773	0.395	0.609	0.142	0.429
init=1.8	0.515	0.086	0.462	0.121	0.853	0.426	0.584	0.222	0.527	0.392	0.600	0.197	0.791	0.400	0.622	0.158	0.435
init=1.9	0.534	0.076	0.474	0.133	0.853	0.426	0.604	0.222	0.527	0.392	0.606	0.200	0.807	0.404	0.631	0.163	0.441
init=2.0	0.550	0.068	0.485	0.141	0.853	0.426	0.644	0.224	0.528	0.392	0.604	0.203	0.820	0.407	0.638	0.169	0.447

NLL	TransE				ComplEx				DistMult				RotatE				Average
	WN18	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	*WN18*	*WN18RR*	*FB15K*	*FB15K-237*	
init=0	-1.714	1.977	-0.694	0.791	2.287	3.210	1.349	2.908	2.841	3.253	1.314	3.099	-3.229	1.540	-0.515	1.145	1.223
init=0.1	-1.422	2.230	-0.296	0.791	2.287	3.210	1.349	2.908	2.841	3.253	1.314	3.099	-2.814	1.620	-0.452	1.150	1.317
init=0.2	-0.209	3.154	0.492	1.014	2.287	3.211	1.356	2.909	2.843	3.253	1.318	3.104	-1.858	1.738	-0.316	1.166	1.591
init=0.3	0.378	3.833	0.493	1.014	2.288	3.211	1.356	2.910	2.843	3.253	1.319	3.104	-1.191	2.238	-0.315	1.165	1.744
init=0.4	0.786	4.312	0.498	1.012	2.288	3.211	1.356	2.910	2.843	3.253	1.319	3.104	-0.549	2.683	1.189	1.166	1.961
init=0.5	1.172	4.722	1.926	1.016	2.288	4.151	1.356	2.910	2.843	4.318	1.319	3.105	-0.187	3.085	1.339	1.166	2.283
init=0.6	1.494	5.037	2.077	2.197	2.288	4.852	1.356	2.910	2.843	5.461	1.319	3.105	0.125	3.460	1.501	1.166	2.574
init=0.7	1.777	5.317	2.191	2.374	2.288	5.459	1.356	2.910	2.843	5.889	1.319	3.105	0.395	3.839	1.682	1.167	2.744
init=0.8	2.037	5.560	2.322	2.542	3.267	6.058	1.356	2.910	2.843	6.328	1.319	3.105	0.637	4.125	1.825	2.429	3.042
init=0.9	2.261	5.773	2.434	2.710	3.937	6.483	1.357	2.910	2.843	6.642	1.319	3.106	0.847	4.456	1.902	2.582	3.223
init=1.0	2.462	5.965	2.536	2.889	4.350	6.965	1.357	2.911	2.843	7.119	1.319	3.106	1.036	4.698	2.033	2.743	3.396
init=1.1	2.651	6.142	2.635	3.006	5.567	7.214	1.357	2.911	2.844	7.429	1.319	3.106	1.219	4.929	2.149	2.727	3.575
init=1.2	2.820	6.302	2.743	3.155	5.739	7.427	2.960	2.911	3.803	7.687	1.319	3.106	1.383	5.144	2.293	3.136	3.871
init=1.3	2.975	6.453	2.844	3.328	6.148	7.606	2.410	2.911	4.791	7.751	1.319	3.106	1.538	5.312	2.407	3.201	4.006
init=1.4	3.121	6.597	2.934	3.421	6.262	7.852	3.194	2.911	5.248	8.071	1.319	3.106	1.688	5.502	2.499	3.216	4.184
init=1.5	3.256	6.729	3.043	3.579	6.591	8.080	2.817	3.398	5.977	8.213	2.415	3.106	1.819	5.678	2.578	3.521	4.425
init=1.6	3.380	6.863	3.137	3.715	6.859	8.228	3.322	4.532	6.350	8.329	2.543	3.107	1.942	5.855	2.668	3.466	4.643
init=1.7	3.502	6.995	3.211	3.834	6.977	8.418	3.058	4.880	6.663	8.459	3.156	4.061	2.056	6.014	2.776	3.708	4.861
init=1.8	3.618	7.108	3.298	3.936	7.242	8.485	2.979	6.043	7.090	8.595	3.186	3.850	2.165	6.183	2.880	3.883	5.034
init=1.9	3.724	7.221	3.376	4.053	7.456	8.669	3.164	5.999	7.228	8.794	3.245	4.284	2.274	6.327	2.960	3.945	5.170
init=2.0	3.822	7.331	3.447	4.136	7.635	8.782	3.690	6.353	7.515	8.882	3.230	5.035	2.375	6.462	3.029	4.035	5.360

Table 16: Effect of different learning rate in KGEC on the performance of various KGE models across multiple datasets. For all the three metrics, the lower the better.

ECE	TransE				ComplEx				DistMult				RotatE				Average
	WN18	WN18RR	FB15K	FB15K-237	WN18	WN18RR	FB15K	FB15K-237	WN18	WN18RR	FB15K	FB15K-237	WN18	WN18RR	FB15K	FB15K-237	
lr=0.001	0.184	0.229	0.469	0.098	0.852	0.423	0.644	0.228	0.528	0.389	0.659	0.220	0.492	0.381	0.498	0.191	0.405
lr=0.002	0.172	0.231	0.517	0.052	0.852	0.423	0.608	0.227	0.527	0.388	0.596	0.218	0.476	0.371	0.456	0.092	0.388
lr=0.003	0.199	0.265	0.534	0.099	0.851	0.423	0.593	0.226	0.527	0.388	0.594	0.212	0.481	0.361	0.467	0.092	0.395
lr=0.004	0.203	0.276	0.535	0.138	0.851	0.422	0.595	0.222	0.526	0.387	0.605	0.191	0.478	0.353	0.480	0.098	0.398
lr=0.005	0.179	0.281	0.521	0.157	0.850	0.422	0.594	0.213	0.525	0.386	0.612	0.178	0.479	0.344	0.476	0.092	0.394
lr=0.006	0.202	0.286	0.520	0.160	0.849	0.421	0.678	0.207	0.522	0.386	0.683	0.178	0.477	0.333	0.483	0.091	0.405
lr=0.007	0.201	0.291	0.497	0.166	0.849	0.421	0.678	0.189	0.517	0.385	0.683	0.178	0.472	0.323	0.462	0.110	0.401
lr=0.008	0.188	0.287	0.491	0.163	0.846	0.419	0.678	0.189	0.495	0.384	0.683	0.178	0.466	0.317	0.455	0.127	0.398
lr=0.009	0.188	0.284	0.478	0.167	0.843	0.419	0.678	0.189	0.446	0.384	0.683	0.178	0.467	0.313	0.463	0.097	0.392
lr=0.010	0.171	0.280	0.459	0.150	0.833	0.418	0.678	0.189	0.446	0.383	0.683	0.178	0.467	0.307	0.466	0.094	0.388
lr=0.020	0.152	0.246	0.380	0.064	0.699	0.392	0.678	0.189	0.446	0.353	0.683	0.178	0.514	0.273	0.560	0.239	0.378
lr=0.030	0.184	0.228	0.460	0.080	0.699	0.374	0.677	0.189	0.446	0.349	0.683	0.178	0.562	0.289	0.606	0.238	0.390
lr=0.040	0.244	0.211	0.507	0.114	0.699	0.374	0.678	0.189	0.446	0.349	0.683	0.178	0.598	0.290	0.638	0.097	0.393
lr=0.050	0.305	0.189	0.530	0.139	0.699	0.374	0.677	0.189	0.446	0.349	0.682	0.178	0.623	0.295	0.652	0.238	0.410
lr=0.060	0.342	0.163	0.554	0.212	0.699	0.364	0.677	0.189	0.446	0.205	0.683	0.178	0.635	0.330	0.674	0.237	0.412
lr=0.070	0.393	0.153	0.569	0.164	0.699	0.374	0.677	0.190	0.526	0.349	0.683	0.178	0.656	0.330	0.679	0.239	0.429
lr=0.080	0.418	0.135	0.577	0.174	0.699	0.407	0.678	0.189	0.446	0.349	0.682	0.178	0.685	0.332	0.680	0.239	0.429
lr=0.090	0.456	0.130	0.584	0.184	0.826	0.374	0.677	0.189	0.446	0.385	0.683	0.178	0.716	0.350	0.693	0.237	0.444
lr=0.100	0.494	0.108	0.590	0.193	0.699	0.374	0.677	0.189	0.527	0.349	0.683	0.178	0.742	0.331	0.696	0.239	0.442

ACE	TransE				ComplEx				DistMult				RotatE				Average
	WN18	WN18RR	FB15K	FB15K-237	WN18	WN18RR	FB15K	FB15K-237	WN18	WN18RR	FB15K	FB15K-237	WN18	WN18RR	FB15K	FB15K-237	
lr=0.001	0.220	0.237	0.276	0.098	0.852	0.423	0.644	0.227	0.527	0.388	0.659	0.219	0.492	0.380	0.498	0.190	0.396
lr=0.002	0.171	0.236	0.248	0.037	0.851	0.423	0.587	0.226	0.527	0.388	0.562	0.216	0.459	0.370	0.456	0.069	0.364
lr=0.003	0.119	0.263	0.247	0.049	0.851	0.423	0.513	0.225	0.526	0.388	0.547	0.205	0.457	0.361	0.445	0.069	0.356
lr=0.004	0.108	0.273	0.252	0.075	0.851	0.422	0.486	0.220	0.525	0.387	0.499	0.198	0.457	0.352	0.437	0.063	0.350
lr=0.005	0.109	0.278	0.256	0.092	0.850	0.422	0.527	0.211	0.524	0.386	0.497	0.199	0.461	0.344	0.440	0.067	0.354
lr=0.006	0.107	0.283	0.264	0.095	0.849	0.421	0.465	0.208	0.521	0.386	0.516	0.199	0.463	0.333	0.435	0.065	0.351
lr=0.007	0.112	0.288	0.264	0.101	0.849	0.421	0.465	0.207	0.514	0.385	0.516	0.199	0.465	0.323	0.448	0.066	0.351
lr=0.008	0.122	0.284	0.272	0.097	0.845	0.419	0.465	0.207	0.488	0.384	0.516	0.199	0.466	0.317	0.452	0.071	0.350
lr=0.009	0.127	0.281	0.284	0.100	0.842	0.419	0.465	0.207	0.457	0.384	0.516	0.199	0.467	0.313	0.463	0.062	0.349
lr=0.010	0.131	0.277	0.293	0.082	0.833	0.418	0.465	0.207	0.457	0.383	0.516	0.199	0.467	0.306	0.466	0.063	0.348
lr=0.020	0.146	0.244	0.400	0.048	0.689	0.392	0.465	0.207	0.457	0.352	0.517	0.199	0.514	0.273	0.560	0.994	0.404
lr=0.030	0.163	0.227	0.471	0.075	0.689	0.160	0.484	0.207	0.457	0.238	0.517	0.199	0.562	0.289	0.606	1.010	0.397
lr=0.040	0.244	0.211	0.511	0.114	0.689	0.160	0.466	0.207	0.457	0.238	0.522	0.199	0.598	0.290	0.638	0.085	0.352
lr=0.050	0.305	0.191	0.535	0.139	0.689	0.160	0.486	0.207	0.457	0.238	0.529	0.199	0.623	0.295	0.652	1.115	0.426
lr=0.060	0.342	0.167	0.554	2.220	0.689	0.364	0.493	0.207	0.457	0.189	0.519	0.199	0.635	0.330	0.674	1.067	0.569
lr=0.070	0.393	0.157	0.569	0.164	0.689	0.160	0.493	0.212	0.526	0.238	0.518	0.199	0.656	0.330	0.679	0.987	0.436
lr=0.080	0.418	0.139	0.577	0.174	0.689	0.407	0.469	0.207	0.457	0.238	0.533	0.199	0.685	0.332	0.680	0.986	0.449
lr=0.090	0.456	0.136	0.584	0.184	0.826	0.160	0.503	0.208	0.457	0.383	0.517	0.200	0.716	0.350	0.693	1.142	0.470
lr=0.100	0.494	0.114	0.590	0.193	0.689	0.160	0.498	0.208	0.527	0.240	0.521	0.200	0.742	0.331	0.696	0.997	0.450

NLL	TransE				ComplEx				DistMult				RotatE				Average
	WN18	WN18RR	FB15K	FB15K-237	WN18	WN18RR	FB15K	FB15K-237	WN18	WN18RR	FB15K	FB15K-237	WN18	WN18RR	FB15K	FB15K-237	
lr=0.001	2.751	6.481	2.469	3.771	6.777	7.749	3.691	7.271	7.279	7.797	4.071	7.207	1.254	5.847	2.159	4.410	5.061
lr=0.002	2.505	6.267	2.341	3.233	6.634	7.665	3.006	6.945	7.073	7.729	2.871	6.351	1.014	5.546	1.997	2.862	4.627
lr=0.003	2.471	6.029	2.309	3.034	6.514	7.599	2.511	6.618	6.818	7.646	2.763	5.263	1.008	5.357	1.961	2.860	4.423
lr=0.004	2.468	5.984	2.311	2.924	6.292	7.493	2.301	5.853	6.526	7.562	2.353	3.975	1.010	5.217	1.934	2.703	4.182
lr=0.005	2.473	5.966	2.331	2.865	6.188	7.455	2.587	4.799	6.230	7.495	2.256	3.106	1.019	5.101	1.942	2.817	4.039
lr=0.006	2.464	5.947	2.353	2.853	5.874	7.356	1.357	4.372	5.670	7.383	1.319	3.106	1.024	4.965	1.927	2.793	3.798
lr=0.007	2.462	5.931	2.406	2.834	5.773	7.280	1.357	2.911	5.093	7.362	1.319	3.106	1.028	4.857	1.970	2.635	3.645
lr=0.008	2.466	5.942	2.444	2.845	5.288	7.094	1.357	2.911	4.088	7.253	1.319	3.106	1.032	4.800	1.984	2.573	3.531
lr=0.009	2.465	5.951	2.473	2.831	4.963	7.113	1.357	2.911	2.843	7.203	1.319	3.106	1.033	4.757	2.026	2.711	3.441
lr=0.010	2.462	5.965	2.536	2.889	4.350	6.965	1.357	2.911	2.843	7.119	1.319	3.106	1.036	4.698	2.033	2.743	3.396
lr=0.020	2.519	6.087	3.006	3.237	2.288	5.673	1.356	2.911	2.844	5.811	1.319	3.106	1.146	4.446	2.456	1.167	3.086
lr=0.030	2.579	6.160	3.362	3.546	2.288	3.211	1.351	2.910	2.844	3.254	1.320	3.106	1.274	4.564	2.756	1.167	2.856
lr=0.040	2.747	6.236	3.661	3.855	2.288	3.211	1.356	2.910	2.844	3.254	1.320	3.106	1.370	4.575	3.027	3.067	3.052
lr=0.050	2.896	6.335	3.839	4.096	2.288	3.211	1.352	2.910	2.844	3.254	1.315	3.106	1.436	4.608	3.173	1.159	2.989
lr=0.060	2.990	6.475	4.068	0.967	2.288	5.092	1.350	2.910	2.844	4.147	1.322	3.106	1.474	4.934	3.466	1.167	3.037
lr=0.070	3.149	6.532	4.254	4.399	2.288	3.211	1.350	2.909	6.683	3.253	1.321	3.106	1.531	4.920	3.546	1.177	3.352
lr=0.080	3.225	6.648	4.370	4.564	2.288	6.227	1.361	2.910	2.844	3.253	1.315	3.105	1.640	4.945	3.557	1.177	3.339
lr=0.090	3.366	6.692	4.495	4.759	4.059	3.211	1.350	2.910	2.844	7.289	1.319	3.105	1.767	5.153	3.803	1.171	3.581
lr=0.100	3.533	6.874	4.587	4.993	2.288	3.211	1.350	2.910	7.271	3.253	1.321	3.105	1.887	4.928	3.867	1.174	3.534